D2.1.1 Knowledge and Provenance Modelling and Stream Reasoning

Nicholas Gibbins (University of Southampton)
Hans-Ulrich Krieger (DFKI)
Thierry Declerck (DFKI)

Abstract.
FP7-ICT Strategic Targeted Research Project (STREP) ICT-2011-287863 TrendMiner
Deliverable D2.1.1 (WP2)

This report constitutes the first deliverable from Task 2.2, and contains three main results: an ontology for representing streaming events, with extensions for provenance and extracted sentiment; domain ontologies for the TrendMiner finance and politics use cases; and an outline description of our proposed semantic stream reasoner.
Keyword list: sentiment, provenance, ontology, stream reasoning

Project	TrendMiner No. 287863
Delivery Date	April 30, 2012
Contractual Date	May 1, 2012
Nature	Report
Reviewed By	Sina Samangooei, Tom Griffith
Web links	-
Dissemination	PU

TrendMiner Consortium

This document is part of the TrendMiner research project (No. 287863), partially funded by the FP7-ICT Programme.

DFKI GmbH
Language Technology Lab
Stuhlsatzenhausweg 3
D-66123 Saarbrcken
Germany
Contact person: Thierry Declerck
E-mail: declerck@dfki.de

University of Southampton
Southampton SO17 1BJ
UK
Contact person: Mahesan Niranjan
E-mail: mn@ecs.soton.ac.uk

Internet Memory Research
45 ter rue de la Rvolution
F-93100 Montreuil
France
Contact person: France Lafarges
E-mail: contact@internetmemory.org

Eurokleis S.R.L.
Via Giorgio Baglivi, 3
Roma RM
00161 Italia
Contact person: Francesco Belini
E-mail: info@eurokleis.com

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

Ontotext AD
Polygraphia Office Center fl.4,
47A Tsarigradsko Shosse,
Sofia 1504, Bulgaria
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Sora Ogris and Hofinger GmbH
Linke Wienzeile 246
A-1150 Wien
Austria
Contact person: Christoph Hofinger
E-mail: ch@sora.at

Hardik Fintrade Pvt Ltd.
227, Shree Ram Cloth Market,
Opposite Manilal Mansion,
Revdi Bazar, Ahmedabad 380002
India
Contact person: Suresh Aswani
E-mail: m.aswani@hardikgroup.com

Changes

Version	Date	Author	Changes
0.1	16.4.2012	Nicholas Gibbins	creation
0.2	24.4.2012	Nicholas Gibbins	added Chapters 2, 3, 5
0.3	28.4.2012	Hans-Ulrich Krieger	added Chapter 4
1.0	30.4.2012	Nicholas Gibbins	final corrections

Executive Summary

Work Package 2 aims to develop novel multilingual ontology-based extraction methods that can be applied to social media streams; the work package has so far been concerned with the application of streaming processing techniques to Semantic Web data in the context of sentiment extraction from social media. This report, the first deliverable from WP2, contains three main results: an ontology for representing streaming events; domain ontologies for the TrendMiner finance and politics use cases; and an outline description of our proposed semantic stream reasoner.

The report begins with a brief state-of-the-art review of stream processing, summarising the key developments in data stream management systems from the database research community, and the nascent work on streaming semantic data from the Semantic Web community. This review informs both the design of our stream reasoning tools, and the design of an ontology for representing social media streams. The representation of streaming social media artifacts is a key concern of the Trendminer project, and to that end we next present an ontology framework for representing streaming social media data, with extensions for provenance and extracted sentiment. Where possible, we have strived to reuse existing ontologies where appropriate; much of the conceptual ground has already been covered by others, and adopting widely-used vocabularies maximises our opportunities to reuse third party data and tools. This ontology is complemented by a pair of ontology sets that aim to represent information from the financial and political use cases; this information includes both subject classifications and some limited relational information. Finally, we give a brief outline of our proposed stream reasoning tools, which will be delivered in Deliverable D2.1.2.

Contents

Chapter 1

Introduction

This document is the first deliverable from Work Package 2, which aims to develop novel multilingual ontology-based extraction methods that can be applied to social media streams; in it, we report on our initial knowledge and provenance modelling work, and provide a sketch of the stream reasoning tools which will constitute Deliverable 2.1.2.

The structure of this document is as follows: in Chapter 2, we give a brief state-of-the-art review of stream processing that we use to inform both the design of our stream reasoning tools, and the design of an ontology for representing social media streams; in Chapter 3, we describe our social media stream ontology and its extensions for provenance and sentiment/opinion; in Chapter 4 we describe the ontologies that will be used to support the finance and politics domains addressed in Work Packages 5 and 6; and finally, in Chapter 5 we give a brief outline of the stream reasoning approach that we will take with the tools that are to be delivered in D2.1.2. In the remainder of this chapter we describe the relevance of this deliverable both to the project outcomes, as given in the Description of Work, and to the research taking place in other Work Packages.

1.1 Relevance to Trendminer

1.1.1 Relevance to project objectives

This deliverable contributes both to project objective 1 ("deliver new real-time trend mining and summarisation methods for stream media") and objective 5 ("deploy a cloud-based infrastructure for real-time data collection, analysis, summarisation, and semantic search").

The ontology described in Chapter 3 of this report will form a common basis for the representation of streaming events from social media services such as Twitter, and so will play a role in support of objective 1.

The outline design of the stream reasoner in Chapter 5 supports objective 5; the second deliverable for Task 2.2 in month 24 (Deliverable 2.1.2) will deliver the reasoner.

1.1.2 Relevance to other work packages

The ontology representation described in Chapter 3 is to be used to represent the extracted sentiments and opinions from the processing tools developed in Work Packages 3 and 4. The current processing infrastructure from these work packages (described in Deliverable 3.1.1) has been designed to handle Twitter data, and so our ontology has been designed with this as a key representational requirement.

In addition, our ontology has been designed to explicitly represent the causal relationship between social media event and extracted information using a profile of the PROV provenance vocabulary (see Section 3.3).

The ontology framework for sentiment and opinion presented in Section 3.2 is relevant to Work Packages 5 and 6 (the financial and political use cases), and is an articulation point between our social media event representation and the domain-specific financial/political ontologies which are described in Chapter 4.

Finally, the stream reasoner described in Chapter 5 will contribute to the real-time stream media platform that is to be developed in Work Package 5.

Chapter 2

Semantic Stream Processing

Traditional database management systems (DBMS) adopt a 'store now, query later' approach in which largely static data is organised in a persistent data set. This is appropriate for an application where the data will be queried repeatedly, and where updates to the data set are small or infrequent; consequently, DBMS typically provide indices or other access structures that improve the efficiency of access to the stored data.

By contrast, many recent applications deal with large volumes of constantly changing dynamic data; these applications are characterised by update event frequencies often in excess of 100,000 events per second, and appear in a variety of domains from online auctions and financial trading systems to sensor networks and social media. The assumptions made by DBMSes, that the amortised cost of building access structures are far outweighed by the savings to be made by using those access structures, no longer hold with such applications, and so different approaches are needed. Moreover, the data rates for some applications are high enough that they may preclude the persistent storage of data in any way.

In the past decade, the database systems community has explored the area of *data stream management systems* (DSMS), special purpose databases which are designed to handle unbounded sequences of time-varying data. These systems are characterised by the use of *continuous queries*, long-lived queries that are matched against streaming data as it is received (as opposed to the one-shot queries that are typical in DBMSes), and by an explicit acceptance of the tradeoff between cost and completeness of results.

Developments in data storage on the Semantic Web over the same period have predominantly focussed on *triplestores* - DBMSes designed for storing RDF triples - and it is only in the last few years that any significant attention has been paid to *streaming semantic data* [della Valle et al., 2009].

In this chapter, we summarise the key developments in data stream management systems from the database research community, and the nascent work on streaming semantic data from the Semantic Web community.

2.1 Data Stream Management Systems

Although work on continuous queries dates back at least as far as the early 1990s [Terry et al., 1992], concerted work on data stream management systems did not start in earnest until a decade later. Babcock et al. [2002] gives a concise survey of the early DSMSes, and we will consider three in more detail as exemplar DSMSes: NiagaraQC, STREAM and Aurora*.

The first of these, NiagaraQC [Chen et al., 2000], is a subsystem of the Niagara project that provides support for continuous queries over multiple distributed XML files. In order to scale effectively, NiagaraQC uses a group optimisation approach, whereby similar queries are grouped together in order to share computation (in this case, the matching of queries against incoming data). This illustrates an important difference between DBMSes and DSMSes; in the former, the majority of query optimisation occurs within each query in isolation, whereas the efficient execution of continuous queries in DSMSes require that a more holistic approach is taken that considers the interactions between queries.

The second system, STREAM [Arasu et al., 2003], takes conventional relational databases as its starting point, rather than the XML-based approach of NiagaraQC. STREAM and its query language CQL have an abstract semantics that defines a data stream as an unbounded set of timestamped tuples (to be compared with the relational algebra, which models relations as sets of tuples). In order to emphasise the connection between relations and streams, STREAM defines operators for converting relations into other relations (effectively the existing relational operators), streams to relations, and relations to streams. Of these, stream to relation operators are of most interest, and typically take the form of a *sliding window* on a stream, such that the instantaneous content of the window may be treated as a relation. This introduces a quality parameter into the system, namely the size of the sliding window; as the window size decreases, the completeness of the results decreases (but the cost of processing also decreases), whereas if infinite-sized windows were chosen, the system would be complete (and behave as a conventional RDBMS).

Aurora* is a federated enhancement of the Aurora DSMS [Cherniack et al., 2003] in which single-node servers cooperate in order to partition a query plan (which perhaps results from the combination of multiple continuous queries) between themselves; individual operators (which in Aurora are roughly equivalent to those discussed in the description of STREAM above) may be migrated between servers according to the availability and distribution of resources (CPU, memory, etc) amongst the servers.

The common feature that all three of these systems share is that they adopt a data flow approach to the processing of incoming data and to the evaluation of continuous queries. Queries are decomposed into fundamental operators that are arranged as the vertices in a directed acyclic graph (the query plan - which may combine multiple queries), the edges of which correspond to the data streams that are the inputs and outputs of those operators. This allows the operators to be executed independently of each other, increasing the

overall flexibility of the system.

2.2 Semantic Stream Processing

Streaming semantic data is a recent development in the Semantic Web community that applies the techniques used in DSMSes to RDF data, and represents a radically different approach to that more usually found in the Semantic Web. The popularity of *linked data* is at least in part due to the assumption of persistence of data (the notion that ”cool URIs don’t change”), whereas streaming data is by its very nature fleeting and ephemeral.

This is in many ways a natural development for the Semantic Web; in the real world, data exists at all points on the spectrum from persistent to ephemeral. Moreover, the application of a DSMS approach to the Semantic Web can be seen as the continuation of a long-standing flow of techniques from the databases community. For example, the development of efficient RDF triple stores has been possible principally because the Semantic Web community has been able to build extensively on forty years of research into relational databases; the semantics of the SPARQL query language [Perez et al., 2009] rely in no small part on the relational algebra.

The transition from DSMSes that operate on streams of arbitrary tuples, to DSMSes that are restricted to operate on RDF triples/quads is therefore a relatively straightforward move that several research groups have made largely independently of each other.

Notable examples of such semantic stream processing are the streaming SPARQL work of Bolles et al. [2008], the C-SPARQL query language of Barbieri et al. [2009] and the EP-SPARQL query language of Anicic et al. [2011]; these languages typically extend the semantics of SPARQL by defining an RDF stream as a set of timestamped subject-predicate-object triples, mostly following the approach made in the STREAM DSMS (with some minor changes; both Bolles and Anicic annotate triples with time intervals, rather than the instants used by STREAM).

2.3 Stream Reasoning

These semantic stream processing approaches may be extended to *stream reasoning*, whereby streams of entailments may be generated from streams of RDF data.One of the earliest references to stream reasoning is by della Valle et al. [2008], who present two conceptual architectures for combining reasoning techniques with data streams. The first of these architectures is based on RDF molecules and reuses existing DSMSes and reasoners by coupling them using a transcoder that converts from the format used by the existing DSMS to timestamped RDF molecules, and a pre-reasoner that incrementally maintains materialised RDF snapshots. These snapshots are passed to conventional SW reasoners that are not aware of time. This may be combined with the incremental materialisation

algorithm described by Barbieri et al. [2010] which maintains the ontological entailments.

This algorithm, based on the *delete and re-derive* approach introduced by Gupta et al. [1993], tags each RDF triple (both inserted and entailed) with an expiration timestamp and applies a sliding window to the stream of timestamped triples. The algorithm then can compute a new complete and correct materialisation by dropping RDF triples that are no longer in the window – effectively temporal truth maintenance.

The second architecture in [della Valle et al., 2008] primarily concerns itself with querying rather than reasoning *per se*, and streams RDF triples (rather than molecules); here, stream operators (like those in the STREAM DSMS) are arranged in query plans. Other similar contemporary approaches include that by Walavalkar et al. [2008], who use the rule-based axiomatisation of RDF Schema to generate a set of continuous queries to be evaluated within the TelegraphQC DSMS [Chandrasekaran et al., 2003], and that by Hoeksema and Kotoulas [2011], who arrange a set of S4 processing elements, whose functionality corresponds to the RDFS entailment rules, to generate a stream of inferred triples which is then fed back into the reasoner (in order to calculate the deductive closure of the stream).

Chapter 3

An Ontology for Streaming Social Media

The representation of streaming social media artifacts is a key concern of the Trendminer project. Work Package 2 aims to develop ontology-based information extraction tools; these tools will require an agreed vocabulary for describing the artifacts. In this chapter, we present an ontology framework for representing streaming social media data: Section 3.1 considers the basic representation of social media events; Sections 3.2 and 3.3 respectively consider the representations of sentiment and provenance.

Where possible, we have strived to reuse existing ontologies where appropriate; much of the conceptual ground has already been covered by others, and adopting widely-used vocabularies maximises our opportunities to reuse third party data and tools. The ontology should also be considered a work in progress; the design in this document is based on our current understanding of the requirements, and we expect the design to evolve as our understanding improves.

3.1 Representing Social Media Events

As mentioned in Chapter 1, our semantic stream processing capability has been designed with streaming social media in mind; the map-reduce processing framework described in Deliverable 3.1.1 has been designed to handle data from the Twitter microblogging service[1], and so we consider the representation of Twitter data to be a key requirement for our ontology.

The input to the processing framework is a stream of JSON objects, each representing an individual tweet, and an abbreviated example of such an object is shown below in Figure 3.1. Much of this data is provided for the convenience of Twitter application

[1] `http://www.twitter.com/`

developers; for example, the `user_mentions` map contains information that could otherwise be parsed out of the text content of the tweet and queried via the Twitter API. For our purposes, the key fields are as follows:

`text` The text of the tweet.

`created_at` The creation time of the tweet.

`user.screen_name` The user name of the creator of the message

`user.id` The id of the creator of the message

`user.lang` The language preference of the creator of the message

`in_reply_to_status_id` The id of the tweet to which this tweet is replying, if any.

`in_reply_to_user_id` The id of the user to which this tweet is being addressed, if any.

Note that the `created_at` field is *not* in RFC2822 date format.

```
{
    "retweeted": false,
    "text": "@CoReyMeKeLL That's good. Do you've any music we could listen to?",
    "coordinates": null,
    "geo": null,
    "created_at": "Sun Oct 10 00:21:14 +0000 2010",
    "retweet_count": 0,
    "favorited": false,
    "contributors": null,
    "in_reply_to_status_id": 26888051316,
    "in_reply_to_user_id": 22946368,
    "entities": {
        "hashtags": [],
        "user_mentions": [{
            "indices": [0, 12],
            "screen_name": "CoReyMeKeLL",
            "name": "Corey Mekell",
            "id": 22946368
        }],
        "urls": []
    },
    "place": null,
    "user": {
        "screen_name": "Music2UPROMO",
        "id": 194873788,
        "lang": "en",
        ...
    },
    "truncated": false,
    "id": 26888968605,
    "in_reply_to_screen_name": "CoReyMeKeLL"
}
```

Figure 3.1: JSON tweet representation

The natural language processing elements within our map-reduce framework further annotate these objects, placing the annotations in the `analysis` map, as shown in Figure 3.2. The fields added to this map are as follows:

`langid` The language detected from the text of the tweet, with an indication of confidence.

`tokens` The tokens identified in the tweet, divided into protected (@ and # tags, , punctuation, emoticons, etc) and unprotected (other words)

`stemmed` A list of stemmed tokens.


```
{
    "retweet_count": 0,
    "in_reply_to_screen_name": "CoReyMeKeLL",
    "text": "@CoReyMeKeLL That\u0027s good. Do you\u0027ve any music we could listen to?",
    "entities": {
        "hashtags": [],
        "user_mentions": [{
            "indices": [0.0, 12.0],
            "screen_name": "CoReyMeKeLL",
            "name": "Corey Mekell",
            "id": 22946368
        }],
        "urls": []
    },
    "user": {
        "screen_name": "Music2UPROMO",
        ...
    },
    "geo": null,
    "coordinates": null,
    "retweeted": false,
    "in_reply_to_status_id": "26888051316",
    "in_reply_to_user_id": "22946368",
    "truncated": false,
    "id": 26888968605,
    "created_at": "Sun Oct 10 00:21:14 +0000 2010",
    "analysis": {
        "stemmed": ["@CoReyMeKeLL", "That\u0027s", "good", ".", "Do", "you\u0027ve",
                    "ani", "music", "we", "could", "listen", "to", "?"],
        "langid": {
            "confidence": 0.006176329312221959,
            "language": "en"
        },
        "tokens": {
            "unprotected": ["good", "Do", "any", "music", "we", "could", "listen", "to"],
            "protected": ["@CoReyMeKeLL", "That\u0027s", ".", "you\u0027ve", "?"],
            "all": ["@CoReyMeKeLL", "That\u0027s", "good", ".", "Do", "you\u0027ve",
                    "any", "music", "we", "could", "listen", "to", "?"]
        }
    },
    "invalid": false
}
```


Figure 3.2: Annotated JSON tweet representation

```
@prefix tm <http://www.trendminer-project.eu/ontology> .
@prefix time <http://www.w3.org/2006/time#> .
@prefix ev <http://purl.org/NET/c4dm/event.owl#> .
:26888968605 a tm:Message ;
  tm:from :22946368 ;
  tm:to :22946368;
  tm:replyTo :26888051316 ;
  tm:text "@CoReyMeKeLL Thats good. Do youve any music we could listen to?" ;
  ev:time [ a time:Instant ;
            time:inXSDDateTime "2010-10-10T00:21:14"^^xsd:dateTime
  ] ;
  tm:lang "en" ;
  tm:langConfidence 0.006176329312221959 ;
  tm:protectedTokens ("@CoReyMeKeLL" "That\u0027s" "." "you\u0027ve" "?") ;
  tm:unprotectedTokens ("good" "Do" "any" "music" "we" "could" "listen" "to") ;
  tm:stemmedTokens ("@CoReyMeKeLL" "That\u0027s" "good" "." "Do" "you\u0027ve"
                    "ani" "music" "we" "could" "listen" "to" "?") .
```

Figure 3.3: Example social media event

There are a number of existing ontologies that could be used to represent tweets, the foremost of which is the SIOC ontology[2]. However, SIOC treats tweets as documents, rather than as events, which

We have chosen to characterise these objects as events in our ontology by focussing on the temporal nature of tweets; this allows us to apply our ontology to a broad variety of social media, from blog posts to status updates on social networking sites. To that end, we use the widely-used Event ontology[3] as a foundation for our ontology; the Event ontology has the added advantage that it in turn builds on the OWL-Time ontology [Hobbs and Pan, 2004], which therefore gives us the ability to perform qualitative reasoning about temporal objects (using the Allen interval calculus). We extend the Event class with a Message class (with the intent that it be used to model messages from systems other than Twitter).

Figure 3.3 gives a usage example for our vocabulary for messages, based on the tweets given in Figures 3.1 and 3.2. Note that we have represented the time of the tweet as a `time:Instant`, with the time itself given as an `xsd:dateTime` (based on ISO8601) rather than as the ad hoc representation in the original JSON representation of the tweet. The screen name and name of the user are not included in the per-tweet representation; users are referred to only by means of their numeric IDs (commuted into URIs). The token lists are represented as RDF collections.

[2] `http://sioc-project.org/`

[3] `http://motools.sourceforge.net/event/event.html`

3.2 Representing Sentiments and Opinions

In addition to representing raw social media events, we must be able to annotate those events with extracted information that represents sentiments and opinions. For example, we would like to be able to associate a tweet that reads *"Loving the new iPad"* with a positive sentiment, and with the fact that it's expressing an opinion about a particular product from a particular company.

Sentiments may be expressed in a number of ways: as a binary value (positive/negative), as a value chosen from a fixed, ordered set (using a Likert scale, for example), as a value from range (-1 for negative to +1 for positive), or simultaneously on a number of orthogonal scales (for example, anxiety, anger, fatigue). In their Twitter-based market prediction system, Bollen et al. [2011] use both a set of keywords derived from the Profile of Mood States (POMS) psychological instrument, and the OpinionFinder[4] system to indicate the sentiment of a tweet.

We are not currently in a position to adopt a single representation for sentiment (nor may we wish to do so at any point), so we do not wish to overly constrain our ontology beyond an general ability to represent multi-dimensional sentiments with quantitative values and associate them with events.

In a similar manner, we do not wish to adopt a single set of terms for the subjects of opinions, given that these are highly domain-specific; within the TrendMiner project alone, we are addressing the worlds of finance (in Work Package 6) and politics (in Work Package 7), and it is our expectation that these Work Packages will choose appropriate vocabularies as the project progresses.

In order to achieve interoperability with a broad range of existing vocabularies and tools, we mandate the use of the SKOS vocabulary [Miles and Bechhofer, 2009] for these opinion and sentiment vocabularies.

In Figure 3.4, we give a usage example of the sentiment and opinion representation. We model sentiments and opinions as annotations of the original message; the `tm:annotates` is used to link the annotation to the message. `http://example.org/sentiment#` and `http://example.org/opinion#` are the URIs for a pair of hypothetical SKOS vocabularies for sentiments and opinions; the resources `s:Mood` and `o:Music` are instances of `skos:Concept`. We model a sentiment as a resource (possibly anonymous) with a dimension (`tm:dimension`) and a value (`tm:value`); the latter is not restricted by a range, in order to support the flexibility required.

3.3 Representing Provenance

Provenance refers to the documented chronology of an object which is used to give the ob-

[4] `http://code.google.com/p/opinionfinder/`

```
@prefix tm <http://www.trendminer-project.eu/ontology> .
@prefix s <http://example.org/sentiment#> .
@prefix s <http://example.org/opinion#> .
:26888968605.1 a tm:Annotation ;
  tm:annotates :26888968605 ;
  tm:sentiment [ tm:dimension s:Mood;
                 tm:value 0.9 ] ;
  tm:subject o:Music .
```

Figure 3.4: Example sentiment and opinion annotation

ject an authoritative status; without an ownership history, a claim that a painting is an old master is likely to be treated with some skepticism. Although it originated in the fine art world, this term has now been applied within many other disciplines, including computer science, where it is used to describe the data produced by computer systems [Moreau et al., 2008]. The provenance of a data object may be considered in three distinct ways:

- Provenance may be considered as a description of the data object's lifecycle history; this sense is close to the traditional fine art meaning of provenance.
- Provenance may be considered as the documentation of some process workflow whose execution led to the creation of the data object. This sense is common in the e-Science community, where provenance is used to support the interpretation of complex distributed experimental workflows that process large volumes of data [Miles et al., 2007]. This provenance information may also capture the difference between planned and actual experiment; documenting such deviations from the intended workflow allows users to understand the weaknesses of provenance [Hughes et al., 2004].
- Provenance may be considered as a form of event annotation. Rather than take a workflow approach that abstracts the history of a data object by considering the functional transformations that have been applied to the object's predecessors, this approach considers the application of those transformations as events from which the past history of the data object may be reconstructed. This is effectively the approach to provenance taken by the CIDOC Conceptual Reference Model [Crofts et al., 2011] for cultural heritage information; CRM distinguishes persistent items (physical objects, for example) from the events that have affected those items.

Given the stream-based nature of the data that we are working with in Trendminer, and the event-based representation described in Section 3.1, the representation of provenance in our ontology is based on the third approach above. In the remainder of this chapter, we briefly discuss two common ontologies for the representation of provenance (the Open Provenance Model, and the W3C PROV ontology), and then give our proposed representation for the provenance of social media data within the Trendminer project.

3.3.1 Open Provenance Model

The Open Provenance Model [Moreau et al., 2011] (OPM) is an abstract model that expresses the provenance of an object as a *causality graph*, a labelled directed graph whose edges represent the causal dependencies between the object and the other entities that were involved in its creation (the nodes in the graph).

OPM defines three kinds of node within the graph:

- Artifact: an immutable piece of state, possibly represented digitally or physically embodied.
- Process: an action or series of actions that are performed on artifacts, and which result in new artifacts.
- Agent: an entity that enables or controls the execution of a process.

The causal dependencies that link nodes to each other (expressed as edges in the causality graph) can be divided into a number of types, based on the types of the nodes that they links together:

- *artifact* used by *process*
- *artifact* generated by *process*
- *process* triggered by *process*
- *artifact* derived from *artifact*
- *process* controlled by *agent*

Both edges and process nodes in the causality graph may be annotated with temporal information, based on the times of a user's observations of the occurrences of those edges and processes. This can be contrasted with the approach taken by the CIDOC CRM, where only entities (viz. nodes in an OPM graph) bear temporal information; CRM therefore reifies key causal dependencies in order to annotate them with temporal information.

The abstract OPM has been encoded as an ontology in OWL, known as OPMO[5] and also as a lightweight RDF Schema vocabulary OPMV[6].

[5] `http://openprovenance.org/model/opmo`
[6] `http://open-biomed.sourceforge.net/opmv/ns.html`

3.3.2 W3C PROV-O

Following the interest in OPM and provenance in general in the Web and e-Science communities, the World Wide Web Consortium (W3C) chartered an incubator group on provenance that published its final report in 2010 [Gil et al., 2010]. The response to this report was such that W3C chartered a working group whose mission is to define a language (PROV) for exchanging provenance information among applications.

This working group is chartered to run until December 2012, and draws on a number of existing vocabularies, including OPM. At the time of writing, the documents produced by the Provenance Working Group are still working drafts, however the likelihood is that the features discussed in this section will not change substantially before the documents reach Recommendation status. As was also the case with OPM, PROV separates the abstract data model (PROV-DM) [Moreau and Missier, 2012] from the encoding of that model as an ontology (PROV-O) [Sahoo and McGuinness, 2011].

The PROV data model is broadly similar to that of OPM, with *entities* of which provenance may be asked (compare with OPM artifact), *activities* which are applied to entities (compare with OPM process) and *agents* that are assigned some degree of responsibility for activities (equivalent to OPM agent). The relations between these classes of object are also similar to those of OPM, but mainly for entity-entity and entity-activity relations.

PROV takes an event-based view of provenance, with generation and usage events for entities, and start and end events for activities; this is a refinement of the temporal approach taken by OPM, but one that is more closely compatible with the event-based approach that we have taken in our ontology. PROV also (currently) makes use of the OWL-Time ontology [Hobbs and Pan, 2004] for temporal information, which fits with our choice of the Event ontology in Section 3.1.

Our proposed ontology for provenance is therefore based on a parsimonious profile of the draft PROV-O ontology from which has been removed those components that are unlikely to be used (principally roles and recipes), and an example of its use is shown in Figure 3.5. This example repeats the annotation from Figure 3.4, but adds the extra information that the annotation was derived from the original message, and that it was generated by a hypothetical extraction process identified by `http://example.org/process/extractor`; we intend to use this property to identify the processing tools described in D3.1.1.

```
@prefix tm <http://www.trendminer-project.eu/ontology> .
@prefix po: <http://www.w3.org/ns/prov-o/> .
@prefix s <http://example.org/sentiment#> .
@prefix s <http://example.org/opinion#> .
:26888968605.1 a tm:Annotation ;
  tm:annotates :26888968605 ;
  po:wasDerivedFrom :26888968605 ;
  po:wasGeneratedBy <http://example.org/process/extractor> ;
  tm:sentiment [ tm:dimension s:Mood;
                 tm:value 0.9 ] ;
  tm:subject o:Music .
```

Figure 3.5: Example provenance annotation

Chapter 4

Domain Ontologies

For domain ontologies, we consider the use of two (sets) of ontologies. The first has been developed at DFKI, in synergy with the European R&D Monnet project (Multilingual ONtologies for NETworked knowledge)[1] and is an integrated set of ontologies which encodes in a class hierarchy, together with associated properties, information from business reporting legislations, information as it is displayed in the web presence of stock exchanges and information about industry activity fields as encoded in standards like IBC and NACE[2]. The work done in synergy with Monnet consists of generating from these information sources an integrated ontology that covers a large set of financial information. While in Monnet the focus was on the issues related to the representation of multilingual information (in the form of `rdfs:label` or `skosxl:literalForm` annotation properties), the focus within TrendMiner is on extending the information base, for example with provenance, opinion and sentiment features, and detailed temporal information. We are currently adding to the integrated ontology the schema we have extracted from the Borsa Italiana (in order to cover also Italian language, see `http://www.borsaitaliana.it/`) and the industry classification GICS, complementary to the ICB.[3] In section 4.1, we give more details about the integrated financial ontology. The second domain knowledge source we are using is "TheSoz" (Thesaurus for the Social Sciences) provided by the GESIS Leibniz Institute for the Social Sciences[4]. While we still have to integrate this knowledge source into our ontology format, the data itself is encoded using already semantic web standards, like SKOS and RDF. A short description of this knowledge source is given in 4.2

[1] See `http://www.monnet-project.eu/`

[2] See `http://www.icbenchmark.com/` and `http://epp.eurostat.ec.europa.eu/portal/page/portal/nace_rev2/correspondence_tables` correspondingly.

[3] ICB stands for "Industrial Classification Benchmark", and GICS for "Global Industrial Classification Standard". See `http://www.icbenchmark.com/` and `http://www.standardandpoors.com/indices/gics/en/us` respectively.

[4] See also `http://lod.gesis.org/pubby/page/thesoz/` for an HTML representation

4.1 The MFO Ontology

The focus of this section lies on the construction of an integrated ontology – the Monnet Financial Ontology (MFO) – that has been assembled from several independent ontologies which are brought together by an interface specification, expressed in OWL. MFO is an ontology that is centered around companies (and people therein) that are listed in, for instance, the DAX or the Euronext 100 indices. MFO also addresses financial reporting through XBRL/XEBR and establishes a bridge between those reports and the stock exchange titles.

Originally, we started in the Monnet project with company snapshots from the DAX index, resulting in the *DAX* ontology which has turned out to be universal for the other Xetra indices MDAX, SDAX, and TecDAX. Later in the project, we extended *DAX* in two ways:

1. We incorporated axioms to store parts of the annual XBRL company reports, and
2. We imported our own OWL version of the NACE taxonomy for characterizing companies against a classification of industry sectors.

At a later stage, further information came in, so that we opted to *separate* independent information from one another, not only by introducing different namespaces, but also by locating this information in distinct ontologies, so that they could be *reused* by other projects and applications. In order to bring these independent ontologies together, we used standard descriptive means from RDFS and OWL, such as `owl:equivalentClass`, `rdfs:subPropertyOf`, or `rdfs:range` in order to state "connection" axioms. These axioms are collected in two separate interface ontologies *IF* and *XEBR2XBRL* described below: *IF* deals with stock exchange integration, whereas *XEBR2XBRL* aligns concepts & properties from XEBR and XBRL.

It is worth noting that the *classification of companies against industry sectors* is of utmost importance in the TrendMiner project, allowing us to check opinions and trends not only for companies, but also for whole industry branches. We have thus decided to view the industry sectors as *subclasses* of the class `Company`, both in the *DAX* (Xetra) and *EN* (NYSE Euronext) ontology. Overall, MFO now provides three, partly overlapping company/industry sector classifications:

1. *DAX* came up with a coarse and flat string-based characterization of 11 industry sectors. We had manually turned these values into direct subclasses of class `dax:Company`. In early 2012, *Deutsche Börse* restructured their pages, adding more sectors and a further layer of sub-sectors.
2. *EN* makes use of the ICB industry sector classification. We have automatically transformed the latest ICB specification into an OWL ontology *ICB* which is used as a subclass hierarchy for `en:Company`.

3. In an older project, we utilized the quite exhaustive NACE industry sector classification from which we have automatically constructed our own OWL ontology *NACE*.

Overall, MFO consists of the **nine**, truly independent sub-ontologies which do not have knowledge of one another. **Two** further ontologies, called *IF* and *XEBR2XBRL* bring them together in this project (see Figure 4.1). Further numbers describing the size of each sub-ontology can be found in Figure 4.2.

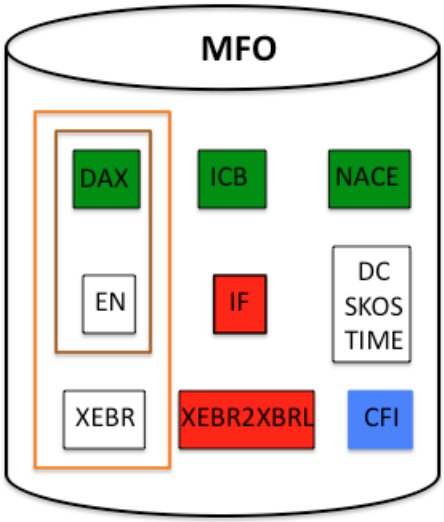

Figure 4.1: The Monnet Financial Ontology consists of 11 sub-ontologies overall. The color encoding refers to ontologies focussing on models of *industry sector classification* (**green**), *stock exchange* (**brown**), *reporting* (**orange**), *financial instruments* (**blue**), and *interface* (**red**). As can be seen from the picture, some of the ontologies even model several aspects of our domain; e.g., *DAX* alone deals with industry sector classification, reporting, and the description of stock exchange listed information.

4.1.1 *DAX*

We have already mentioned above that the 2011 DAX version came with a small set of 11 industry sectors which we turned into direct subclasses of `dax:Comapny`. The 2012 version replaced the 11 categories by 18, but also added a further layer of 63 subsectors. The 81 subclasses are furthermore assigned an informal description in English and German through the use of the RDFS annotation property `rdfs:label`. *DAX* mainly focusses on *companies*, but also on *people*, since information about the executive and supervisory board, as well as human shareholders are mentioned. Financial numbers are listed (e.g., the total capital stock), as also are other master data, such as the ISIN number, the foundation year, or the end of the business year.

4.1.2 *EN*

The Euronext ontology *EN* does invent with its own industry classification, but makes use of ICB (see below). In a certain sense, *EN* reduplicates the stock exchange ontology *DAX*, but uses different names for classes and properties. However, more financial numbers are given here, even for three successive years.

Let us take an example. *Credit Agricole*'s revenues for 2011, 2010, and 2009 are listed today as 20,783,000, 20,129,000, and 17,942,000 Euros. Even at the end of this year, these numbers will be the same. However in 2013, the number for 2009 will no longer be listed, but instead, we will then find only numbers for 2012, 2011, and 2010. Clearly, we do *not* want to extend the ontology with new property names every time a new business year starts, thus we must avoid properties such as `hasRevenue2012`, `hasRevenue2011`, `hasRevenue2010`, etc.

In order to address this and to proper represent the numbers against the varying date when the numbers where taken, we use a simple "trick" here: we always use exactly the three properties `hasRevenue-1`, `hasRevenue-2`, and `hasRevenue-3`. The hyphen - now should be interpret as a minus sign −, thus, e.g., the value stored under `hasRevenue-`**2** actually refers to the revenue *two* (**2**) years ago *relative* to the actual business year when the company snapshot was taken (the business year is stated elsewhere).

4.1.3 *NACE*

NACE (*Nomenclature statistique des Activités économiques dans la Communauté Européenne*) provides a huge industry sector classification which we hooked up with *DAX* through a connection axiom in *IF*. NACE is a four-layered taxonomy of 996 concepts that has been turned into an OWL ontology in a former project. The NACE sectors were represented as *instances* of the single class `IndustrySector` and were connected through the object property `subSectorOf`. Two further datatype properties `hasCode` and `hasLevel` were also defined on `IndustrySector`. An informal description of the instances in English, German, and Italian had been realized through the use of `rdfs:label`.

As explained above, we view industry sectors as subclasses of `Company`, so we have turned the NACE instances into *classes*, connected this time through the obvious subsumption relation `rdfs:subClassOf`. Thus, there is no longer a need to use `subSectorOf` here. In order to keep the information associated with `hasCode` and `hasLevel`, these properties have now been declared as annotation properties. Since NACE does not state a unique top sector, we use the synthetical superclass `IndustrySector` to head the 21 top-level sectors in NACE. Finally, the multilingual information has been transferred from the instances to the classes.

4.1.4 *ICB*

ICB, the *Industry Classification Benchmark*, is a further industry classification used worldwide at many places (e.g., NYSE in New York). ICB is proprietary to FTSE International and Dow Jones. Euronext (as operated by NYSE) makes use of ICB's four level deep classification in its description of titles. Given the ICB terminology stated in a document, we have auto-generated an OWL ontology *ICB* that arranges the 186 industry sectors in a subsumption hierarchy. *ICB* is connected to *EN* through an axiom from *IF* and comes up with an informal sector description for English, German, and Spanish, together with a further multi-lingual "definition" of the most specific concepts (actually, even 11 languages are available). In order to address these definitions on the class level properly, we make use of a further annotation property called `definition`.

4.1.5 *CFI*

CFI, the *Classification of Financial Instruments* specifies a taxonomy that cross-classifies financial instruments, e.g., tradable assets of any kind. CFI is an ISO standard and is used to further classify Euronext titles. CFI descriptions are six-letter codes: the first letter describes a *category* (such as Equities), followed by a so-called *group* description, together with four additional special attributes.

The CFI ontology is quite simple in our case: instead of using a subclass hierarchy, we have opted for a single class `CFI`, together with six datatype properties, representing the six letters of a CFI code. This approach nicely complements the fact that CFI codes need not always be six letters long and can even be underspecified (using the special * character).

4.1.6 *XEBR*

XEBR is our OWL version of the XEBR taxonomy (version 3). *XEBR* has been auto-generated from the XEBR specification and essentially model a part-of hierarchy of shared "presentations" of XBRL financial reports. For instance, a *fixed assets presentation* is part of an *assets presentation* which is part of the *key balance sheet figures* which finally is part of an *XEBR report*. Presentations are modelled by OWL classes, whereas the actually values are accessible through functional OWL datatype properties, usually mappings to monetary values, defined on those presentations. For example, the class `TangibleFixedAssetsPresentation` comes with two properties: `hasOther-TangibleAssets` and `hasTangibleFixedAssetsTotal`.

In order not to reduplicate the fixed part-of hierarchy each time for every document, we have abandoned extensional parthood in favor of an intensional representation: instead of defining the part-of hierarchy over instances of presentation classes, we have opted to use the classes directly. In order to define the membership of an instantiated presentation

with a specific report, a special `belongsTo` property is employed here. We finally note that the temporal information (the time, the report talks about) is associated with the report itself through properties `starts` and `ends`.

4.1.7 *DC*

The manually constructed ontology *DC* makes use of the concept `dc:Resource` from *Dublin Core* ontology, together with several relations such as `dc:date` or `dc:source` which we have modelled as annotation properties in OWL. *IF* furthermore states that `xebr:Report` is a subclass of `dc:Resource`. At the moment, *DC* is solely used in XEBR ABox data.

4.1.8 *SKOS*

SKOS, as a reminiscence of the *Simple Knowledge Organization System* makes use of object and annotation properties, originally defined in SKOS to express "matching" relations separate from equivalence and equality, and to allow further labels beyond `rdfs:label`, such as `skos:prefLabel` and `skos:altLabel`. We use some of these properties in the *XEBR2XBRL* interface specification below and in XEBR ABox data.

4.1.9 *TIME*

TIME is a small, manually-constructed ontology that allows us to cross-classify relations as being either *synchronic* or *diachronic*, i.e., to state whether they stay constant or change over time. E.g., the functional property `dax:foundedIn` is synchronic, since the founding year (like the date of birth of a person) will not change over time:

```
dax:foundedIn rdf:type owl:FunctionalProperty .
dax:foundedIn rdf:type time:SynchronicProperty .
```

However, the relational property `dax:executiveBoard` is diachronic, because people resign from an executive board and other come in at different times.

TIME (meta-)models synchronic and diachronic properties through specialized OWL-like property characteristics, such as `owl:FunctionalProperty`: we have defined two pairwise disjoint classes `time:SynchronicProperty` and `time:DiachronicProperty` as *subclasses* of the class `rdf:Property`. Other ontologies might then use these specialized classes to type their properties.

4.1.10 *IF*

As already explained, *IF* "binds" *DAX*, *EN*, *ICB*, *NACE*, *CFI*, and parts of *XEBR* together. To achieve this, *IF* makes use of *DC*, *SKOS* and *TIME*, and the standard axioms constructors from RDFS and OWL, viz.,

- `owl:equivalentClass`
- `rdfs:subClassOf`
- `owl:equivalentProperty`
- `rdfs:subPropertyOf`
- `owl:sameAs`
- `rdfs:domain`
- `rdfs:range`
- `rdf:type`

Here are some examples, using description logics (DL) syntax.

Classes & Properties

`dax:Company`, `en:Company`, and `nace:IndustrySector` can be used interchangeably; `xebr:Report` is a subclass of `dc:Resource`; the properties `dax:portrait` and `en:activity` are equivalent:

$$\text{dax:Company} \equiv \text{en:Company}$$

$$\text{dax:Company} \equiv \text{nace:IndustrySector}$$

$$\text{xebr:Report} \sqsubseteq \text{dc:Resource}$$

$$\text{dax:portrait} \equiv \text{activity}$$

Domain/Range Restrictions & Typing

XEBR reports are linked to companies via the diachronic functional object property `if:hasReport`:

$$\top \sqsubseteq \forall \text{if:hasReport}^{-} \,.\, \text{dax:Company}$$

$$\top \sqsubseteq \forall \text{if:hasReport} \,.\, \text{xebr:Report}$$

$$\text{if:hasReport} : \text{owl:FunctionalProperty}$$

$$\text{if:hasReport} : \text{owl:ObjectProperty}$$

$$\text{if:hasReport} : \text{time:DiachronicProperty}$$

Meta-Modelling

The class `DiachronicProperty` is a subclass of `rdf:Property`; the property `partOf` is a property connecting OWL classes, *not* instances.

time:DiachronicProperty $\sqsubseteq$ rdf:Property

$\top \sqsubseteq \forall$xebr:partOf^{-} . owl:Class

$\top \sqsubseteq \forall$xebr:partOf . owl:Class

4.1.11 *XEBR2XBRL*

XEBR2XBRL interfaces parts of XEBR with XBRL: concepts and properties are made equivalent, and multilingual labels are assigned, using constructs from *SKOS*. For instance:

xebr:AssetsPresentation $\equiv$ xbrlbe:AssetsTitle

xebr:AssetsPresentation $\equiv$ xbrles:ActivoPresentacion

xebr:hasAssetsTotal $\equiv$ xbrles:hasTotalActivo

4.2 TheSoZ

The Thesaurus contains 12.747 German labels and 11.816 English labels.[5] Bi-lingual labels are easily extractable. The concepts in TheSoZ and the associated terminology (contained in the skosxl:literalForm annotation properties) are related to the large field of social and political sciences and so give a vocabulary against which we can write match content of streaming data, as analysed by tools described in Deliverable D3.1.1, trying to better classify the resulting information, in term of a class hierarchy and associated properties. An example taken from TheSoz in the social field, all the information related to the concept ”10036355”:

```
<rdf:Description rdf:about="concept/10036355">
   <rdf:type rdf:resource="../thesoz/ext/Descriptor"/>
</rdf:Description>
<rdf:Description rdf:about="concept/10036355">
   <skos:inScheme rdf:resource="http://lod.gesis.org/thesoz/"/>
</rdf:Description>
<rdf:Description rdf:about="concept/10036355">
   <skosxl:prefLabel rdf:resource="term/10036355"/>
</rdf:Description>
<rdf:Description rdf:about="concept/10036355">
```

[5] A French version is now also available, and we will consider it in near future.

	CFI	DAX	DC	EN	ICB	IF	NACE	SKOS	TIME	XEBR	XEBR to XBRL	MFO
#concepts	1	97	1	9	186	19	997	1	3	63	11	1366
#eq class axioms	--	--	--	--	--	4	--	--	--	--	7	11
#object properties	--	7	--	6	--	37	--	6	--	6	4	61
#eq obj prop axioms	--	--	--	--	--	17	--	--	--	--	2	19
#datatype properties	6	22	--	42	--	--	--	--	--	154	123	295
#eq data prop axioms	--	--	--	--	--		--	--	--	--	71	71
#individs	--	40	--	48	--	7	--	--	--	60	137	288
#annotat properties	--	1	4	--	1	--	2	3	--	--	--	11
#explicit triples (= #axioms)	28	744	11	278	1,370	81	6,983	31	10	1,164	783	11,483
size [KB]	4	111	4	41	229	12	967	8	4	180	127	1,687

Figure 4.2: Numbers describing the size of each sub-ontology in terms of the number of classes, properties, and axioms.

```
      <skosxl:altLabel rdf:resource="term/10034998"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355">
      <skosxl:literalForm xml:lang="de">Arbeitsloseninitiative
      </skosxl:literalForm>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355">
      <rdf:type rdf:resource="http://www.w3.org/2008/05/skos-xl#Label"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355-en">
      <skosxl:literalForm xml:lang="en">unemployed person
      initiative</skosxl:literalForm>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355-en">
      <rdf:type rdf:resource="http://www.w3.org/2008/05/skos-xl#Label"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355-en">
```

```
      <ext:hasTranslation rdf:resource="term/10036355"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355">
      <ext:hasTranslation rdf:resource="term/10036355-en"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355">
      <skos:notation>1.2.01</skos:notation>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355">
      <skos:broader rdf:resource="classification/1.2.01"/>
   </rdf:Description>
   <rdf:Description rdf:about="classification/1.2.01">
      <skos:narrower rdf:resource="term/10036355"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355-10034998">
      <rdf:type rdf:resource="ext/EquivalenceRelationship"/>
      <ext:UF rdf:resource="term/10034998"/>
      <ext:USE rdf:resource="term/10036355"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10034998">
      <ext:EquivalenceRelationship rdf:resource=
      "term/10036355-10034998"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10036355">
```

And an example taken from the political field:

```
 <rdf:Description rdf:about="concept/10044148">
      <rdf:type rdf:resource="../thesoz/ext/Descriptor"/>
   </rdf:Description>
   <rdf:Description rdf:about="concept/10044148">
      <skos:inScheme rdf:resource="http://lod.gesis.org/thesoz/"/>
   </rdf:Description>
   <rdf:Description rdf:about="concept/10044148">
      <skosxl:prefLabel rdf:resource="term/10044148"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148">
      <skosxl:literalForm xml:lang="de">freie Whlervereinigung
      </skosxl:literalForm>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148">
      <rdf:type rdf:resource="http://www.w3.org/2008/05/skos-xl#Label"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148-en">
      <skosxl:literalForm xml:lang="en">independent voters'
```

```
      association</skosxl:literalForm>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148-en">
      <rdf:type rdf:resource="http://www.w3.org/2008/05/skos-xl#Label"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148-en">
      <ext:hasTranslation rdf:resource="term/10044148"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148">
      <ext:hasTranslation rdf:resource="term/10044148-en"/>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148">
      <skos:scopeNote xml:lang="en">in local elections
      </skos:scopeNote>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148">
      <skos:notation>4.4.04</skos:notation>
   </rdf:Description>
   <rdf:Description rdf:about="term/10044148">
      <skos:broader rdf:resource="classification/4.4.04"/>
   </rdf:Description>
   <rdf:Description rdf:about="classification/4.4.04">
      <skos:narrower rdf:resource="term/10044148"/>
   </rdf:Description>
```

As mentioned above, GESIS is a thesaurus, but already makes use of many Semantic Web features and standards. We are currently "upgrading" this knowledge source into an ontology, which will be integrated with other ontologies, as this is currently the case for the ontologies we have for the financial domain. In addition, we will also seek to integrate information about time, provenance, and opinion and sentiment features.

Chapter 5

A Rete-based Stream Reasoner

In this chapter, we briefly describe the high-level design of the stream reasoner which is to be delivered in Deliverable 2.1.2.

Rather than reuse an existing reasoner, as in the RDF molecule stream reasoning approach described in Section 2.3, we have chosen to develop a RDF triple stream reasoner. This approach several aspects of the systems described above, specifically the fine-grained streaming of [della Valle et al., 2008], the rule-based nature of [Walavalkar et al., 2008] and the re-entrant streams of [Hoeksema and Kotoulas, 2011].

Key to this approach is the development of an efficient query plan that corresponds to the entailment rules that are in use. We intend to use the widely-used and well-understood Rete pattern matching algorithm [Forgy, 1979] to transform the rule bodies into a query plan. The Rete algorithm has long been used to improve the efficiency of matching facts against productions in production systems, and is in essence a data flow system; partial matches are propagated through a network of alpha and beta nodes (effectively select and join operators) in such as way as to minimise the number of times that each new fact is matched against a pattern. In adapting the Rete algorithm to streams of RDF triples, the alpha and beta memories become respectively streams and windows on streams (beta memories appearing immediately before beta or join nodes); this is effectively the approach taken by Jin et al. [2005] in the ARGUS stream processing system.

In order to support entailment rules beyond those in the axiomatisation of the ontology language, we propose to accept Datalog rules expressed in the BLD dialect of the Rule Interchange Format[Boley and Kifer, 2010]; these will be compiled to a query plan using Rete and deployed such that the output stream from the rule network (the merge of the streams resulting from the terminal nodes that represent the head of each rule) is fed back into the network in order to calculate the deductive closure of the entailment rules, as in [Hoeksema and Kotoulas, 2011]. The output stream can also be fed into a second Rete network, this time for the continuous queries that have been registered with the system. This partitioning of the Rete network simplifies its management; the rule network is relatively static (the entailment rules are expected to persist for the lifetime of the system),

whereas the query network is more dynamic (although long-lived, the continuous queries do not necessarily persist as do the entailment rules). An outline sketch of the reasoner from a data flow perspective is shown in Figure 5.1.

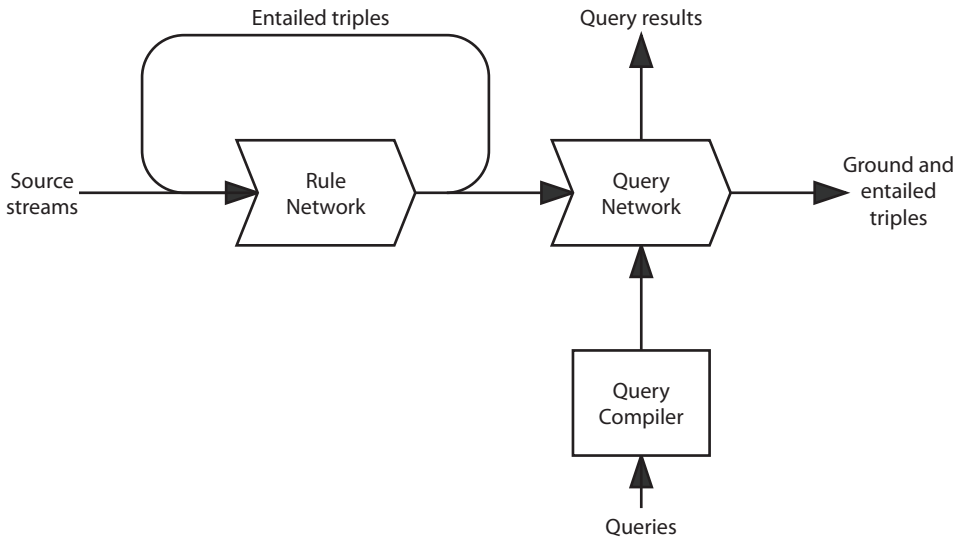

Figure 5.1: Stream Reasoner data flow

The implementation platform for the reasoner has yet to be decided, but in order to make best use of project resources we intend to use an existing stream processing framework in order to reduce the complexity of implementation; we are currently considering a number of frameworks including ESPER[1].

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

Bibliography

Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. EP-SPARQL: a unified language for event processing and stream reasoning. In *WWW '11: Proceedings of the 20th international conference on World Wide Web*, 2011.

Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Rajeev Motwani, Itaru Nishizawa, Utkarsh Srivastava, Dilys Thomas, Rohit Varma, and Jennifer Widom. STREAM: The Stanford Stream Data Manager. *IEEE Data Engineering Bulletin*, 26 (1):19–26, 2003.

Brian Babcock, Shivnath Babu, Rajeev Motwani, and Jennifer Widom. Models and Issues in Data Stream Systems. In *Proceedings of the 21st ACM Symposium on Principles of Database Systems*, pages 1–16, January 2002.

Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele della Valle, and Michael Grossniklaus. C-SPARQL: SPARQL for continuous querying. In *WWW '09: Proceedings of the 18th international conference on World wide web*. ACM, April 2009.

Davide Francesco Barbieri, D Braga, S Ceri, Emanuele della Valle, and M Grossniklaus. Incremental reasoning on streams and rich background knowledge. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC'10)*, pages 1–15. Springer, 2010.

Harold Boley and Michael Kifer. RIF Basic Logic Dialect. W3C Recommendation REC-rif-bld-20100622, World Wide Web Consortium, June 2010.

Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, March 2011.

A Bolles, M Grawunder, and J Jacobi. Streaming SPARQL - Extending SPARQL to process data streams. *Proceedings of the 5th European Semantic Web Conference (ESWC2008)*, page 15, March 2008.

S Chandrasekaran, O Cooper, A Deshpande, MJ Franklin, JM Hellerstein, W Hong, S Krishnamurthy, Samuel Madden, F Reiss, and MA Shah. TelegraphCQ: Continuous Dataflow Processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. *SIGMOD Record*, 29(2):379–390, June 2000.

Mitch Cherniack, H Balakrishnan, and M Balazinska. Scalable Distributed Stream Processing. In *Proceedings of the 1st Biennial Conference on Innovative Data Systems (CIDR)*, January 2003.

Nick Crofts, Martin Doerr, Tony Gill, Stephen Stead, and Matthew Stiff. Definition of the CIDOC Conceptual Reference Model. Technical report, CIDOC CRM Special Interest Group, November 2011.

Emanuele della Valle, S Ceri, Davide Francesco Barbieri, and D Braga. A first step towards stream reasoning. In *Proceedings of Future Internet - FIS 2008*, pages 72–81, January 2008.

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

Charles Forgy. *On the Efficient Implementation of Production Systems*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, February 1979.

Yolanda Gil, James Cheney, Paul Groth, Olaf Hartig, Simon Miles, Luc Moreau, and Paulo Pinheiro da Silva. Provenance XG Final Report. Incubator Group Report XGR-prov, World Wide Web Consortium, dec 2010.

Ashish Gupta, Inderpal Singh Mumick, V S Subrahmanian, Ashish Gupta, Inderpal Singh Mumick, and V S Subrahmanian. Maintaining views incrementally. *ACM SIGMOD Record*, 22(2):157–166, June 1993.

Jerry R Hobbs and Feng Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing*, 3(1):66–85, March 2004.

Jesper Hoeksema and Spyros Kotoulas. High-performance Distributed Stream Reasoning using S4. In *Proceedings of the First International Workshop on Ordering and Reasoning (OrdRing2011)*, 2011.

G. Hughes, H. Mills, D. De Roure, J. G. Frey, L. Moreau, m. c. schraefel, G. Smith, and E. Zaluska. The semantic smart laboratory: a system for supporting the chemical escientist. *Organic & Biomolecular Chemistry*, 2(22):3284–3293, 2004. doi: http://dx.doi.org/10.1039/B410075A.

Chun Jin, Jaime Carbonell, and Phil Hayes. ARGUS: Rete+ DBMS= Efficient Persistent Profile Matching on Large-Volume Data Streams. In *Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems (ISMIS2005)*, pages 156–170, 2005.

Alistair Miles and Sean Bechhofer. SKOS Simple Knowledge Organisation System Reference. W3C Recommendation REC-skos-reference-20090818, World Wide Web Consortium, 2009.

Simon Miles, Ewa Deelman, Paul Groth, Karan Vahi, Gaurang Mehta, and Luc Moreau. Connecting scientific data to scientific experiments with provenance. In *Proceedings of the third IEEE International Conference on e-Science and Grid Computing (e-Science'07)*, pages 179–186, Bangalore, India, November 2007. IEEE Computer Society. doi: http://dx.doi.org/10.1109/E-SCIENCE.2007.22.

Luc Moreau and Paolo Missier. The PROV Data Model and Abstract Syntax. Working Draft WD-prov-dm-20120202, World Wide Web Consortium, feb 2012.

Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and Laszlo Varga. The Provenance of Electronic Data. *Communications of the ACM*, 51(4):52–58, April 2008.

Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, June 2011. doi: http://dx.doi.org/10.1016/j.future.2010.07.005.

Jorge Perez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3):1–45, August 2009.

Satya Sahoo and Deborah McGuinness. The PROV Ontology: Model and Formal Semantics. Working Draft WD-prov-o-20111213, World Wide Web Consortium, dec 2011.

Douglas Terry, David Goldberg, David Nichols, Brian Oki, Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous queries over append-only databases. *ACM SIGMOD Record*, 21(2):321–330, June 1992.

O Walavalkar, A Joshi, Tim Finin, and Y Yesha. Streaming Knowledge Bases. In *Proceedings of the Fourth International Workshop on Scalable Semantic Web Knowledge Base Systems*, 2008.