Proceedings of

DeRiVE 2015

4th International Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web

Co-located with the 12th Extended Semantic Web Conference (ESWC 2015), 31 May – 4 June, Portoroz, Slovenia

Preface

This volume contains the papers presented at DeRiVE2015: 4th Workshop on Detection Representation and Exploitation of Events in the Semantic Web held on Sunday May 31, 2015 in Portoroz (Co-located with the 12th Extended Semantic Web Conference - ESWC 2015)

In recent years, researchers in several communities involved in aspects of information science have begun to realise the potential benefits of assigning an important role to events in the representation and organisation of knowledge and media-benefits which can be compared to those of representing entities such as persons or locations instead of just dealing with more superficial objects such as proper names and geographical coordinates. While a good deal of relevant research for example, on the modeling of events has been done in the semantic web community, much complementary research has been done in other, partially overlapping communities, such as those involved in multimedia processing, information extraction, sensor processing and information retrieval research. However, these areas often deal with events with a different perspective. The attendance of DeRiVE 2011, DeRiVE 2012 and DeRiVE 2013 proved that there is a great interest from many different communities in the role of events. The results presented in there also indicated that dealing with events is still an emerging topic. The goal of this workshop is to advance research on the role of events within the information extraction and semantic web communities, both building on existing work and integrating results and methods from other areas, while focusing on issues of special importance for the semantic web.

We have defined questions for the two main directions that characterise current research into events on the semantic web. Orthogonal to that, we have identified a number of application domains in which we will actively seek contributions.

Question 1: How can events be detected and extracted for the semantic web?

- How can events be detected, extracted and/or summarized in particular types of content on the web, such as calendars of public events, social media, semantic wikis, and regular web pages?
- What is the quality and veracity of events extracted from noisy data such as microblogging sites?
- How can a system recognise a complex event that comprises several sub-events?
- How can a system recognise duplicate events?

Question 2: How can events be modelled and represented in the semantic web?

- How are events currently represented on the Web? In particular, how deployed is the schema.org Event class? Should scheduled events versus breaking events be represented the same way?

- To what extent can the many different event infoboxes of Wikipedia be reconciled? How to deal with the numerous Timeline of xxx topics in knowledge bases?
- How can existing event representations developed in other communities be adapted to the needs of the semantic web? To what extent can/should a unified event model be employed for different types of events?
- How do social contexts (Facebook, Twitter, etc.) change the implicit content semantics?

Application Domains: Research into detection (question 1) and representation (question 2) of events is being implemented in various application domains. Known application domains that we target are:

- Personal events
- Cultural and sports events
- Making something out of "raw" events
- Historic events and events in news and other media
- Scientific observation events
- Supply chain events

Among the submissions we received, 6 papers were selected for full presentation at the workshop:

- Jean-Paul Calbimonte and Karl Aberer - *Reactive Processing of RDF Streams of Events*
- Selver Softic, Laurens De Vocht, Erik Mannens, Martin Ebner and Rik Van de Walle - *COLINDA: Modeling, Representing and Using Scientific Events in the Web of Data*
- Michael Färber and Achim Rettinger - *Toward Real Event Detection*
- Gregory Katsios, Svitlana Vakulenko, Anastasia Krithara and Georgios Paliouras - *Towards Open Domain Event Extraction from Twitter: REVEALing Entity Relations*
- Loris Bozzato, Stefano Borgo, Alessio Palmero Aprosio, Marco Rospocher and Luciano Serafini - *A Contextual Framework for Reasoning on Events*
- Jacobo Rouces, Gerard de Melo and Katja Hose - *Representing specialized events with FrameBase*

We would like to thank the members of the program committee and the additional reviewers for their time and efforts. All papers included here have been revised and improved based on your valuable feedback, thus setting the basis for an exciting workshop programme.
Finally, we would like to thank our sponsor, the Newsreader (www.newsreader-project.eu) FP7 European Project, for funding the workshop.

May 31, 2015

Marieke Van Erp
Raphaël Troncy
Marco Rospocher
Willem Robert Van Hage
David A. Shamma

Program Committee

Eneko Agirre	University of the Basque Country, Spain
Stefano Borgo	LOA - CNR, Italy
Loris Bozzato	Fondazione Bruno Kessler
Christian Hirsch	University of Auckland, New Zealand
Jane Hunter	University of Queensland, Australia
Tomi Kauppinen	Aalto University, Finland
Azam Khan	Autodesk Research, Canada
Erik Mannens	Ghent University – IBBT, Belgium
Ingrid Mason	Intersect Australia Ltd
Diana Maynard	University of Sheffield, UK
Adrian Paschke	Freie Universiteit Berlin, Germany
Giuseppe Rizzo	EURECOM, France
Ansgar Scherp	Kiel University and Leibniz Information Center for Economics, Kiel, Germany
Ryan Shaw	University of North Carolina at Chapel Hill, USA
Thomas Steiner	Google Inc, Germany
Kerry Taylor	CSIRO & Australian National University
Denis Teyssou	Agence France-Presse

Reactive Processing of RDF Streams of Events

Jean-Paul Calbimonte and Karl Aberer

Faculty of Computer Science and Communication Systems
EPFL, Switzerland.
`firstname.lastname@epfl.ch`

Abstract. Events on the Web are increasingly being produced in the form of data streams, and are present in many different scenarios and applications such as health monitoring, environmental sensing or social networks. The heterogeneity of event streams has raised the challenges of integrating, interpreting and processing them coherently. Semantic technologies have shown to provide both a formal and practical framework to address some of these challenges, producing standards for representation and querying, such as RDF and SPARQL. However, these standards are not suitable for dealing with streams for events, as they do not include the concpets of streaming and continuous processing. The idea of RDF stream processing (RSP) has emerged in recent years to fill this gap, and the research community has produced prototype engines that cover aspects including complex event processing and stream reasoning to varying degrees. However, these existing prototypes often overlook key principles of reactive systems, regarding the event-driven processing, responsiveness, resiliency and scalability. In this paper we present a reactive model for implementing RSP systems, based on the Actor model, which relies on asynchronous message passing of events. Furthermore, we study the responsiveness property of RSP systems, in particular for the delivery of streaming results.

1 Introduction

Processing streams of events is challenging task in a large number of systems in the Web. Events can encode different types of information at different levels, e.g. concerts, financial patterns, traffic events, sensor alerts, etc., generating large and dynamic volumes of streaming data. Needless to say, the diversity and the heterogeneity of the information that they produce would make it impossible to interpret and integrate these data, without the appropriate tools. Semantic Web standards such as RDF[1] and SPARQL[2] provide a way to address these challenges, and guidelines exist to produce and consume what we know as Linked Data. While these principles and standards have already gained a certain degree of maturity and adoption, they are not always suitable for dealing with data streams. The lack of order and time in RDF, and its stored and bounded characteristics contrast with the inherently dynamic and potentially infinite nature

[1] RDF 1.1 Primer`http://www.w3.org/TR/rdf11-primer/`
[2] SPARQL 1.1 `http://www.w3.org/TR/sparql11-query/`

of the time-ordered streams. Furthermore, SPARQL is governed by one-time semantics as opposed to the continuous semantics of a stream event processor. It is in this context that it is important to ask *How can streaming events can be modeled and queried in the Semantic Web?*. Several approaches have been proposed in the last years, advocating for extensions to RDF and SPARQL for querying streams of RDF events. Examples of these RDF stream processing (RSP) engines include C-SPARQL [4], SPARQL$_{stream}$ [6], EP-SPARQL [3] or CQELS [11], among others.

Although these extensions target different scenarios and have heterogeneous semantics, they share an important set of common features, e.g. similar RDF stream models, window operators and continuous queries. There is still no standard set of these extensions, but there is an ongoing effort to agree on them in the community [3]. The RSP prototypes that have been presented so far focus almost exclusively in the query evaluation and the different optimizations that can be applied to their algebra operators. However, the prototypes do not consider a broader scenario where RDF stream systems can reactively produce and consume RDF events asynchronously, and deliver continuous results dynamically, depending on the demands of the stream consumer.

In this paper we introduce a model that describes RSP producers and consumers, and that is adaptable to the specific case of RSP query processing. This model is based on the *Actor Model*, where lightweight objects interact exclusively by interchanging immutable messages. This model allows composing networks of RSP engines in such a way that they are composable, yet independent, and we show how this can be implemented using existing frameworks in the family of the JVM (Java Virtual Machine) languages. In particular, we focus on specifying how RSP query results can be delivered in scenarios where the stream producer is faster than the consumer, and takes into account its demand to push only the volumes of triples that can be handled by the other end. This dynamic push delivery can be convenient on scenarios where receivers have lower storage and processing capabilities, such as constrained devices and sensors in the IoT. The remainder of the paper is structured as follows: we briefly describe RSP systems and some of their limitations in Section 2, then we present the actor-based model on Section 3. We provide details of the dynamic push delivery on Section 4, and the implementation and experimentation are described in Section 5. We present the related work on Section 6 before concluding in Section 7.

2 RSP Engines, Producers and Consumers

In general RSP query engines can be informally described as follows: given as input a set of RDF streams and graphs, and a set of continuous queries, the RSP engine will produce a stream of continuous answers matching the queries (see Figure 1). This high-level model of an RSP engine is simple yet enough to describe most stream query processing scenarios. Nevertheless, this model,

[3] W3C RDF Stream Processing Community Group `http://www.w3.org/community/rsp`

and the existing implementations of it, does not detail how stream producers communicate with RSP engines, and how stream consumers receive results from RSP engines. This ambiguity or lack of specification has resulted in different implementations that may result in a number of issues, especially regarding responsiveness, elasticity and resiliency.

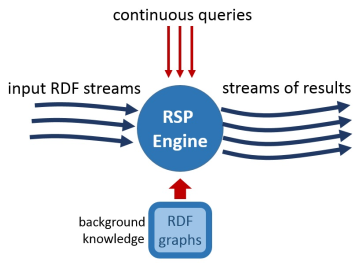


Fig. 1: Evaluation of continuous queries in RDF Stream Processing. The data stream flows through the engine, while continuous queries are registered and results that match them are streamed out.

2.1 RSP Query Engines

To illustrate these issues, let's consider first how streams are produced in these systems. On the producer side, RDF streams are entities to which the RSP engine subscribes, so that whenever a stream element is produced, the engine is notified (Figure 2). The issues with this model arise from the fact that the RSP engine and the stream producer are tightly coupled. In some cases like C-SPARQL or SPARQL$_{stream}$, the coupling is at the process level, i.e. both the producer and the engine coexist in the same application process. A first issue regards scalability: it is not possible to dynamically route the stream items from the producer to a different engine or array of engines, since the subscription is hard-wired on the code. Moreover, if the stream producer is faster than the RSP engine, the subscription notifications can flood the latter, potentially overflowing its capacity. A second issue is related to resilience: failures on the stream producer can escalate and directly affect or even halt the RSP engine.

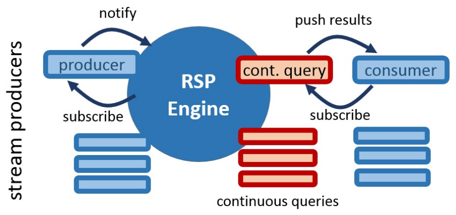


Fig. 2: Implementation of an RSP query engine based on tightly coupled publisher and subscribers.

Looking at the stream consumer side, the situation is similar. The continuous queries, typically implemented as SPARQL extensions, are registered into the RSP engine, acting as subscription systems. Then, for each of the continuous queries, a consumer can be attached so that it can receive notifications of the continuous answers to the queries (see Figure 2). Again, we face the problem of tightly coupled publisher and subscribers that have fixed routing configuration and shared process space, which may hinder the scalability, elasticity and resiliency of the system. Added to that, the delivery mode of the query results is fixed and cannot be tuned to the needs of the consumer.

It is possible to see these issues in concrete implementations: for instance in Listing 1 the C-SPARQL code produces an RDF stream. Here, the stream data structure is mixed with the execution of the stream producer (through a dedicated thread). Even more important, the tightly coupled publishing is done when the RDF quad is made available through the `put` method. The engine (in this case acting as a consumer) is forced to receive quad-by-quad whenever the RDF Stream has new data.

```
public class SensorsStreamer extends RdfStream implements Runnable {
  public void run() {
    while(true){
      RdfQuadruple q=new RdfQuadruple(subject,predicate,object,
                                      System.currentTimeMillis());
      this.put(q);
    }
  }
}
```

Listing 1: Example of generation of an RDF stream in C-SPARQL.

A similar scenario can be observed on query results recipient. The continuous listener code for the CQELS engine in Listing 2 represents a query registration (`ContinuousSelect`) to which one or more listeners can be attached. The subscription is tightly coupled, and results are pushed mapping by mapping, forcing the consumer to receive these updates and act accordingly.

```
String queryString =" SELECT ?person ?loc "
ContinuousSelect selQuery=context.registerSelect(queryString);
selQuery.register(new ContinuousListener() {
  public void update(Mapping mapping){
    String result="";
    for(Iterator<Var> vars=mapping.vars();vars.hasNext();){
       result+=" "+context.engine().decode(mapping.get(vars.next()));
       System.out.println(result);
    }
  }
});
```

Listing 2: Example of generation of an RDF stream in CQELS.

2.2 Results Delivery for Constrained Consumers

In the previous section we discussed some of the general issues of current RSP engines regarding producing and consuming RDF streams. Now we focus on the particular case where a stream consumer is not necessarily able to cope with the rate of the stream producer, and furthermore, when the stream generation rate fluctuates. As an example, consider the case of an RDF stream of annotated geo-located triples that mobile phones communicate to stationary sensors that detect proximity (e.g. for a social networking application, or for public transportation congestion studies), In this scenario the number of RDF stream producers can greatly vary (from a handful to thousands, depending on how many people are nearby in a certain time of the day), and also the stream rate can fluctuate.

In this and other examples the assumption that all consumers can handle any type of stream load does not always hold, and RSP engines need to consider this fact. Some approaches have used load shedding, eviction and discarding methods to alleviate the load, and could be applicable in these scenarios [1,9]. Complementary to that, it should be possible for stream producers to regulate the rate and the number of items they dispatch to a consumer, depending on the data needs and demand of the latter.

3 An Actor Architecture for RDF Stream Processing

A central issue in the previous systems is that several aspects are mixed into a single implementation. An RDF stream in these systems encapsulates not only the stream data structure, but also its execution environment (threading model) and the way that data is delivered (subscriptions). In distributed systems, one of the most successful models for decentralized asynchronous programming is the *Actor model* [2,10]. This paradigm introduces *actors*, lightweight objects that communicate through messages in an asynchronous manner, with no-shared mutable state between them. Each actor is responsible of managing its own state, which is not accessible by other actors. The only way for actors to interact is through asynchronous and immutable messages that they can send to each other either locally or remotely, as seen in figure 3.

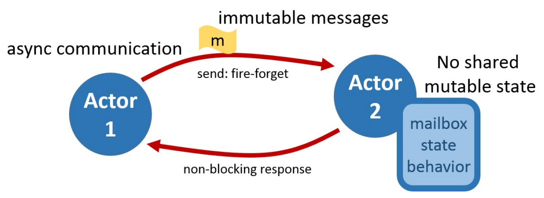


Fig. 3: Actor model: actors communicate through asynchronous messages that arrive to their mailboxes. There is no shared mutable state, as each actor handles its own state exclusively.

We can characterize an actor A as a tuple: $A = (s, b, mb)$, where s is the actor state, b is the actor behavior and mb is its message box. The state s is accessible and modifiable only by the actor itself, and no other Actor can either read or write on it. The mailbox mb is a queue of messages m_i that are received from other actors. Each message $m_i = (a_i^s, a_i^r, d_i)$ is composed of a data item d_i, a reference to the sender actor a_i^s, and a reference to the receiver actor a_i^r. The behavior is a function $b(m_i, s)$ where m_i is a message received through the mailbox. The behavior can change the actor state depending on the message acquired. Given a reference to an actor a, an actor can send a message m_i through the $send(m_i, a)$ operation. References to actors can be seen as addresses of an actor, which can be used for sending messages.

We propose a simple execution model for RDF stream processing that is composed of three generic types of actors: a stream producer, a processor and a consumer, as depicted in Figure 4. A producer actor generates and transmits messages that encapsulate RDF streams to the consumer actors. The processor actor is a special case that implements both a producer (producer of results) and a consumer (consumes the input RDF streams), as well as some processing logic. Following the above definitions the data d_i of a message m_i emitted by a producer actor, or received by a consumer actor, is a set of timestamped triples. This model does not prevent these actors to receive and send also other types of messages.

In this model there is a clear separation of the data and the execution: the data stream is modeled as an infinite sequence of immutable event messages, each containing a set of RDF triples. Communication between producers and consumers is governed through asynchronous messaging that gets to the mailboxes of the actors. In that way, the subscribers are not tightly coupled with the producers of RDF streams, and in fact any consumer can feed from any stream generated by any producer. Moreover, this separation allows easily isolating failures in either end. Failures on consumers do not directly impact other consumers nor the producers, and vice-versa.

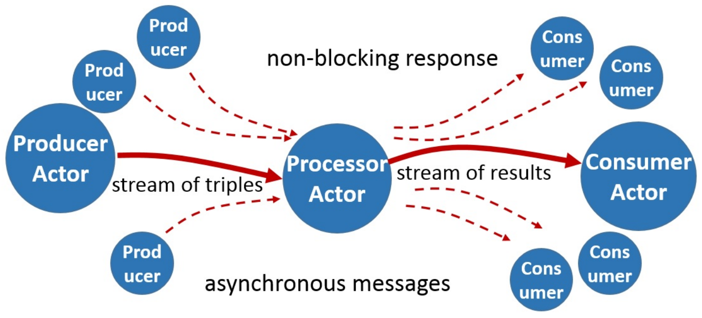


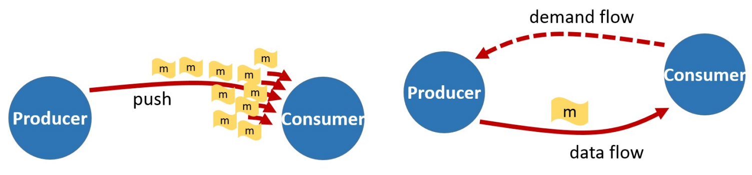
Fig. 4: RSP actors: RDF stream producers, processors and consumers. All actors send the stream elements as asynchronous messages. An RSP query engine is both a consumer (receives an input RDF stream) and a producer (produces a stream of continuous answers).

Event-driven asynchronous communication within RSP actors, as well as avoiding blocking operators, guarantees that the information flow is not stuck unnecessarily. Also, adaptive delivery of query results using dynamic push and pull, can prevent data bottlenecks and overflow, as we will see later. By handling stream delays, data out of order and reacting gracefully to failures, the system can maintain availability, even under stress or non-ideal conditions. Similarly, elasticity can boost the system overall responsiveness by efficiently distributing the load and adapting to the dynamic conditions of the system. The actor model results convenient for RDF stream processing, as it constitutes a basis for constructing what is commonly called a *reactive system*[4]. Reactive systems are characterized for being *event-driven*, *resilient*, *elastic*, and *responsive*.

[4] The reactive manifesto `http://www.reactivemanifesto.org/`

4 Dynamic Push Delivery

In RSP engines there are typically two types of delivery modes for the stream of results associated to a continuous query: *pull* and *push*. In pull mode, the consumer actively requests the producer for more results, i.e. it has control of when the results are retrieved. While this mode has the advantage of guaranteeing that the consumer only receives the amount and rate of data that it needs, it may incur in delays that depend on the polling frequency. In the push mode, on the contrary, the producer pushes the data directly to the consumer, as soon as it is available. While this method can be more responsive and requires no active polling communication, it forces the consumer to deal with bursts of data, and potential message flooding. In some cases, when the consumer is *faster* than the producer, the push mode may be appropriate, but if the rate of messages exceeds the capacity of the consumer, then it may end up overloaded, causing system disruption, or requiring shedding or other techniques to deal with the problem (see Figure 5a).



(a) Push: overload if the producer pushes too fast. **(b)** Dynamic push: demand on the side of the consumer.

Fig. 5: Delivery modes in RSP engines.

As an alternative, we propose using a dynamic push approach for delivering stream items to an RDF stream consumer, taking into consideration the capacity and availability of the latter (see Figure 5b). The dynamic mechanism consists in allowing the consumer to explicitly indicate its demand to the producer. This can be simply done by issuing a message that indicates the capacity (e.g. volume of data) that it can handle. Then, knowing the demand of the consumer, the stream producer can push only the volume of data that is required, thus avoiding any overload on the consumer side. If the demand is lower than the supply, then this behavior results in a normal push scenario. Otherwise, the consumer can ask for more data, i.e. pull, when it is ready to do so. Notice that the consumer can at any point in time notify about its demand. If the consumer is overloaded with processing tasks for a period of time, it can notify a low demand until it is free again, and only then raise it and let the producer know about it.

5 Implementing RSP Dynamic Push

In order to validate the proposed model, and more specifically, to verify the feasibility of the dynamic push in a RSP engine, we have implemented this mechanism on top of an open-source RSP query processor. We have used the Akka library[5], which is available for both Java and Scala, to implement our

[5] Akka: `http://akka.io/`

RSP Actors. Akka provides a fully fledged implementation of the actor model, including routing, serialization, state machine support, remoting and failover, among other features. By using the Akka library, we were able to create producer and consumer actors that receive messages, i.e. streams of triples. For example, a Scala snippet of a consumer is detailed in Listing 3, where we declare a consumer that extends the Akka Actor class, and implements a `receive` method. The receive method is executed when the actor receives a message on its mailbox, i.e. in our case an RDF stream item.

```
class RDFConsumer extends Actor {
  def receive ={
    case d:Data =>
      // process the triples in the data message
  }
}
```

Listing 3: Scala code snippet of an RDF consumer actor.

To show that an RSP engine can be adapted to the actor model, we have used CQELS, which is open source and is written in Java, as it has demonstrated to be one of the most competitive prototype implementations, at least in terms of performance [12]. More concretely, we have added the dynamic push delivery of CQELS query results, so that a consumer actor can be fed with the results of a CQELS continuous query.

To show the feasibility of our approach and the implementation of the dynamic push, we used a synthetic stream generator based on the data and vocabularies of the SRBench [15] benchmark for RDF stream processing engines. As a sample query, consider the CQELS query in Listing 4 that constructs a stream of triples consisting of an observation event and its observed timestamp, for the last second.

```
  PREFIX omOwl: http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#.
  CONSTRUCT {?observation <http://epfl.ch/stream/produces> ?time}
  WHERE {
     STREAM <http://deri.org/streams/rfid> [RANGE 1000ms] {
       ?observation omOwl:timestamp ?time
     }
  }
```

Listing 4: Example of generation of CQELS query over the SRBench dataset.

In the experiments, we focused on analyzing the processing throughput of the CQELS dynamic push, compared to the normal push operation. We tested using different processing latencies, i.e. considering that the processing on the consumer side can cause a delay of 10, 50, 100 and 500 milliseconds. This simulates a slow stream consumer, and we tested its behavior with different values for the fluctuating demand: e.g. from 5 to 10 thousand triples per execution. The results of these experiments are depicted in Figure 6, where each plot corresponds to a different delay value, the Y axis is the throughput, and the X axis is the demand of the consumer.

As it can be seen, when the demand of the consumer is high, the behavior is similar to the push mode. However if the consumer specifies a high demand

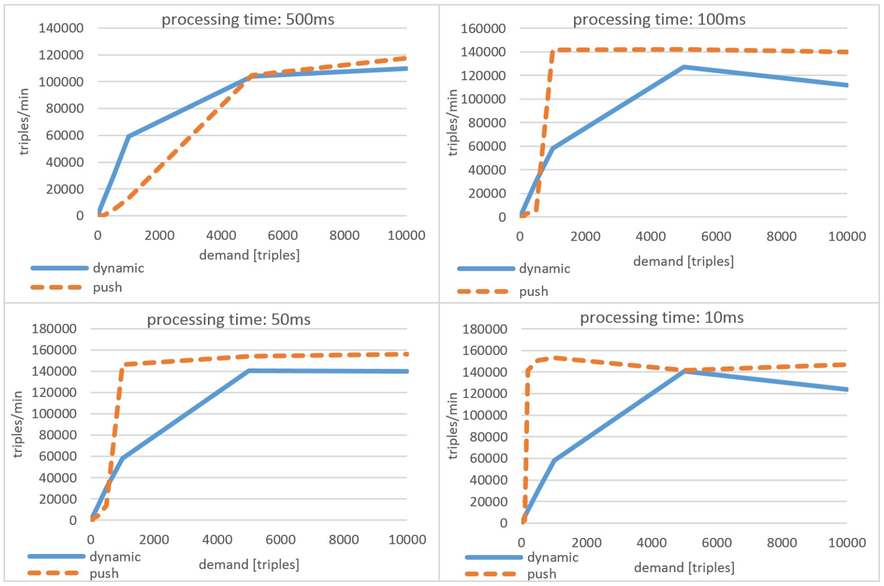


Fig. 6: Results of the experimentation: throughput of the results delivery after query processing for bot dynamic and normal push. The delay per processing execution is of 500, 100, 50, 10 milliseconds from left to right, top to bottom. The Y axis is the throughput, and the X axis is the demand.

but has a slow processing time, the throughput is slowly degraded. When the processing time is fast (e.g. 10 ms), the push delivery throughput is almost constant, as expected, although it is important to notice that in this mode, if the supply is greater than the demand, the system simply drops and does not process the exceeding items. In that regard, the dynamic push can help alleviate this problem, although it has a minor penalty in terms of throughput.

6 Related Work & Discussion

RDF stream processors have emerged in the latest years as a response to the challenge of producing, querying and consuming streams of RDF events. These efforts resulted in a series of implementation and approaches in this area, proposing their own set of stream models and extensions to SPARQL [5, 11, 7, 3, 9]. These and other RSP engines have focused on the execution of SPARQL streaming queries and the possible optimization and techniques that can be applied in that context. However, their models and implementation do not include details about the stream producers and consumers, resulting in prototypes that overlook the issues described in Section 2.

For handling continuous queries over streams, several Data Stream Management Systems (DSMS) have been designed and built in the past years, exploiting the power of continuous query languages and providing pull and push-based data access. Other systems, cataloged as complex event processors (CEP), emphasize on pattern matching in query processing and defining complex events from basic ones through a series of operators [8]. Although none of the commercial CEP solutions provides semantically rich annotation capabilities on top of their query interfaces, systems as the ones dexfibed in [14, 13] have proposed different types of semantic processing models on top of CEPs.

More recently, a new sort of stream processing platforms has emerged, spinning off the massively parallel distributed Map-Reduce based frameworks. Examples of this include Storm[6] or Spark Streaming[7], which represent stream processing as workflows of operators that can be deployed in the cloud, hiding the complexity of parallel and remote communication. The actor based model can be complementary to such platforms (e.g. Spark Streaming allows feeding streams from Akka Actors on its core implementation).

7 Conclusions

Event streams are one of the most prevalent and ubiquitous source of Big Data on the web, and it is a key challenge to design and build systems that cope with them in an effective and usable way. In this paper we have seen how RDF Stream Processing engines can be adapted to work in an architecture that responds to the principles of reactive systems. this model is based on the usage of lightweight actors that communicate via asynchronous event messages. We have shown that using this paradigm we can avoid the tight coupled design of current RSP engines, while opening the way for building more resilient, responsive and elastic systems. More specifically, we have shown a technique for delivering the continuous results of queries in an RSP engine through a dynamic push that takes into consideration the demand of the stream consumer. The resulting prototype implementation, on top of the well known CQELS engine, shows that is feasible to adapt an RSP to include this mode, while keeping a good throughput.

When processing streams of data, whether they are under the RDF umbrella or not, it is important to take architectural decisions that guarantee that the system aligns with the characteristics of a reactive system. Otherwise, regardless of how performant a RSP engine is, if it is not able to be responsive, resilient to failures and scalable, it will not be able to match the challenges of streaming applications such as the Internet of Things. we have seen that there are many pitfalls in systems design that prevent most of RSP engines to be *reactive*, in the sense that they do not always incorporate the traits of resilience, responsiveness, elasticity and message driven nature. We strongly believe that these principles have to be embraced at all levels of RDF stream processing to be successful.

As future work, we plan to extend the reactive actor model to all aspects of an RSP engine, including the stream generation, linking with stored datasets and dealing with entailment regimes. We also envision to use this architecture to show that different and heterogeneous RSP engines can be combined together, forming

[6] `http://storm.apache.org/`

[7] `https://spark.apache.org/streaming/`

a network of producers and consumers that can communicate via messaging in a fully distributed scenario.

Acknowledgments Partially supported by the SNSF-funded Osper and Nano-Tera OpenSense2 projects.

References

1. Abadi, D.J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. The VLDB Journal 12(2), 120–139 (August 2003)
2. Agha, G.: Actors: A model of concurrent computation in distributed systems. Tech. rep., MIT (1985)
3. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW, pp. 635–644 (2011)
4. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: SPARQL for continuous querying. In: WWW, pp. 1061–1062 (2009)
5. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Proc. 7th Extended Semantic Web Conference, pp. 1–15 (2010)
6. Calbimonte, J.P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: ISWC, pp. 96–111 (2010)
7. Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Enabling query technologies for the semantic sensor web. International Journal On Semantic Web and Information Systems (IJSWIS) 8(1), 43–63 (2012)
8. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Computing Surveys 44(3), 15:1–15:62 (2011)
9. Gao, S., Scharrenbach, T., Bernstein, A.: The clock data-aware eviction approach: Towards processing linked data streams with limited resources. In: ESWC, pp. 6–20. Springer (2014)
10. Karmani, R.K., Shali, A., Agha, G.: Actor frameworks for the jvm platform: a comparative analysis. In: Proceedings of the 7th International Conference on Principles and Practice of Programming in Java. pp. 11–20. ACM (2009)
11. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: ISWC, pp. 370–388 (2011)
12. Le-Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. Web Semantics: Science, Services and Agents on the World Wide Web 16, 42–51 (2012)
13. Paschke, A., Vincent, P., Alves, A., Moxey, C.: Tutorial on advanced design patterns in event processing. In: DEBS. pp. 324–334. ACM (2012)
14. Taylor, K., Leidinger, L.: Ontology-driven complex event processing in heterogeneous sensor networks. In: ISWC, pp. 285–299. Springer (2011)
15. Zhang, Y., Duc, P., Corcho, O., Calbimonte, J.P.: SRBench: A Streaming RDF/SPARQL Benchmark. In: ISWC, pp. 641–657. Springer (2012)

COLINDA: Modeling, Representing and Using Scientific Events in the Web of Data

Selver Softic[1], Laurens De Vocht[2], Martin Ebner[1],
Erik Mannens[2], and Rik Van de Walle[2]

[1] Graz University of Technology, Inffeldgasse 16c, 8010 Graz, Austria
{selver.softic,martin.ebner}@tugraz.at
[2] Ghent University, iMinds - Multimedialab,
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
{laurens.devocht,erik.mannens,rik.vandewalle}@ugent.be

Abstract. Conference Linked Data (COLINDA)[3], a recent addition to the LOD (Linked Open Data) Cloud[4], exposes information about scientific events (conferences and workshops) for the period from 2002 up to 2015. Beside title, description and time COLINDA includes venue information of scientific events which is interlinked with Linked Data sets of GeoNames[5], and DBPedia[6]. Additionally information about events is enhanced with links to corresponding proceedings from DBLP (L3S)[7] and Semantic Web Dog Food [8] repositories. The main sources of COLINDA are WikiCfP[9] and Eventseer[10]. The research questions addressed by this work in particular are: how scientific events can be extracted and summarized from the Web, how to model them in Semantic Web to be useful for mining and adapting of research related social media content in particular micro blogs, and finally how they can be interlinked with other scientific information from the Linked Data Cloud to be used as base for explorative search for researchers .

Keywords: Linked Data, Scientific Events, Linked Science, Research 2.0

1 Introduction and Motivation

COLINDA[11] contains information about scientific events worldwide (including location and proceedings references), published as Linked Data. The data contained in COLINDA is extracted and accumulated from the data dumps of WikiCfP , which are published yearly and freely available on request for research[12] purposes, and from data

[3] http://colinda.org
[4] http://lod-cloud.net/
[5] http://www.geonames.org/
[6] http://dbpedia.org
[7] http://dblp.l3s.de/d2r/
[8] http://data.semanticweb.org/
[9] http://www.wikicfp.com/
[10] http://eventseer.net/
[11] Available at: http://colinda.org/, see also http://datahub.io/dataset/colinda
[12] http://www.wikicfp.com/cfp/data.jsp

gathered via JSON interface from Eventseer. WikiCfP and Eventseer are two very popular online scientific event archives. WikiCfP contains calls for paper for about approximately 30.000 conferences and has approximately 100.000 registered users. Eventseer contains according the latest information[13] calls for around 21000 events and serves more then 1 million users. We also track the Twitter[14] feeds of both sites integrating on the fly arrival of upcoming scientific events using the Twitter API[15] to recieve the data from Twitter profiles of Wiki CfP and Eventseer. Currently COLINDA includes data about more than 15000 conferences. Event instances are enriched with information from Linked Data proceedings repositories DBLP (L3S)[16] and Semantic Web Dog Food[17] as well by location information from Geonames and DBPedia. Primary intention of COLINDA was to provide hashtag based identification system for scientific events in Twitter in the manner of the "5-star" quality Open Data[18]. Researchers are using very often hashtags, while they are discussing on Twitter. Specially during scientific events, they are using hashtags as abbreviated reference to the event they are attending [6]. E.g. ISWC (International Semantic Web Conference) 2012 is often referred as "iswc12" or "iswc2012". DBLP (L3S) Linked Dataset and Semantic Web Dog Food also use this kind of notation to reference the event of conference proceedings[19],[20]. The overall idea of COLINDA is to serve as mining reference for creation of semantically driven microblog data Mash Ups for Research 2.0 and as interlinking hub for other science relevant sources from the LOD cloud in order to enhance explorative search for researchers. Efforts made in this field using COLINDA will be introduced in detail in section 3.

2 Extraction, Modeling, Creation and Publishing of Linked Scientific Events

COLINDA data covers generally three domains: The first domain originates from WikiCfP and Eventseer and describes the **Conference** as basic scientific event with a start date, location, description, label and link to the event web page. Second domain is the **Location** of the event with geographic parameters resolved using the GeoNames and DBPedia data set in interlinking process. Each location contains reference to the city, country and coordinates of the location. Further as extension and third domain we have **Proceedings** of the conference represented by the links from DBLP (L3S) or Semantic Web Dog Food.

2.1 Linked Scientific Events Creation Process

The data creation process comprises the following steps:

[13] http://eventseer.net/data/
[14] http://www.twitter.com/
[15] COLINDA
[16] http://datahub.io/dataset/l3s-dblp
[17] http://datahub.io/dataset/semantic-web-dog-food
[18] http://5stardata.info/
[19] e.g. for 'iswc2012' at DBLP(L3S): http://dblp.l3s.de/d2r/page/publications/conf/ISWC/2012
[20] e.g. for 'iswc2012' at SW Doog Food: http://data.semanticweb.org/conference/iswc/2012/

- Extraction - extraction and pre-processing of data sources (Subsection 2.2)
- Modeling of Events using SWRC Ontology - concept coverage (Subsection 2.3)
- Triplification - creating RDF data triples (Subsection 2.4)
- Interlinking - connection to other Linked Data sets (Subsection 2.5)

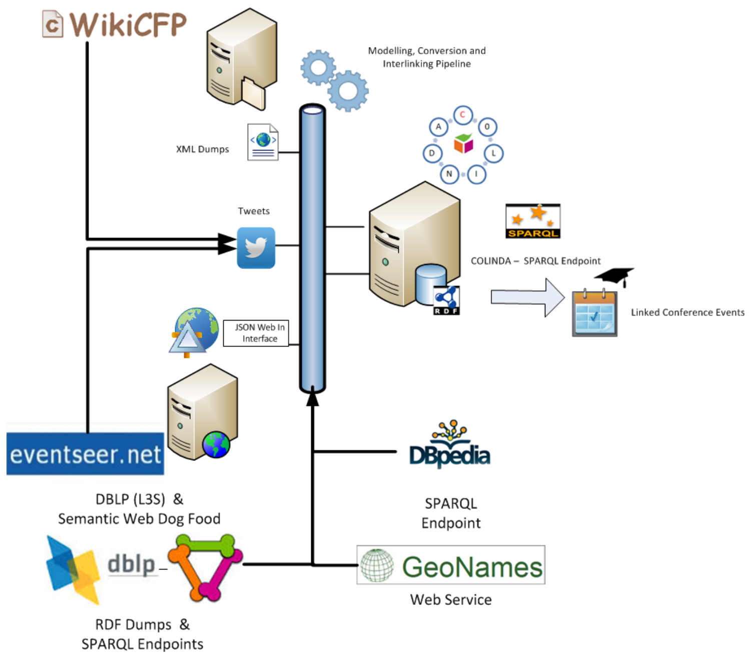

WikiCFP
Modelling, Conversion and Interlinking Pipeline
XML Dumps
Tweets
SPARQL
COLINDA – SPARQL Endpoint
RDF
Linked Conference Events
JSON Web In Interface
eventseer.net
DBpedia
DBLP (L3S) & Semantic Web Dog Food
SPARQL Endpoint
dblp
GeoNames
Web Service
RDF Dumps & SPARQL Endpoints

Fig. 1. Creation process of linked scientific events.

2.2 Data Extraction

COLINDA is constructed from variously structured sources. Therefore we defined a minimal set of properties that describe the **Conference** concept for a single RDF instance. During extraction, all properties from sources are being mapped to defined normalized set in order to harmonize the federated data. The **Location** and **Proceedings** concepts related to conference events as such are considered as optional enrichment which will be treated in the interlinking process. We made this decision having in mind that all conference descriptions do not explicitly include the venue information. The quality of source data depends on the users that provide the information. Thus such data sources implicitly exclude assumption of completeness. Table 1 represents the minimal set of properties a **Conference** and **Location** instance should include. The Extraction process includes steps of either pre-processing of XML dumps from WikiCfP or JSON from Tweets and Eventseer into the temporary tables of values formatted as Comma

Separated Value (CSV). During the pre-processing cycle data fields like e.g. date or labels are being normalized to achieve uniform representation, and to provide easier processable input for triplification step which converts the extracted values from temporary tables into RDF formatted instances of Linked Data.

Table 1. Harmonized COLINDA - minimal properties set. Entries denoted with * are optional.

Concept	*Property*
Conference	*label*
	title
	description
	*date**
	*link**
	*location**
Proceedings	*proceedings**
Location	*placename*
	city
	country
	longitude
	latitude

2.3 Modeling Scientific Events in the Web of Data

Basic representation of scientific events was well elaborated in previous research work about the SWRC ontology introduced by Sure et al [7]. This practice has been already approved and adapted by the implementation of Linked Data proceedings repositories DBLP (L3S) and Semantic Web Dog Food. We also followed the good practice of re-using existing vocabularies before we define our own. Minimal field set defined in table 1 for RDF instance generation matches well the range of SWRC concepts. Therefore, we have chosen the SWRC Ontology[21] and basic RDFS Schema[22] as established vocabularies to describe **Conference** instances. The same approach was applied for **Location** concept; needed set of geographical features to describe conference venues is well covered by elements from GeoNames[23] and Basic Geo (WGS84) Vocabulary[24]. Complete model with interlinked properties (proceeding and location) can be seen in figure 2, where a single complete and interlinked instance of a conference (ISWC2012) is depicted. Matchings between features and the vocabulary properties is shown in table 2.

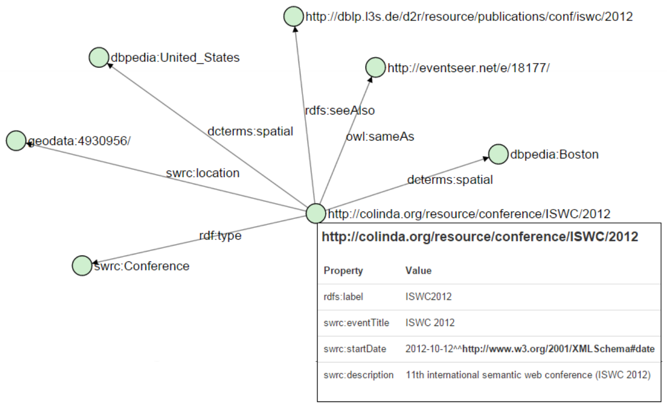
2.4 Triplification - Creation of RDF Instances of Scientific Events

The triplification[25] process uses as input temporary data tables in CSV like format generated in extraction and pre-processing step. Input generated in this way represents tab-

[21] http://ontoware.org/swrc/
[22] http://www.w3.org/TR/rdf-schema/
[23] http://www.geonames.org/ontology/
[24] http://www.w3.org/2003/01/geo/wgs84_pos#
[25] Under 'triplification' we understand 'triple-wise' creation of Linked Data instances as RDF graphs

Table 2. COLINDA concept to ontology model mapping (note: geonames - GeoNames Ontology, geo - W3C GEO Vocabulary, swrc - SWRC Ontology). Entries denoted with * are optional.

Concept/Property	***RDF Class/Property***
Conference	swrc:Conference
label	rdfs:label
title	swrc:eventTitle
description	swrc:description
*date**	swrc:startDate
*link**	owl:sameAs
*location reference**	swrc:location
*location reference**	dcterms:spatial
Proceedings*	rdfs:seeAlso
Location*	geo:SpatialThing
*placename**	geonames:P
*city**	geonames:name
*country**	geonames:countryName
*longitude**	geo:long
*latitude**	geo:lat



http://colinda.org/resource/conference/ISWC/2012

Property	Value
rdfs:label	ISWC2012
swrc:eventTitle	ISWC 2012
swrc:startDate	2012-10-12^^**http://www.w3.org/2001/XMLSchema#date**
swrc:description	11th international semantic web conference (ISWC 2012)

Fig. 2. Sample interlinked **Conference** RDF instance of ISWC 2012 generated by Visual RDF.

ular set of values compatible with properties from table 1. This input is then parsed line by line and conference instance is generated as single RDF graph using the vocabulary properties defined in table 2. Each conference instance is accessible via REST (Representational State Transfer) call as described in subsection 2.6. To make them accessible by SPARQL endpoint, background batch process loads the conference instances into the ARC2[26] RDF triple store running on the server.

2.5 Interlinking to Other Interesting Sources

In order to provide 5-star data and led by the design issues described in [1], we used ***swrc:location*** as interlinking property in order to interlink the location data with GeoNames. The interlinking process uses GeoNames query service to resolve geographical

[26] https://github.com/semsol/arc2/

information and retrieve coordinates. Although usually ***owl:sameAs*** is used to interlink to other data set we used this property to resolve the connection to the conference web page and since ***swrc:location*** seems regarding the GeoNames to be more appropriate choice. How this connection looks like can be seen in the sample depicted in figure 2 as well as online[27],[28]. Further we use dumps of DBPedia and Semantic Web Dog Food to enhance the instances with DBPedia location info using the ***dcterms:spatial*** property and for interlinking the proceedings from DBLP (L3S) ans Semantic Web Dog Food we match the conference's ***rdfs:label*** to the corresponding labels in those data sets via SPARQL queries. In matching case a link is established with correlating results using the ***rdfs:seeAlso*** property.

2.6 URI Design and Public Accessibility

Access to instances of COLINDA is possible via URIs with following pattern:

– http://colinda.org/resource/conference/{label}/{year}

All responses from COLINDA are formatted as RDF/XML fragment. Other supported formats are: HTML, Text, N3, NTRIPLES format[29]. Alternative access offers the SPARQL [30] endpoint. Current endpoint supports up to 250000 result triples per query and delivers results in different formats like: JSON, RDF/XML, XML, TSV etc. How to query the endpoint is shown by simple example in listings 1.1. Results from the query return the COLINDA link, city, country and the geo-location of ISWC 2012 conference. Recently, a dump of COLINDA was made available as Linked Data Fragments[31]. COL-

Listing 1.1. Sample SPARQL query for retrieval of conference (geo) location.

```
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX gn: <http://www.geonames.org/ontology#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf—schema#>
SELECT DISTINCT ?x ?city ?country ?long ?lat
{
 ?x rdfs:label "ISWC2012";
    swrc:location ?loc.
    OPTIONAL
    {
               ?loc gn:name ?city;
                  gn:countryName ?country;
                  geo:lat ?lat;
                  geo:long ?long.
    }
}
```

INDA RDF data dumps are also accessible via the CKAN Registry[32] of LOD Cloud.

[27] http://www.colinda.org/resource/conference/ISWC/2012?format=html
[28] http://graves.cl/visualRDF/?url=www.colinda.org/resource/conference/ISWC/2012
[29] e.g. http://www.colinda.org/resource/conference/ISWC/2012?format=html
[30] http://colinda.org/endpoint.php
[31] http://data.linkeddatafragments.org/colinda#dataset
[32] http://datahub.io/dataset/colinda

2.7 Actuality of Data

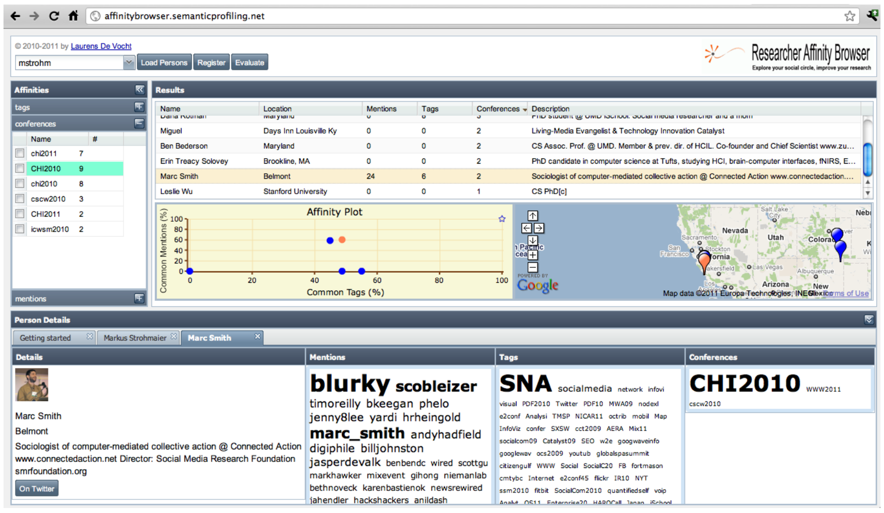
COLINDA is kept up-to-date by a daily cron job which grabs the newest event announcements over the Twitter API for accounts of WikiCfP and Eventseer. The cron job parses, creates, interlinks and synchs new events into the triple store. Each tweet also includes information about the call page link which allows retrieval of the extended information about events via web (WikiCfP) or available JSON (Eventseer) interface during the update task. Additionally to the automated job, also manual updates are ran as soon as the fresh dumps from both sites are available.

3 Applications and Use Cases

Both use cases introduced in following subsections address the challenges of Research 2.0. Research 2.0 as adaptation of the Web 2.0 for researchers defines researchers as main consumers of the information. The purpose of these research activities is to offer a set of tools and services which researcher can use to discover resources, such as publications or events they might be interested in, as well as to collaborate with each other via the web. These tools and services, according to the specifications of Research 2.0, are considered as Mash Ups, APIs, publishing feeds and specially designed interfaces based on social profiles [5, 8]. The role of COLINDA is addressed separately in application description.

3.1 Affinity Browser

The "Researcher Affinity browser" was developed as a tool to demonstrate semantically driven aggregation of microbolog data for Research 2.0. (use of Web 2.0 tools in scientific research). In this context, COLINDA was used as mining source for the faceted detection of similar scientist Twitter profiles based upon conferences they visited as special affinity criteria. This is done by matching the COLINDA tags with the hashtags of the Twitter user. Adequate demo video showing the "Researcher Affinity Browser" in action can be also viewed online[33]. The "Researcher Affinity Browser Application" [4] is depicted in figure 3. At the beginning it retrieves a list of relevant users. Those results represent a current snapshot which means that every time users produce new tweets on Twitter, the analysis result evolves with it. The relevance is measured according to the number of common conceptual affinities. Different affinity facets are displayed on the left. Users can explore three types of affinities: conferences, tags and mentions. Activation of a certain affinity filters the list of matching persons. There is the result table that displays detailed information about each person and how many affinities are shared. Further there is a map view and an affinity plot synchronized with the result table. The purpose of the map is to get a better impression of where the affiliations of the found persons would lie. The affinity plot visualizes in a quick overview affinity correspondence between the analyzed profile and other profiles in the system.


affinitybrowser.semanticprofiling.net
© 2010-2011 by Laurens De Vocht
mstrohm
Load Persons
Register
Evaluate
Researcher Affinity Browser
Explore your social circle, improve your research
Affinities
tags
conferences
mentions
Results
Affinity Plot
Common Mentions (%)
Common Tags (%)
Person Details
Getting started
Markus Strohmaier
Marc Smith
Details
Mentions
Tags
Conferences
On Twitter
blurky scobleizer timoreilly bkeegan phelo jenny8lee yardi hrheingold marc_smith andyhadfield digiphile billjohnston jasperdevalk benbendc wired scottgu markhawker mixevent gihong niemanlab bethnoveck karenbastienok newsrewired jahendler hackshackers anildash
SNA socialmedia network infovi visual PDF2010 Twitter PDF10 MWA09 nodexl e2conf Analysi TMSP NICAR11 octrib mobil Map InfoViz confer SXSW cct2009 AERA Mix11 socialcom09 Catalyst09 SEO w2e googwaveinfo googlewav ocs2009 youtub globalspasummit citizengulf WWW Social SocialC20 FB fortmason cmtybc Internet e2conf45 flickr IR10 NYT ssm2010 fitbit SocialCom2010 quantifiedself voip
CHI2010 WWW2011 cscw2010

Name	#
chi2011	7
CHI2010	9
chi2010	8
cscw2010	3
CHI2011	2
icwsm2010	2

Name	Location	Mentions	Tags	Conferences	Description
Miguel	Days Inn Louisville Ky	0	0	2	Living-Media Evangelist & Technology Innovation Catalyst
Ben Bederson	Maryland	0	0	2	CS Assoc. Prof. @ UMD. Member & prev. dir. of HCIL. Co-founder and Chief Scientist www.zu...
Erin Treacy Solovey	Brookline, MA	0	0	2	PhD candidate in computer science at Tufts, studying HCI, brain-computer interfaces, fNIRS, E...
Marc Smith	Belmont	24	6	2	Sociologist of computer-mediated collective action @ Connected Action www.connectedaction....
Leslie Wu	Stanford University	0	0	1	CS PhD[c]

Marc Smith
Belmont
Sociologist of computer-mediated collective action @ Connected Action www.connectedaction.net Director: Social Media Research Foundation smrfoundation.org

Fig. 3. "Researcher Affinity Browser Application" snapshot.

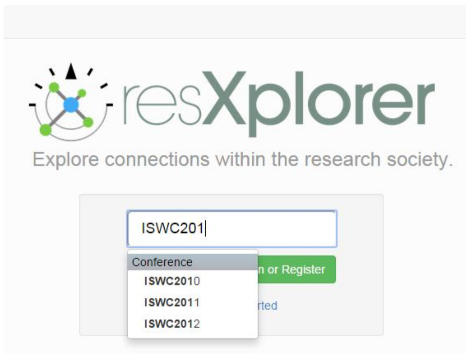


Fig. 4. Mapping of keywords

3.2 ResXplorer

"ResXplorer"[34] is an Research 2.0 [8] aggregated interface for search and exploration of the underlying Linked Data Knowledge Base. A demo video explaining the interface shown in figure 5 is available online[35]. Data from Linked Data Knowledge Base originates from: DBLP (L3S)[36] which is a bibliography of computer science conference proceedings, COLINDA[37] which is a main binding hub data set including informa-

[33] http://www.youtube.com/watch?v=A25DrP3Mv8w
[34] http://www.resxplorer.org
[35] https://www.youtube.com/watch?v=tZU97BQxE-0
[36] http://dblp.l3s.de/
[37] http://colinda.org

tion about scientific events and links to venue and proceedings, common Linked Data Knowledge Base DBPedia[38] and Open Linked Data repository with geographical information GeoNames[39]. The role of COLINDA is to act as a hub which connects all data sets in the knowledge base by pointing with links to other data sets. In this context it is used both for keyword matching together with other data sets and for enabling the algorithms in back-end to find better connections and paths between the terms visualized in the interface as well as for their expansion. Within ResXplorer interface a real-time keyword disambiguation guides researchers by expressing their needs. User selects the correct meaning from a typeahead drop down menu. Query expansion of terms happens in real-time. Figure 4 shows the typeahead expansion of "ResXplorer" in action. At the same time background modules also fetch neighbor links which match the selected suggestion. As result, selection of various resources is then presented to the researchers within radial interface. In case they have no idea which object or topic to investigate next, they get an overview of possible objects of interest (like points of interest on a street map e.g. figure 5). As shown in figure 5 features like color, shape and size of the items are used to enhance the guidance of the user during the search and exploration process [2]. Different shapes and colors represent different entities like conferences, persons, publications or proceedings. The explored items are marked black, and relations are marked red and clearly highlight the context and history of a search. Each presented resource is somehow related with some of other resources. This is expressed through lines and description of the relation which is a RDF property. The path distance in hops over links is expressed through the orbital layers.

As additional feature in ResXplorer is that researchers, when they sign in with their Twitter account, they can either use the mentions and hashtags automatically for search setup instead of typing keywords or to check visually the status of their network. This happens through visualization of recent collaboration and interactions based upon data from their Twitter accounts [3] (link to video on this procedure [40]). Figure 6 depicts the network of a researcher. The size of the scholar is in the middle between the minimum and maximum size of a node. As much as possible users are placed around the focused researcher. The more publications someone coauthored with the scholar, the bigger the node. Several visual aspects aid the user in focusing and exploring the current state of their network:

– *Spatial*: the number of *co-authorships* determines distance to center, a higher number results in a closer distance.
– *Size*: a higher frequency of being *mentioned* together on social media (i.e. Twitter) increases the size.
– *Color*: green, already in their Twitter network; red, not in their Twitter network.
– *Tooltip*: displays facts about the collaborations (e.g. co-authorships and mentions), i.e. the number of mentions for a specific conference (where conference identification is done over user hashtags which are automatically matched with COLINDA conference labels) and the the number of co-publications. The co-autorships are re-

[38] http://dbpedia.org
[39] http://geonames.org
[40] `http://youtu.be/QopnPvWIFzw`

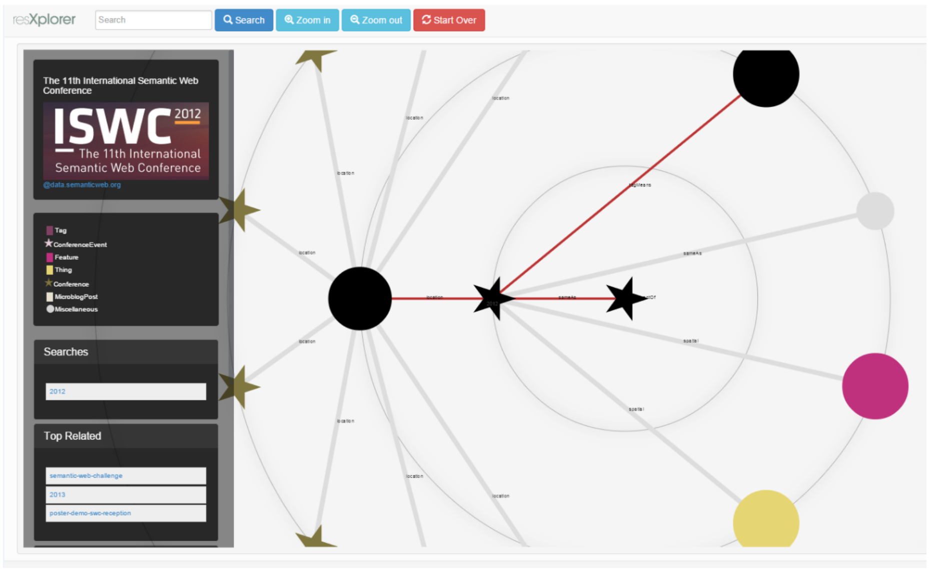


Fig. 5. ResXplorer - discovering scholar artifacts like conferences (represented as stars), miscellaneous related resources such as locations or microblog posts (represented as dots in different colors) etc. The distance to central node represents the intensity of relation.

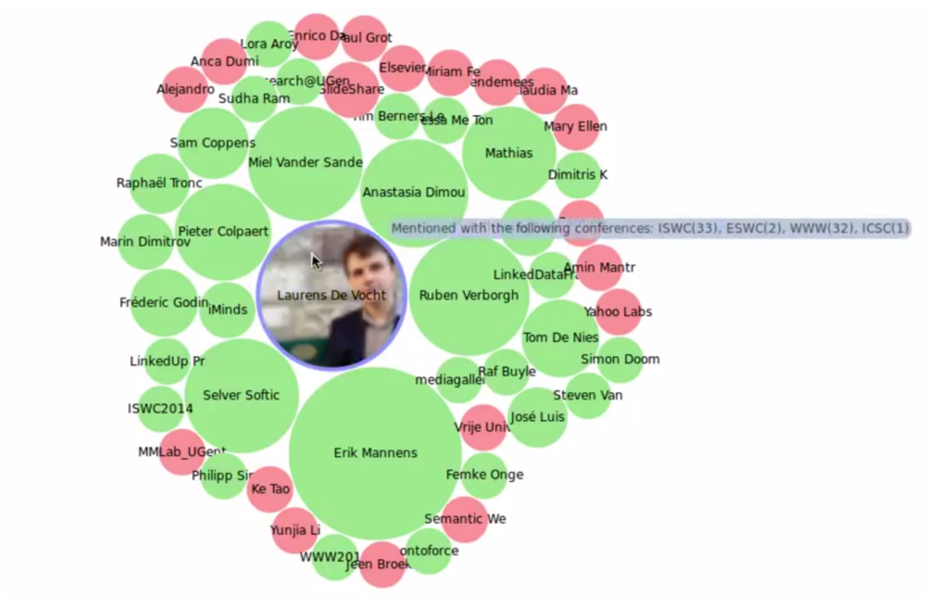


Fig. 6. The scholar is centered in the middle and the network is visualized in nodes around the central (blue) node.

solved by bibliographic records from DBLP which are matched pair-wise between the users.

Users who whether have no co-autorships or common mentions and conference hashtags with central user profile are not included in visualization.

4 Conclusion and Outlook

In this work we described how we extract, model and create scientific events as Linked Data from known conference portals. We showed also how those events can be enhanced with additional relevant information and applied as as mining source for generation and enhancement of *Researcher Affinity Browser* as well as main interlinking hub for discovery of research related artifacts for the *ResXplorer*. This potential has been also recognized by the LinkedUp Challenge at the ISWC 2014[41] and upcoming Semantic Publishing Challenge 2015[42] at ESWC 2015 where COLINDA is nominated as reference Linked Data set for scientific events. As one of the future efforts we also want to implement a DBPedia Lookup[43] and Spotlight[44] like service for detection and identification of scientific events with COLINDA. We also want to link the instances to WorldCat URIs of the published proceeding volumes and to he Crossref DOIs of the published conference articles to make it more useful for the library linked data community. Finally, to verify the quality of COLINDA we will run in the future an evaluation against Linked Data Integration Benchmark (LODIB)[45].

Acknowledgments.

The research activities that have been described in this paper were funded by Ghent University, the Social Learning Department at Graz University of Technology, iMinds (an independent research institute founded by the Flemish government to stimulate ICT innovation), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

References

1. Berners-Lee, T.: Linked data (2006), `http://www.w3.org/DesignIssues/LinkedData.html`
2. De Vocht, L., Mannens, E., Van de Walle, R., Softic, S., Ebner, M.: A search interface for researchers to explore affinities in a linked data knowledge base. In: Proceedings of the 12th International Semantic Web Conference Posters & Demonstrations Track. pp. 21–24. CEUR-WS (2013)

[41] http://data.linkededucation.org/linkedup/catalog/browse/
[42] https://github.com/ceurws/lod/wiki/SemPub2015
[43] http://lookup.dbpedia.org
[44] http://dbpedia-spotlight.github.io/demo/
[45] http://lodib.wbsg.de/

3. De Vocht, L., Softic, S., Dimou, A., Verborgh, R., Ebner, M., Mannens, E., Van de Walle, R.: Visualizing collaborations and online social interactions at scientific conferences for scholarly networking. In: Proceedings of the Workshop on Semantics, Analytics, Visualisation: Enhancing Scholarly Data; 24th International World Wide Web Conference (May 2015)
4. De Vocht, L., Softic, S., Ebner, M., Mühlburger, H.: Semantically driven social data aggregation interfaces for research 2.0. In: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies. pp. 43:1–43:9. i-KNOW '11, ACM, New York, NY, USA (2011), `http://doi.acm.org/10.1145/2024288.2024339`
5. Parra Chico, G., Duval, E.: Filling the gaps to know More! about a researcher. In: Proceedings of the 2nd International Workshop on Research 2.0. At the 5th European Conference on Technology Enhanced Learning: Sustaining TEL,. pp. 18–22. CEUR-WS (Sep 2010)
6. Reinhardt, W., Ebner, M., Beham, G., Costa, C.: How people are using Twitter during conferences. In: Hornung-Prähauser, V., Luckmann, M.(Hg.): 5th EduMedia conference, Salzburg. pp. 145–156 (2009)
7. Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The SWRC ontology - Semantic Web for Research Communities. Progress in Artificial Intelligence pp. 218–231 (2005), `http://dx.doi.org/10.1007/11595014\_22`
8. Ullmann, T.D., Wild, F., Scott, P., Duval, E., Vandeputte, B., Parra Chico, G.A., Reinhardt, W., Heinze, N., Kraker, P., Fessl, A., Lindstaedt, S., Nagel, T., Gillet, D.: Components of a research 2.0 infrastructure. In: Lecture Notes in Computer Science,. pp. 590–595. Springer (2010)

Toward Real Event Detection

Michael Färber* and Achim Rettinger

Karlsruhe Institute of Technology (KIT), Institute AIFB, Karlsruhe, Germany
{michael.faerber,rettinger}@kit.edu

Abstract. News agencies and other news providers or consumers are confronted with the task of extracting events from news articles. This is done i) either to monitor and, hence, to be informed about events of specific kinds over time and/or ii) to react to events immediately. In the past, several promising approaches to extracting events from text have been proposed. Besides purely statistically-based approaches there are methods to represent events in a semantically-structured form, such as graphs containing actions (predicates), participants (entities), etc. However, it turns out to be very difficult to automatically determine whether an event is real or not. In this paper, we give an overview of approaches which proposed solutions for this research problem. We show that there is no gold standard dataset where *real events* are annotated in text documents in a fine-grained, semantically-enriched way. We present a methodology of creating such a dataset with the help of crowdsourcing and present preliminary results.

Keywords: Event Detection, Information Extraction, Factuality.

1 Motivation

News agencies and other digital media publishers publish each day news articles in the magnitude of dozens of thousands. They also process the news for further business tasks such as trend prediction and market change detection. This is still mainly done manually today. Even if knowledge workers at news agencies have access to all this information, it is infeasible for them to read all the news and to determine, whether the articles contain information which is not only interesting for people in their domains, but which contain real events and, hence, have a significant, immediate impact on business such as financial operations (shares) and political happenings. Consider for example the first sentence of a news article:

"Apple may acquire Beats Electronics next week." (1)

* This work was carried out with the support of the German Federal Ministry of Education and Research (BMBF) within the Software Campus project *SUITE* (Grant 01IS12051).

Here, it remains unclear whether Apple is really going to acquire Beats (and does not cancel it in the last minute) or whether this is just a rumor. The sentence

"Apple confirmed that it acquired Beats Electronics on Wednesday." (2)

in contrary, reveals that the acquisition already happened (besides the confirmation which is an event per se). This demonstrates the differentiating characteristic between real events and events in general. As humans we can estimate that the first article is not a trigger for immediate shifts in the stock market (besides psychological effects), but maybe the second mentioned article. Machines, in contrast, have their difficulties in distinguishing real events from other events.

We envision building a decision support tool for agents like stockbrokers. The aim of the system is to inform the user quickly and automatically when some detected event has really happened and hence might influence the invested assets of the user. The user should also have the possibility to store purely real events in his database. For such purposes, an event extraction system would consist of two steps: i) It extracts events in a structured, semantically enriched representation and ii) determines based on linguistic cues whether the event is real or not.

Research on real event detection has been very limited so far. In this paper, we present an approach to define events and real events in a setting as described. Since no suitable gold standard for evaluating a real event detection system exists, we present our setting of creating one using crowdsourcing. Preliminary results regarding this gold standard are presented, as well as challenges which we came across.

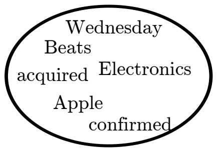
The remainder of this paper is organized as follows: First we present definitions of event detection in Section 2, before considering definitions of real event detection in Section 3. After discussing our setup of creating a gold standard for real event detection in Section 4, we conclude in Section 5.

2 General Event Definitions

2.1 Event Definitions in Use

We can distinguish between the following classes of event representation (see also Fig. 1 for examples):

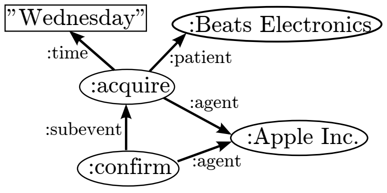
1. *Something happened*: In this event representation, events are only roughly covered. There are no types and deeper meanings gathered, only what topic the document/sentence is about. This topic is often characterized by the words occurring in the document (bag-of-words model) and/or by the set of recognized named entities.
2. *This happened*: For this representation, the event type of the event is detected. The event type can be quite generic such as *earthquake*. The number of events which can be detected is often very limited. Events may



(a) Event Representation Class 1

Event type	Acquisition
Participant	Apple
Participant	Beats Electronics

(b) Event Representation Class 2



(c) Event Representation Class 3

Fig. 1: Examples of event representations for the different event representation classes regarding the example sentence "Apple confirmed that it acquired Beats Electronics on Wednesday."

have attributes or slots which are pre-defined for the single event types. Instead of predefined entity types such as *earthquake* or *accident* sometimes only the entity types PER, LOC, ORG, and MISC are used.

3. *This happened to these objects in this way*: If we use this representation format, we have a deeper understanding in the actual event. Events of this class are quite specific and include not only specific actions, but also participants, and maybe time, place, and manner of the action. Often linguistic theories such as Semantic Role Labeling provide the basis for event representations of this class.

Related work using event definitions corresponding to the first event representation class do not define events at all [1,2,3,4,5]. This is due to the fact that here it must be only known *that* something happened (something that is, for instance, different to what has been seen so far), but not *what*. Events do not need to be represented on its own; instead, events are indirectly represented by the document in which they are expressed. Documents are compared against each other, either by using the bag-of-words model [2,3,4] or in addition by taking detected named entities (with the classical entity types PER, LOC, ORG, MISC) into account [1,5].

Approaches using the second event definition have in common that coarse-grained events such as accidents or earthquakes are represented. Each event has therefore an event type. Property-value-pairs can be assigned to the events, whereas the assignable properties are pre-defined for all event types. Often templates are used for storing the information about events [6].

In case of event representations of the third kind, structural representations of fine-grained events are extracted from text – here, typically from single sentences or clauses. Research based on this event class usually does not introduce a new definition of events, but instead either uses linguistic definitions of events where events consist of happenings with agents, locations, time, etc. [7,8,9] or abstracts from it to a certain, but limited extend [10]. Bejan [10] characterizes an event as a happening at a given location and in a specific time interval. Each event has semantic relations to agents, to a location, time, etc. as parts of the event. These are the semantic/thematic roles of an event in the linguistic understanding. Events can contain several sub-events. Events of an event scenario (as higher-order structure) are connected by event relations. An example is the *cause* relation where one event causes another event. Xie et al. [7] propose two approaches which are based on Semantic Frames – constructed by the tool *SEMAFOR*. Also, Wang et al. [8] use semantic parsing which is based on PropBank in order to represent events. Yeh et al. [9] regard events as similar to frames in FrameNet. Each event encodes knowledge about the participants, where (and when) the event occurred and the events which are caused by this event. A *buy* event, for instance, is about the object bought, the donor, and the recipient.

2.2 Event Definition

In this paper we focus on the detection and semantically-structured representation of real events of the third-mentioned event class, which is the most expensive one. More specifically, an event in our scenario is characterized by

– specific participants (agents or objects)
– situations (events or states) which are described within the event
– taking place at a specific place and/or time
– being not a state.

States are hereby defined as lasting for an indefinite period of time and which are not really observable. Given the example sentence 2 in Section 1 we can extract two events from it: i) The event that Apple confirmed something (which is an event itself) and ii) the event that Apple acquired Beats Electronics.

Fig. 1c shows how these events can be represented as a semantically-structured graph. Hereby, Event ii) can either be part of Event i) (as depicted in the figure) or be stored as a separate graph. Nodes in each event graph can be either predicate nodes (representing actions), entity nodes (representing participants), or literal nodes (representing the time, etc.). Predicate and entity nodes can be linked to entries in knowledge bases such as DBpedia (for entities) and WordNet (for predicates). This enables having unique identifiers for resources and to resolve ambiguities. The edges in these event graphs arise from the semantic roles assigned by a Semantic Role Labeling tool. In the depicted figure, the semantic roles are grounded as RDF predicates.

3 Real Event Detection

3.1 Definitions of Real Events

We define *real event detection* as the task of determining whether a given event expressed in text is real. Real events are events according to the definition in Section 2.2 and have already happened or are happening. Thus, the definition of events is extended by this aspect. We can split the task of real event detection therefore into two subtasks: 1. Determining if the situation described in the text is about an event according to our definition. 2. Determining if the event already happened or is currently happening.

Regarding the first subtask, we can refer to two areas of linguistic work: i) The distinction of events from states, and ii) the identification of factuality of events. In the following, we amplify these two areas with respect to our goal of real event detection. We hereby use the term *situation* as a generic concept which encompasses both event and state (cf. [11]).

Ad i) The classification of situations can be traced back to Aristotle who distinguished between verbs that have a defined end or result, and others that do not [12]. Vendler [13] distinguished situations into four *aspectual classes* (also called *aktionsarten*) and performed empirical experiments. The aspectual classes are based on the temporal structure of events. These classes are namely: *state*, *activity*, *accomplishment*, and *achievement*. A state is something in which an entity remains for a longer, often unspecified period of time (e.g., "Jack knows the answer"). The three other classes in the aspectual classification cover different types of events in the narrower sense. An event is characterized as something which happens or occurs in a definite time interval or at a specific point in time. It often comes along with predicates such as "write", "push", etc. An event usually causes some state change.

To determine which aspectual class a given situation belongs to, we can differ between telic, dynamic, and durative situations (see Table 1). Telic situations always have a culmination point beyond which the situation cannot continue. Dynamic situations consist of internal sub-events which change over time and are, hence, intrinsically heterogeneous. For instance, walking consists of several alternating subevents. Durative situations (e.g., eating) last for a specifiable period in time and are not punctual.

Table 1: Vendler's four-way distinction between verbs based on their aspectual features [13].

Class	Telic	Dynamic	Durative
state	-	-	✓
activity	-	✓	✓
accomplishment	✓	✓	✓
achievement	✓	✓	-

In our case we want to distinguish events from states. But how can we determine which aspectual class holds for a given situation? For Vendler [13] and others who worked on top of his theories it became apparent that it is not trivial to determine the class automatically. See [13,11,14,15,16] for more details on linguistic rules for that purpose.

Moens and Steedman [14] propose another classification of situations. Here, situations are also either states or events. Events are sub-classified by two dimensions: 1. Events are either atomic or durative events. 2. Entities of events are in a consequent state or not. We refer to [14] for more information.

Ad ii) Other researchers have focused on determining the factuality of events, i.e. to recognize whether events are presented in the sentences as corresponding to real situations in the world, as situations that have not happened, or as situations of uncertain status. The focus is, hence, the trustfulness of events in text. Factuality can be characterized by two dimensions: Polarity and epistemic modality. Polarity – more concrete: polarity on actuality and not subjective polarity – is a discrete category and can be either positive or negative. Epistemic modality, in contrast, expresses the speaker's degree of commitment to the truth of a proposition [17]. It ranges from uncertain (also called "possible") to absolutely certain (also called "necessary"). According to Horn [18], modality is a continuous category. Sauri [19] spans the factuality values space from positive, negative, to unknown for the polarity dimension, and certain, probable, possible, to unknown for the modality dimension. *Unknown* is true for cases of uncommitment. In this way, a tuple of polarity value and epistemic modality value states the factuality of the event.

How is factuality expressed in the text? This is done by lexical markers as well as syntactic markers. Lexical modal markers are modal auxiliaries (e.g., "could", "may", "must"), as well as clausal/sentential adverbial modifiers (e.g., "maybe", "likely", "possibly"). Examples of lexical polarity markers are adverbs (e.g., "not", "until"), quantifiers (e.g., "no", "none"), and pronouns (e.g., "nobody"). Syntactic constructs are necessary to consider since often one clause is embedded in another. Considerable are in this context especially relative clauses and that-clauses as in the example sentences.

What are the challenges to determine the factuality? Factuality markers interact with each other. The local modality and polarity operators (e.g., of the current clause) are therefore not enough. Instead, a global consideration is necessary. For instance, in case of that-clauses, the factuality of the inner event is dependent on the factuality of the outer event. Furthermore, what makes the factuality much more complex is the fact that the source of an event is often not only the author. These additional sources are introduced by means of predicates of reporting (such as "say" or "tell"), knowledge and opinion (such as "believe", "know"), psychological reaction (such as "regret"), etc. Sauri and Pustejovsky [19] calls these predicates due to their role *Source Introducing Predicates (SIPs)*. The difficulty is that the status of the other sources often differs from the author. The reader does not have direct access to the factual assessment of these other sources. In the sentence, "The Guardian wrote that the G-7 leaders pretended everything was OK in Russia's economy.", the reader cannot assess directly the "frame of mind" of The Guardian with respect to the factuality of the event of "pretended". However, the factuality assessment has to be relative to the relevant sources.

3.2 Requirements of a Gold Standard for Real Event Detection

According to our event definition in Section 2.2 and the additional aspect of factuality addressed in Section 3.1 we can list the following requirements a gold standard dataset for the evaluation of a real event detection system must fulfill:

1. Each mention of an action within an event (e.g., “wrote”) is annotated.
2. There is a distinction between events and states, so that all events in the strict sense are annotated.
3. There is no restriction to specific event types.
4. The factuality of the event is annotated (being positive or negative).
5. All participants and participating objects are annotated.
6. All participants and participating objects are linked to prevalent knowledge bases.
7. Subevents of events are annotated and linked.
8. Mentions of place and time of each event are annotated.

This gold standard is also suitable when it comes to extracting real events according to the Event Representation Classes 1 and 2 (see Section 2.1). In these cases, the information about the structural representation of events can be neglected. Additional filtering can achieve that only events of specific types such as accidents are detected.

3.3 Datasets for Real Event Detection

In the following, we review existing corpora where event factuality was annotated to some degree.

The *Multi-Perspective Question Answering (MPQA) corpus* [20] provides news articles annotated for opinions and other private states such as beliefs or thoughts. It was designed for subjectivity and sentiment research and does not provide any structured representation of (real) events. At most, it might be applicable as negative corpus in a scenario where situations written in text are approved to be not real events.

The *Penn Discourse TreeBank (PDTB)* [21] is a corpus where discourse connectives are annotated along with their arguments (e.g. $arg1 “– even though” $arg2). On top of the original annotation scheme, an extended annotation scheme was released for marking the attribution of abstract objects such as propositions, facts and eventualities associated with discourse relations and their arguments annotated in the PDTB. The events described in the arguments are, however, not transformed into a structured event representation.

TimeBank 1.2 [22] is a corpus which was annotated with TimeML [23]. TimeML is a language for representing temporal and event information. TimeBank is suitable for event factuality learning since it uses grammar markers as well as annotations of predicates. Events are classified into *occurrence*, *state*, *reporting*, *immediate-action*, *immediate-state*, *aspectual*, and *perception*. TimeBank does not contain a structured event representation where all participating objects are annotated. In addition, the event definition is somehow different to our proposed definition: A huge fraction (25,7%) of phrases annotated as events are not verbs, but nouns, adjectives, etc. Not all phrases that should be regarded as event predicates are annotated.

FactBank [19] is a corpus which was built on top of TimeBank and a subset of the documents in the AQUAINT TimeML Corpus (A-TimeML Corpus). It comes along with annotations of explicitly factual information about events. FactBank has the same obstacles as TimeBank.

ACE 2005 [24] from the *Automatic Content Extraction (ACE) technology evaluation* is a dataset dedicated to the detection of events in text. The task was limited to the detection of specific event types which are: *Life*, *Movement*, *Transaction*, *Business*, *Conflict*, *Contact*, *Personnel*, and *Justice*. Each type has one to 13 subtypes so that each event is assigned to one main event type and one subtype of it. The limitation to these event types is the main obstacle why ACE 2005 cannot be used in our setting directly. Four attributes are attached to each annotated event: *Modality*, *Polarity*, *Genericity*, and *Tense*. In accordance with the event type, specific slots (*argument roles* called here; such as entities, values, and times) can be assigned. ACE entities are categorized in specific classes (namely, Person, Organization, Location, Geo-political entity, Facility, Vehicle, and Weapon) and their subclasses, but are not linked to any knowledge base.

In summary, we can state that none of the mentioned corpora contains semantically-structured representations of events to the extent it is needed to evaluate a real event detection system where events are defined as in Section 2.2. Thus, in the following section we provide experiments on how to build a gold standard which fulfills all our requirements.

4 Experiments for Building a Gold Standard Dataset

Very first crowdsourcing experiments revealed that letting users annotate real events as described in Section 3.2 at once is too complex for any crowdsourcing job. Therefore, we arranged subtasks where the following questions are answered separately for each event:

1. Which are the actions/predicates inducing a real event?
2. Which are the participating objects?
3. What is the time and place?
4. Which sub-events are contained?

In the following we present our approach regarding the first subtask, namely identifying *real events* and naming the central predicates of them. We performed two crowdsourcing jobs which differ in their methodology.[1]

Run 1 The crowd was asked to read a given sentence, to look for real events (as defined above), and to enter the action verbs of these events as written in the sentence.

[1] The crowdsourcing job descriptions and evaluation data is available online at `http://www.aifb.kit.edu/web/Toward_Real_Event_Detection`

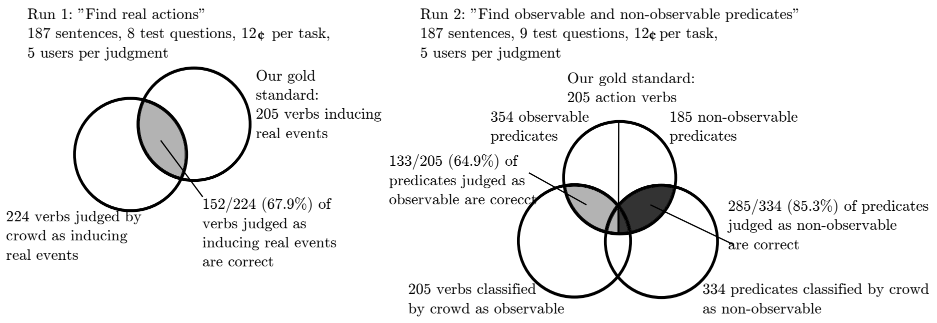

Run 1: "Find real actions"
187 sentences, 8 test questions, 12¢ per task,
5 users per judgment
Our gold standard: 205 verbs inducing real events
224 verbs judged by crowd as inducing real events
152/224 (67.9%) of verbs judged as inducing real events are correct
Run 2: "Find observable and non-observable predicates"
187 sentences, 9 test questions, 12¢ per task,
5 users per judgment
Our gold standard: 205 action verbs
354 observable predicates
185 non-observable predicates
133/205 (64.9%) of predicates judged as observable are corecct
285/334 (85.3%) of predicates judged as non-observable are correct
205 verbs classified by crowd as observable
334 predicates classified by crowd as non-observable

Fig. 2: Results of two crowdsourcing runs where the predicates of real events were annotated in English sentences. In both runs, the confidence value of the answers had to be above 0.5 in order to be considered.

Run 2 For this second run, the crowd was asked to read each given sentence, look for all verbs, and categorize them into either *observable* or *not-observable*.

Observable events/facts were defined as follows:[2] An observable fact can be an occurrence (e.g., "arrive", "destroy"), a reporting (e.g., "report"), or an immediate action (e.g., "approve"). Observable facts are characterized by the fact that they could be observed or confirmed by third persons directly (e.g., in case of "say") or indirectly (e.g., in case of "confirm"). Non-observable facts describe states which characterize persons or objects, but which are not observable by other persons than the persons involved. Such non-observable facts are states which last for an indefinite/unspecified period of time (e.g., "be happy"), immediate states (e.g., "believe", "worried"), aspects (e.g., "start", "continue"), or perceptions (e.g., "feel"). The categorization into observable vs. non-observable facts is here done independently of the fact whether the event has happened (or the state is) for sure or not. The categorization into the past/presence or future is performed in a separate crowdsourcing task.

As dataset we used all first sentences of news articles which were published on one day (2014/05/28) by the news agency Bloomberg and where the news articles contained some information about Apple Inc. In total we manually annotated 187 sentences to assess the performance of our crowdsourcing tasks. Crowd sourcing was performed on the platform Crowdflower.[3] In Run 1 (Run 2), users had to answer 8 (9) quiz test questions before entering the actual task. In both runs, users got 12 cent per task consisting of 4 questions each. For each question we gained results from 5 users and took the answers where there was an inter-rater agreement of at least 50%.

The results of our crowdsourcing annotation experiments are summarized in Fig. 2. It became apparent that completing the crowdsourcing tasks requires high cognitive efforts in comparison to other crowdsourcing tasks. A considerable amount of users did not pass the test questions at the beginning. Even if we

[2] The definition is based on the TimeBank annotation guidelines.
[3] `http://crowdflower.com`

admit only users who worked on our job in the past sufficiently well, creating a big annotated corpus is tricky. As Run 2 shows, already the distinction between observable events, i.e. events showing up in the real world, and not-observable events is hard to perform. Although we put much effort in refining the task descriptions the question arises whether a better approach to annotating the factuality of events is achievable.

5 Conclusions

If events are extracted from text in a fine-grained manner, huge amounts of events are gathered, but only a fraction of them represent real events and, hence, are worthwhile to process further on. In this paper, we gave an overview of existing linguistic work about the detection of *real events*. In order to evaluate a proposed system which extracts semantically-structured, real events from text, we defined requirements and proposed a methodology to create a gold standard dataset. Preliminary experiments with crowdsourcing showed that the annotation of text with factual information is non-trivial. Still, we believe that the creation of such a dataset is necessary for many event detection systems in the future.

References

1. Gabrilovich, E., Dumais, S., Horvitz, E.: Newsjunkie: providing personalized newsfeeds via analysis of information novelty. WWW '04, New York, NY, USA, ACM (2004) 482–490
2. Karkali, M., Rousseau, F., Ntoulas, A., Vazirgiannis, M.: Efficient Online Novelty Detection in News Streams. In Lin, X., et al., eds.: Web Information Systems Engineering – WISE 2013. Springer Berlin Heidelberg (2013) 57–71
3. Zhang, Y., Callan, J., Minka, T.: Novelty and Redundancy Detection in Adaptive Filtering. SIGIR '02, New York, NY, USA, ACM (2002) 81–88
4. Zhang, K., Zi, J., Wu, L.G.: New Event Detection Based on Indexing-tree and Named Entity. SIGIR '07, New York, NY, USA, ACM (2007) 215–222
5. Li, X., Croft, W.B.: Novelty Detection Based on Sentence Level Patterns. CIKM '05, New York, NY, USA, ACM (2005) 744–751
6. Kosmerlj, A., Belyaeva, J., Leban, G., Fortuna, B., Grobelnik, M.: Crowdsourcing Event Extraction. In: NewsKDD – Workshop on Data Science for News Publishing at KDD 2014. (2014)
7. Xie, B., Passonneau, R.J., Wu, L., Creamer, G.G.: Semantic Frames to Predict Stock Price Movement. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. (2013) 873–883
8. Wang, D., Li, T., Zhu, S., Ding, C.: Multi-document Summarization via Sentence-level Semantic Analysis and Symmetric Matrix Factorization. SIGIR '08, New York, NY, USA, ACM (2008) 307–314
9. Yeh, P.Z., Puri, C.A., Kass, A.: A Knowledge Based Approach for Capturing Rich Semantic Representations from Text for Intelligent Systems. Int. J. Adv. Intell. Paradigms **2**(1) (November 2010) 33–48
10. Bejan, C.A.: Learning event structures from text. PhD thesis, The University of Texas at Dallas (2009)

11. Bach, E.: The Algebra of Events. Linguistics and Philosophy (1986) 5–16
12. Dowty, D.R.: Word Meaning and Montague Grammar: the semantics of verbs and times in generative semantics and in Montague's PTQ. Reidel (1979)
13. Vendler, Z.: Linguistics in Philosophy. Cornell University Press (1967)
14. Moens, M., Steedman, M.: Temporal Ontology and Temporal Reference. Computational Linguistics **28**(3) (1988) 15–28
15. Pustejovsky, J.: The syntax of event structure. Cognition **41** (1991) 47–81
16. Dorr, B.J., Olsen, M.B.: Deriving Verbal and Compositonal Lexical Aspect for NLP Applications. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) (1997) 151–158
17. Palmer, F.: Mood an Modality. Cambridge University Press (1986)
18. Horn, L.: A Natural History of Negation. University of Chicago Press (1989)
19. Sauri, R., Pustejovsky, J.: From structure to interpretation: A double-layered annotation for event factuality. Proceedings of the Second Linguistic Annotation Workshop (2008)
20. Wiebe, J., Wilson, T., Cardie, C.: Annotating expressions of opinions and emotions in language. Language Resources and Evaluation **39**(2) (2005) 165–210
21. Miltsakaki, E., Prasad, R., Joshi, A., Webber, B.: The Penn Discourse Treebank. Proceedings of LREC 2004 (2004)
22. Pustejovsky, J., et al.: The TIMEBANK Corpus. Proceedings of Corpus Linguistics 2003 (2003) 647–656
23. Pustejovsky, J., Knippen, R., Littman, J., Saurí, R.: Temporal and event information in natural language text. Language Resources and Evaluation **39**(2) (2005) 123–164
24. Walker, C., Strassel, S., Medero, J., Maeda, K.: ACE 2005 Multilingual Training Corpus LDC2006T06 (2006)

Towards Open Domain Event Extraction from Twitter: REVEALing Entity Relations

G. Katsios[1], S. Vakulenko[2], A. Krithara[1], G. Paliouras[1]

[1] Institute of Informatics and Telecommunications, NCSR Demokritos, Greece
[2] MODUL University Vienna, Austria

Abstract. In the past years social media services received content contributions from millions of users, making them a fruitful source for data analysis. In this paper we present a novel approach for mining Twitter data in order to extract factual information concerning trending events. Our approach is based on relation extraction between named entities, such as people, organizations and locations. The experiments and the obtained results suggest that relation extraction can help in extracting events in social media, when combined with pre and post-processing steps.

Keywords: Event extraction, social media analysis, relation extraction, Twitter

1 Introduction

Social media attracts millions of users, and has evolved to become a source of various kinds of information. In Twitter for example, more than 255 million active users publish over 500 million 140-character "tweets" every day[3]. Evidently it has become an important communication medium. More and more people use social media to communicate their ideas and thoughts, as well as to spread important news. Given the enormous size of information exchange happening every day, it is a rather challenging task to process these data and filter out the important and relevant information.

Twitter data is part of the Big Data paradigm and is characterized by high Velocity, Veracity and Volume ("the 3 Vs") [12]. The topics on Twitter span across multiple domains from private issues to important public events in the society. Therefore, filtering out the important or relevant to the user information poses the first challenge for automated processing of tweets.

Twitter provides user-generated content in real time. The data is stored in a form of short text messages called tweets. Each tweet has a body that contains text of the message itself, but also a variety of metadata associated with it, e.g. date of creation, author, user mentions, location etc. However, what makes Twitter texts unique is its word count limitation which causes extensive usage

[3] `https://about.twitter.com/company`

of acronyms and other abbreviations. Moreover, users often use colloquial words and phrases in tweets, which require context for interpretation.

The goal of this research is to develop tools that extract and efficiently summarize trending events, the so-called “breaking news“, mined from social media, e.g. Twitter. This task is especially relevant for the professional journalists helping them to utilize social media as an information source helping to cope with the information overload.

This research was conducted in the context of two European 7th Framework projects, REVEAL and DecarboNet. The projects aim at developing new tools and approaches to automatically process digital media content, extracting important information and summarizing it.

This paper is reporting on the results of the initial round of experiments, where we combined the current state-of-the-art methods and tools available, and further evaluated them for the task of event extraction from social media. We also enhanced the pipeline with pre- and post-processing procedures in order to adopt it to the specific requirements stemming from the nature of social media data, e.g. spam detection, mention disambiguation and relation selection. These initial investigation and prototyping results aim to reveal the pitfalls and shortcomings of the current state-of-the-art approaches and suggest directions for the future work.

The definition of an event itself might appear rather blurry and controversial from the first sight. We adopt the wide definition of an 'event', which goes beyond scheduled events, like a music concert, conference or a football match. In general, we consider any action, which can be observed in the physical world, to constitute an event [21].

Events are often communicated through social media, e.g. *"Chelsea won today"*, *"We are going to a bar"*. Due to the abundance of such event reports on social media we define the notion of an 'important event', i.e. an event, information about which is of a potential value to a user of the system. For example, information about an international political summit involving famous politicians may be considered as important for the journalist, while the content of a lunch meal of an average twitter user is likely to be of no particular value.

In this work we focus on extracting the factual information about an event, e.g. its location, time and participants. It is important to separate the factual information from the content that expresses an opinion or an emotion related to the event, such as feelings and thoughts of an individual or a group. This can be a rather tricky task, because sentences that are lexically very similar can convey semantically opposite facts. For example: *"Chelsea won today"* versus *"I wish Chelsea won today"* versus *"I wish Chelsea wins today"*.

In order to extract event-related information from tweets we adopt and enhance existing state-of-the-art approaches to automated information extraction, taking into account the unique properties of social media data. We implement and apply the proposed approach to several datasets, evaluate and discuss the results, outlining further directions for the future work.

2 Related Work

Existing algorithms for news monitoring typically detect events by grouping together words with similar burst patterns (i.e. words or phrases showing burst in appearance count [24]). They rely on clustering or topic modeling techniques [3, 10, 13]. The draw-back of these approaches is that the resulting bag-of-words representation of the clusters/topics is often not descriptive enough.

More sophisticated and precise approach is information extraction on the level of events. Event extraction involves parsing of natural language text with the aim of extracting event-related information. The usual suspects for the event facets are the named entities that belong to actor/place/time classes in Simple Event Model (SEM) [21]. Therefore, many approaches to event extraction include entity recognition stage [5, 19]. In our work we also utilize the assumption that many events are centered around named entities as in [19]. Still open remains the question of how to connect the event-related entities, e.g. persons, locations, dates. Most of the approaches use NLP-methods involving a set of regular expressions to extract verbs that are assumed to constitute an event and feed it together with the related entities into the event model [1, 8, 18, 19, 22].

On the contrary, in our approach we utilize the state-of-the-art method for relation extraction [6], that has already been successfully applied to news articles. Relation extraction is the task of identifying relations that hold between entities in text data. Up to now relation extraction systems were only evaluated on news collections, but not on social media data. Therefore, the novelty of the proposed approach is testing the suitability of relation extraction methods for the task of event extraction on Twitter. We also make several modifications in order to adapt the relation extraction approach to the specific nature of social media data and further enhance it to extract event-related relations between the frequent named entities from tweets.

There have been a number of projects aiming at extracting events specifically from tweets [5, 20, 23]. Tweets are specific in nature and require special treatment, different from the news articles. Therefore, Twitter-oriented systems often include methods to detect spam, reduce noise and eliminate uninformative messages [5, 20].

Domain-specific event extraction, such as [5, 23], allow fine-tuned event detection, but require a set of keywords or event types to be manually predefined. In this work we focus on extracting trending events, i.e. events which are most popular among the users and are most frequently discussed. This approach also allows us to be domain-agnostic and catch previously unknown events.

In this respect, our approach is most similar to TwiCal [20]. However, instead of training classifier for event extraction on in-domain training data we utilize already trained extractor from ClausIE [6]. The goal of TwiCal is constructing a calendar of upcoming events. Therefore, it extracts only scheduled events accompanied with explicit date mention. We are primarily interested in information concerning recent or current events, where explicit date annotation is often omitted.

3 Our Approach

We adopt the state-of-the-art approach to relation extraction [6] and further enhance it for the task of event extraction from tweets. In our approach we consider any action, which can be observed in the physical world, to constitute an event [21]. We assume that events are indicated by nonstative (dynamic) verbs. Dynamic verbs describe an action, such as 'kick', 'meet', 'visit', as opposed to stative verbs, such as 'believe', 'like', 'consider', etc.

Relation extraction approach enables us to extract predicates from a sentence (corresponding to the verbs indicating events) together with their subjects and objects. For example, the sentence: *"The match starts on Sunday"* will result in the following relation: *The match (Subject) - starts (Predicate) - on Sunday (Object).*

Objects of the relations often contain event facets that uniquely characterize events in spatial, temporal and social dimensions (e.g. place, date, organizers, participants). Thus, this approach allows for more fine-grained event extraction as opposed to clustering or topic modeling-based approaches which operate with the bag-of-words model, which tend to blend together several lexically similar events.

We have extended the initial approach to relation extraction with a few pre-processing steps in order to clean the input data and annotate it with named entities. After the pre-processing we extract relations, link them to named entities and rank according to their frequencies. The resulting pipeline summarizing our approach is presented in Figure 1. In the rest of this section, the different modules of our approach are described in details.

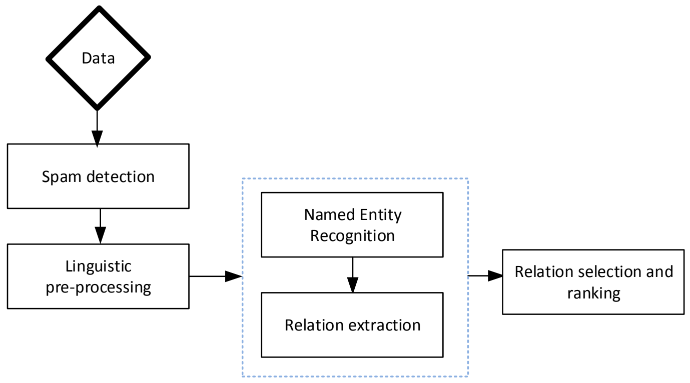


Fig. 1: System's pipeline

3.1 Spam detection

Here we define spam as useless uniformative or malformated messages, which are unlikely to provide us with any meaningful information. Our goal is to pre-process the raw data from Twitter and deliver to the end user only useful and relevant information. Therefore, we attempt to filter out meaningless and misleading messages already on the first stage of our pipeline.

In the first place, we use a freely distributed black-list of domain names [4] in order to exclude tweets containing links that point to the untrusted web sites. Next, we calculate a "spam score" for each of the remaining tweets and exclude the tweets that receive the score higher than the empirically learned threshold value. The **"spam score"** is calculated as the number of spam-associated tokens [4, 16] divided by the total number of tokens in the tweet:

$$spam\_score = \frac{|U| + |H| + |L| + |S| + |N|}{|T|} \tag{1}$$

where:

- $|U|$: number of user mentions (e.g. @themichaelowen);
- $|H|$: number of hashtags (e.g. #DavidGill);
- $|L|$: number of web links (e.g. `http://t.co/my55ZOoAko`);
- $|S|$: number of spam words (from the predefined list[5], e.g. dutyfree, poker, casino);
- $|N|$: number of non-word characters (e.g. %, !);
- $|T|$: total number of tokens in the tweet.

The bigger the value of the "spam score", the more likely that the tweet contains spam. We conducted an experiment spanning numerous trials to choose the optimal threshold value for the spam score and arrived at the value of 0.74. Further one, we identified 3% of the tweets in our datasets as spam and, therefore, excluded them from the next stages in our pipeline.

3.2 Linguistic Pre-processing

All the tweets that passed through the Spam Detection module, are further considered in the Linguistic Pre-processing module. The pre-processing steps include tokenization, user mentions resolution, further text cleaning and sentence splitting.

Tokenization is used to identify the tokens that will be replaced or removed from the text, such as URLs, user mentions, etc. First, we exploit tweet metadata to resolve user mentions to their canonical names. In particular, each tweet that contains user mentions, carries a list of the corresponding full user names from the Twitter database. Thus, we substitute the user mentions in the tweet text with the corresponding full names using the tweet metadata. For example, *@themichaelowen* is resolved to *Michael Owen*.

[4] `http://www.squidguard.org/blacklists.html`

[5] `http://notagrouch.com/wp-content/uploads/2009/12/wordpress-blacklist-words.txt`

3.3 Named Entity Recognition

In this module, we identify named entities mentioned in the text of the tweet, as well as their types. For example, the tweet containing the following snippet: "@DavidGill walks out of FIFA meeting in Sao Paulo", gets annotated with the named entities: *David Gill - Person*, *FIFA - Organization* and *Sao Paulo - Location.*

We used Stanford Named Entity Recognizer (Stanford NER) [15] for detecting named entities in tweets. According to the benchmark evaluation reported in [7], Stanford NER achieves highest average precision on all three datasets of tweets, when compared with other state-of-the-art Twitter-tailored algorithms.

Stanford NER detects the following types of named entities: Location, Person, Organization, Date, Money, etc.[6]. Due to our pre-processing procedure we also detect the entities "hidden" within the user mentions and hashtags (e.g. *@DavidGill*). This would not be feasible, when applying the Stanford NER on the original tweets.

3.4 Relation Extraction

The core of our approach is based on extracting relations from the pre-processed tweets. Relation is a triple that consist of subject, predicate and object. Subject and object are entities, predicate is the relation between these entities. For example, the sentence: *"The match starts on Sunday"* will result in the following relation: *The match (Subject) - starts (Predicate) - on Sunday (Object).*

We considered three state-of-the-art systems for the task of relation extraction: ReVerb [9], Ollie [14] and ClausIE [6]. ClausIE was reported to significantly outperform Ollie by the number of propositions extracted [6]. However, it has not been previously applied to social media data. Therefore, we ran our own experiments to compare the results returned by ReVerb and ClausIE. Subsequently, we chose ClausIE as the best-suited baseline system.

In ClausIE relation triples are extracted from clauses, parts of a sentence that express coherent pieces of information [6]. The clauses are identified based on the results from the dependency parser that helps to reveal the syntactic structure of an input sentence. In particular, ClausIE is using Stanford unlexicalized dependency parser [11].

Additionally, ClausIE has an option to return n-ary predicate by decomposing the object of the relation into several arguments. This option can be useful for extracting complex relations, that consist of several independent but overlapping parts, such as place and time relations. For example, the sentence: "The match starts on Sunday at Wembley" will result in the following relation: *The match (Subject) - starts (Predicate) - "on Sunday", "at Wembley" (Object).*

We made several modifications to the original implementation of ClausIE in order to adapt it to the task of extracting the relations describing events. Specifically, we enforce omitting the following types of clauses from the relation extraction process:

[6] `http://nlp.stanford.edu/software/CRF-NER.shtml`

– conditional clauses (If-clauses), e.g. “If @Chelsea wins I will celebrate till morning!!!!!!!!”
– clauses rooted in a stative verb, e.g. ”I believe @Chelsea is the actual winner!”

Conditional clauses are used to speculate about what might happen, what could have happened, and what we wish to happen. Stative verbs describe mental state of an agent, but do not signify any action. For example, the following verbs are stative: hate, love, believe, prefer, want, suppose, etc.

3.5 Relation Selection

We designed a post-processing step for selecting relations that will appear in the final results. For this we chose the Frequent Pattern Mining approach that helps us to reveal the recurrent information patterns following the assumption that input data from Twitter is often abundant and redundant. Additionally, we employ the following heuristic technique: for the relation to be selected it has to contain popular (frequently occuring) named entities. In this way we get rid of the trivial resuls, e.g. *”I - ate pizza - for breakfast”*, but retain the relations such as: *”President Obama - ate pizza - for breakfast”*, if they are reported by a considerable number of tweets.

Therefore, we combine the results from Relation Extraction (RE) and Named Entity Recognition (NER) modules produced on the previous stages. In particular, we select only those relations that contain named entities in subject and/or object of the relation. The intuition behind this approach enriching relations with NER annotations is that events in real-life are often associated with the corresponding named entities: dates, places and participants.

Hints about importance of the relations and named entities are given from their frequencies count. We assume that widely discussed news are more likely to be of importance and interest to the users of our system. Therefore, in order to link NER and RE results we identify frequent named entities and then select frequent relations, in which these entities occur. We use several approaches to select relations between the named entities described below.

Firstly, we detect the named entities that occur most frequently in the tweets (~ 10 entities for each of the datasets), e.g. Chelsea, Drogba, Ramires. We also identify the most frequently co-occurring pairs of named entities (~ 5 pairs per dataset), e.g. Chelsea and Liverpool, Putin and Ukraine. Then, we identify the following relations that hold between named entities:

1. Relations in which the most frequently occurring entities appear in subject or object of the relation;
2. Relations that hold between pairs of the most frequently co-occurring entities;
3. Relations for every combination of entity types pairs from the set: [Person, Organization, Location, Date], e.g. between Person and Organization, Person and Person, Location and Organization, Person and Date etc.

Finally, we calculate the support for each of the selected relations, i.e. number of tweets from which the same relation was extracted, and use it for ranking of the relations. The topmost relations are reported in the final results.

4 Experimental Evaluation

4.1 Datasets

We conducted experiments using three different Twitter datasets (see Table 1). All datasets are centered around one or several major events discussed on social media. We have deliberately selected the datasets containing event-related tweets for our evaluation with the goal to uncover the details surrounding these events using our approach.

The FACup dataset was created within the Social Sensor project[7] and covers the events during the last match of the Football Association Challenge Cup [2]. The SNOW dataset [17] is an attempt to capture the footprint in the social media regarding several important international events: uprising in Ukraine (#ukraine, #euromaidan), protests in Venezuela (#Venezuela), major Bitcoin exchange theft (#bitcoin), etc. The third dataset was collected in June 2014 and contains $\sim$ 270.000 tweets, that were extracted using the hashtag #WorldCup2014.

Dataset	**# Tweets**	**Hashtags**
FA Cup	$\sim$ 20.000	#FACupFinal
SNOW	$\sim$ 1.000.000	#ukraine, #euromaidan, #Venezuela, #bitcoin
World Cup	$\sim$ 270.000	#WorldCup2014

Table 1: Datasets

4.2 Evaluation Method

We manually evaluated the results by annotating the relations returned on the last stage of our pipeline (section 3.5). Each of the annotators (3 in total) independently considered perceived correctness and usefulness (importance) of the relations by looking up the original text of a sample tweet, from which the relation was extracted by the system.

The relation was marked as *Correct*, if the information it provides naturally follows from the original text of the tweet and does not contradict the message conveyed in it. Negation handling is a good example for potential errors in the results returned by the system. If the original tweet reports, that Chelsea did not play better than Liverpool, the relation has to communicate the same fact and not the opposite. For example, *Chelsea - play better - than Liverpool* relation should be marked as *Incorrect* in this case.

Furthermore, all correct relations were further evaluated with respect to perceived importance for the end user of the system. The importance of a relation is harder to evaluate than its correctness, because of the complexity and subjectivity in the notion of importance with respect to an information piece. In general, a relation is considered *Important*, when it is perceived as being descriptive and

[7] `http://www.socialsensor.eu/`

potentially useful. Meaningless and uninformative relations are marked as *Not important*, respectively.

Collective discussion of the individual annotations resulted in a consensus and a single final evaluation table was constructed. Afterwards, we summarized our evaluation results by counting the number of relations for each of the classes: *Correct & Important*, *Correct & Not important* and *Incorrect* relations (see Table 2). We calculated the ratios and the total number of evaluated relations separately for each of the datasets. The last row of the evaluation table highlights the average precision values across the three datasets.

Dataset	Incorrect	Correct	
		Not important	Important
FA Cup	0.17 (8)	0.17 (8)	0.66 (32)
SNOW	0.1 (21)	0.14 (32)	0.76 (168)
World Cup	0.1 (18)	0.19 (35)	0.71 (134)
Average	**0.12 (47)**	**0.17 (75)**	**0.71 (334)**

Table 2: Precision of the evaluation results: fraction (total) of relations

4.3 Discussion and Future Directions

The average precision of our approach was estimated at 88% taking into account all correctly extracted relations. However, less that 3/4 of the relations returned by the system were considered as potentially valuable for the end users of the system (see *Correct & Important* in Table 2).

The most frequent relations that were selected using our approach from FA Cup dataset are listed in Table 3. These 5 relations provide a short summary of the event by revealing the names of the teams, the place where the game took place, the winner and the final score, as well as the player, who scored. The timestamps of the tweets can disambiguate the mentions "now", reveal date of the event and indicate the "hot spots" on the game timeline, such as the last relation in Table 3.

Relations extracted from the SNOW dataset are less homogeneous containing various political statements, business and sport announcements, as well as snapshots of historical events. Sample relations (with their support): *Ukraine's leaders - warn - "of Crimea separatism threat"* (106); *Chelsea fans - attending - "the Galatasaray match", "on 26 Feb"* (84).

World Cup dataset is another noisy collection containing many tweets not related to the football championship. Nevertheless, the three top-most relations reveal the major conflict in the football association: *director David Gill - walks out - "of FIFA meeting in Sao Paulo"* (902); *director David Gill - says - "Sepp Blatter should stand down"* (901); *FA Vice-Chairman David Gill - calls on - "Sepp Blatter not to stand for re-election as FIFA President"* (481).

In general, due to our broad definition of 'event' (as any kind of action reflected in a physical world) relations can be extracted from virtually any collection of tweets. However, in order to achieve comprehensive results the tweets need to be previously clustered according to the common topic, e.g. using a set of hashtags.

Subject	Predicate	Object	Count	Sample tweet
The Chelsea players	are throwing	"Robbie Di Matteo high in the air"	129	RT @chelseafc: What celebrations! The Chelsea players are throwing Robbie Di Matteo high in the air. And catching ...
Chelsea	have won	"17 major trophies", "now"	58	RT @chelseafc: Chelsea have now won 17 major trophies. We've caught Tottenham who are on the same total.
Liverpool	are out	"for the second half"	27	RT @chelseafc: Liverpool are out for the second half, and Chelsea are on the way. #CFCWembley #FACupFinal (SL)
Chelsea	beat	"Liverpool 2-1 to win the FA Cup at Wembley"	24	RT @premierleague: Chelsea beat Liverpool 2-1 to win the FA Cup at Wembley, their fourth win in six years in the competition. #cfc #lfc ...
Liverpool	is	"much", "pretty", "giving every Chelsea fan a heart attack right now"	21	RT @espn: Liverpool is pretty much giving every Chelsea fan a heart attack right now: http://t.co/MGxAkv94

Table 3: Results from FA Cup dataset

We performed only limited experimental evaluation for the proof-of-concept of our approach and can not quantatively compare our results with other approaches to event extraction. Moreover, the relation extraction algorithm is currently computationally rather expensive, which might prevent us from running the system on Twitter stream data in real time.

Nevertheless, our initial results provide further motivation and help to outline directions for the future work:

1. Linking relations that convey the same information. Disambiguating and clustering these relations will help to improve quality of the results by increasing support of the frequent relations and removing semantic duplicates. This can be achieved by:
   – grouping the predicates into semantic groups using existing lexical resources, such as FrameNet (e.g. verbs related to communication, cognition, perception: *say = tell = report*, *believe = think = consider*);
   – disambiguating and linking named entities contained in subjects and objects of the relations (e.g. *President Obama = Barack Obama*, *next month = June 2015*)
2. Linking relations that describe the same event. This can be achieved by building an event knowlege model, e.g. an event ontology, that will incorporate and meaningfully combine event facets extracted from different sources.

3. Linking events between each other. This task will help to reveal patterns within spatial/temporal/social dimensions by projecting the events on a timeline or a geographic map. This approach may help to learn the common-sense rules useful for reasoning and inference over the event data, such as the 'finish' event follows the 'start' event, but also reveal non-trivial patterns and the outliers.

5 Conclusion

We presented a novel approach to event extraction from Twitter, which builds upon current state-of-the-art relation extraction techniques. We manually evaluated the quality of extracted relations in terms of precision on three real-world datasets. Most of the results returned by the system are correct (88%) and contain descriptive and potentially useful event-related information (71%). However, recall and computational performance of the system was out of scope of this intial evaluation run.

Acknowledgments

This work was supported by REVEAL (http://revealproject.eu/) and DecarboNet (www.decarbonet.eu) projects, which have received funding by the European Unions 7th Framework Program for research, technology development and demonstration under the Grant Agreements No. FP7-610928 and 610829, respectively.

References

1. Puneet Agarwal, Rajgopal Vaithiyanathan, Saurabh Sharma, and Gautam Shroff. Catching the Long-Tail: Extracting Local News Events from Twitter. In *ICWSM*, 2012.
2. L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. In *Multimedia*, volume 15, 2013.
3. Hila Becker, Mor Naaman, and Luis Gravano. Beyond Trending Topics: Real-World Event Identification on Twitter. *ICWSM*, 2011.
4. F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, 2010.
5. Smitashree Choudhury and John G. Breslin. Extracting semantic entities and events from sports tweets. In *'Making Sense of Microposts': Big Things Come in Small Packages*, 2011.
6. L. Del Corro and R. Gemulla. Clausie: clause-based open information extraction. In *WWW*, 2013.
7. Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2), 2015.

8. Peter Exner and Pierre Nugues. Using semantic role labeling to extract events from Wikipedia. In *Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE)*, 2011.
9. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.
10. Yuheng Hu, Ajita John, Dore Duncan Seligmann, and Fei Wang. What Were the Tweets About? Topical Associations between Public Events and Twitter Feeds. In *ICWSM*, 2012.
11. D. Klein and C. D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on ACL*, 2003.
12. D. Laney. 3D data management: Controlling data volume, velocity, and variety. Technical report, February 2001.
13. Jimmy Lin, Rion Snow, and William Morgan. Smoothing techniques for adaptive online language models: Topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.
14. M., M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.
15. C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the ACL*, 2014.
16. M McCord and M Chuah. Spam detection on twitter using traditional classifiers. In *Autonomic and Trusted Computing*. Springer, 2011.
17. S. Papadopoulos, D. Corney, and L. M. Aiello. Snow 2014 data challenge: Assessing the performance of news topic detection methods in social media. In *SNOW-DC@WWW*, 2014.
18. Thomas Ploeger, Maxine Kruijt, Lora Aroyo, Frank De Bakker, Iina Hellsten, and Antske Fokkens. Extractivism: Extracting activist events from news articles using existing NLP tools and services. In *The 12th International Semantic Web Conference (ISWC)*, 2013.
19. Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, 2011.
20. Alan Ritter, Oren Etzioni, Sam Clark, and others. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.
21. Willem Robert Van Hage, Vronique Malais, Roxane Segers, Laura Hollink, and Guus Schreiber. Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2), 2011.
22. Willem Robert van Hage, Vronique Malais, Marieke Van Erp, and Guus Schreiber. Linked open piracy. In *Proceedings of the sixth international conference on Knowledge capture*, 2011.
23. Guido Van Oorschot, Marieke Van Erp, and Chris Dijkshoorn. Automatic extraction of soccer game events from twitter. In *Proc. of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web*, 2012.
24. Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR*, 1998.

A Contextual Framework for Reasoning on Events

Loris Bozzato[1], Stefano Borgo[2], Alessio Palmero Aprosio[1], Marco Rospocher[1], and Luciano Serafini[1]

[1] Fondazione Bruno Kessler,
Via Sommarive 18, 38123 Trento, Italy
[2] Laboratory for Applied Ontology, ISTC CNR,
Via alla Cascata 56 C, 38123 Trento, Italy

{bozzato,aprosio,rospocher,serafini}@fbk.eu, stefano.borgo@cnr.it

Abstract. In this paper we investigate the contextualized representation of events. In particular, we re-interpret the formalization of events and situations adopted in the NewsReader Event and Situation Ontology (ESO) according to the notions of context and module proposed in the Contextualized Knowledge Repositories (CKR) framework. This contextualized formalization sets the basis for exploiting logical reasoning on events and situations, enabling to automatize tasks such as: recognizing incompatible event descriptions or inconsistent situations, inferring missing or implicit events.

1 Introduction

With the growth of technologies managing the extraction of events and their participants from texts (e.g. [1,7]), interest has spread in using these low level tools as the base for event processing and reasoning in a higher level abstraction for events. The idea is to be able to discover, starting from low level descriptions of events extracted from text, complex events and their succession (viz. *stories*) with respect to their participant entities.

This is one of the goals of the *NewsReader project*[3]. In this project, Natural Language Processing (NLP) technologies, such as Named Entity Recognition (NER) and Semantic Role Labelling (SRL), are exploited to process large streams of news in multiple languages, to extract events and event participants, locations, dates. At the end of the extraction and processing, events and related information are represented using RDF. In order to define and reason over the features and effects of events, an OWL ontology has been defined for such data, called the *Event and Situation Ontology (ESO)* [11].

In this paper we propose to reinterpret and formalize the event model defined by the ESO ontology using a context-based framework for representation of Semantic Web data, the *Contextualized Knowledge Repositories (CKR)* framework [12,2,4,5]: this enables to exploit the structure and reasoning possibilities offered by the contextual framework in order to perform complex inferences about and inside knowledge associated with events.

Intuitively, CKR is a description logics (DL) based framework defined as a two-layered structure: a lower layer contains a set of knowledge bases representing each context, while the upper layer contains context independent knowledge and meta-data

[3] www.newsreader-project.eu

defining the structure of contexts. The CKR framework has not only been presented as a theoretical framework, but also actual implementations based on its definitions [5,3] have been proposed. In particular, in [5] we presented an implementation for the CKR framework over state-of-the-art tools for storage and inference over RDF graphs: intuitively, the CKR architecture can be implemented by representing the global context and the local object contexts as distinct RDF named graphs, while inference inside (and across) named graphs is implemented as SPARQL-based forward rules.

From a formal ontology point of view, in the approach we propose in this paper we clearly distinguish *ontology* from *knowledge*. The upper layer of our system contains the underlying ontology, that is, the description of the organization of the world. This part lists types of entities, relations and constraints that are assumed to exist or to simply be possible. More generally, the upper layer should be thought as including two elements: a general (foundational or domain) ontology describing what can exist, and the organization of a set of knowledge modules characterizing roles and relationships in (typically social) standardized scenarios like economic transactions or soccer games. Regarding the first, we do not make a specific commitment towards one ontology or another although we explicitly commit to the existence of physical and social objects, like people and organizations, and temporal happenings like weddings and thunderstorms. These entities come with their usual physical and temporal properties: weight, shape, duration and so on. Regarding the modules, we do explore their role in the system to infer new facts from given knowledge and thus we will present their general setting and how they are used. The lower layer of the system, on the other hand, collects claims about what happens in the world, that is, claims about how things are or change at some time. Throughout the paper these are what we call *events*, but note that they are not events in the ontological sense, rather they are descriptions of happenings.

Each event is associated with three state-like entities, namely happenings characterized by some continuously holding property or relation like "begin married", "being running" or "being employed by" (e.g. see the notion of *stative perdurant* in DOLCE [9]). These state-like entities are called *situations* in our system; more specifically the situation holding before the event is called *pre-situation*, the one holding after *post-situation* and the one holding during the event itself is the *during-situation*. Informally, these situations are used to make explicit relevant properties that (*a*) persist during the whole event, or (*b*) hold (or don't hold) before/after the event and their truth values change "because" of the event. We will use situations to formalize (and reason on) the preconditions for an event of a certain type to happen, the postconditions (or consequences) of its happening, and what can be taken as stable during the whole event of that type.

Although we take the received events at face value, it is possible that the news are imprecise or incomplete, that the text processing used to acquire the news is faulty and misinterprets them, that statements extracted from different sources about the same happenings contradict each other. For this reason, we take each piece of information extracted from an outside source as a contextual perspective on the world. This means that the content of a news is not equated to absolute knowledge (it is not indisputable). Indeed, the extracted information constitutes a context annotated with its original source (possibly with different degrees of reliability which can change over time). This allows to integrate pieces of information coming from different sources into large and articulated event descriptions by integrating different contexts (in turn, news statements). These extrapolated events reconstruct how an entity like a person or an organization, changes over time by changing its status (size, capacity, death) or its relationships with other entities (property acquisition, employment, marriage). Furthermore, missing information on these complex events can be detected via logical reasoning based on the ontology at the upper layer, leading to infer changes that have not been reported (or detected) in the news. Finally, note that, when a contradiction arises, the system can isolate the conflicting pieces of information and establish which contexts are to be kept apart and need to be verified.

In this paper we propose the first sketch and example of use for the contextualized ESO model: in the next section we briefly present the ESO model and provide an example of event modelling; in Section 3, we summarize the base definitions for the CKR framework; in the following section we describe the *CKR-ESO model*, that is our contextual based realization of ESO, we show how it models the previously presented event example and provide some insights in the advantages of such representation; we conclude by outlining some of the current and future directions for our work.

2 Events and Situations

One of the objectives in the NewsReader project is the representation of events and of their effects on the entities participating in them. For example, in a "giving" event, we have at least two actors (people, organizations) and an object, e.g., a person A giving something to another person B at some time T. The event also describes a change in these entities: at the time T, person A owns the object, while person B does not (aka, *pre-situation*); after T, the opposite is true (aka, *post-situation*).

To achieve this objective, the Event and Situation Ontology (ESO) [11] has been developed. It defines two main classes of entities: *events* and *situations*. An *event* describes an happening, typically a change, in the world (for instance, person A giving an object to person B at time T), has a certain number of participants (here, the two people and the object) and an associated period of time (here, T). A *situation* describes a state, i.e., takes a set of statements as describing (part of) the world at some point or interval of time. In the example "person A gives an object to person B at time T", we can identify a *pre-situation* (the state of the world at the initial time point of T):

– person A owns the object,
– person B does not own the object

and a *post-situation* (the state of the world at the ending time point of T):

– person A does not own the object
– person B owns the object.

For instance, in the sentence: "*The chairman of India's Tata Group has confirmed that his company is acquiring the Jaguar and Land Rover businesses from Ford.*", an event is described: the acquisition by Tata Group of Jaguar and Land Rover from Ford. The event is represented in ESO terminology as follows:

⟨*:evID, rdf:type, sem:Event*⟩
⟨*:evID, rdf:type, eso:Getting*⟩
⟨*:evID, rdfs:label, acquire*⟩
⟨*:evID, eso:possession-owner_1, dbp:Ford*⟩
⟨*:evID, eso:possession-owner_2, dbp:Tata_Group*⟩
⟨*:evID, eso:possession-theme, :Jag_and_L._Rover*⟩
⟨*:evID, sem:hasTime, #tmxID*⟩

where *#tmxID* is the RDF representation of “August 26th, 2007”.

According to ESO, the event *eso:Getting* has both *pre-* and *post-situation*, containing *eso:hasInPossession* and *eso:notHasInPossession* assertions, respectively. In the *pre-situation* of our example, Ford has in possession Jaguar and Land Rover, and Tata Group does not have in possession them; in the *post-situation*, the contrary holds. In the ESO model, such facts specific to a situation are stored in a RDF named graph identified by the URI of the situation. Therefore, the following set of *n*-tuples is generated:[4]

⟨*:evID, eso:hasPreSituation, :evID_pre*⟩
⟨*:evID_pre, rdf:type, eso:Situation*⟩
⟨*:evID_pre, sem:hasTime, #tmxID*⟩
⟨*dbp:Ford, eso:hasInPossession, :Jag_and_L._Rover, :evID_pre*⟩
⟨*dbp:Tata_Group, eso:notHasInPossession, :Jag_and_L._Rover, :evID_pre*⟩

⟨*:evID, eso:hasPostSituation, :evID_post*⟩
⟨*:evID_post, rdf:type, eso:Situation*⟩
⟨*:evID_post, sem:hasTime, #tmxID*⟩
⟨*dbp:Ford, eso:notHasInPossession, :Jag_and_L._Rover, :evID_post*⟩
⟨*dbp:Tata_Group, eso:hasInPossession, :Jag_and_L._Rover, :evID_post*⟩

In NewsReader, all the events and related information, instantiated according to the ESO metamodel, are stored, together with the original news article from where they were extracted in the KnowledgeStore [6], a scalable, fault-tolerant, and Semantic Web grounded storage system to jointly store, manage, retrieve, and query both structured and unstructured data.

3 Contextualized Knowledge Repositories

In the following we provide an informal summary of the definitions for the CKR framework: for a formal and detailed description and for complete examples, we refer to [5].

A CKR is a two layered structure: (1) the upper layer consists of a knowledge base $\mathfrak{G}$, called *global context*, containing (a) *meta-knowledge*, i.e. the structure and properties of contexts, and (b) *global* (context-independent) *object knowledge*, i.e., knowledge that applies to every context; (2) the lower layer consists of a set of *(local) contexts* that contain locally valid facts and can refer to what holds in other contexts. The intuitive structure of a CKR knowledge base is depicted in Figure 1: in the following we detail its formal components and interpretation.

Syntax. The meta-knowledge of a CKR is expressed in a DL language containing the elements that define the contextual structure: the *meta-vocabulary* Γ is a DL signature containing, in particular, the sets of symbols for *context names* $\mathbf{N}$, *module names* $\mathbf{M}$ and *context classes* $\mathbf{C}$, including the class of all contexts Ctx. Intuitively, modules represent pieces of knowledge specific to a context or a context class. The role mod defined on $\mathbf{N} \times \mathbf{M}$ expresses associations between contexts and their modules. The *meta-language* $\mathcal{L}_\Gamma$ of a CKR is a DL language over Γ.

The knowledge in contexts of a CKR is expressed via a DL language $\mathcal{L}_\Sigma$, called *object-language*, based on an object-vocabulary Σ. The expressions of the object language are evaluated locally to each context, i.e., contexts can interpret each symbol

[4] Whenever applicable, default named graph is omitted.

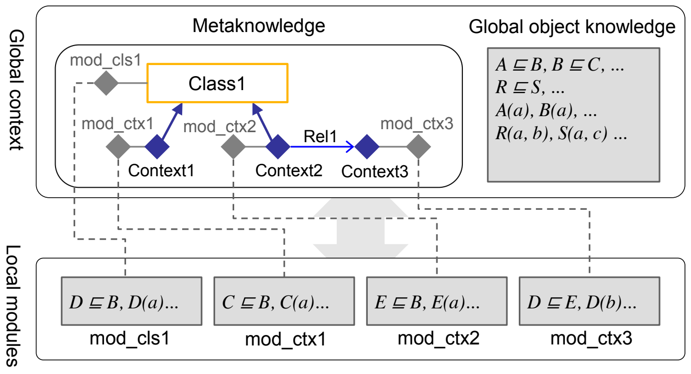


Fig. 1. CKR structure

independently. To access the interpretation of expressions inside a specific context or context class, we extend $\mathcal{L}_\Sigma$ to $\mathcal{L}^e_\Sigma$ with *eval expressions* of the form $eval(X, \mathsf{C})$, where X is a concept or role expression of $\mathcal{L}_\Sigma$ and C is a concept expression of $\mathcal{L}_\Gamma$ (with $\mathsf{C} \sqsubseteq \mathsf{Ctx}$). Intuitively, $eval(X, \mathsf{C})$ can be read as "the interpretation of X in all the contexts of type C".

We define a *Contextualized Knowledge Repository (CKR)* as a structure $\mathfrak{K} = \langle \mathfrak{G}, \{\mathrm{K}_\mathsf{m}\}_{\mathsf{m} \in \mathbf{M}} \rangle$ where: (i) $\mathfrak{G}$ is a DL knowledge base over $\mathcal{L}_\Gamma \cup \mathcal{L}_\Sigma$; (ii) every K_m is a DL knowledge base over $\mathcal{L}^e_\Sigma$, for each module name $\mathsf{m} \in \mathbf{M}$. We note that the knowledge in a CKR can be expressed by means of any DL language: in our work, we consider $\mathcal{SROIQ}$-RL [5] as language of reference. $\mathcal{SROIQ}$-RL is a restriction of $\mathcal{SROIQ}$ syntax corresponding to OWL RL [10].

Semantics. The semantics of CKR basically extends the usual model-based semantics of DL knowledge bases to the two layered structure of the framework. A *CKR interpretation* is a structure $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ s.t.: (i) $\mathcal{M}$ is a DL interpretation of $\Gamma \cup \Sigma$ (respecting the intuitive interpretation of Ctx as the class of all contexts); (ii) for every context $x \in \mathsf{Ctx}^\mathcal{M}$, $\mathcal{I}(x)$ is a DL interpretation over Σ (with same domain and interpretation of individual names of $\mathcal{M}$). The interpretation of ordinary DL expressions on $\mathcal{M}$ and $\mathcal{I}(x)$ is as usual while *eval* expressions are interpreted as follows: for every $x \in \mathsf{Ctx}^\mathcal{M}$, $eval(X, \mathsf{C})^{\mathcal{I}(x)}$ represents the union of all elements in $X^{\mathcal{I}(\mathsf{e})}$ for all contexts e in $\mathsf{C}^\mathcal{M}$.

A CKR interpretation $\mathfrak{I}$ is a *CKR model* of $\mathfrak{K}$ iff: (i) for $\alpha \in \mathcal{L}_\Sigma \cup \mathcal{L}_\Gamma$ in $\mathfrak{G}$, $\mathcal{M} \models \alpha$; (ii) for $\langle x, y \rangle \in \mathsf{mod}^\mathcal{M}$ with $y = \mathsf{m}^\mathcal{M}$, $\mathcal{I}(x) \models \mathrm{K}_\mathsf{m}$; (iii) for $\alpha \in \mathfrak{G} \cap \mathcal{L}_\Sigma$ and $x \in \mathsf{Ctx}^\mathcal{M}$, $\mathcal{I}(x) \models \alpha$. Intuitively, this means that $\mathfrak{I}$ verifies the contents of global and local modules associated with contexts and global object knowledge has to be propagated to local contexts.

Materialization calculus. Reasoning in CKR has been formalized as a materialization calculus [8], a datalog-based calculus for instance checking in $\mathcal{SROIQ}$-RL CKRs.

Intuitively, the calculus is based on a translation to datalog of the input CKR. It has three components: (i) the *input translations* I_{glob}, I_{loc}, I_{rl}, where given an axiom α and $\mathsf{c} \in \mathbf{N}$, each $I(\alpha, \mathsf{c})$ is a set of datalog facts or rules encoding the contents of input global and local DL knowledge bases; (ii) the *deduction rules* P_{loc}, P_{rl}, which are sets of datalog rules representing the inference rules for the instance-level reasoning over the translated axioms; and (iii) the *output translation* O, where given an axiom α and

$\mathsf{c} \in \mathbf{N}$, $O(\alpha, \mathsf{c})$ is a single datalog fact encoding the ABox assertion α that we want to prove to be entailed by the input CKR (in the context c).

Intuitively, $\mathcal{SROIQ}$-RL input I_{rl} and deduction P_{rl} rules provide the translation and interpretation of $\mathcal{SROIQ}$-RL axioms from the input CKR. Global input rules in I_{glob} encode the interpretation of Ctx in the global context. Similarly, local input rules I_{loc} and deduction rules P_{loc} provide the translation and rules for the local *eval* expressions. The rules in O provide the translation of ABox assertions that can be verified to hold in a context c by applying the rules of the final program.

The translation of a CKR $\mathfrak{K}$ to its datalog program $PK(\mathfrak{K})$ proceeds in four steps: we first translate $\mathfrak{G}$ in the *global program* $PG(\mathfrak{G})$ by applying input rules I_{glob} and I_{rl} to $\mathfrak{G}$ and adding deduction rules P_{rl}; then, for every context name $\mathsf{c} \in \mathbf{N}$ appearing in $PG(\mathfrak{G})$, we compute its knowledge base K_{c} as the set of modules $\mathrm{K}_{\mathsf{m}} \in \mathfrak{K}$ s.t. $\mathsf{mod}(\mathsf{c}, \mathsf{m})$ is proved by $PG(\mathfrak{G})$; we translate each *local program* $PC(\mathsf{c})$ by applying input rules I_{loc} and I_{rl} to K_{c} and adding deduction rules P_{loc} and P_{rl}; the final *CKR program* $PK(\mathfrak{K})$ is then obtained as the union of $PG(\mathfrak{G})$ with all local programs $PC(\mathsf{c})$. We say that $\mathfrak{K}$ *entails* an axiom α in a context $\mathsf{c} \in \mathbf{N}$ if $PK(\mathfrak{K}) \models O(\alpha, \mathsf{c})$. We can show (see [5]) that the presented rules and translation process provide a sound and complete calculus for instance checking in $\mathcal{SROIQ}$-RL CKR.

CKR implementation on RDF. We recently presented a prototype [5] that implements the forward reasoning procedure over CKR defined by the materialization calculus. The prototype accepts RDF input data expressing OWL-RL axioms and assertions for global and local knowledge modules: these different pieces of knowledge are represented as distinct named graphs, while we encoded in a OWL vocabulary the CKR contextual primitives (e.g. the class $\mathsf{Context}$ of all context individuals, the class Module of all modules and the property $\mathsf{hasModule}$ corresponding to the role mod). The prototype is based on an extension of the Sesame RDF Framework[5] and structured in a client-server architecture: the main component, called *CKR core* and residing in the server-side part, offers the ability to compute and materialize the inference closure of the input CKR, add and remove knowledge and execute queries over the complete CKR structure.

The distribution of knowledge in different named graphs asks for a component to compute inference over multiple graphs in a RDF store, since inference mechanisms in current stores usually ignore the graph part. This component has been realized as a general software layer called *SPRINGLES* (*SParql-based Rule Inference over Named Graphs Layer Extending Sesame*) [5]. Intuitively, SPRINGLES provides methods to demand a closure materialization on the RDF store data: rules are encoded as (named graphs aware) SPARQL queries and it is possible to customize both the used ruleset and the evaluation strategy.

In our case, the ruleset basically encodes the rules of the presented materialization calculus. The rules are evaluated with a strategy that follows the same steps of the translation process defined for the calculus. The plan goes as follows: (i) we compute the inference closure on the graph for global context $\mathfrak{G}$, by a fixpoint on rules corresponding to P_{rl}; (ii) we derive associations between contexts and their modules, by adding dependencies for every assertion of the kind $\mathsf{hasModule}(\mathsf{c}, \mathsf{m})$ in the global closure; (iii) we compute the closure of the contexts, by applying rules encoded from P_{rl} and P_{loc} and resolving *eval* expressions by the metaknowledge information in the global closure.

[5] `http://www.openrdf.org/`

4 Representing events in CKR: CKR-ESO ontology

We can now describe how we translated and implemented a first prototype of the ESO model in the form of a contextualized ontology for the CKR, that we call the *CKR-ESO ontology*.

In this model, the event and situation structures are modelled in the metaknowledge. Similarly to the ESO model, each event instance is associated in the metaknowledge with its pre-, during- and post-situations using the object properties hasPreSituation, hasPostSituation and hasDuringSituation, subproperties of hasSituation. Situation elements associated with events can be generated automatically by SPRINGLES rules when importing an event.

Each event is represented in the metaknowledge as an instance of the class Event: in particular, each event is associated, analogously to the ESO model, with a subclass of the Event class that determines the type of associated situations: in particular, DynamicEvents (e.g. *ChangeOfPossession*, *Constructing*) are typically characterized by their pre- and post-situations, while StaticEvents (e.g. *BeingOperational*) by their during-situations.

This classification is provided by restrictions over the definition of such classes. For example, for the ChangeOfPossession event class, the CKR-ESO ontology states that:

$$\mathsf{ChangeOfPossession} \sqsubseteq \forall \mathsf{hasPreSituation.Pre\_ChangeOfPossession}$$
$$\mathsf{ChangeOfPossession} \sqsubseteq \forall \mathsf{hasPostSituation.Post\_ChangeOfPossession}$$

Each event individual is associated with a knowledge module that corresponds to the RDF graph of the event in the ESO model. This association is represented in the metaknowledge by the property hasEventModule. The following chain axiom is defined over this property, asserting that situations related to an event inherit the facts asserted in the event module: $(\mathsf{hasSituation})^{-} \circ \mathsf{hasEventModule} \sqsubseteq \mathsf{hasModule}$. As defined by the ESO model, we expect to find in the event module the instantiation for all the required roles involved in the event.

The class Situation is defined as a subclass of the Context class in the CKR vocabulary: in other words, in our model we consider situations and their local knowledge as contexts. The particular (pre, post and during) situations associated with event types are modelled by specific context classes. Thus, for example, we have that the pre- and post-situations for events of type ChangeOfPossession are classified as members of the classes Pre_ChangeOfPossession and Post_ChangeOfPossession. The association between such type of situations and their local axioms (i.e. what its modelled in the ESO ontology by situation assertions) is performed by linking specific knowledge modules to these context classes. For example, in CKR-ESO we declare that every pre-situation of ChangeOfPossession is associated with the knowledge module pre_change-of-possession-m and post-situations to post_change-of-possession-m:

$$\mathsf{Pre\_ChangeOfPossession} \sqsubseteq \exists \mathsf{hasModule}.\{\mathsf{pre\_change\text{-}of\text{-}possession\text{-}m}\}$$
$$\mathsf{Post\_ChangeOfPossession} \sqsubseteq \exists \mathsf{hasModule}.\{\mathsf{post\_change\text{-}of\text{-}possession\text{-}m}\}$$

Situation assertions are thus encoded inside these specific modules: the assertions can be basically translated to chain axioms across the roles specified in the event. For example, assertions for pre-situations of ChangeOfPossession stating that:

$$hasInPossession(possession\text{-}owner\_1, possession\text{-}theme)$$
$$notHasInPossession(possession\text{-}owner\_2, possession\text{-}theme)$$

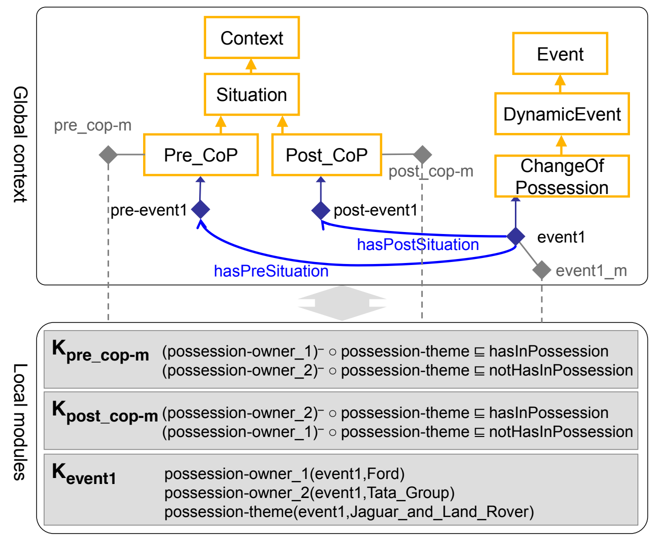

Global context
Context
Situation
Event
DynamicEvent
ChangeOf Possession
pre_cop-m
Pre_CoP
Post_CoP
post_cop-m
pre-event1
post-event1
event1
hasPostSituation
hasPreSituation
event1_m
Local modules
K_pre_cop-m (possession-owner_1)⁻ ∘ possession-theme ⊑ hasInPossession
(possession-owner_2)⁻ ∘ possession-theme ⊑ notHasInPossession
K_post_cop-m (possession-owner_2)⁻ ∘ possession-theme ⊑ hasInPossession
(possession-owner_1)⁻ ∘ possession-theme ⊑ notHasInPossession
K_event1 possession-owner_1(event1,Ford)
possession-owner_2(event1,Tata_Group)
possession-theme(event1,Jaguar_and_Land_Rover)

Fig. 2. Example event in CKR-ESO model.

is translated in CKR-ESO to these chain axioms across role properties:

$$(possession\text{-}owner\_1)^{-} \circ possession\text{-}theme \sqsubseteq hasInPossession$$
$$(possession\text{-}owner\_2)^{-} \circ possession\text{-}theme \sqsubseteq notHasInPossession$$

We now can show how to represent our example event from Section 2 using the CKR-ESO model: we depict this modelling in Figure 2. In the global context, we define event1 to be of type ChangeOfPossession and associate it with its situations:

$$\mathsf{ChangeOfPossession}(\mathsf{event1})$$

$$\mathsf{hasPreSituation}(\mathsf{event1}, \mathsf{pre\text{-}event1})$$
$$\mathsf{hasPostSituation}(\mathsf{event1}, \mathsf{post\text{-}event1})$$

By the above axioms for such event type, we know that the pre- and post-situations of event1 have to be of type Pre_ChangeOfPossession and Post_ChangeOfPossession. By metalevel reasoning, this implies that:

$$\mathsf{hasModule}(\mathsf{pre\text{-}event1}, \mathsf{pre\_change\text{-}of\text{-}possession\text{-}m})$$
$$\mathsf{hasModule}(\mathsf{post\text{-}event1}, \mathsf{post\_change\text{-}of\text{-}possession\text{-}m})$$

and thus the situation assertions associated with the pre- and post-situations of ChangeOfPossession are imported in the two situations[6]. Moreover, the graph associated with the original event is now defined as a module associated with event1 in the metalevel: hasEventModule(event1, event1_m). The event1_m module (i.e. the associated RDF graph) now contains the following facts that are shared with all the situations associated with this event:

$$possession\text{-}owner\_1(\mathsf{event1}, Ford)$$
$$possession\text{-}owner\_2(\mathsf{event1}, Tata\_Group)$$
$$possession\text{-}theme(\mathsf{event1}, Jaguar\_and\_Land\_Rover)$$

[6] In Figure 2 we abbreviate classes of pre- and post-situations of ChangeOfPossession with Pre_CoP and Post_CoP and their modules with pre_cop-m and post_cop-m.

Using the situation assertions in the module associated with the pre-situation, the CKR thus derives the following facts in the context of pre-event1:

$$hasInPossession(Ford, Jaguar\_and\_Land\_Rover)$$
$$notHasInPossession(Tata\_Group, Jaguar\_and\_Land\_Rover)$$

Similarly, in the context of post-event1 we obtain:

$$notHasInPossession(Ford, Jaguar\_and\_Land\_Rover)$$
$$hasInPossession(Tata\_Group, Jaguar\_and\_Land\_Rover)$$

We note that representation of the described event can be completed with its associated during-situation: among the facts that are known to hold during the event, for example, we can assert the existence of the actors and of the possession theme (using the *exists* property in the ESO).

This contextual re-interpretation of the ESO model can bring several advantages from the point of view of reasoning capabilities inside and across events. First of all, every aspect of the reasoning procedure is now strictly ruled by logical reasoning: situation assertions and their association with the type of situation are now directly modelled by the CKR structure and local axioms, without demanding an external reasoner to consider the situation rules and the local reasoning inside situations. Furthermore, the propagation of global object knowledge to local knowledge allows the use of context independent background knowledge in local reasoning. In our example, we can assert in the global knowledge that both actors (*Ford* and $Tata\_Group$) are classified as car companies and their features can be used in local reasoning. More in general, the advantages of an explicit and structured representation of contexts (as the one offered by the CKR) with respect to a modelling based on reification have been shown in [2].

On the other hand, the clear separation of meta and object level reasoning can be exploited to exchange information across the two levels. For example, depending on the type and specific patterns of situation and events, by adding custom SPRINGLES rules it is possible to generate implicit events that have to occur for the completion of the event sequence in a story. In our example, if we have a second event event2 representing another ChangeOfPossession of $Jaguar\_and\_Land\_Rover$ between two companies $Company1$ and $Company2$, different than the two companies from event1, and event2 has a timestamp greater than event1, then we can infer that there have been another two events (possibly being the same one): in one $Jaguar\_and\_Land\_Rover$ has been sold from $Tata\_Group$ and in the other it has been acquired from $Company1$. Similarly, we can recognize cases in which we can assert the equality of certain situations: this can be used to compile sequences of events in a story.

Metalevel information for situations and events can be derived from local reasoning: we might recognize incompatible descriptions of the same event from different news. For example, let us suppose a different representation of the scenario shown in the example in Figure 2: assume that event1 is now classified as Buying (subclass of ChangeOfPossession) while another event event2 is extracted as Selling (also subclass of ChangeOfPossession), but they both represent the same conceptual event (i.e. the acquisition of $Jaguar\_and\_Land\_Rover$ by $Tata\_Group$ from *Ford*). Thus, at the level of the metaknowledge, the two events are modelled as:

```
Buying(event1)
hasPreSituation(event1, pre-event1)
hasPostSituation(event1, post-event1)

Selling(event2)
hasPreSituation(event2, pre-event2)
hasPostSituation(event2, post-event2)
```

Since they represent the same happening, the event modules event1_m and event2_m basically share the same contents: that is, the actors are the same and they take the same role. However, suppose that, due to the extraction from different news sources, the metamodel property *sem:hasTime* associated to post-event1 has value *"August 26th, 2007"* while the value associated to pre-event2 is *"August 28th, 2007"*. Then, using this metalevel information and the local contents of the event modules, we can easily write a reasoning rule that recognizes that the two events are incompatible and adds the assertion event1 incompatibleWith event2 in the global context. Similarly, we can recognize inconsistent situations by local reasoning: this can be used both to exclude further inferences from inconsistent contexts, by marking as "inconsistent" the situation individual in the metaknowledge, but also to repair (possibly with some ad-hoc rules) the local axioms. We note that, on the other hand, this kind of reasoning requires to define ad-hoc rules to recognize such different situations.

Another interesting possibility is the one of having inter-situation knowledge propagation. For example, if two situations or two events are recognized as consequent in a story, unmodified knowledge from the previous situations can be propagated to subsequent situations (e.g. the marital status of Obama did not change when he was elected US president). This clearly presents problems of non-monotonicity, since one has to consider which knowledge can be seamlessly propagated without incurring in contradictory states. In this regard, we recently introduced in CKR a notion of *defeasible axioms* and their overriding across different contexts [3].

5 Conclusions and future works

In this paper we introduced the model of the CKR-ESO ontology, a re-interpretation of the Event and Situation Ontology under the contextual view of knowledge offered by the CKR framework. We discussed informally the advantages of such representation and demonstrated its application by means of an example.

We are currently completing the translation of the full ESO ontology to its contextualized version: we remark that, given the direct translation across the two models, we can easily automatize this transformation.

Our goal is to be able to apply some of the proposed complex reasoning services to the events currently represented in the KnowledgeStore of the NewsReader project: to this aim, we plan to integrate the RDF-based CKR implementation with the KnowledgeStore and encode such reasoning services with respect to CKR contextual model.

Acknowledgments. The research leading to this paper was supported by the European Union's 7th Framework Programme via the NewsReader Project (ICT-316404).

References

1. Björkelund, A., Hafdell, L., Nugues, P.: Multilingual semantic role labelling. In: Proceedings of CoNLL-2009. Boulder, CO, USA (2009)
2. Bozzato, L., Ghidini, C., Serafini, L.: Comparing contextual and flat representations of knowledge: a concrete case about football data. In: K-CAP 2013. pp. 9–16. ACM (2013)
3. Bozzato, L., Eiter, T., Serafini, L.: Contextualized Knowledge Repositories with Justifiable Exceptions. In: DL2014. CEUR-WP, vol. 1193, pp. 112–123. CEUR-WS.org (2014)
4. Bozzato, L., Homola, M., Serafini, L.: Towards More Effective Tableaux Reasoning for CKR. In: DL2012. CEUR-WP, vol. 824, pp. 114–124. CEUR-WS.org (2012)
5. Bozzato, L., Serafini, L.: Materialization Calculus for Contexts in the Semantic Web. In: DL2013. CEUR-WP, vol. 1014. CEUR-WS.org (2013)
6. Corcoglioniti, F., Rospocher, M., Cattoni, R., Magnini, B., Serafini, L.: Interlinking unstructured and structured knowledge in an integrated framework. In: Proc. of 7th IEEE International Conference on Semantic Computing (ICSC), Irvine, CA, USA (2013), (to appear)
7. Das, D., Schneider, N., Chen, D., Smith, N.: Probabilistic frame-semantic parsing. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. HLT'10 (2010)
8. Krötzsch, M.: Efficient inferencing for OWL EL. In: JELIA 2010. Lecture Notes in Computer Science, vol. 6341, pp. 234–246. Springer (2010)
9. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: DOLCE: a descriptive ontology for linguistic and cognitive engineering. WonderWeb Project, Deliverable D17 v2 1 (2003)
10. Motik, B., Fokoue, A., Horrocks, I., Wu, Z., Lutz, C., Grau, B.C.: OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C (Oct 2009), http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/
11. Segers, R., Vossen, P., Rospocher, M., Serafini, L., Laparra, E., Rigau, G.: ESO: a Frame based Ontology for Events and Implied Situations. In: Proceedings of the MAPLEX 2015 Workshop (2015)
12. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. J. of Web Semantics 12 (2012)

Representing Specialized Events with FrameBase

Jacobo Rouces[1], Gerard de Melo[2], and Katja Hose[1]

[1] Aalborg University, Denmark
jrg@es.aau.dk, khose@cs.aau.dk
[2] Tsinghua University, China
gdm@demelo.org

Abstract. Events of various sorts make up an important subset of the entities relevant not only in knowledge representation but also in natural language processing and numerous other fields and tasks. How to represent these in a homogeneous yet expressive, extensive, and extensible way remains a challenge. In this paper, we propose an approach based on FrameBase, a broad RDFS-based schema consisting of frames and roles. The concept of a frame, which is a very general one, can be considered as subsuming existing definitions of events. This ensures a broad coverage and a uniform representation of various kinds of events, thus bearing the potential to serve as a unified event model. We show how FrameBase can represent events from several different sources and domains. These include events from a specific taxonomy related to organized crime, events captured using schema.org, and events from DBpedia.

1 Introduction

The surge of research on large-scale knowledge bases in recent years has largely been driven by the availability of new sources of information about entities. While structured data about millions of places, people, or companies are very valuable, there have been comparably few new results on capturing events of various sorts. Most existing event-oriented ontologies have introduced only a few abstract classes of events, and typical knowledge bases tend to describe just a small number of specific types of events.

Often, however, there is a need to talk about a broad range of very specific sorts of events. For instance, one might want to distinguish battles from both gunfights and from wars, and capture the class-specific details of such events. We adopt a broad notion of events here. This includes the prototypical cases, e.g. local happenings such as concerts, gatherings, or competitions, and world events such as those reported in the news. It also encompasses the more general abstract definition of events, for instance as "happenings in the real world" [15], which would include, e.g., the birth of a person or a commercial transaction between two people. Clearly, such events make up an important aspect of the world that is relevant in knowledge representation, natural language processing, and numerous other fields and tasks. Occasionally, the term *eventuality* is used to denote a broader notion of events that explicitly includes states, e.g. two people knowing each other.

In this paper, we address this challenge of representing many different notions of events under a common schema, from the very prototypical cases to the very abstract, in a way that has both a broad coverage yet supplies sufficient detail to model event-specific properties. For this, we present a new approach for representing event information that is based on FrameBase [12], a broad RDFS-based schema made of frames and their roles. FrameBase provides a predefined vocabulary with event-specific properties for thousands of different kinds of events. For instance, FrameBase's schema accounts for the fact that a battle takes place in a certain time and place and normally involves two parties. For this, the schema draws on two linguistic resources, FrameNet [2] and WordNet [6]. As these describe important fragments of the English lexicon, their coverage is quite substantial. Additionally, as we illustrate later on, FrameBase can be easily extended.

In the following, we prove the suitability of FrameBase for representing different kinds of events by creating rules that integrate instances from different domains:

- A taxonomy of event classes relating to organized crime from the EU FP7 project ePOOLICE[3]. In the project, the event classes in the taxonomy are used as types of entities that are extracted from documents crawled from the web, as part of a strategic early-warning system. The taxonomy was originally captured using the Conceptual Graphs formalism [17]. We use and integrate the event taxonomy as it is, without ad-hoc modifications to the schema.
- The subclasses and properties of the "Event" class in schema.org, which "provides a collection of schemas that webmasters can use to markup HTML pages in ways recognized by major search providers, and that can also be used for structured data interoperability" [1].
- The subclasses and properties of the "Event" class in DBpedia [4], which are extracted from the infoboxes in Wikipedia.
- We conclude with a more general overview of how salient aspects of events [15] can be mapped into FrameBase.

This paper is structured as follows. After describing previous approaches and research in Section 2, a brief overview of the FrameBase schema is given in Section 3. Section 4 then shows how we can rely on the FrameBase schema to represent events from several different sources and domains. Finally, Section 5 provides concluding remarks and describes avenues for future research and applications of our work.

2 Related Work

Considering their importance and unique characteristics, events have been included in numerous upper-level ontologies and vocabularies. In [15], existing event models are reviewed, but these define very broad abstract categorizations or meta-models. Only few example specializations or vocabularies for narrow domains exist, and their overall size is relatively small.

For instance, the Simple Event Model (SEM) Ontology [18] introduces the four types *Event*, *Actor*, *Place*, and *Time*. While it provides a mechanism to create more specific ones by extending these, it does not actually define any specific kinds of events itself. Similarly, the LODE (Linking Open Descriptions of Events) model [16] provides very general concepts, such as the four just mentioned. The event model E [14] proposes a generic structure for the definition of events, but a specific vocabulary is provided

[3] `https://www.epoolice.eu/`

only for the domain of media events with sensor data. The Event Ontology [11] defines a single event class, for which time, place, agents, factors, products, and meronymic relations can be specified, and the domain of focus is music events. Likewise, the Context Ontology (CONON) is limited to the domain of pervasive computing environments [19].

FrameBase's schema instead aims at a broader coverage of many domains by building on natural language resources. Previous work has made use of natural language processing techniques to extract events from text. For instance, one study [5] relies on semantic role labelling (SRL) in conjunction with VerbNet to collect events from text and convert them to the LODE vocabulary mentioned above. Another system [10] extracts events both from text and from semi-structured data. We believe that such automatic extraction methods would benefit from being able to use a standardized wide-coverage representation schema for their output.

3 The FrameBase Schema

The FrameBase schema [12] consists of classes representing frames, and properties representing frame elements. A *frame* describes any kind of situation, state or action, in which several elements, participants (agents, patients, etc.) or properties are involved. Examples include commercial exchanges, marriages, or the act of stomping. The *frame elements* refer to the participants or properties that are involved in a particular frame instance. Common general frame elements include those of agent, patient, time, and location, but not all frames involve these. Frame elements are sometimes also referred to as semantic roles, roles, or theta roles, especially when they are very general.

The frames and the frame elements in FrameBase are organized in hierarchies of classes (based on subclass relationships) and of properties (based on subproperty relationships), respectively. There are three kinds of frames in FrameBase: LU-microframes, synset-microframes and non-lexical frames. Non-lexical frames are very general and are situated in the upper part of the hierarchy. LU-microframes (lexical unit microframes) descend from non-lexical frames, but are much more specific by being associated with the meaning of particular words (the lexical units). They come from FrameNet [7, 13]. Synset-microframes allow an intermediate level of granularity connecting synonymous LU-microframes, e.g. for *marriage* and *matrimony*. These are based on WordNet [6], and thus also have allowed us to extend the coverage of FrameBase beyond that of FrameNet. In the field of linguistics, frames are said to be evoked by words: for example, both the verb *to create* and the noun *creation* evoke the `Creation` frame.

FrameBase additionally provides direct binary predicates to directly connect certain values for elements of a given frame. For example, in a creation event, the agent and the place are directly connected via the `establishesInPlace` relation. This enables more concise queries and representations when only two elements are involved in a particular frame. The frame patterns and the direct binary predicates are logically connected by means of definite clauses that can be used with different kinds of inference systems.

For interoperability with existing resources, FrameBase relies on the standard RDF model [9], which has become a common choice for representing knowledge. This is particularly true in the context of the Linked Data [3], a large Web of datasets referring to and reusing each other's elements. The RDF model uses subject-predicate-object triples to represent statements. Each triple can also be seen as an edge in a directed labelled entity-relationship graph. SPARQL [8] is the standard query language for RDF, which is what we use in order to integrate other event representations into FrameBase.

Event frames are specific kinds of frames, subsuming a range of different notions of events, from the very abstract (e.g., "a natural abstraction of happenings in the real world" [15]) to notions with a notably narrower scope, such as that of widely-known events [10]. Frame elements correspond to what are referred to as *aspects* in the event literature [15]. However, frames can also be more general, and include what the event model E categorizes separately as entities [14]. For example, FrameNet, from which FrameBase is derived, includes a frame `People` that is evoked by lexical units (LUs) such as the noun *man*, and with frame elements such as `Age` and `Origin`.

We believe that the advantage of FrameBase over the existing event models lies on the fact that while extensible as the others, it already provides a broad-coverage vocabulary out of the box in order to bootstrap widespread adoption. Besides, its connection to natural language provides potential advantages, like interfacing with text for question answering or text mining.

FrameBase includes, from FrameNet, an Event frame, which inherits from the `Change of state scenario` frame, and includes a relatively rich hierarchy below for events like creation and destruction events (including more specific ones such as births and deaths), and some others. However, not every event must necessarily fall below this event frame, nor does doing so preclude it from being mapped to other frames that represent other conceptualizations for events, or reflect other perspectives of the frame that stress different aspects than the eventive one. Therefore, the representation of events in FrameBase is not confined to the Event frame and its subframes. We will see examples of this in the next section.

4 Integrating Events

In the first subsections of this section, we present manually built rules for integrating events from three different sources into FrameBase. Later, we add further explanations about these rules and discuss the complexity of the integration rules, and the challenges they present, in particular when they are to be established automatically.

4.1 Representing Events about Organized Crime

The following list of integration rules shows, for each instance of an event class in the organized crime conceptual graph (in bold), the corresponding representation in RDF that it would have in FrameBase. In particular, the main event instance is represented by the anonymous node `_:e`. The default prefix indicates elements that already existed in the core FrameBase schema created from FrameNet and WordNet.

- **Event** `_:e a :frame-Event-event.n`
  - **Act** `_:e a :frame-Intentionally_act-act.n`
    - **Arrest** `_:e a :frame-Arrest-arrest.n`
      - **Drug Possession Arrest** `_:e a :frame-Arrest-arrest.n .`
        `_:e :fe-Arrest-Offense _:e2 .`
        `_:e2 a :frame-Offenses-possession.n`

Human Trafficking Arrest `_:e a :frame-Arrest-arrest.n .`
`_:e :fe-Arrest-Offense _:e2 .`
`_:e2 a :frame-Commerce_scenario-trafficker.n .`
`_:e2 :fe-Commerce_scenario-Goods :frame-People-human.n`

Metal Theft Arrest `_:e a :frame-Arrest-arrest.n .`
`_:e :fe-Arrest-Offense _:e2 .`
`_:e2 a :frame-Theft-theft.n .`
`_:e2 :fe-Theft-Goods :frame-Substance-metal.n .`
`_:e2 a :frame-Offenses-theft.n`

Buy `_:e a :frame-Commerce_buy-buy.v`

Crime `_:e a :frame-Committing_crime-crime.n`

Illegal Drug Use `_:e a :frame-Ingest_substance-use.v`

Consume `_:e a :frame-Ingestion-consume.v`

Inhale `_:e a :frame-Ingest_substance-sniff.v`

Inject `_:e a :frame-Ingest_substance-inject.v`

Possession `_:e a :frame-Offenses-possession.n`

Smoke `_:e a :frame-Ingest_substance-smoke.v`

Organised Crime
`_:e a fbe:frame-Organization-criminal%20organization.n`

Theft `_:e a :frame-Theft-theft.n .`
`_:e a :frame-Offenses-theft.n`

Metal Theft `_:e a :frame-Theft-theft.n .`
`_:e :fe-Theft-Goods :frame-Substance-metal.n .`
`_:e a :frame-Offenses-theft.n`

Trafficking `_:e a :frame-Commerce_scenario-trafficker.n`

Drug Trafficking
`_:e a :frame-Commerce_scenario-trafficker.n .`
`_:e :fe-Commerce_scenario-Goods :frame-Intoxicants-drug.n`

Human Trafficking
`_:e a :frame-Commerce_scenario-trafficker.n .`
`_:e :fe-Commerce_scenario-Goods :frame-People-human.n`

Seizure `_:e a :frame-Taking-seizure.n`

Drug Seizure `_:e a :frame-Taking-seizure.n .`
`_:e :fe-Taking-Theme :frame-Intoxicants-drug.n`

Sell `_:e a :frame-Commerce_sell-sell.v`

Transaction `_:e a :frame-Commercial_transaction-transaction.n`

Crime Transaction
`_:e a :frame-Commercial_transaction-transaction.n .`
`_:e a :frame-Committing_crime-crime.n`

Drug Trafficking Transaction
`_:e a :frame-Commercial_transaction-transaction.n .`
`_:e a :frame-Committing_crime-crime.n .`
`_:e :fe-Commercial_transaction-Goods :frame-Intoxicants-drug.n`

Human Trafficking Transaction

```
_:e a :frame-Commercial_transaction-transaction.n .
_:e a :frame-Committing_crime-crime.n .
_:e :fe-Commercial_transaction-Goods :frame-People-human.n
```

Metal Theft Transaction

```
_:e a :frame-Commercial_transaction-transaction.n .
_:e a :frame-Committing_crime-crime.n .
_:e :fe-Commercial_transaction-Goods :frame-Substance-metal.n
```

The hierarchy in the original ontology is not necessarily consistent with the hierarchy in FrameBase. Only in certain cases does a superclass relationship between two elements of the source also exist between the two elements' respective translations to FrameBase. Therefore, for each translation of an original class of event, the translations of the parents in the original ontology can be added to the set of instances (ABox) in FrameBase, and this will provide additional knowledge that would not always be inferred by the FrameBase schema alone.

We minimize the need for declaring new frames and frame elements for specialized domains by making use of the compositionality of most specialized terms, creating complex structures that combine the semantics of simpler, basic elements. For instance, the translation for the event of type "Drug Possession Arrest" declares an event of type arrest, and specifies that it is about drug possession by assigning drug possession (`Offenses-possession.n`) as the offence.

Owing to this flexibility, we merely needed to mint one single new entity that had not existed in the core FrameBase schema (the microframe `Organization-criminal%20-organization.n`, with the prefix `fbe:` denoting that this is an extension). This exemplifies the potential of FrameBase to represent events from relatively specialized domains, but at the same time the capacity to be extended to fill any possible gaps.

For representing timelines, the frame `Individual_history-history.n` can be used. Each timeline can be represented with one instance of that frame. This instance can be linked with the frame element `Individual_history-Domain` to the topic, which is preferably an entity (or alternatively, a literal or an anonymous node or dummy entity named with a literal). The instance can also be linked with the frame element `Individual_history-Event` to each of the elements in the timeline. Additional frame elements are available in FrameBase, originating from FrameNet, for expressing participants, total duration, etc.

Then, complex queries such as retrieving all events in a given timeline between two given dates, can be built in SPARQL. Similarly, sub-events can be represented with the property path: `^:fe-Part_whole-Part/:fe-Part_whole-Whole`.

4.2 Representing Events from DBpedia.org

We now turn to the `Event` class in DBpedia, and its subclasses, showing how these can be integrated into FrameBase. The integration is implemented using SPARQL CONSTRUCT rules because DBpedia is already in RDF. We only add a couple of subclasses, but most of the properties belong to the parent `Event` class itself.

Top event

```
CONSTRUCT {
  ?e a :frame-Event-event.n .
  ?e :fe-Event-Time _:timePeriod .
    _:timePeriod a fbe:frame-Timespan-period.n ;
      fbe:fe-Timespan-Start ?o1 ; fbe:fe-Timespan-End ?o2 .
  _:e2 a :frame-Relative_time-preceding.a ;
    :fe-Relative_time-Landmark_occasion ?e ;
    :fe-Relative_time-Focal_occasion ?o3 .
  _:e3 a :frame-Relative_time-following.a ;
    :fe-Relative_time-Landmark_occasion ?o3 ;
    :fe-Relative_time-Focal_occasion ?e .
  _:e4 a :frame-Relative_time-following.a ;
    :fe-Relative_time-Landmark_occasion ?e ;
    :fe-Relative_time-Focal_occasion ?o4 .
  _:e5 a :frame-Relative_time-preceding.a ;
    :fe-Relative_time-Landmark_occasion ?o4 ;
    :fe-Relative_time-Focal_occasion ?e .
  ?e :fe-Event-Reason ?o5 .
  ?e a :frame-Social_event-meeting.n ;
    :fe-Social_event-Attendee ?o8 .
} WHERE {
  ?e a dbpedia-owl:Event .
  OPTIONAL{?e dbpedia-owl:startDate ?o1}
  OPTIONAL{?e dbpedia-owl:endDate ?o2}
  OPTIONAL{?e dbpedia-owl:previousEvent ?o3}
  OPTIONAL{?e dbpedia-owl:followingEvent|dbpedia-owl:nextEvent ?o4}
  OPTIONAL{?e dbpedia-owl:causedBy ?o5}
  OPTIONAL{?e dbpedia-owl:duration ?o6}
  OPTIONAL{?e dbpedia-owl:numberOfPeopleAttending ?o7} #Omitted
  OPTIONAL{?e dbpedia-owl:participant ?o8}
}
```

For sub-classes of dbpedia-owl:Event

```
CONSTRUCT {
  ?e a :frame-Social_event-meeting.n .
} WHERE {?e a dbpedia-owl:SocietalEvent}
```

For sub-classes of dbpedia-owl:SocietalEvent

```
CONSTRUCT {
  ?e a :frame-Project-project.n .
  ?e :fe-Project-Activity dbpedia:Space_exploration .
} WHERE {?e a dbpedia-owl:SpaceMission}
```

For sub-classes of dbpedia-owl:SocietalEvent

```
CONSTRUCT {
  ?e a fbe:frame-Social_event-convention.n .
} WHERE {?e a dbpedia-owl:Convention}
```

Out of the 9 properties of the class `Event`, the only omitted one was `numberOfPeopleAttending`, because the class `Event` is too general for it, as it has sub-classes such as `NaturalEvent` (`SolarEclipse`) and `PersonalEvent` (`Birth`, etc.). The `SocietalEvent` class appears more appropriate for this.

4.3 Representing Events from schema.org

Finally, we present the translation of the Event class in schema.org. Again, SPARQL CONSTRUCT rules are used because schema.org can be expressed using RDFa, and SPARQL offers a standard way of representing knowledge graph transformations. Due to space restrictions, we omit the subclasses here, but these have very few genuine properties, and therefore the specialization is relatively simple. Besides, the taxonomy of schema.org events has some inconsistency issues that makes its use complex: the Event class is defined as capturing events such as concerts, lectures, and festivals, with properties such as "typical age range", but there are sub-events such as UserInteraction and UserPlusOnes that actually represent a more general kind of events.

```
CONSTRUCT {
  ?e a :frame-Social_event-meeting.n .
  ?e :fe-Social_event-Time _:timePeriod .
    _:timePeriod a fbe:frame-Timespan-period.n ;
      fbe:fe-Timespan-Start ?Osta ; fbe:fe-Timespan-End ?Oend .
  ?e :fe-Social_event-Duration ?Odur . ?e :fe-Social_event-Place ?Oloc .
  ?e :fe-Social_event-Attendee ?Oatt . ?e :fe-Social_event-Host ?Oorg .
  ?e :fe-Social_event-Occasion ?Osup . ?Osub :fe-Social_event-Occasion ?e .
  ?Ooff a :frame-Offering-offer.v ;
    :fe-Offering-Theme ?e .
  ?e a :frame-Performing_arts-performance.n ;
    :fe-Performing_arts-Performer ?Oper ;
    :fe-Performing_arts-Performance ?Owor .
  _: a :frame-Recording-record.v ;
    :fe-Recording-Phenomenon ?e ;
    :fe-Recording-Medium ?Orec .
} WHERE {
  ?e a sch:Event .
  # Unambiguous translation
  OPTIONAL{?e sch:startDate ?Osta}     OPTIONAL{?e sch:endDate ?Oend}
  OPTIONAL{?e sch:duration ?Odur}      OPTIONAL{?e sch:location ?Oloc}
  OPTIONAL{?e sch:attendee ?Oatt}      OPTIONAL{?e sch:organizer ?Oorg}
  OPTIONAL{?e sch:superEvent ?Osup}    OPTIONAL{?e sch:subEvent ?Osub}
  OPTIONAL{?e sch:offers ?Ooff}        OPTIONAL{?e sch:performer ?Oper}
  OPTIONAL{?e sch:workPerformed ?Owor} OPTIONAL{?e sch:recordedIn ?Orec}
  # Ambiguous translation
  OPTIONAL{?e sch:doorTime ?Odoo}
  # No translation
  OPTIONAL{?e sch:eventStatus ?Oeve}
  OPTIONAL{?e sch:typicalAgeRange ?Otyp}
  OPTIONAL{?e sch:previousStartDate ?Opre}

}
```

The only extension of the FrameBase schema used here was the frame `:frame-Timespan-period.n` with the start and end frame elements, used to denote periods of time. This, however, is not an ad-hoc extension motivated by a particular need of only one source, but a very general one. Out of the 16 properties of the `Event` class, 12 were translated without loss of meaning. One was translated with partial loss of meaning (`doorTime`, translated as a generic start time) and 3 of them were not translated. Whether these can be integrated too, by means of more complex structures, is something we are investigating.

4.4 Mapping Event Aspects to Frame Elements

The survey by Scherp and Mezaris [15] proposes a classification of salient aspects of events. We use this classification to show in a more general way how event aspects can relate to frame elements in the FrameNet-based schema of FrameBase.

- **Time** and **Space**: When applicable, frames include frame elements `Time` and `Place`.
- **Participation**: The classification defines this as "participation of objects in event, where objects can be any living as well as non-living things and include people, buildings, and other even intangible objects like the roles a person plays in a specific situation" [15]. FrameBase provides a large inventory of more specific roles to capture such participants. Often, these correspond to what are sometimes called the proto-agent and proto-patient roles, whose realization in FrameBase depends on the frame. Some examples are `:fe-Commerce_buy-Buyer`, `:fe-Destroying-Destroyer` and `:fe-Destroying-Undergoer`, which are subproperties of `:fe-Getting-Recipient`, `:fe-Transitive_action-Agent` and `:fe-Transitive_action-Patient`, respectively.
- Relations between events.
  - **Mereology**: The relation between two events, when one is part of another. Some frames will have a frame element that will fill this role, like `:fe-Social_event-Occasion` in the example of the Event class in schema.org. In other cases, an additional frame instance of type `:frame-Part_whole` can be used.
  - **Causality**: One event is the cause of another. Some frames will have a frame element that will fill this role, like `:fe-Event-Reason` in the example for the Event class in DBpedia. In other cases, an additional frame instance of type `:frame-Causation` can be used.
  - **Correlation**: When "two (or more) events have a common cause, but this common cause cannot be explained". If we can assume there is a common cause as in the definition, then the causal relationships can be represented with two instances of `:frame-Causation` connecting with an anonymous node for the unknown cause.
- **Documentation**: Events can be "documented using some media like photos or videos captured during the event". This relation is between an event and such documentation. It can be expressed connecting the events by an additional frame of type `:frame-Recording-document.v`, `:frame-Recording-record.v`, and `:frame-Recording-register.v`, or some extension if needed.
- **Interpretation**: This aspect aims at capturing "subjectivity that may exist on the other aspects of events". This is a very broad category that may include different phenomena. The perspectivization relation in FrameNet [13] connects frames representing objective events with frames describing them from a particular perspective. For instance, `:frame-Commerce_Sell` and `:frame-Commerce_Buy` are perspec-

tivizations of :frame-Commerce_Scenario. In other cases, an additional frame instance of a pertinent type can be used, for instance :frame-Becoming_aware.

4.5 Complex Transformations

Most of the integration rules we have described follow a pattern which involves an event *class* in the source being translated as a frame class, and each of their outgoing properties being mapped to individual frame elements. However, there are multiple ways in which the rules can differ from this basic pattern.

1. Sometimes, a class integration rule may need to instantiate multiple frames rather than just a single one. We distinguish two main types of this phenomenon.

   a) The instantiated frame instances may be connected by frame elements. Examples of this include the frame :frame-Timespan-period.n created to represent time periods, and the subframes of Relative_time to express precedence between events (all in the example for dbpedia-owl:Event). The same applies when a frame element is used to specify a frame beyond the lexical unit (see the rule for dbpedia:Space_exploration).

   b) Several frames can also be evoked separately, without the instances being directly connected by any frame element. When these frames describe different perspectives of the same event, there is the possibility that FrameNet links them by means of *perspectivization*, and therefore FrameBase can infer one from another. For example, classes :frame-Commerce_buy-buy.v and :frame-Commerce_sell-sell.v, which are used for classes Buy and Sell in the organized crime taxonomy, are both perspectivizations of :frame-Commerce_goods-transfer. In this case, inference is possible because RDFS subclass and subproperty properties are used in FrameBase to reflect the perspectivization relation between frame classes and frame elements respectively. Another example are :frame-Receive_visitor_scenario and :frame-Visit_host, which are perspectives of :frame-Visitor_and_host. However, in other cases one cannot rely on existing inference. For instance, see how the rule to translate Event from schema.org, besides frames Event-event.n and Timespan-period.n, also instantiates Performing_arts-performance.n, Recording-record.v and Offering-offer.v when certain properties are present.

2. Another possible source of complexity is that frame elements can be inverted. In this case, the integration rules need to invert the order of the arguments, like in the second appearance of :fe-Social_event-Occasion in the integration rule for the class Event in schema.org.

3. Oftentimes, a *property* (rather than a class) in the source can be translated as evoking a frame on its own. In this case, the two involved entities become connected to the new frame by means of frame elements. This would be the case for a property like fightAgainst, which might evoke an event or frame of type armed conflict, about which additional information could be added. None of the examples we have

covered above are of this kind, because we use sources that explicitly represent, or *reify*, events. In other sources, however, this phenomenon appears quite frequently.

Arbitrary combinations of these phenomena are possible (e.g. the rule integrating the Event class from schema.org). Overall, this makes automatic generation of the integration rules a very hard task, because it generates so many free variables that any attempt to train a system would face extreme sparsity. In some cases, it may thus make sense to sacrifice some recall, developing a system that only covers simpler transformations.

4.6 Representational Flexibility

Finally, another potential challenge for data integration is that even when a homogeneous schema such as FrameBase is used, certain kinds of knowledge can still be expressed in multiple possible ways.

- One example is that there are several ways of narrowing down the meaning of a frame instance. One is creating a new sub-microframe associated with a new lexical unit. Another one is assigning a value to a frame element (see example for `SpaceMission`), as mentioned above. This may lead to divergent choices of representation even within the core part of the schema that comes from FrameNet.
- Another example of this is when a frame element needs to be reified, i.e. represented as a frame instance, to express something additional about it (as would be the case of the property `previousStartDate` in schema.org), or when there is no direct frame element available and creating it would lead to a combinatorial explosion in the size of the schema. An example of the latter is the difference between our proposal for using the frame `Part_whole` for expressing sub-event relations, and how we used the frame element `Occasion` for the frame `Social_event`, but this is a particularity of that frame. Again, this may lead to an incoherent representations in the knowledge base. One potential way of addressing this would be extending the reification–dereification mechanism of FrameBase [12].

5 Conclusion

We have shown how events from specialized domains can be represented with the FrameBase schema under a unified model, integrating events in the prototypical sense with more general kinds of events in the sense of abstract happenings or situations. This model has proven to have a high degree of coverage because it needed just few extensions to accommodate the integrated knowledge, and we have illustrated how these extensions can be performed when needed. We have also discussed the various challenges and problems one faces when the integration rules from disparate structured sources of event information are to be built automatically.

Extremely specialized domains, such as quantum physics, may produce lower coverage and need more extensions, although in some cases the creators of FrameNet have also been involved in projects that led to the inclusion of specific scientific and technical domains.

The integration rules that we produce can be used in the future as gold standards for training and testing automatic methods for creating rules from other schemas. We are currently performing research on these methods to integrate further sources such as YAGO2s, Freebase, and Wikidata.

Please refer to `http://framebase.org` for information on using FrameBase and the integration rules.

Acknowledgments The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. FP7-SEC-2012-312651 (ePOOLICE project). as well as China 973 Program Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 61450110088.

References

1. Schema.org. http://schema.org.
2. C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet Project. ICCL '98, pages 86–90, 1998.
3. C. Bizer, T. Heath, and T. Berners-Lee. Linked data–the story so far. *IJSWIS*, 5(3):1–22, 2009.
4. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia-A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.
5. P. Exner and P. Nugues. Using semantic role labeling to extract events from Wikipedia. DeRiVE '11, 2011.
6. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.
7. C. J. Fillmore, C. R. Johnson, and M. R. Petruck. Background to Framenet. *International journal of lexicography*, 16(3):235–250, 2003.
8. S. Harris and A. Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, W3C Consortium, Mar. 2013.
9. P. Hayes and P. Patel-Schneider. RDF 1.1 semantics. Technical report, W3C, 2014. http://www.w3.org/TR/rdf11-mt/.
10. E. Kuzey and G. Weikum. Extraction of temporal facts and events from wikipedia. In *Proceedings of the 2nd Temporal Web Analytics Workshop*, pages 25–32. ACM, 2012.
11. Y. Raimond and S. Abdallah. The event ontology. Technical report, Oct. 2007. http://motools.sf.net/event.
12. J. Rouces, G. De Melo, and K. Hose. FrameBase: Representing N-ary Relations using Semantic Frames. In *Proceedings of the 12th Extended Semantic Web Conference, ESWC*, 2015.
13. J. Ruppenhofer, M. Ellsworth, M. R. Petruck, C. R. Johnson, and J. Scheffczyk. *FrameNet II: Extended Theory and Practice*. ICSI, 2006.
14. A. Scherp, S. Agaram, and R. Jain. Event-centric media management. In *Electronic Imaging 2008*, pages 68200C–68200C. International Society for Optics and Photonics, 2008.
15. A. Scherp and V. Mezaris. Survey on modeling and indexing events in multimedia. *Multimedia Tools and Applications*, 70(1):7–23, 2014.
16. R. Shaw, R. Troncy, and L. Hardman. LODE: Linking Open Descriptions of Events. In *ASWC '09*, Lecture Notes in Computer Science, pages 153–167, 2009.
17. J. F. Sowa. Conceptual graphs. In *In Handbook of Knowledge Representation*, pages 213–237. Elsevier, 2008.
18. W. R. Van Hage, V. Malaisé, R. Segers, L. Hollink, and G. Schreiber. Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):128–136, 2011.
19. X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18–22. Ieee, 2004.