DELIVERABLE SUBMISSION SHEET

To: Susan Fraser *(Project Officer)*

EUROPEAN COMMISSION
Directorate-General Information Society and Media
EUFO 1165A
L-2920 Luxembourg

From:

Project acronym:	PHEME	Project number:	611233
Project manager:	Kalina Bontcheva		
Project coordinator	The University of Sheffield (USFD)		

The following deliverable:

Deliverable title:	PHEME Integrated Veracity Framework - v 2.0
Deliverable number:	D6.1.3
Deliverable date:	31 January 2017
Partners responsible:	ATOS
Status:	☒ Public ☐ Restricted ☐ Confidential

is now complete. ☒ It is available for your inspection.
☒ Relevant descriptive documents are attached.

The deliverable is:

☐ a document
☐ a Website (URL: .....)
☐ software (.....)
☐ an event
☒ other (Prototype)

Sent to Project Officer:	Sent to functional mail box:	On date:
Susan.Fraser@ec.europa.eu	*CNECT-ICT*-611233*@ec.europa.eu*	31 January 2017

FP7-ICT Strategic Targeted Research Project PHEME (No. 611233)

Computing Veracity Across Media, Languages, and Social Networks

D6.1.3 PHEME Integrated Veracity Framework - v 2.0

Tomás Pariente Lobo, Mateusz Radzimski (ATOS) / Kalina Bontcheva, Leon Derczynski (USFD) / Georgi Georgiev, Ivelina Nicolova, Atanas Popov, Laura Tolosi (ONTO) / Thierry Declerck, Piroska Lendvai (USAAR) / Arno Scharl (MODUL) / Anna Kolliakou (KCL)

Abstract

FP7-ICT Strategic Targeted Research Project Pheme (No. 611233)
Deliverable D6.1.3 (WP 6)

This document and the associated software and running demos summarise the activities carried out related to the data and software integration of PHEME

Keyword list: Integration, PHEME architecture, acquisition framework

Nature: Prototype Dissemination: PU
Contractual date of delivery: 31/01/2017 Actual date of delivery: 31/01/2017
Reviewed By: Kalina Bontcheva
Web links:http://www.pheme.eu

PHEME Consortium

This document is part of the PHEME research project (No. 611233), partially funded by the FP7-ICT Programme.

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP, UK
Tel: +44 114 222 1930
Fax: +44 114 222 1810
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

Universitaet des Saarlandes
Language Technology Lab
Campus
D-66041 Saarbrücken
Germany
Contact person: Thierry Declerck
E-mail: declerck@dfki.de

MODUL University Vienna GMBH
Am Kahlenberg 1
1190 Wien
Austria
Contact person: Arno Scharl
E-mail: scharl@modul.ac.at

Ontotext AD
Polygraphia Office Center fl.4,
47A Tsarigradsko Shosse,
Sofia 1504, Bulgaria
Contact person: Georgi Georgiev
E-mail: georgiev@ontotext.com

ATOS Spain SA
Calle de Albarracin 25
28037 Madrid
Spain
Contact person: Tomás Pariente Lobo
E-mail: tomas.parientelobo@atos.net

King's College London
Strand
WC2R 2LS London
United Kingdom
Contact person: Robert Stewart
E-mail: robert.stewart@kcl.ac.uk

iHub Ltd.
NGONG, Road Bishop Magua Building
4th floor
00200 Nairobi
Kenya
Contact person: Rob Baker
E-mail: robbaker@ushahidi.com

SwissInfo.ch
Giacomettistrasse 3
3000 Bern
Switzerland
Contact person: Peter Schibli
E-mail: Peter.Schibli@swissinfo.ch

The University of Warwick
Kirby Corner Road
University House
CV4 8UW Coventry
United Kingdom
Contact person: Rob Procter
E-mail: Rob.Procter@warwick.ac.uk

Executive summary

This document describes and delivers the software associated with the final version of the PHEME integrated framework. The document focuses on reporting on the data and software integration aspects. The initial and intermediate integration versions of this document were explained in the PHEME deliverable D6.1.1 (restricted) and D6.1.2 (public), while the initial evaluation was presented in the deliverable D6.2.1 and the results of its final evaluation will be explained in D6.2.2.

One of the most challenging issues from the integration perspective is to enable different components to work with streaming data in real time. Therefore, the development of the software prototype has been conducted by making sure that the system and all its components are able to fulfil the project requirements. These requirements include processing social network data in a streaming fashion for real-time rumour classification and detection, dealing with cross-language and cross-media information, and providing timely results.

From the data perspective, the document reports on the pipelines and data flows as well as in the components and repositories that allow PHEME to acquire, pre-process, enrich, store and visualise social media data. The document describes how data from different social networks (mainly Twitter and Reddit) and different languages (English, German and Bulgarian) are processed and aggregated to take care of the challenging cross-media and cross-language aspects tackled in Pheme.

From the software integration perspective, once the decision was made in the second year of the project of using Apache Kafka as our main integration middleware, this year witnessed several integration iterations of different components developed in the technical work packages. In particular, the main effort has been directed towards the integration of the components to create processing pipelines for the journalist (WP8) and medical (WP7) use cases. The focus has been therefore on defining pipelines, integrating components, identify issues and bottlenecks, facilitating component owners an easy deployment of their components.

Contents

Index of Figures

1 Relevance to PHEME

1.1. Purpose of this document

This document is a software deliverable. The purpose of the document is to provide an overview of the architecture, data and software integration achievements of the PHEME Integrated Framework. This document covers the following aspects:

- PHEME architecture and integration approach.
- Data Integration prototype.
- Software integration prototype.
- Conclusion
- Annexes containing a brief overview of the components integrated.

1.2. Relevance to project objectives

Data and software integration are key aspects of any project, and more in a data-intensive project such as PHEME. Projects that do not pay attention to devise pragmatic and clear integration plans are likely to fail to deliver sound common results, even if the research and the separate components and tools developed within the project are of very good quality. In consequence, project partners have been aware since the inception of the project of the integration procedures and plans, encouraging all to participate actively in the integration discussions.

Therefore, this work package provided in previous documents the integration guidelines to the rest of the project partners to ensure that the data, components and tools provided by all partners can work nicely together in the different aspects they should interact. This deliverable reports on the final outcomes of the integration. It is a software deliverable, so besides the document it delivers software and a set of working prototypes which results can be accessed either via open software repositories (PHEME GitHub repository) or the integrated prototypes delivered by the use case work packages and dashboards (WP5, WP7 and WP8).

1.3. Relation to other work packages

As stated in the previous paragraphs, this deliverable describes the integration glue for all the technical work done in the project. Therefore, it has relation to all of the technical and use case work packages. It also has connections to the exploitation aspects, as the PHEME integrated framework is one of the main exploitation outputs of the project and its sustainability will depend heavily on how well the integration is done.

1.4. Structure of the document

The document is organised as follows:

- Section 1 gives a brief introduction, outlines the major purpose of the document and explains its relevance to PHEME.
- Section 2 provides a general background of the PHEME architecture and the approach followed for integration.
- Section 3 reports the data flow and repositories.
- Section 4 reports the software integration prototype.
- Section 5 concludes with consolidated findings so far and the next steps.

- References are listed in section 6.
- Annexes with extra information about specific components and repositories are given in section 7 and 8.

2 PHEME Veracity Framework Architecture and Integration Approach

2.1. Overview of the architecture

As originally devised, the PHEME Veracity Framework was designed to be able to integrate the veracity intelligence tools and services developed in the rest of the technical and use case work packages. The emphasis was on storing and processing large volumes of historical data, coupled with real-time analysis of incoming new content. A view representing the functional blocks in relation to the repositories is given in Figure 1.

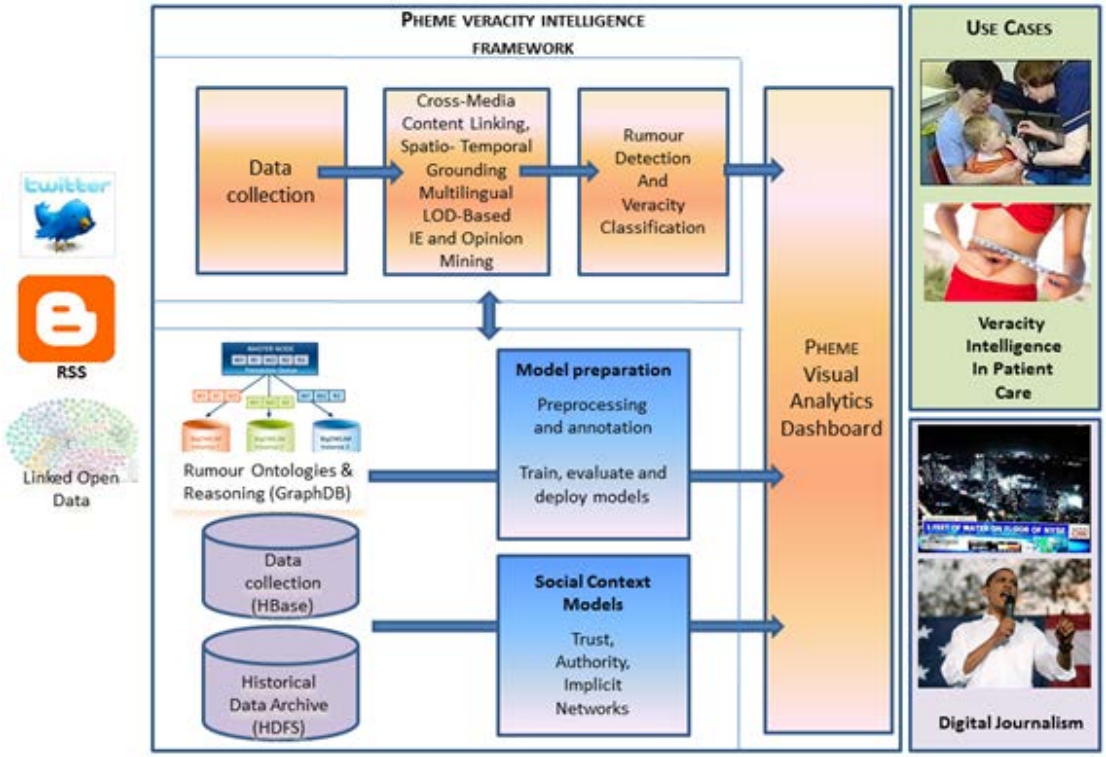

Figure 1. PHEME Veracity Framework Functional blocks perspective.

The figure above presents the main functional elements of the PHEME Veracity Framework:

- The data value chain for veracity checking: At the top of the figure, the different modules represent different steps in the collection, processing and analysis of data. It comprises components for data acquisition from Social Media resources, analytics (i.e. algorithms for cross-media and cross-language analysis, Information Extraction, etc.), and detection and classification of rumours. This is the core of the project and represents work done in most of the technical work packages of the project (WP2, WP3, WP4 and WP6). This core analytical layer was meant to be supported by stream and batch processing engines and service or pipelining frameworks,
- The model preparation: It comprises all the steps and processes aiming at annotate, train and evaluate the veracity models in PHEME. This layer is also core to the preparation of the data and models and it is mainly reflected in the

technical work done in WP2. Therefore it is not the main subject of integration work, but rather preparatory work for it.

- The usage layer: On the right hand side of the figure, it is the application provided to the end users. According to the different use cases defined in the project (WP7 and WP8), different applications can be built to provide adapted user interfaces according to the specific use case requirements. The Application Layer deals with the configuration of the user interface and the management of the user interaction to dispatch the events towards the internal system (Service Layer). The PHEME Visualization Dashboard (WP5) is in charge of displaying different visualization perspectives to end users to understand how rumours form, spread and die, among other interesting visualization paradigms. This dashboard, although generic in nature, has been used in the scope of WP7 and tailored for the medical domain, and as such integrated at the end of the medical pipeline. On the other hand, a Journalist Dashboard has been developed WP8 to take care of the specific visualization paradigms required by journalists. This dashboard is also integrated at the end of the Journalist pipelines.
- The data persistence layer: This layer is in charge of the mechanisms and models to store information. In PHEME, the main repository is the Knowledge repository, where the processed and data processed in the pipelines containing raw data and annotations added by the components in the pipelines is stored and further enriched. There are other repositories used for different purposes, such as raw data collection, annotated datasets, or even dashboard-specific repositories for fast visualization. These extra repositories are integral part of the architecture of the solution, but are not the main subject of integration. Therefore, this document only gives a brief mention to them, focusing on the knowledge repository and its interactions instead.

A high-level architectural diagram of the PHEME framework appears in Figure 2.

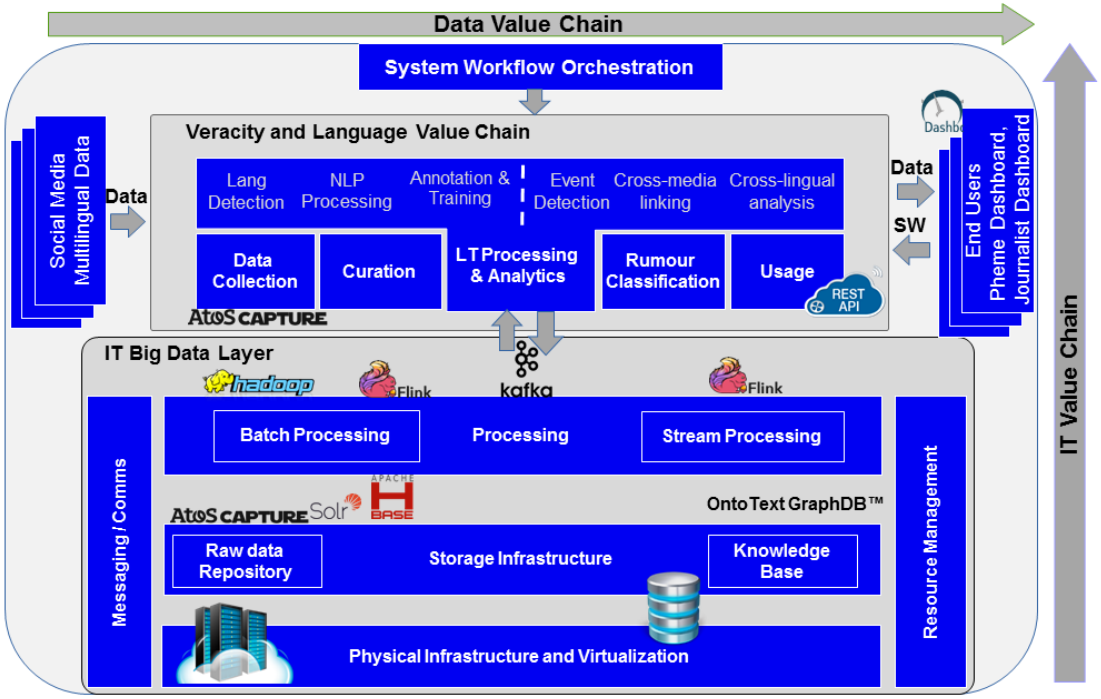

Figure 2. PHEME Veracity Framework Architecture.

Figure 2 shows the reference architecture for PHEME. This figure is following closely the blueprint provided for language technologies by the Coordination and Support Action MLi (MLi Deliverable D2.1). MLi proposes a Reference Architecture divided in two main axes:

- On the vertical axis "IT Value Chain" the value is created by providing more abstract access to the system functionalities, starting from low-level infrastructural services, through LT/MT/NLP domain-specific services until the high-level workflows and LT marketplace services for support of the business services and information and process integration within the Business Entity.
- On the horizontal "Language Value Chain" the value is created by combining LT/MT/NLP services and components into more complex and higher level features that fulfil requirements of the end user.

At first glance, it is clear that PHEME follows closely the MLi Hub reference architecture. This has been done intentionally as common partners of the two projects were interested in collaborating in these architectural aspects. In more detail, the PHEME implementation shows the following:

IT Infrastructure

PHEME implements the IT Infrastructure of the MLi Hub following the same building blocks. The Physical Infrastructure and Virtualization layer is not specified in the diagram further, but it uses virtualised servers and several installations for different use cases. Therefore, specific infrastructure providers can deploy PHEME in multiple environments. During the project there are two separate Physical Infrastructure instantiations, one in Sheffield that provides the main setup for hosting the integrated pipelines for both use cases (WP7 and WP8), and a second one in KCL mainly for data collection and analysis related to WP7 objectives.

One difference between the MLi Reference architecture and PHEME resides in the storage layer. In PHEME, the social media data and the semantic knowledge base are placed in the storage layer instead of the upper processing layer as proposed by MLi. This decision was taken on purpose to separate storage from the processing framework, although conceptually is very similar to the approach followed in the MLi Hub. PHEME proposes NoSQL databases (HBase[1]) and indexed repositories (Solr) for the raw data, while using a semantic database (GraphDB) as Knowledge Base. As a matter of example, the HBase and Solr repositories have been switched-off on the Sheffield infrastructure, as it was important not to duplicate storage but rather store the results of the processed data in the Knowledge repository. However, in the KCL infrastructure it was the other way around, as the objective was to use that data for internal annotation and studies. We also have an instance in Sheffield to gather raw data for annotation purpose that is only activated on demand. This gives an idea of the flexibility of the architecture proposed. The architecture also provides Apache Kafka[2], Apache Flink[3] and Apache Hadoop[4] as underlying streaming and batch processing engine and integration mechanisms. In the final implementation, the decision was to rely heavily

[1] http://hbase.apache.org/

[2] http://Kafka.apache.org/

[3] https://flink.apache.org/

[4] http://hadoop.apache.org/

on Apache Kafka as the middleware and streaming backbone for messaging and integration using its highly-scalable publish-subscription paradigm.

Language Value Chain

The Language Value Chain (LVC) in PHEME comprises all functional processes from the collection of social media data to its analysis and visualization. It is composed of processes and Language Technology components, services and sophisticated algorithms for event detection, cross-language and cross-media linking, along with training of veracity models for tumour detection and classification. The result of the analysis made in the LVC is used by the use cases (digital journalism and health) and visualised in dedicated dashboards. This can be done either by the visualization dashboard subscribing to the streaming data coming from the pipelines or by querying the data stored in the knowledge repository.

2.2. Software Integration approach

This section reports on the infrastructure used within the project life-span and the software integration approach used to pipeline the components developed within the project. Note that some components were not subject of integration, as they performed auxiliary or autonomous tasks such as training, data annotation, etc. These components have been developed in the scope of their respective work packages and have been reported in other deliverables.

PHEME is a research project that aims to deliver a set of tools as integrated as possible to push the state of the art in rumour and veracity detection. It was not the aim of the project to deliver a near-to-market toolset. However, in the light of what has been discussed so far and the attention raised in the market, within PHEME we paid huge attention to provide an integration framework capable of delivering scalable pipelines that could be potentially tailored for commercial scenarios. This means that the integration approach has to be flexible, scalable and simple.

2.2.1. PHEME IT infrastructure

As hinted in the previous section, PHEME implements the IT Infrastructure using a physical Infrastructure and a virtualization layer in a couple of installations for different use cases. This is an example on how specific infrastructure providers can deploy PHEME in multiple environments. Within the project there are two separate physical infrastructure instantiations:

- Main PHEME deployment infrastructure in Sheffield: This infrastructure is deployed on servers residing in Sheffield. It comprises four different servers, one acting as head node and two acting as working nodes. This infrastructure provides enough flexibility to be virtualised and hosts the main components and the underlying software infrastructure required for integration. This installation is used to showcase the project results and it is where the bulk of the work on integration has been done.
- Secondary PHEME deployment infrastructure in KCL: KCL deployed a similar infrastructure at the end of Y2 to test and showcase the results of their case study. It is completely separated from the previous infrastructure and it is mainly used for retrieving data for WP7 experiments.

These two instantiations, especially the one in Sheffield, are examples of how the PHEME Veracity Framework could be hosted and deployed. This big data infrastructure allows enough flexibility to achieve the necessary scalability. For instance, new nodes could be added and the workload of the different components and pipelines could be parallelised to improve the response time and avoid bottlenecks if necessary. This has been tested in some particular scenarios. This means that although the project has a limited budget in terms of infrastructure, nothing prevents achieving better throughput in a potential commercial scenario by horizontally scaling the infrastructure.

At the end of the project, a new instantiation has been produced in OntoText premises to be tailored for potential exploitation avenues.

2.2.2. Integration using Apache Kafka

PHEME components covered in this document need to be chained in a process in order to accomplish certain tasks, such as language detection, text pre-processing, processing in several languages, etc. These components are heterogeneous, developed by different partners often using different programming languages (mainly Java and Python) and sometimes even hosted remotely. These facts pose requirements to the integration approach followed in the project. This real-time processing integration follows the concept of pipelines. Pipelines allow the addition of multiple components in a process.

From the integration perspective, the main goal is to ensure that the whole system and all its components are able to fulfil the project requirements of processing social network data in a streaming fashion for real-time rumour classification and detection, cross-language and cross-media and providing timely results. Some of the components will perform other tasks by themselves, such as training the Machine Learning systems. In these cases, integration with other components is not required, and therefore not explained in detail in this document. Where needed, the integration approach should allow an easy and loosely coupled integration and communication for batch processing.

For programming language independence, messaging systems support multiple platforms and programming languages, which represent a clear solution to the integration problem. There are many popular messaging systems, such as ZeroMQ[5], RabbitMQ[6], Apache Flume[7] and Apache Kafka among others. However, as explained in PHEME deliverables D6.1.1 and D6.1.2, the Capture module provides the infrastructure for messaging and several other integration points that PHEME may take advantage of. Capture is built on top of a big data-enabled infrastructure that provides batch and streaming processing engines. In particular Capture uses Apache Kafka pub-sub mechanism for message exchange. This ability has been exploited during the second year of the project as baseline for data and component integration for the veracity framework.

Apache Kafka is a high-throughput publish/subscribe messaging system that provides very handy features for real-time processing systems and it is frequently used to enable

[5] http://zeromq.org/

[6] https://www.rabbitmq.com/

[7] https://flume.apache.org/

data pipelining. It is therefore an adequate approach to be used in the integration of the PHEME real-time components. Kafka is used in PHEME as the main integration middleware.

Kafka is designed to allow a single cluster to serve as the central data backbone for a large organisation. It can be elastically and transparently expanded without downtime. Data streams are partitioned and spread over a cluster of machines to allow data streams larger than the capability of any single machine and to allow clusters of coordinated consumers.

Apache Kafka differs from traditional messaging systems in that:

- It is designed as a very easy to scale out distributed system.
- It offers high throughput for both publishing and subscribing operations.
- It supports multi-subscribers and automatically balances the consumers during failure.
- It persists messages on disk and thus can be used for batched consumption such as ETL, in addition to real time applications.

Kafka is a general purpose publish-subscribe model messaging system, which offers strong durability, scalability and fault-tolerance support. For the pipelining approach of integration of PHEME components, Apache Kafka allows to pass streams (e.g. Twitter streams) from one component to the next. PHEME components and algorithms can therefore publish and subscribe to data streams to decouple the different processes, thus creating pipeline of loosely-coupled components. Figure 3 shows the publish-subscribe mechanism in Kafka.

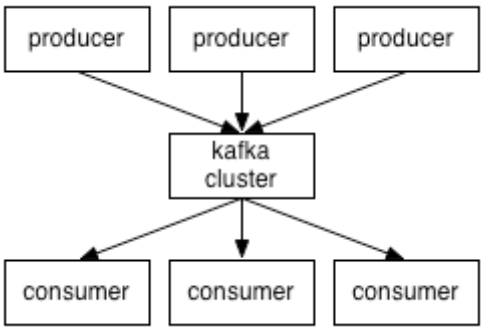

Figure 3. The publish-subscribe paradigm in Kafka

The Capture module provides an Apache Kafka installation that allows the subscription from PHEME components to the social networks data streams collected in real time by Capture. As an example, implementing a new Kafka Consumer for the streaming incoming flow is just a matter of creating a new Kafka Topic and subscribing to it, as shown in Figure 4 in an example taken from Capture.

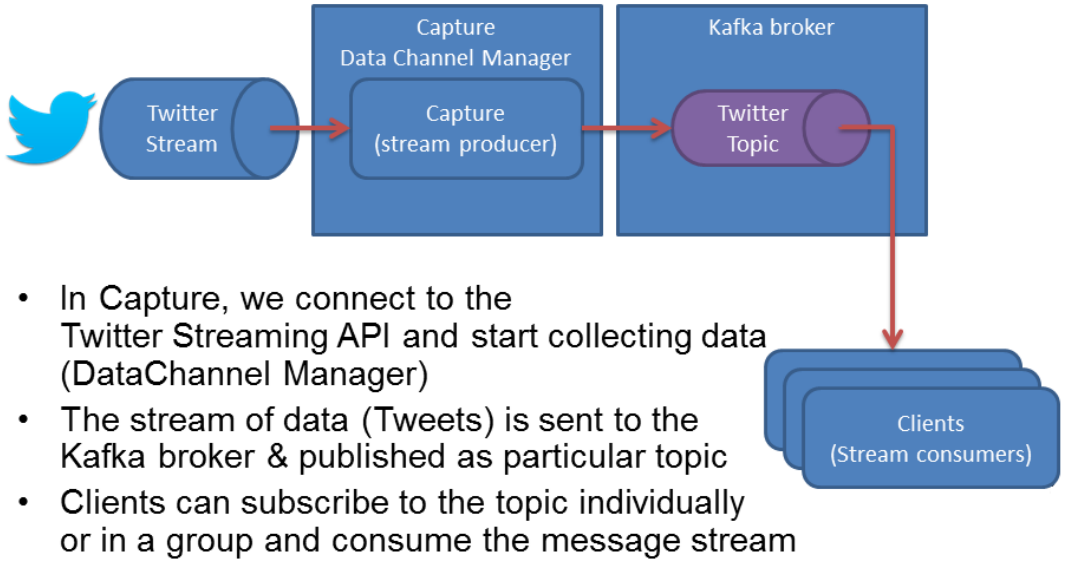

Figure 4. An example of a Kafka consumer reading from Twitter

2.2.3. Other possible integration mechanisms

It is also possible to use directly APIs from different modules to achieve a more tight integration between components. Examples of these are:

- Capture REST API in order to execute basic and/or faceted queries over the collected data. Components from PHEME will be able to fetch data using the available services. This is only possible if the data is stored in the raw data repository (switched-off in the Sheffield infrastructure, but available in the KCL infrastructure). Some of the methods are nevertheless always available, for instance to initiate new data collections, sample results, etc.
- Dashboard Document API: API to insert documents in the PHEME Dashboard as discussed in section 7 (7.3). In the integrated framework, the dashboard is subscribed to one of the pipelines (in this case, the medical pipeline) and the data streamed from there is stored and indexed in the dashboard repository for further visualization purposes.
- Using GraphDB queries (SPARQL): The social media data enriched with annotations and further enhanced in GraphDB, can be accessed via a SPARQL endpoint. In the case of the journalist pipeline, the Journalist Dashboard uses specific SPARQL queries to retrieve and show the data from GraphDB. More details about GraphDB can be found in section 7 (7.2).

2.2.4. Scalability and performance

The PHEME Integrated Framework is meant to scale. In order to ensure the scalability of the solution the project is on the one hand improving the performance and scalability of the different individual components as reported in previous sections. On the other hand, from the integration perspective, the project is following a global integration strategy to ensure the performance and scalability of the overall system. This global integration strategy presents project-wide approaches orthogonal to the individual scaling plans and common for most technical components. The focus is on integration aspects to provide an infrastructure to integrate components while enabling big data

scaling techniques, such as scaling up, scaling out, parallelisation, etc. This global integration strategy takes into account limits of individual components to align them into a common plan.

From the integration and scalability perspectives, pipelines should be able to increase the throughput and decrease the latency as much as possible. In order to do that, enabling parallelisation means processing of several inputs coming from components in a pipeline with other identical components that work in parallel. In an optimal scenario, it is simply adding more processing units for the same components that work slower compared to other components in a pipeline. More processing units can be provided to components by scaling horizontally or vertically. Scaling horizontally is achieved by adding more nodes (computers) to a system, while vertical scaling can be achieved by running the whole pipeline on a faster machine.

In the current version of the pipelines, and due to the time and hardware constraints of a research project such as PHEME, we couldn't afford the actual implementation of massive parallelization. Therefore, we detected some bottlenecks (i.e. components of the pipeline that cannot cope with certain peaks) that could have been easily overcome by adding extra cores, but we didn't do it as the current setting was enough to probe the project and use case needs. This was a good exercise to warn component-owners to fine-tune their components to achieve the desired throughput, although in some cases parallelization would be the way to go. Therefore this is a hint on how to proceed in case of going for a commercial approach, where the pipeline throughput should be maximised as much as possible.

Deliverable D6.2.2 (Evaluation report) will report on the scalability achieved by the different pipelines, bottlenecks and ways to improve the results of the project in future commercial settings.

3 Data flow and repositories

3.1. Introduction

Within WP6, task 6.2 is in charge of investigating how PHEME's diverse kinds of content and knowledge can be stored and accessed. From the perspective of the data usage within PHEME, the project differentiates between the social media data with very little pre-processing (Social Media content), and the more semantic and annotated content (Knowledge content). This section reports on the approach followed to integrate data for the project needs.

PHEME provides two main storage systems for social media and knowledge content in order to guarantee the data flow and data integration for the different services and components developed in PHEME.

The social media content data layer provides storage and access to the raw data acquired from social media. It is based on the Capture module developed in the scope of task 6.1 and it is based on state-of-the-art big data technology to guarantee the scalability of the solution. More details of this module can be found in section 7. As it has been already mentioned, PHEME proposes NoSQL databases (HBase) and indexed repositories (Solr) for the raw data. However these repositories are not being used on the Sheffield infrastructure, as it was important not to duplicate storage but rather store the results of the processed data in the Knowledge repository. Therefore, the focus on this section is more on the knowledge integration.

On the other hand, from the technical perspective the data flow in PHEME is achieved by using Apache Kafka as the main middleware to create pipelines. PHEME components are actively producing and consuming data from different Kafka topics using the pub-sub paradigm. In this section the concrete technical solution used in PHEME for data flow integration is presented.

3.2. Data flow in PHEME

3.2.1. Data flow

The integration of the PHEME infrastructure is dataflow-driven and stream-oriented in order to deal with large amounts of data in a timely fashion. The shift from service-oriented architectures towards the data-flow driven (e.g. message-oriented) architectures allows many obstacles to be overcome overcoming many obstacles of traditional approaches and focusing on relevant requirements. From the point of view of integration, the set of desired requirements towards the architecture are the following:

- High throughput
- Availability and resilience through infrastructure distribution and failover mechanisms (e.g. automatic replication, node failure handling)
- Shift from less efficient data polling (e.g. REQ/REP[8]) to more efficient data pull, with client-side subscription to relevant message channels

[8] https://en.wikipedia.org/wiki/Request%E2%80%93response

- Distributed deployment for consumer/producer or publisher/subscriber data processing paradigms.
- Message queues/logs for dealing with real-time, buffered data streams

PHEME partners are following a common approach for the format of the Kafka topics. The idea is to follow a pipeline approach enriching the original data from the social media feeds with extra annotations provided by the different components of the pipeline, thus enabling an incremental addition of annotations of the original Kafka message. The first component in the pipeline (Capture) publishes stream of tweets and Reddit posts in Kafka topics. Other component subscribe to those topics to do their job, produce new annotations (i.e. language of the tweet) and publish them in a new Kafka topic. This incremental approach allows for a very easy integration, as components just need to publish a new stream with the addition of the new annotation made. New components just need to subscribe to those queues to consume the streams.

This approach has been followed in the design of the main pipelines that need near-real time processing (see section 4. where the different pipelines developed in PHEME are explained in more detail). In those pipelines, the first component (Capture) sends raw messages to the first Kafka topic. The message passes through various components. After every stage of the processing the enriched message ends up in a new Kafka topic, ready to be consumed by the next component in the pipeline.

At the end of the pipelines the data flows to the two dashboards developed for the Medical and Journalist use cases. In the case of the Medical pipeline a final component passes the enriched data to the PHEME Dashboard developed in WP5. This dashboard uses an internal database to render the results in a web based interface. In the case of the Journalism dashboard, the web-based interface developed in WP8 queries both Capture and GraphDB APIs and SPARQL endpoint in order to render the enriched data gathered via the Journalist pipeline.

3.2.2. Kafka message format guidelines

The guidelines for this particular integration pipeline approach involving processing of social media data and the addition of annotations are the following:

- Messages are passed through Kafka topics in JSON format, encoded as simple text strings.
- At the beginning of the pipeline, each JSON object has a set of common baseline properties:
    - As Twitter, our main source, Reddit and other sources all follow the convention of placing the document text in a top-level key called "text".
    - The social media message identified should be a top-level "id_str" object, following Twitter convention.
    - To distinguish between different social media document types (e.g. Twitter, Reddit, etc.) each message has "pheme_source" property that denotes one of supported type. At this moment those are: “twitter”, “reddit”. Other types will be included later.
    - In case of Twitter, the raw twitter object is contained in “raw_json” property. Other SN may have similar raw data.

- Changes to the JSON object message are incremental, that is, each component may add new annotations (in a form of JSON properties), but should not remove previous information if it is not necessary. In other words, components re-publish the entire original JSON object and add additional keys.
- New annotations created by the pipeline components are added as top-level properties to the existing JSON object. Examples of some of these annotations are the following properties:
    - Event extraction: "event_cluster", long unsigned int
    - Language: "langid", an array consisting of two elements: [string, float], where string is a two-letter lowercase country code and float the confidence in [0..1]. This is not to be confused with the “lang” field based on the Twitter language identification that comes from the Twitter API and is contained in messages.
    - Tokens: "tokens", an array of tokens in a form of array: [int, int], giving token start/end offsets.
    - Named entities, spatio-temporal entities: "pheme_entities", an array of entities in a form of array: iint, int, string, corresponding to string start offset, end offset, and entity type; e.g. "person", "timex".
    - Support/Deny/Query: "pheme_sdq", array of [string, float] where string is support|query|deny and the float the confidence in [0..1].
    - Spatial grounding: "pheme_location", an object of {key: value}, where the source key is "latlong", "dbpedia", "nuts" or "geonames", and the value is either a string reference or an array [float, float] for latitude and longitude. All keys are optional.
- If there is a chance of colliding with a social media source's top level key, or a key name is non-descriptive, prefix it with "pheme_"

An example document from the first stage of the pipeline:

{
"**text**": "Obama is doing well",
"**pheme_source**": "twitter",
"**id_str**": "921349812834098012313256586018634",
"**raw_json**": { *raw JSON object, as received from the Twitter API* }
}

After passing through the pipeline components, the message might look like this:

{
"**text**": "Obama is doing well",
"**pheme_source**": "twitter",
"**id_str**": "921349812834098012313256586018634",
"**event_cluster**": 8721364871239,
"**langid**": ["en", 0.879],
"**tokens**": [
[0,5], [6,8],
[9,14],
[15,19]],
"**pheme_entities**": [
[6,8, "person"],
[9, 14, ”event”]],
"**pheme_sdq**": ["support", 0.566],
"**pheme_location**": {

"dbpedia": "United_States",
"latlong": [35.5, 98.0] },
"**raw_json**": { *raw JSON object, as received from the Twitter API* }
}

Note that other social media document types (coming from social media sources other than Twitter) will follow similar guidelines. In this sense, the core fields remain the same (such as "text", "pheme_source", "id_str"), while other specific fields can be added based on particular data type needs.

3.2.3. PHEME Knowledge Integration

The resource description framework (RDF) is a world wide web consortium standard (W3C standard) that defines a graph based models that allows integration of flexible and dynamic data from virtually limitless sources. In PHEME we have requirements to integrate various highly dynamic sources including but not limited to Twitter and Reddit. There is also need of real time indexing and support for standard query mechanisms and language such as the SPEQL language and SPARQL endpoint it became obvious that we needed a native triple store as our graph database component. Triplestores also allow the definition of an ontology that can serve as a contract or an API layer between different components in the system. Ontologies can also help provide inference mechanisms at scale. The semantic repository GraphDB was chosen as it is native triple store with scalable inference layer. It is capable for storing, querying and managing large amount of structured and semi-structured data and allows for interlinking it with LOD resources. In combination with its powerful inference mechanisms GraphDB provides excellent tooling capabilities for data management and exploration.

From a technical perspective, the Knowledge repository in PHEME is covered by GraphDB. A detailed description of GraphDB was given in T4.1. (Deliverable D4.1.1, "LOD-based Reasoning about Rumours: Initial Prototype"). The data model is presented in Deliverable D4.1.2. "LOD-based reasoning" and is schematically demonstrated on Fig. 1.5. In addition to the properties presented there, in the later phases of the project, new properties for tweet veracity and cross-media linking are introduced. All these could be observed in the sample RDF representations of the four main PHEME concept types - Tweet, PhemeMention, UserAccount, CrossMediaLinking available in section 8.

In D2.2 "Linguistic Pre-processing Tools and Ontological Models of Rumours and Phemes" we described in the PHEME ontology, that models veracity in social media. The ontology describes rumours, misinformation, disinformation and disputed claims, together with important network information such as authors (receivers and diffusers), events, and relations. Also, it easily integrates knowledge from LOD datasets, including DBPedia, Wikidata and GeoNames.

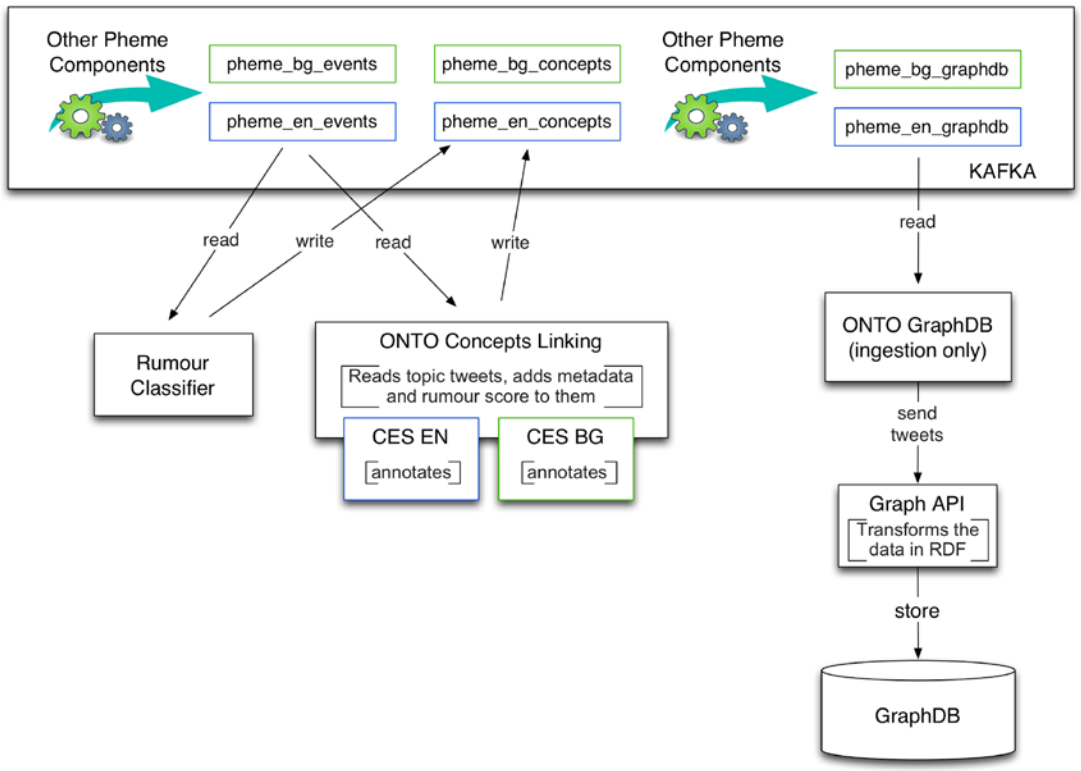

Figure 5. Components for tweets enrichment and for GraphDB integration

On Figure 5 are shown the Ontotext components for Concept Tagging, Rumour detection and the APIs responsible for transforming the JSON messages to RDF according to a predefined ontology as well as the API used for writing RDF representations of the messages into GraphDB. The process of storing RDF data in GraphDB is explained in detail in section 7.

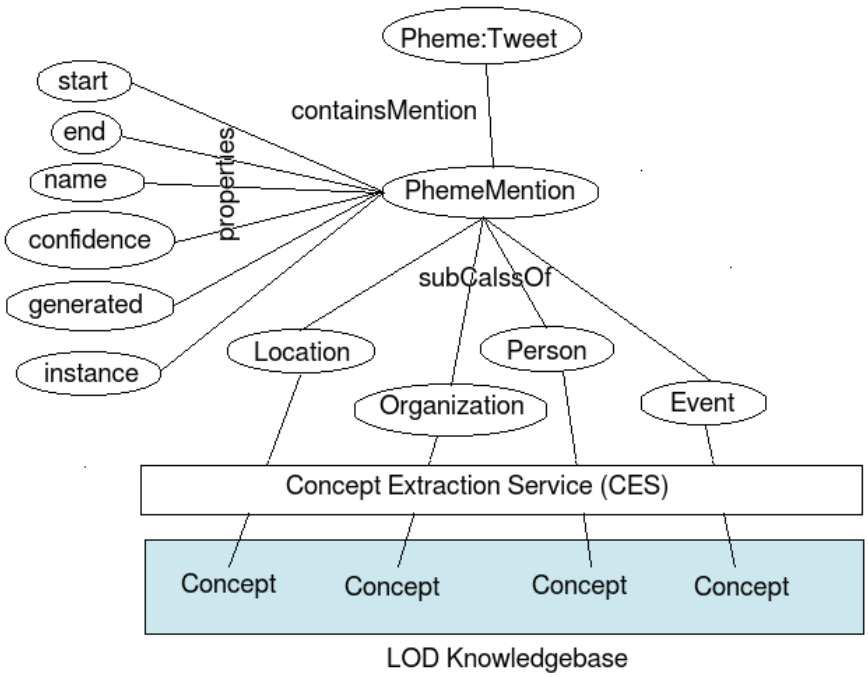

Figure 6. Integration of LOD data with PHEME ontology

The integration of LOD data with the PHEME ontology is implemented in the Ontotext's Concepts Linking (CL) component (shown on Figure 4) and they are interlinked as given on Figure 5. The CL component enriches the textual content of every tweet (or other information resource) represented in the PHEME ontology with LOD concepts, called PhemeMentions. It has several modules executed in a row on the input text. It starts with text preprocessing where basic linguistic analysis is performed: tokens and sentence boundaries are detected, part-of-speech labels are added to the words and shallow parsing is also applied. The next phase consists of logic that generates key-phrase candidates, assigns relevance scores to the candidates, and classifies them into positive or negative instances via a specialized processing resource for supervised classification. After that the text is enriched with content from gazetteers and entity candidates are matched. Given the named entity candidates discovered by the Linked Data Gazetteer as a prerequisite, the next phase conducts classification of each candidate into a positive or negative instance, based on the article context. As a result, the ambiguity associated with the existence of overlapping gazetteer lookups is eliminated – at most one named entity remains per document position, and the redundant named entity candidates are removed. At the disambiguation phase each entity is being linked with an existing concept in our predefined knowledge base. The knowledge base comprises 3 main linked data sources - DBpedia[9], WikiData[10] and Geonames[11]. Currently, the component supports disambiguation of named entities belonging to any of the following 4 classes: "Person", "Location", "Organization" and "Event". Each mention has properties: start (the position where it starts in the text, in characters), end (the position where it ends), name (the name of the mention), confidence (a probability estimate of the confidence of the tag, given by the Concept Extraction model), generated (a Boolean value that is false if the entity is found as a concept in the LOD knowledge repository and true, if it is only predicted by the model, but no corresponding concept was found in the LOD knowledgebase) and instance (the URI of the concept in the LOD knowledgebase, if not generated).

The PHEME knowledge graph contains the PHEME ontology and all concepts from DBpedia, GeoNames and WikiData with labels in English, Bulgarian and German. The PHEME graph is used to ground the small and rather simple text of a tweet, in the world knowledge and linked open data. For instance, if a tweet is automatically categorised (enriched) with Donald Trump and an event happening in London, GraphDB can infer new facts leveraging the rich hierarchy in GeoNames and the Person's class properties in DBpedia. The system can associate the tweet with the Republican Party since Mr Trump is linked to this concept via a DBpedia property, and to the location UK, since London is a sub region of the UK. Such inferences provide richer and deeper context around the tweets and facilitate information retrieval, query performances and navigating in the data.

[9] http://wiki.dbpedia.org/
[10] https://www.wikidata.org/
[11] http://www.geonames.org/

3.3. Cross-media and cross-language data integration approach

3.3.1. Cross-media

The PHEME pipeline schema integrated in WP6 is designed to support multiple social networks. In the previous version of this deliverable (D6.1.2), it was successfully tested with the Twitter data. The central idea for providing cross-media interoperability on the message level is the definition of common attributes of social media messages, and providing social network-specific extensions as complementary information. In this way, component operating on general social media data can still consume new format, while additional information (social media specific) can be taken into account when performing natural language classification tasks.

The top-level raw properties that are common across social media are the following:

- Text (or otherwise original content of the post)
- Date (Unix timestamp)
- Id (unique message identifier)
- User (author of the post)
- Source (type of message, a social network that is the source of this message)

3.3.1.1 Twitter data

PHEME has access for research purposes to historical Twitter data. However, due to the need to identify and follow in near-real time the events happening in Twitter, a number of continuous data collection tasks were established related to Twitter. To that extent, Atos provided an initial version of the Capture framework, developed in the scope of commercial proof of concepts with Atos customers. This initial version has been tailored and extended for further project needs. More detailed information about Capture can be found in section 7.

Twitter data can be searched in the whole Twitter. Capture had an initial data model enabling the acquisition of tweets using the Twitter Search[12] and Streaming[13] APIs. In particular, the Twitter model in Capture was also extended to be able to not only retrieve the pure json representation of a tweet, but also Twitter user profiles, list of followers and followees, etc. using the existing Twitter open APIs. The Capture model for Twitter data is characterized by the followings attributes:

- TweetID: The representation of the unique identifier for the Tweet.
- CreatedAt: UTC time when this Tweet was created.
- FavouriteCount: Indicates approximately how many times this Tweet has been "favorited" by Twitter users.
- HashTags: Tweet hash tags.
- InReplyToId: If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's ID. Needed for the conversation chains.
- Latitude: The latitude of the Tweet's location.

[12] https://dev.twitter.com/rest/public/search

[13] https://dev.twitter.com/streaming/public

- Longitude: The longitude of the Tweet's location.
- OriginalTweetId: Original identifier of the Tweet.
- Place: It is a specific, named location with corresponding geo coordinates.
- RetweetCount: Number of times this Tweet has been retweeted.
- Source: Utility used to post the Tweet, as an HTML-formatted string. Tweets from the Twitter website have a source value of web.
- Text: Tweet text
- UserDescription: The user-defined text describing their account.
- UserFollowers: The number of followers this account currently has. Under certain conditions of duress, this field will temporarily indicate "0."
- UserFollowees: The number of users this account is following. Under certain conditions of duress, this field will temporarily indicate "0.
- UserID: The representation of the unique identifier for the User.
- UserName: User name.
- UserScreenName: The screen name of the user.
- RawTweet: The whole content of the raw tweet

Capture implements a list of search queries into Data Channels. The Twitter data gathered by Capture for each of the data channel is streamed into the pipeline including the data channel ID in order to differentiate from which query the tweets come from and it is then analysed and enriched by the rest of the components of the pipelines.

3.3.1.2 Reddit data

Reddit is another social network that is oriented around the concepts of sub-reddits: focused groups typically centred on a common topic of interest. There are plenty of sub-reddits on almost all categories, such as education, entertainment, technology, news, and many others. Users on Reddit subscribe to their sub-reddit of choice and submit items (such as text postings, images, URLs) or comment on other user's items. In this sense Reddit resembles bulletin board systems or internet forums. The discussions on Reddit are flourishing and are source of many data on almost any conceivable topic.

For the sake of the PHEME project, we aim at two main categories:

- Journalism dashboard: News-related sub-reddits concerning Switzerland and world news (general topics)
- Medial dashboard: sub-reddits on mental health and depression.

Contrary to the Twitter social, the relevant data cannot be simply searched across the whole Reddit, but rather we aim at monitoring concrete chosen sub-reddits. This is done on a continuous fashion, by monitoring:

- New postings: beginning of new conversations.
- Comments on new and ongoing conversations.

The acquisition of Reddit posts is done through the Reddit API[14]. The structure of data resembles the tree-like structure of Reddit conversations.

[14] API Reference https://www.reddit.com/dev/api/

Each conversation is also rated based on the number or upvotes/downvotes and number of comments below the post. Individual comments can be also rated by other users and higher rated comments tend to appear closer to the top.

Another difference in comparison with Twitter is the definition of the single message. In case of Twitter a single message is a “Tweet”. In case of Reddit this can be either a whole conversation or a single post, depending on the concrete scenario. For the sake of PHEME processing pipeline we aimed at the latter: each message is an individual comment, with necessary metadata that locates such comment in a concrete conversation in a sub-reddit.

Capture has been also adapted to carry out the Reddit data collection.

3.3.2. Cross-language

As in the case for cross-media, the PHEME pipelines are helping to solve the cross-language issues. Basically the approach consists of getting the raw data from social networks and then splitting the language-dependant part of the process in different pipelines using different Kafka topics for each language. Components for English, German and Bulgarian processing are then doing their job before converging into the final stages of the pipeline.

Section 4 explains in more detail the concrete implementation of the pipelines done in the project.

4 Components and pipelines

This section reports on the current version of the integrated prototype released in the final stages of the project. It also gives an overview of the main processes, components and data delivered, focusing in the two main pipelines delivered for the PHEME use cases subject of integration work.

4.1. Overview of the components

Table 1 shows the components developed in different PHEME work packages that have been integrated within WP6. The table describes in brief the status of the each component, its availability and pointers either within this document or to specific deliverables where more information can be found.

Note that in the scope of the project there are two main pipelines: The so-called Medical and Journalist pipelines, integrating components useful for WP7 and WP8 respectively. Due to their business requirements, the Medical pipeline tracks tweets only in English and shows the final results in the PHEME dashboard developed in WP5, while the Journalist pipeline is multilingual and the results are shown in the Journalist dashboard developed in WP8. Therefore, some of the components listed in the table below are common to both cases, while others are related to just one of them. In particular we listed separately the components specific to the Medical pipeline and of the other languages other than English.

Table 1. PHEME components

Component	Responsible Partner	Brief Description	Extended Description

Capture	ATOS	Data collection tool developed in WP6
		Reads From: journalists and medical data channels (internal way of retrieving social media info in Capture)
		Output To: journalists_capture and med_capture kafka topics
		Annotations added: **"dc_id"**, "**text**", "**userID**", "**userScreenName**", "**createdAt**", "**id**", "**id_str**", "**lang**", "**source_type**", "**raw_json**"
		Usage: Capture is a tool that must be configured to get social media data using specific data channels (queries to social media). In Pheme it has been configured to get data for the medical and journalist use cases.
		E.g.
		{
		"dc_id": "xyz",
		"**text**": "Obama is doing well",
		"**userID**": "xxxxxx",
		"**userScreenName**": "xxx xxxx",
		"**createdAt**": "2016-04-18T11:14:24Z",
		"**id**": "92134981283409801231325658601863 4",
		"**id_str**": "92134981283409801231325658601863 4",
		"**lang**": "en",
		"**source_type**": "twitter",
		"**raw_json**": { *raw JSON object, as received from the Twitter API* },
		}
		Availability: The code needed for Pheme is open sourced and available in the Pheme GitHub https://github.com/project-pheme/capture
			Section 7.1.
			Used as the main ingestion point of Twitter and Reddit data both for the Journalist and Medical pipelines

Component	Responsible Partner	Brief Description	Extended Description
Language Detection	USFD	Algorithm to classify incoming messages according to their language (focus is on EN, DE, BG). It's critical to identify the language of content before sending it for linguistic processing. It's also worthwhile ignoring non-project-languages for event clustering, in order to reduce load and ease development. For this, we use state-of-the-art language ID to screen and split incoming content. Analysis earlier in the project covered the capability of language identification over social media content, where terseness of messages provides a real challenge to traditionally n-gram based approaches. Langid.py, with its *LD* feature selection, reduces the impact of noisy features and identifies language-discriminatory features that are stable across shifts of genre – perfectly suited to social media. It performed best overall in our trials. Twitter's recently-upgraded language ID is still not yet powerful enough for the task, working over a balance of existing non-social-media language ID solutions. Therefore, we wrap langid.py with Kafka input and split-output interfaces, taking in any social media content and providing three language-specific streams for English, German and Bulgarian. Langid.py is entirely Python, using an internally-encoded model, and so as platform-independent as that language. We have deployed this on Linux systems using the Kafka-python library. **Reads From**: pheme_capture OR med_capture kafka topics **Output To**: pheme_en pheme_de, pheme_bg OR med_en kafka topics **Annotations added**: “langid” **Usage**: The module is stored in the GATE Extras repository (gate-twitter/stream/). It reads from a given Kafka topic, consuming there and mapping Capture output into the Pheme/Twitter JSON format agreed and described in D6.2.1. Language specific messages are published on three Kafka topics: pheme_en, pheme_bg and pheme_de. The remainder of content is discarded.	Details in D2.2 First component for the Journalist and Medical pipelines

Component	Responsible Partner	Brief Description	Extended Description
		E.g.: `./langid_Kafka.py` **Availability:** Available in the Pheme GitHub at `https://github.com/project-pheme/langid`	
English Event Detection	USFD	Event detection aims to detect newsworthy events from a stream of tweets. It adds annotation to the tweets indicating the ID of the cluster of events the tweet is assigned to. Event detection aims to group together claims, or substories, that are made up of incoming messages. We have upgraded the technology here beyond that in D3.3.1. We now achieve this grouping through a cognitive-based machine learning approach. The participants, events, and places are extracted from incoming messages, as well as URLs, hashtags, and username mentions. Through a neural network trained to identify same-substory messages, this information is used to judge which of all the clusters currently being tracked the incoming message belongs to. If it belongs to none of them, it goes on its own to form a new cluster. The cluster id is available through the "event_cluster" field of the output Kafka The HDP algorithm requires one to specify the input Kafka stream to read from and the output Kafka stream to write. It uses a separate pre-processor for tweets from different languages and the same HDP algorithm works for tweets from different languages. It considers the "in_reply_to_status_id" and other "related document" fields of the incoming message to group together tweets belonging to the same conversational thread. The algorithm is developed using the Python programming language running in the Linux operating system environment. Python 3.4 or higher is required to run the algorithm. The algorithm is integrated in the Pheme project using the Apache Kafka framework. Further, access to NLTK and the Keras deep learning library are required, as well as GenSim for its word embedding loading functions.	Deliverable 3.3.1 For both Journalist and Medical pipelines (English)

Component	Responsible Partner	Brief Description	Extended Description
		The algorithm can be invoked by calling the Python program clusterer.py from the command line and providing the Kafka stream to read and write, as well as a machine learning model. This program comes with bundled help. Machine learning models can be trained using nn_cluster_keras.py over data generated using train.py. **Reads From**: pheme_en_entities OR med_advert kafka topics **Output To**: pheme_en_events OR med_events kafka topics **Annotations added**: **event_cluster** **Usage**: `/home/leon/pheme3/bin/python ./clusterer.py -m prce.negE-1.tsv.model.20160929132617.h5 -ki pheme_en_entities -e glove.twitter.27B.25d.txt -n 800 -ko pheme_en_events -f journalism.clusters -t 0.03 -T -D dc_id` **Availability:** Available in Pheme GitHub after publication through blind review	
English Entity Tagging	USFD	Named entity, spatial, and temporal extraction are all run as independent instances of the entity-recognition class, taken from the Python3 PIP package which was created earlier in Pheme. The NE and spatial variants use non-generalised Brown clusters from English tweets, with c=2500 and T=250 million; the spatial variant uses non-generalised Brown clusters induced over a blend of Reuters newswire and English tweets, with c=6000 and 64 million tokens of each genre (128 million total). This connects to the Kafka broker, consuming JSON documents one by one and re-publishing them with overlaid tokens, mapping tokens to pairs of character offsets, and also a pheme_entities field containing a list of entities, for each the entity types and bounding token offsets. **Reads From**: pheme_en OR med_en **Output To**: pheme_en_entities OR med_entities **Annotations added**: pheme_entities	

Component	Responsible Partner	Brief Description	Extended Description
		Usage: `entities_kafka.py <in topic> <out topic>` **E.g.** `/home/leon/pheme3/bin/python3 entities_kafka.py pheme_en pheme_en_entities` **Availability:** Core library available in the Pheme GitHub at `https://github.com/leondz/entity_recognition`	
English Concept Tagging	ONTO	This component is concerned with the identification of entities in text in English language. It is composed of several modules executed in row on the input text. It starts with text preprocessing where basic linguistic analysis is performed: tokens and sentence boundaries are detected, part-of-speech labels are added to the words and shallow parsing is also applied. The next phase consists of logic that generates keyphrase candidates, assigns relevance scores to the candidates, and classifies them into positive or negative instances via a specialized processing resource for supervised classification. After that the text is enriched with content from gazetteers and entity candidates are matched. Given the named entity candidates discovered by the Linked Data Gazetteer as a prerequisite, the next phase conducts classification of each candidate into a positive or negative instance, based on the article context. As a result, the ambiguity associated with the existence of overlapping gazetteer lookups is eliminated – at most one named entity remains per document position, and the redundant named entity candidates are removed. At the disambiguation phase each entity is being linked with an existing concept in our predefined knowledge base. The knowledge base comprises 3 main linked data sources - DBpedia (http://wiki.dbpedia.org/), Wikidata (https://www.wikidata.org/) and Geonames (http://www.geonames.org/). Currently, the component supports disambiguation of named entities belonging to either of the following 4 classes: “Person”, “Location”, “Organization” and “Event”.	

Component	Responsible Partner	Brief Description	Extended Description

		This component is run twice, once on the text of the message and second on the user.location tweet property which is also a free text added by the twitter user. The second time we aim to identify the location the user is suggesting.
		
		Reads From: pheme_en_events, pheme_de_events, pheme_bg_events OR med_events
		Output To: pheme_en_entities, pheme_de_entities, pheme_bg_entities OR med_entities
		Annotations added:
		The component appends annotations in `pheme_concepts` and `pheme_user_location` properties. The user location is extracted from the user.location tweet property which is a free text added by the twitter user.
		
		pheme_concepts": [
		{
		"name": "Trump",
		"type": "Keyphrase",
		"features": {
		"isGenerated": "true",
		"confidence": 0.5,
		"relevanceScore": 0.896551724137931,
		"inst": "http://data.ontotext.com/publishing/topic/Trump",
		"class": "http://ontology.ontotext.com/taxonomy/Keyphrase"
		},
		"startOffset": 13,
		"endOffset": 18
		},
		{
		"name": "Political",
		"type": "Keyphrase",
		"features": {
		"isGenerated": "false",
		"confidence": 0.8362389173713288,
		"relevanceScore": 0.689655172413793,	

Component	Responsible Partner	Brief Description	Extended Description
		"inst": "http://ontology.ontotext.com/resource/tsk85t0bq70g", "class": "http://ontology.ontotext.com/taxonomy/Thing" }, "startOffset": 31, "endOffset": 40 }] **Usage**: The component is packaged as a standalone web application. It registers itself to kafka stream and listens to messages in the configured topics. **Availability:** Available in the Pheme GitHub	
Cross-Media and Cross-Language linking	USAAR	Language independent. Linking to press-media from Twitter text. Extraction of an extended set of keywords/terms The core of the algorithm is language-independent string alignment between social media texts (in our first prototype, Twitter posts) and user-linked web documents (i.e., classical online media articles). The algorithm can additionally connect to media repositories for further cross-media access. In the currently implemented version, the algorithm accesses a third-party web service, EventRegistry (http://eventregistry.org/), a product of the FP7 Project "Xlike" (http://www.xlike.org/). Usability evaluation is planned in cooperation with WP8 (Digital Journalism Use Case) partners in Y3. **Reads From**: pheme_en_concepts, pheme_de_concepts **Output To**: pheme_en_media, pheme_de_ media **Annotations added**: Below we have a display of the currently added annotation, consisting in a listing of linked document with a summary, a list of keywords extracted from the original tweet and the tweet external text, and with the information if a contradiction has been found between the original tweet text and the data pointed at by the url.	The component "Cross-Media and Cross-Language Linking" has been developed in the context of Task 3.1, and has been described in detail in D3.1 "Cross-Media and Cross-Language

Component	Responsible Partner	Brief Description	Extended Description
		{'summary_of_linked_article': u'Doubling down on his goal to upend the established world order and remake it in his own image\u2014one that looks particularly like told two foreign newspapers over the weekend that he would consider lifting sanctions on Russia and believes that the NATO alliance, put in place to check Russian influence in the wake of WWII, is \u201cobsolete.\u201d', 'linked_url': u'http://www.politico.com/story/2017/01/russia-us-relations-trump-kremlin-233673?utm_source=dlvr.it&utm_medium=twitter', 'keyword_candidates_for_tweet': [u'Lavrov: We may improved relations U.S. Trump', u'relations U.S.'], 'cleaned_tweet_text': u'Donald Trump Just Offered Putin Exactly What He Wants URL', 'presence_of_contradiction': 'False', 'tweet_id': u'821334259545690116', 'presence_of_linked_article': 'True', 'linked_article_domain': 'www.politico.com', 'related_articles_headlines': [{'deu': []}, {u'eng': [u'Lavrov says Russia keen for dialogue with Trump', u"Donald Trump promises post-Brexit Britain a 'fair' trade deal", u"Trump is Putin's mouthpiece", u"Why Europe Is Worried About Donald Trump's Latest Remarks", u"Trump calls NATO 'obsolete,' pitches Russia nuke deal, hits Merkel over refugee crisis	The Japan Times"]}], 'linked_article_headlines': u'Lavrov: We may have improved relations with U.S. under Trump - POLITICO'} **Usage**:python xmedia.py **Availability:** Available in the Pheme GitHub	Linking Algorithm"
Rumour stance and veracity classification	USFD	Having enriched, event-clustered, language-specific text, the goal is to identify stance expressed in each tweet belonging to the pheme, as well as determine the overall veracity of the pheme/rumour. Tweet-level stance is classified as supporting, denying, querying or commenting (SDQC). The implementation details appear in D4.3.2. This component	D4.3.2.

Component	Responsible Partner	Brief Description	Extended Description
		depends on tokenisation, POS tagging, named entity recognition, and emotion detection. It also requires the GATE learning framework to perform the classification. **Reads From**: pheme_en_ concepts **Output To**: pheme_en_preprocessed **Annotations added**: The output parameter added is `pheme_sdqc` which contains both the predict label and also confidence, and `pheme_veracity`, containing the estimated veracity and confidence in that estimate, as a tuple **Usage**: Invoke a GATE application using the package in `gate-extras/pheme/gate-stream` **Availability:** Available in the Pheme GitHub	
Rumour Classification	ONTO	Topic-agnostic, language-agnostic model, based entirely on the textual features of the Twitter/Reddit message as wording, punctuation, length, capitalization etc. It does not make use of any metadata features thus works independently of the message source. The Rumour Classificator (RC) is a machine learning model (classification tree) that predicts the probability that a tweet is a rumour. As a result the service gives back the rumour coefficient of the specific tweet. The RC returns a probability, which quantifies the likelihood that the tweet is a rumour. Therefore, it is a real value between 0 and 1; a value of 1 means that the tweet is a rumour with certainty and a value of 0 means it is not a rumour, with certainty. **Reads From**: pheme_en_entities, pheme_en_events. **Output To**: pheme_en_events, pheme_de_events **Annotations added**: The component produces a “pheme_rumour_confidence” which is attached to the final document.	Deliverable D4.3.1 Section 7.1.2.

Component	Responsible Partner	Brief Description	Extended Description
		Usage: The component is packaged as a standalone web application. It registers itself to kafka stream and listens messages in the configured topics. **E.g**. { "pheme_rumour_confidence": 0.72 } **Availability:** Available in the Pheme GitHub at **. . .**	
Pheme Dashboard Ingestion	ATOS	This component is in charge of streaming the data from the medical pipeline to the Pheme Dashboard using the Pheme Dashboard and API developed in WP5 **Reads From**: med_processed **Output To**: Stream to be handled by the Pheme Dashboard (to be persisted in their database) **Annotations added**: None **Usage**: It is used in the scope of the medical pipeline as main output of the pipeline. The results are streamed to the Pheme Dashboard to be used and visualized. **Availability:** Available in the Pheme GitHub	
GraphDB Ingestion	ONTO	This component is responsible for collecting the final JSON enriched by the rest of the components during the previous steps and for converting it into RDF document following the ontology model (WP 2.2). Next the RDF document is imported into the Knowledge repository (GraphDB). **Reads From**: pheme_en_graphdb, pheme_de_graphdb, pheme_bg_graphdb **Output To**: GraphDB RDF database. No further output to kafka. **Annotations added**: None	

Component	Responsible Partner	Brief Description	Extended Description
		Usage: The component is packaged as a standalone web application. It registers itself to kafka stream and listens to messages in the configured topics. **Availability:** Available in the Pheme GitHub	
Medical pipeline specific			
Advert, Anti-stigma and Suicide Detection	KCL	In social media, large quantities of messages are sent that contain content advertising particular medications. Similarly, many messages is countering stigma about mental health and correcting stigmatizing statements. To classify tweets into whether they were advertising medication and counteracted stigma, two algorithms that make use of natural language processing (NLP) have been developed. Taking as input the text content of tweets, the advert algorithm classifies the message as to whether it advertises psychotropic medication or not. The anti-stigma algorithm, on the other hand, applies a set of linguistic rules to the messages and classifies them as either anti-stigma or not anti-stigma. Finally, the suicide application classifies tweets based on whether they are literal references to suicide or not. These three algorithms have been packaged together in this component. The algorithms have been developed in python and do not use any external NLP libraries. The application inputs the form of Kafka messages, processes their text and writes out as JSON to a Kafka output, the classification of advert, anti-stigma and suicide (whether or not the message is an advert, a literal reference to suicide or refers to anti-stigma). **Reads From**: med_ en **Output To**: med_advert **Annotations added**: anti-stigma (0=not anti-stigma, 1=anti-stigma), advert (0=not advertisement, 1=advertisement) and suicide (0=not relevant, 1=relevant)	

Component	Responsible Partner	Brief Description	Extended Description
		Usage: The module, test.py, is stored in the PHEME directory of the server Gatestorm1. It reads from a given Kafka topic, consuming there and mapping ATOS Capture output into the PHEME/Twitter JSON format. **E.g**. Testing components over annotated datasets, respectively, showed the applications to perform highly with 0.90 P and 0.92 R for the advert app and 0.98 P and 0.31 R for the anti-stigma app and 0.92 P and 0.58 R for the suicide app. **Availability:** Available in the Pheme GitHub	

Stance Classification	USFD	Having enriched, event-clustered, language-specific text, the goal is to identify candidate rumours. This is done by placing messages into the class of either support, deny, query or comment (SDQC). We implement this using a Gaussian process model, implemented in Gpy, and using the technology from D2.4 and WP4.[9]
		The tool has a number of Python dependencies which require Python version 2.7 and certain platform-specific tweaks; it is deployed on Linux.
		
		Reads From: pheme_en_ concepts
		Output To: pheme_en_preprocessed
		Annotations added: The output parameter added is `pheme_sdqc` which contains both the predict label and also confidence.
		Usage: Run:
		`./Kafka_consumer_misinformation.py TOPIC1 TOPIC2 TRAININGDATAPATH MODELPATH`
		to read content JSON from TOPIC1 and output resulting content JSON to TOPIC2. Recommended training data and models are
		`data/twoPHEME_datasets_as_events_041015.csv`
		and	

Component	Responsible Partner	Brief Description	Extended Description
		`results/store_models_test/BROWNGPjoinedfeaturesPooledLIN0.pick`. **Availability:** Available in the Pheme GitHub at `https://github.com/project-pheme/pheme-sdqc-gaussian`	
Bulgarian pipeline specific			
Bulgarian Entity Tagging	ONTO	This component is concerned with the identification of entities in text in Bulgarian language. It is composed of several modules executed in row on the input text. It starts with text preprocessing where basic linguistic analysis is performed: tokens and sentence boundaries are detected, part-of-speech labels are added to the words and shallow parsing is also applied. The next phase consists of logic that generates keyphrase candidates, assigns relevance scores to the candidates, and classifies them into positive or negative instances via a specialized processing resource for supervised classification. After that the text is enriched with content from gazetteers and entity candidates are matched. Given the named entity candidates discovered by the Linked Data Gazetteer as a prerequisite, in the next phase we conduct classification of each candidate into a positive or negative instance, based on the article context. As a result, the ambiguity associated with the existence of overlapping gazetteer lookups is eliminated – at most one named entity remains per document position, and the redundant named entity candidates are removed. Currently, the component supports disambiguation of named entities belonging to either of the following classes: “Person”, “Location”, “Organization” and “Event”. **Reads From**: pheme_bg_events **Output To**: pheme_bg_entities	

Component	Responsible Partner	Brief Description	Extended Description
		Availability: Available in the Pheme GitHub	

Bulgarian Concept Tagging	ONTO	Based on the entity tagging output and a type disambiguation procedure, this component links each entity to an existing concept in our predefined knowledge base. The knowledge base comprises 3 main linked data sources - DBpedia (http://wiki.dbpedia.org/), Wikidata (https://www.wikidata.org/) and Geonames (http://www.geonames.org/).
		
		Annotations added: The component appends annotations in `pheme_concepts` and `pheme_user_location` properties. The user location is extracted from the user.location tweet property which is a free text added by the twitter user.
		pheme_concepts": [
		{
		"name": "Тръмп",
		"type": "Keyphrase",
		"features": {
		"isGenerated": "true",
		"confidence": 0.5,
		"relevanceScore": 0.896551724137931,
		"inst": "http://data.ontotext.com/publishing/topic/Trump",
		"class": "http://ontology.ontotext.com/taxonomy/Keyphrase"
		},
		"startOffset": 13,
		"endOffset": 18
		},
		{
		"name": "Политик",
		"type": "Keyphrase",
		"features": {
		"isGenerated": "false",
		"confidence": 0.8362389173713288,
		"relevanceScore": 0.689655172413793,
		"inst": "http://ontology.ontotext.com/resource/tsk85t0bq70g",
		"class": "http://ontology.ontotext.com/taxonomy/Thing"
		},
		"startOffset": 31,	

Component	Responsible Partner	Brief Description	Extended Description
		"endOffset": 40 }] **Usage**: The component is packaged as a standalone web application. It registers itself to kafka stream and listens to messages in the configured topics. **Availability:** Available in the Pheme GitHub	
German pipeline specific The German pre-processing pipe-line is operational, and the steps needed to transform classical text processing software to be applicable to social media documents (in the current version, Twitter text) have been described in D2.2 "Linguistic Pre-processing Tools and Ontological Models of Rumours and Phemes". Recent work has been dedicated to integrating this component in the Pheme-Kafka pipeline, which is described in the current document. The German processing tools also benefit an updated version of the language independent Cross-Media and Cross-Language algorithm, via lemmatisation of both the social media and the online media texts, which results in improved string alignment. What in principle were separate components have been integrated in a single component for performance reasons (Event Detection, Entity Tagging and Concept Tagging). Therefore the German pipeline differs from the English pipeline. All functionalities (Entity Tagging, Event Detection and Concept Tagging are thus reading from “pheme_de” and outputs directly to pheme_de_entities, pheme:de_events and pheme_de_concepts			
German Event Detection (including German Entity Tagging Detection and German Concepts)	USAAR	Is now a functionality of one unique component (see introduction of the German pipe-line above). Below in the example we can see how we transform a verb to its lemma (“aufnahmen” in the text => “aufnehmen” in the annotation. **Reads From**: pheme_de **Output To**: pheme_de_entities and pheme_de_concepts **Annotations added**: The detected event (for now delivering only the verbal expression) and the related tokenization indices (relevant info in bold face):	Builds on and further develops D2.2 "Linguistic Pre-processing Tools and Ontological Models of

Component	Responsible Partner	Brief Description	Extended Description
		"lang": "de", "in_reply_to_status_id_str": null, "in_reply_to_screen_name": null, "coordinates": null, "contributors": null, "truncated": false}, "is_quote_status": false, "retweeted": false, "text": "RT @fabikde: Als die USA einmal 10.000e syrische Flüchchtlinge aufnahmen... Hab für SpOn über New Yorks \\"Little Syria\\" geschrieben. https://t\\u2026", "source_type": **"twitter", "pheme_events": [[63, 72, "aufnehmen"], [122, 133, "schreiben"]]**, "metadata": {"iso_language_code": "de", "result_type": "recent"}, "in_reply_to_user_id": null, "created_at": "2017-01-12T19:08:56Z",... **Usage**: python pipeline_kafka_de **Availability:** Available in the Pheme GitHub	Rumours and Phemes
German Entity Tagging (including German Event Detection and German Concepts)	USAAR	Is now a functionality of one unique component (see introduction of the German pipe-line above) **Reads From**: pheme_de **Output To**: pheme_de_entities and pheme_de_concepts **Annotations added**: In this case we add an annotation of the offsets in the text leading to the detection of entities and the entities themselves. The relevant annotation is in boldface. We notice that the entities have been mapped onto two different (compatible) classes, due to the type of query we address to DBpedia **Examples**: "in_reply_to_user_id_str": null, "id": 819622207538294784, **"lang": "de"**, "in_reply_to_status_id_str": null, "in_reply_to_screen_name": null, "geo": null, "coordinates": null, **"pheme_entities_texts": ["USA", "USA", "Syria", "Syria"], "pheme_entities": [[22, 25, "Country"], [22, 25, "Place"], [114, 119, "Country"], [114, 119, "Place"]]**, "favorited": false, "user_screen_name": "Monaflor"}'b'{"langid": ["de", 1.0],	Builds on and further develops D2.2 "Linguistic Pre-processing Tools and Ontological Models of Rumours and Phemes

Component	Responsible Partner	Brief Description	Extended Description
		Usage: python pipeline_kafka_de **Availability:** Available in the Pheme GitHub	
German Concept Tagging (including German Event Detection and German Entity tagging)	USAAR / ONTO	Is now a functionality of one unique component (see introduction of the German pipe-line above) **Reads From**: pheme_de **Output To**: pheme_de_entities and pheme_de_concepts **Annotations added**: The addition here is a the annotation of concepts that make their ways into GraphDB, following the specifications delivered by the partner ONTO **Examples:** "pheme_concepts": [{"name\" : "USA","startOffset": 22,"endOffset": 25,"type": "Country","features": "{"class": "http://schema.org/Country",}}, {"name" : "USA","startOffset": 22,"endOffset": 25,"type": "Place","features": "{"class": "http://schema.org/Place",}}, {"name" : "Syria","startOffset": 114,"endOffset": 119,"type": "Country","features": "{"class": "http://schema.org/Country",}}, {"name" : "Syria","startOffset": 114,"endOffset": 119,"type": "Place","features": "{"class": "http://schema.org/Place",}}"], "dc_id": "d6dacda2", "retweeted_status": {"place": null, "is_quote_status": false, "retweeted": false, "text": "Als die USA einmal 10.000e syrische Flu00fcchtlinge aufnahmen... Hab **Usage**: python pipeline_kafka_de **Availability:** Available in the Pheme GitHub	Builds on and further develops D2.2 "Linguistic Pre-processing Tools and Ontological Models of Rumours and Phemes

4.2. Integrated pipelines

4.2.1. Overview of the main processes and pipelines

This section provides a very high-level overview of some of the main processes envisaged for the project, with special attention to the integration between different components and the data repositories. The main pipelines implemented as part of the integration work in the project are shown in Table 2:

Table 2. PHEME main processes

Process / Pipeline	Responsible Partner	Brief Description
Journalists pipeline (stream)	SWI / ATOS	This pipeline is the main integrated process for the Journalist use case. The pipeline process streams of social media data with the aim of detecting events, veracity scores, etc. The Journalists pipeline is available for several languages and Twitter and Reddit feeds. In that sense, the Journalist pipeline makes use of the so-called German pipeline, which in reality is a subset of language-dependant components tailored for German processing, but it is not a separated pipeline itself. In this sense, the initial set of language-dependant components was tailored for English, but we don't refer to these components as English pipeline. It involves integration of functionality from many components and algorithms developed by partners. More info can be found in section 4.
Medical pipeline (stream)	KCL / ATOS	This pipeline is the main integrated process for the medical use case. The pipeline process streams of social media data with the aim of detecting events, veracity scores, anti-stigma, stance, etc. It involves integration of functionality from many components and algorithms developed by partners, including specific components for the medical use case such as anti-stigma or stance detection, among others. More info can be found in section 4.

Only processes that need a certain degree of integration are listed here, meaning that processes that involve several components but are tightly integrated have not been considered as part of the integration work carried out within WP6. For clarity sake, Table 3 below shows a non-exhaustive list of these kinds of processes:

Table 3. Example of other processes in PHEME

Process	Responsible Partner	Brief Description
Data Collection	ATOS	Based mainly in Capture. Available for Twitter and Reddit.
Data Annotation	UWAR	Annotation tool available for identification of rumours and non-rumours Crowdsourcing methodology established for the annotation of tweets within conversations, using the annotation scheme developed at PHEME More info can be found in Deliverable 2.4
Rumor / Non-Rumor Classification model	ONTO	This process classifies tweets as rumour or non-rumour, by computing the rumour probability according to a tree classifier trained on the Journalism data. More technical details on the classifier can be found in Deliverable 4.3.1
Support / Denying / Question Classification model	USFD	This is a process that involves the training of a model for SDQ classification More info in deliverable D4.3.2.
Veracity Classification model	USFD	This is a process that involves the training of a model for veracity classification More info in deliverable D4.3.2.

4.2.2. Journalists' pipeline

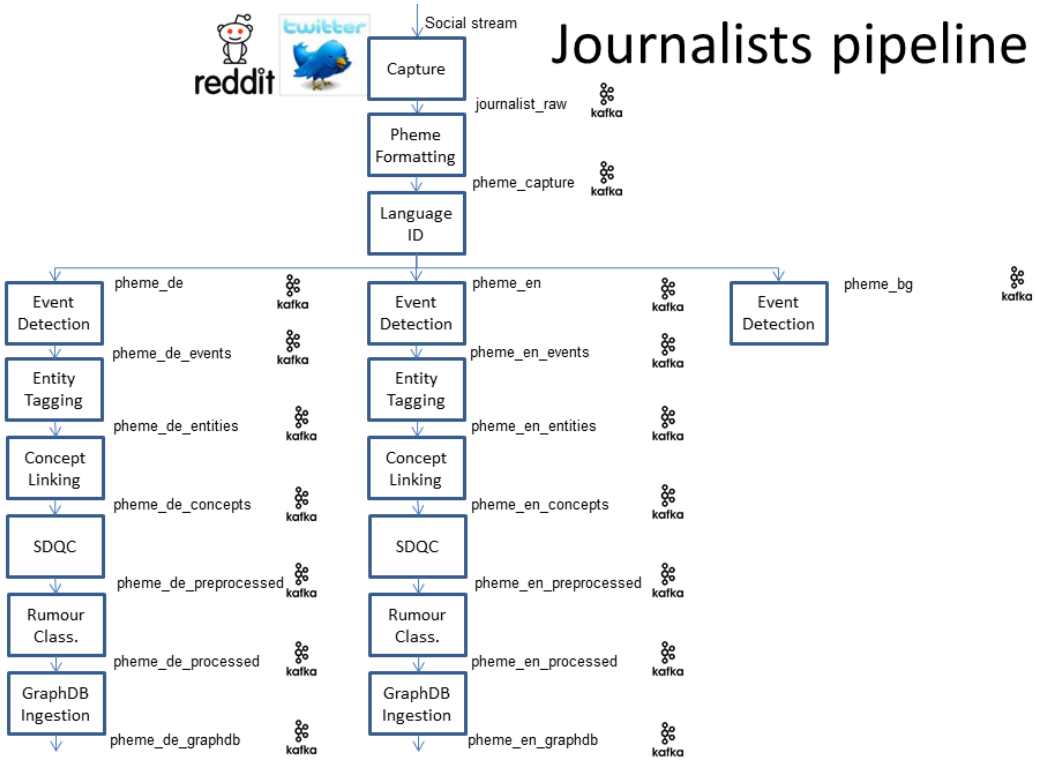

Figure 7. Journalist pipeline components

4.2.3. Medical pipeline

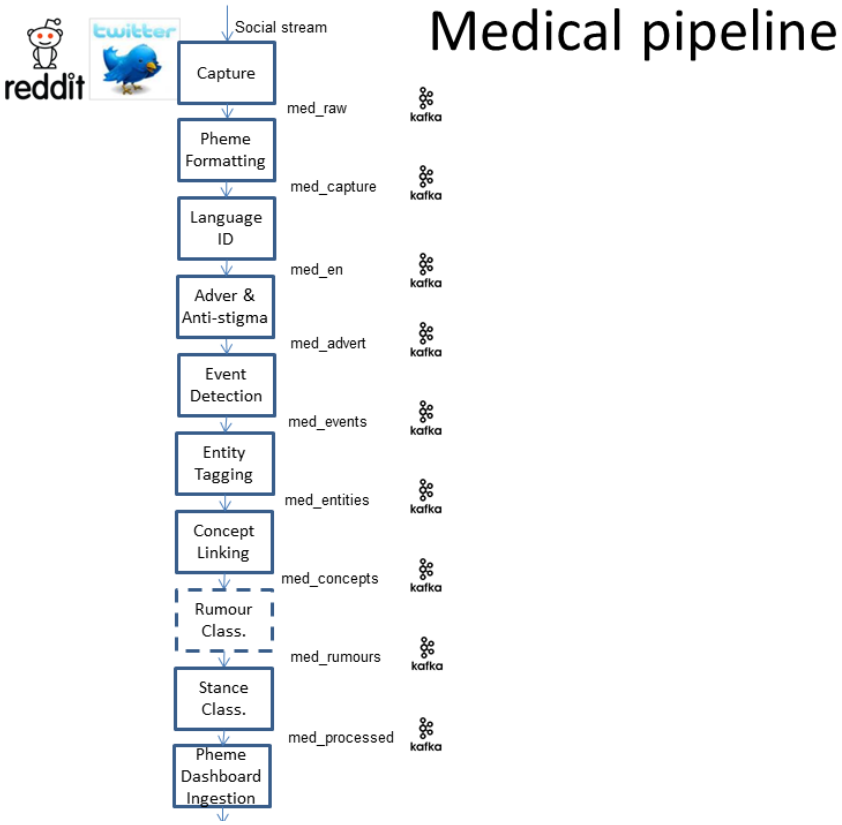

Figure 8. Medical pipeline components

4.3. Pipeline monitor

WP6 has developed a simple user interface to monitor the usage of the pipelines based on monitoring the elements of the pipelines and the Kafka topics from where they subscribe or publish data. This monitor allows in one single page to check whether the components of the pipeline are working fine as well as providing insights in the number of documents processed by component in several timeline-based widgets. The monitor offers also warning functionalities sending emails to component owners in case the components are down, offering thus a way of reducing the response time of correcting failures.

The monitor has been tailored for the final versions of the main PHEME pipelines, although it can be easily configured for future extensions in terms of Kafka topics and components.

Journalist pipeline monitoring:
http://pheme-capture.gate.ac.uk/pheme-stream-statistics/overview

Figure 9 below shows a partial view of the monitor for the journalist pipeline.

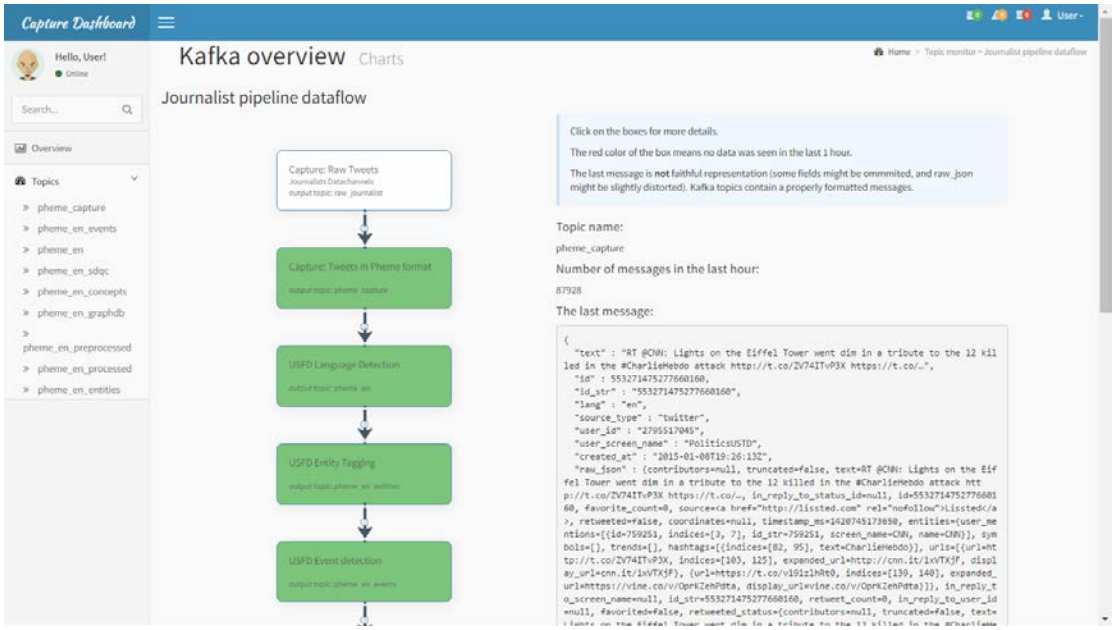

Figure 9 Journalist pipeline monitor overview

By clicking on the name of the Kafka topics on the left hand side of the monitor, several details about the number of messages passing through the component in the pipeline are shown, as exemplified in Figure 10:

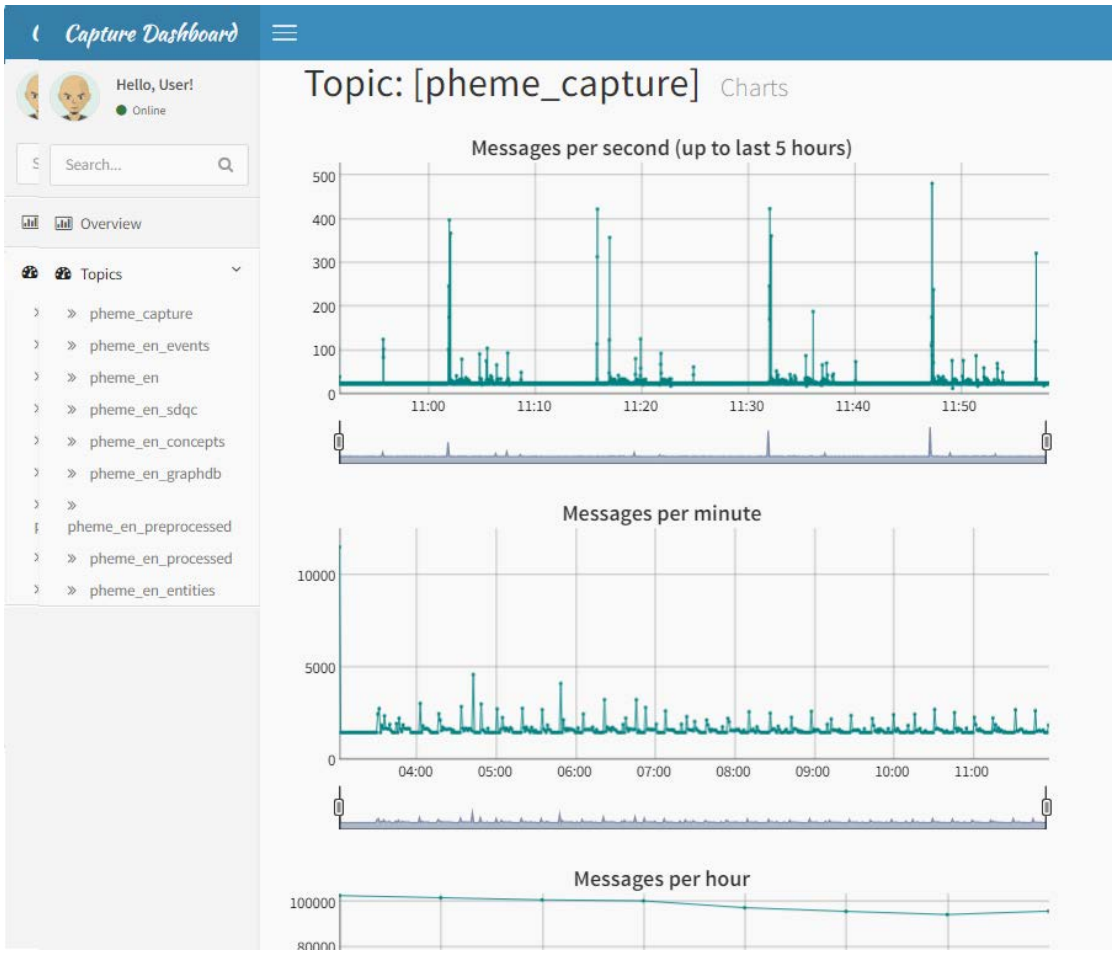

Figure 10. Journalist pipeline monitor - Component usage timeline statistics

A similar approach has been followed for the Medical pipeline monitoring: http://pheme-capture-2.gate.ac.uk/pheme-stream-statistics/overview

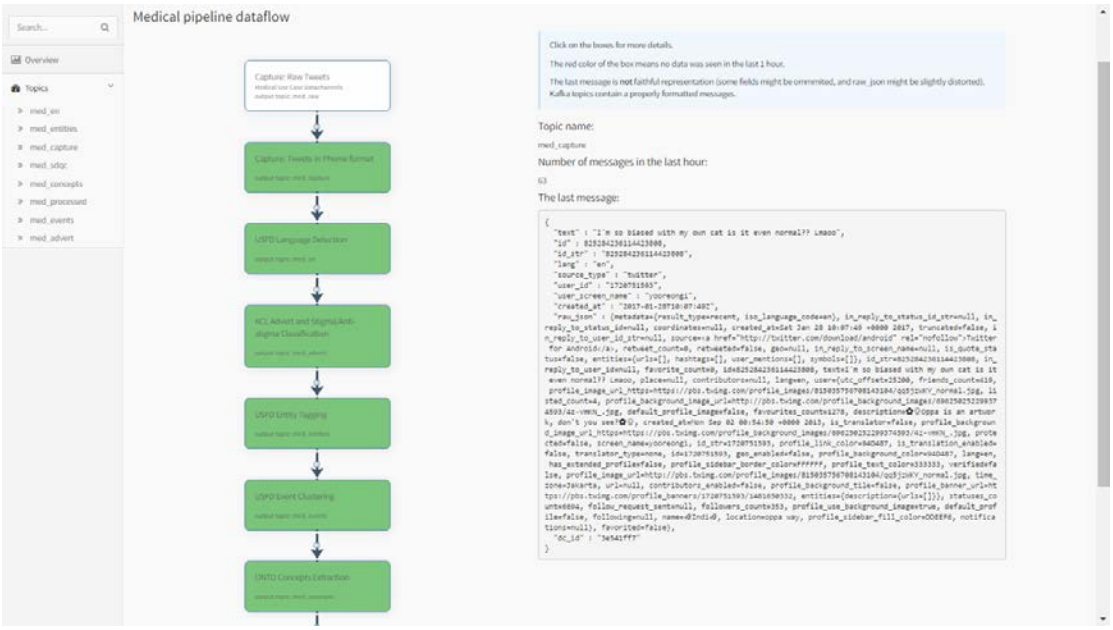

Figure 11. Medical pipeline monitor overview

The monitors are password-protected to ensure that only project partners will be granted access to the monitoring tool. http://pheme-capture.gate.ac.uk/pheme-stream-statistics/stats

5 Conclusion

The document is a working prototype and the associated description of the main integration work done in the scope of the PHEME project, especially regarding real-time stream pipelines tailored for the user cases combining work done by several partners. This document therefore describes and delivers the software associated with the final version of the Pheme integrated framework.

Special care has been taken to the integration of the different components that work with streaming data in real time. These near-real time requirements of the use cases have been taken into account in order to deliver the final software prototype. These requirements include processing social network data in a streaming fashion for real-time rumour classification and detection, dealing with cross-language and cross-media information, and providing timely results.

From the data perspective, the document reports on the way pipelines are connecting components using Apache Kafka as main public-subscription mechanism. This integration approach have been followed by all partners developing components and provided an easy, reliable and scalable framework for integration. Data flows have been implemented that allow PHEME to acquire, pre-process, enrich, store and visualise social media data for both use cases of the project. The document describes how data from different social networks (mainly Twitter and Reddit) and different languages (English, German and Bulgarian) are processed and aggregated to take care of the challenging cross-media and cross-language aspects tackled in Pheme.

This integration effort paid off and at the end of the project the running prototypes are fully functional and ready for evaluation.

6 Bibliography and references

Pariente Lobo T., Radzimski, M., 2015 MLi Deliverable D2.1- Early design of the reference architecture and the Hub. http://mli-project.eu/wp-content/uploads/2014/11/MLi-D2.1-Early-design-of-the-reference-architecture-and-the-Hub.pdf

7 Annex 1. Component descriptions

7.1. Data collection framework: Capture

7.1.1. Description

ATOS provides in the context of PHEME a data collection tool named "*Capture*". From the conceptual point of view, the main element in Capture is the ***Data Channel***. A data channel is the way users can group query results below a single umbrella. Data channels allow defining several queries or *Data sources* to social networks. All results can be stored and indexed associated to the channel and/or streamed to an Apache Kafka pipeline for further processing (pipeline approach).

The second element that enriches the definition of Data Channel is the ***Data Source***. A Data Source represents a specific web resource (i.e. Twitter, Reddit) and the definition of queries or filters that the system will perform. Conceptually, a Data Channel could be composed of 1 to N Data Sources, giving the possibility of making several queries to the same or different resources in the same data channel (i.e. two different queries to Twitter and Reddit grouped in the same data channel). However, in PHEME for simplicity we are creating different Data Channels for each social network we are using.

In the case of Twitter, Data Channels can be defined to target the Twitter Search[15] or Twitter Streaming[16] APIs. These APIs are open to developers that want to use Twitter data feeds, and subject to the Twitter terms and conditions stated in their specifications, The Capture Twitter acquisition module follows the guidelines stated on those terms and conditions.

In the case of Reddit, the convention followed in PHEME is that for each sub-Reddit a new Data Source with the required filters (keywords) will be generated. A Data Channel may contain several of these Data Sources therefore handling several sub-Reddits under the same Data Channel. The Capture Reddit acquisition module takes care of accessing Reddit using the Reddit API[17] respecting the limitations imposed by the social network.

7.1.2. Technical Perspective

Capture is a solution for social data collection that is based on big data technologies. It relies on the concept of gathering content using dedicated data channels. A data channel listens to data from different data sources (i.e. Twitter), meaning that data channels implement configurable user queries (based on keywords, hashtags, locations, etc.) to gather data (tweets) associated to the channel. The users are therefore able to set up several data channels for different purposes (i.e. to listen to specific events, or search for mentions of legal highs in Twitter), providing that the search limits provided by the APIs of the social networks (i.e. the limits of the public Twitter search and/or streaming APIs) are respected. Therefore, the data can be collected in very flexible ways. Capture

[15] https://dev.twitter.com/rest/public/search

[16] https://dev.twitter.com/streaming/overview

[17] https://www.reddit.com/dev/api/

also provides a search API to query for the data collected. It is also worth mentioning that Capture enables pipelining different analytical components both in batch or real time, giving an extra integration flavour to the data collection tool.

Figure 12 depicts the main building blocks of the Capture module.

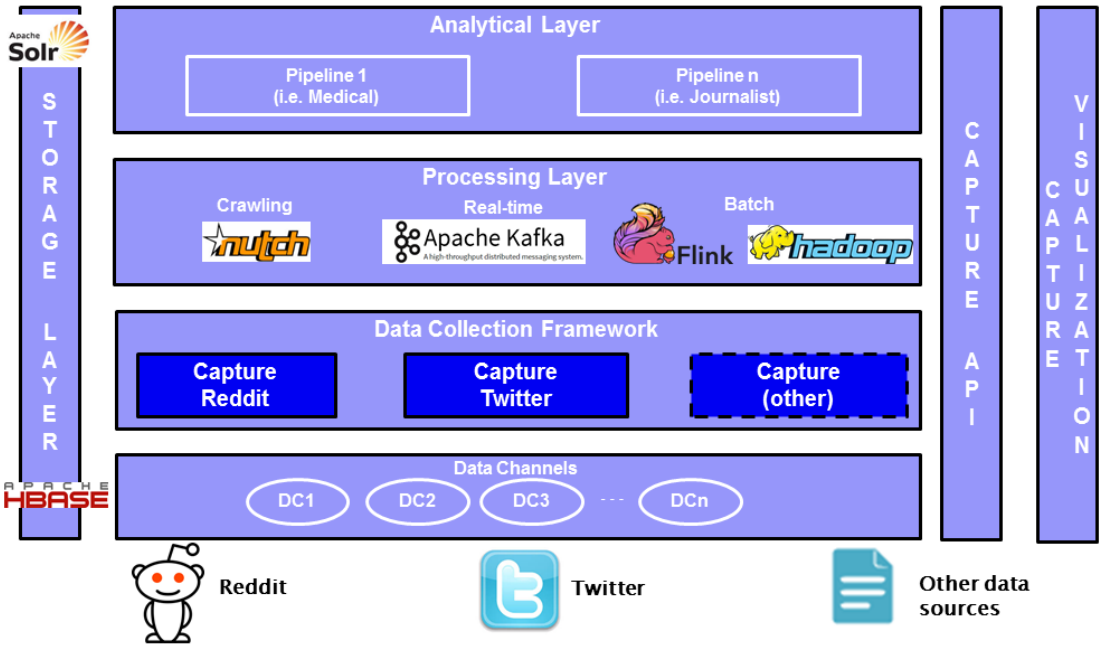

Figure 12. Detailed overview of the Data Collection framework (Capture)

As mentioned in this document, in the Sheffield infrastructure the persistence of the raw social media data is switched off and the social media data is only streamed and further enriched by other components before being stored in the Knowledge Repository (GraphDB). However, in the KCL infrastructure, the data is only stored in the raw data repository and not streamed to Kafka, as KCL needs that data for creating datasets and do further experiments. Capture is using Apache HBase as storage repository combined with a powerful indexing mechanism based on Apache Solr. This way, fast information retrieval for the collected data (i.e. tweets) is ensured. The interaction with the repository is done using a RESTful service layer (the Capture API). This API layer provides methods for the management of the data channels and search, The Capture REST API can be found in the PHEME GitHub repository.

7.1.3. Enhancements of Capture done in PHME

The Capture module has been developed based on previous work done by ATOS coming from previous EU projects and internal funding. The result of that was the launching of the Capturean[18] Social Media monitoring tool and service in the commercial offering of ATOS in the first quarter of 2015.

Capturean was developed completely separated from PHEME, giving as a result a big data framework for social media data analytics. At the beginning of the project PHEME partners' agreed to use the underlying Capturean big data infrastructure and integration framework based on Apache Kafka for integration purposes. Therefore, The Capture

[18] http://capturean.com/

module became the initial component in the pipelines for data acquisition, while the Apache Kafka framework became the main messaging and public-subscription mechanism to communicate different components that have to be chained together forming the project pipelines. To this extent, a specific deployment of the Capturean infrastructure was deployed in a private cloud setup to this effect at the University of Sheffield as it is explained in section 2.

The Capturean commercial tool was therefore developed completely separated from PHEME. However, following a logical transference process related to the exploitation of results, some of the enhancements done in PHEME to the Capture module eventually provided an upgrade of the Capturean commercial tool, as it happened the other way around as well.

The main work strictly related to PHEME done in the scope of the Capture component and the Capturean-based PHEME integration frameworks can be summarized as follows:

- User profiling: API to get the Twitter user profiles of some user handlers retrieved from the datasets acquired, including followers and followings for further
- New specific Twitter and Reddit data models for PHEME: The data models used by the components integrated in the pipelines are completely different to the ones used natively by Capture. The format is explained in section 3.
- PHEME Integration Framework: Although inherited from the underlying infrastructure of Capturean, the way components are integrated in PHEME is not the same as in Capturean, Capturean uses Apache Kafka for specific messaging purposes, but the integration of its internal component is managed in an Apache Flink cluster. In the case of PHEME, the integration framework is done basically decoupling the components using the potential of Apache Kafka as a public-subscription framework. PHEME components can be therefore completely decoupled from the rest and developed in different programming languages (i.e. Java or Python). All the work on the PHEME Integrated Framework is therefore PHEME-specific and did not revert to Capturean.
- Inclusion of Reddit data.
- Improvements on scalability using Kafka
- More clever usage of Twitter APIs (to cover WP8 requirements)
    - Search preview of tweets while setting the queries of the Data Channel
    - Ordering data from Twitter Search API
    - Getting a sample of the historical data to speed up the process of starting a new Data Channel
    - Improvements of the REST API, including sampling for the dashboard
    - API to generate Data Channels on demand (by the Journalist Dashboard)

7.1.4. Deployment Environment

Capture is deployable in any UNIX environment with the following characteristics:

- Tomcat 6-7 or Jetty 9
- Java 7
- Apache Kafka 0.8.2.2 or later,
- Apache HBase 1.1.1 or later,
- Hadoop 2,

- Apache Solr 5 or later
- Apache Flink 0.10 or later

The recommended minimal physical architecture is: 4 CPU cores, 8 GB RAM and 500 GB HD (extensible if the storage needs grows in time)

Capture expose two sets of user interfaces: (1) Capture REST API and (2) Capture Web GUI.

7.1.5. Invocation guidelines

The Data Collection tool Capture is deployed as RESTful services. The signature of the service is provided here.

Deliverable D6.1.2 showed examples on how to use the Capture REST API. The current version of the Capture REST API available for PHEME can be found in the PHEME GitHub.

7.2. Knowledge repository: GraphDB

7.2.1. Description

The Knowledge repository has been explained in detail in section 2. This section expands the GraphDB description with its deployment and invocation guidelines.

7.2.2. Deployment Environment

GraphDB is deployable at any Linux environment with:

- Tomcat 7
- Java 8

The minimal physical architecture required is: 8 cores, 20 GB RAM, 100 GB SSD.

GraphDB expose two sets of user interfaces: (1) Openrdf-WorkBench and (2) GraphDB Workbench and a SPARQL endpoint.

7.2.3. Invocation Guidelines

GraphDB is currently deployed at ONTO with Openrdf-WorkBench, GraphDB Workbench and SPARQL endpoint, and can be accessed at http://pheme-repo.ontotext.com.

The following ontology files are deployed:

- <file://PhemeOntology-v2.ttl>
- <file://Travis-Allen-Twitter-Ontology2.owl>
- <file://atc2en_de_es.ttl>

After opening the repository link one may select a predefined query under SPARQL/Query.

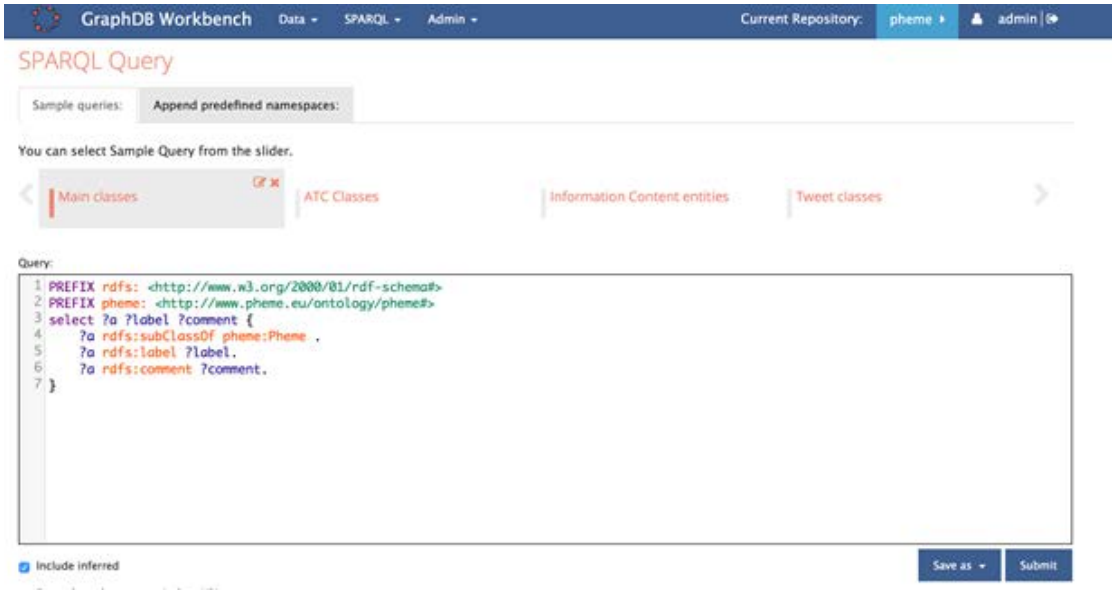

Figure 13. SPARQL query interface.

GraphDB reveals the results in a set of interfaces for RDF navigation and exploration, as shown in Figure 14 and Figure 15 below.

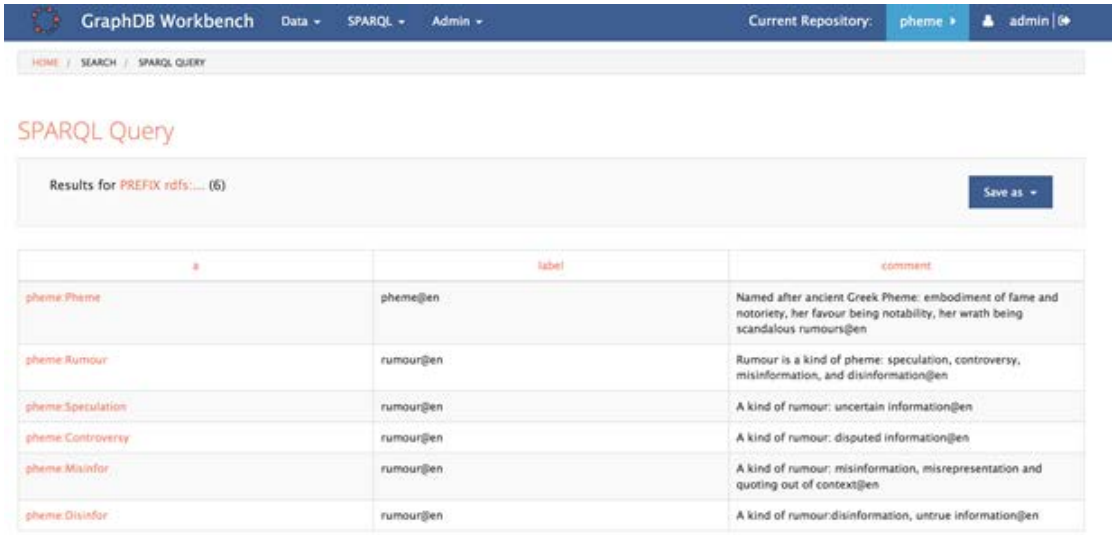

Figure 14. RDF query result set.

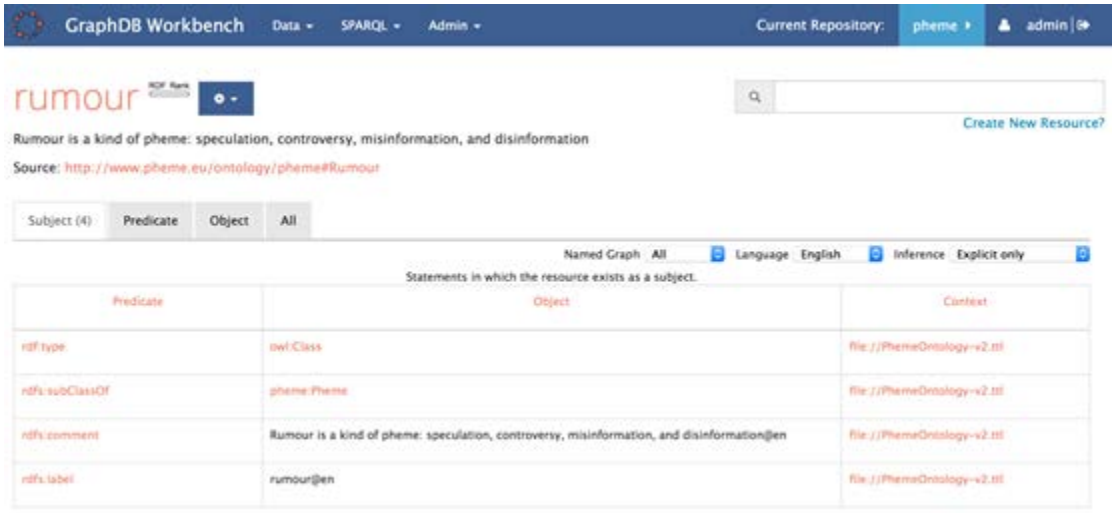

Figure 15. PHEME ontology navigation.

Sample queries that would help to explore further the currently deployed ontologies, including the rumour classification scheme and the tweeter metadata vocabulary, are shown below.

```
###Pheme classes
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pheme: <http://www.pheme.eu/ontology/pheme#>
select ?a ?label ?comment {
?a rdfs:subClassOf pheme:Pheme .
?a rdfs:label ?label.
?a rdfs:comment ?comment.
}

###ATC Classes
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select * {
?a rdfs:subClassOf atc:A01.
?a rdfs:label ?label.

}

###Information Content entities
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select * {
?a rdfs:subClassOf <http://purl.obolibrary.org/obo/IAO_0000030>.
?a rdfs:comment ?comment
}

###All Tweet subclasses
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select * {
```

```
?a rdfs:subClassOf
<http://www.semanticweb.org/travis/ontologies/2013/10/ontologicalengineeringtwitte
r#entity>.
OPTIONAL {
?a rdfs:comment ?comment.
}
OPTIONAL {
?a rdfs:label ?label.
}
}
```

Example request to the SPARQL endpoint of GraphDB that is equivalent to "Information Content entities" query is narrated below.

Sparql request:

```
http://pheme-
repo.ontotext.com/repositories/pheme?query=PREFIX+rdfs%3A+%3Chttp%3A%2F
%2Fwww.w3.org%2F2000%2F01%2Frdf-
schema%23%3E%0D%0Aselect+*+{%0D%0A++++%3Fa+rdfs%3AsubClassOf++
%3Chttp%3A%2F%2Fpurl.obolibrary.org%2Fobo%2FIAO_0000030%3E.%0D%0A
++++%3Fa+rdfs%3Acomment+%3Fcomment%0D%0A}&implicit=true
```

Response (partial):

```
<?xml version='1.0' encoding='UTF-8'?>
<sparql xmlns='http://www.w3.org/2005/sparql-results#'>
  <head>
        <variable name='a'/>
          <variable name='comment'/>
  </head>
  <results>
        <result>
        <binding name='comment'>
        <literal>an information content entity is an entity that is generically dependent
on some artifact and stands in relation of aboutness to some entity -IAO</literal>
        </binding>
        <binding name='a'>
<uri>http://purl.obolibrary.org/obo/IAO_0000030</uri>
        </binding>
        </result>
```

Below we show an example of Sparql query that illustrates the value of integrating LOD knowledge with the PHEME ontology. We want to retrieve Tweets that are related to the location New Jersey. Ideally, we expect to retrieve not only Tweets that mention New Jersey directly, but also Tweets that mention cities located in the New Jersey area.

```
PREFIX pubid: <http://ontology.ontotext.com/resource/>
PREFIX pheme: <http://www.pheme.eu/ontology/pheme#>
PREFIX pub: <http://ontology.ontotext.com/taxonomy/>
PREFIX geo-ont: <http://www.geonames.org/ontology#>
```

```
PREFIX dlpo: <http://www.semanticdesktop.org/ontologies/2011/10/05/dlpo>
select * {
        #Select New Jursey state
        ?parent pub:preferredLabel "New Jersey"@en.
        ?parent pub:exactMatch ?geoparent.
        ?geoparent a geo-ont:Feature.

        #Now select tweets mentioned with places inside New Jersey.
        ?t pheme:containsMention ?a.
        ?a pheme:inst ?pub.
        ?pub pub:preferredLabel ?pubLabel.
        ?pub pub:exactMatch ?geo.
        ?geo geo-ont:parentADM1 ?geoparent.
        ?t
<http://www.semanticdesktop.org/ontologies/2011/10/05/dlpo#textualContent> ?text.
} limit 100
```

Availability: ONTO granted access to GraphDB to project partners during the project duration in the PHEME installations. The access will continue at least for one year after the duration of the project. In case of commercialization, it will be done based on specific agreements between partners and customers.

7.3. Dashboard API

The PHEME visual dashboard reported in Deliverable D5.2.2 is based on a multiple coordinated view approach to explore the veracity intelligence extracted by the content analytics methods from WP2, WP3 and WP4. The dashboard provides modular and scalable update mechanisms, advanced query capabilities to reveal supportive and critical voices, and visual tools to reveal the context and diffusion of emerging rumours.

To foster collaboration and leverage synergies between PHEME and the ASAP FP7 project,[19] the data interchange between the dashboard and other PHEME work packages is based on an extended version of the webLyzard API (originally published as part of the ASAP Deliverable 6.2). From the API components, the following are relevant in the context of PHEME:

- Document API – ingests unstructured data from social media sources from T6.1 The main API objects are Documents, Sentences and Annotations – the latter are provided by WP2 and WP3, using PHEME-specific API extensions to support the required metadata elements.
- Search API – returns a set of query results in the form of unstructured text documents. The main object is Query.
- Embeddable Visualization API – represents a standardized way to integrate individual visualizations in third-party applications (as compared to using the

[19] ASAP = *Adaptive Scalable Analytics Platform* (www.asap-fp7.eu); The use of the webLyzard API not only avoids redundant efforts, but also supports joint exploitation efforts; specific opportunities arise from pursuing a *Visualization-as-a-Service* (VaaS) approach, where Pheme components such as the cluster map and the keyword graph can be offered as part of an integrated framework.

full dashboard). The main object is Visualization, which is typically rendered based on the results of a Query.

To model cross-referencing between documents and represent threaded dialogs, we have extended the document model to support document relations of various types, and have exposed the new structure via the Document API. With this addition, one can specify linkage information related to a document to be ingested into the PHEME dashboard by providing the relation type and the target URL through the JSON payload. On request, the API tries to match the provided target URL against existing documents contained in the PHEME metadata repository, resolving link targets to internal IDs where available. The document relation extension has been designed in a generic manner, enabling reasoning on the metadata document level – e.g., outgoing and incoming links for opinion mining, information diffusion paths via temporal linkage analysis, etc.

To facilitate API usage, the documentation[20] has been published using the *Swagger* toolkit.[21] This page represents a central hub to connect the documentation for all API endpoints in a standardized way, providing clear and reliable guidance for the technical partners working on WP2-WP6. In addition, code examples allow developers to quickly test and validate client code against the API endpoints.

Below is an example document representation (`pheme_document.json`) including the Pheme enrichments encoded as 'document features' prepared for ingestion in the PHEME Dashboard.

```
{
  "content" : "Cat called at in my workout gear walking to my car. Of
course. I ignore and they cat call even louder like I'm the one being an as
shole",
  "content_type" : "text/plain",
  "repository_id" : "pheme.weblyzard.com/api",
  "uri" : "https://twitter.com/KSchmerbach/status/807050199617118208",
  "title" : "Tweet by KSchmerbach",
  "features" : {
    "anti_stigma" : 0,
    "advert" : 0,
    "event_cluster" : [ 167956 ],
    "sdqc_type" : "support",
    "sdqc_confidence" : 0.5073472531352273,
    "veracity_score" : 0.0,
    "veracity_confidence" : 0.8022541519302714
--
  "meta_data" : {
    "published_date" : "2016-12-09T02:32:16Z",
    "user_screen_name" : "KSchmerbach",
    "tweet_id" : "807050199617118208",
    "twitter_lang_id" : "en"
  }
}
```

The CURL command to push this document to the webLyzard repository is as follows:

[20] api.weblyzard.com
[21] www.swagger.io

```
password = "superSecretPassword"
asap_token = curl -i -u api@pheme.weblyzard.com:$password
https://api.weblyzard.com/0.1/token
curl -H "Authorization: Bearer "$asap_token -H "Content-Type:
application/json" -X POST --data @pheme_document.json
https://api.weblyzard.com/0.1/documents/asap.weblyzard.com/api
```

8 Annex 2. RDF properties of important PHEME concepts

8.1. RDF description of the PhemeMention concept type

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://www.pheme.eu/ontology/pheme#phemeMention-
253b0906a771423a8a8b506ce853c043">
        <confidence xmlns="http://www.pheme.eu/ontology/pheme#"
                rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.0</confidence>
        <endnode xmlns="http://www.pheme.eu/ontology/pheme#">91</endnode>
        <generated xmlns="http://www.pheme.eu/ontology/pheme#">pheme</generated>
        <mentionType xmlns="http://www.pheme.eu/ontology/pheme#">LOCATION</mentionType>
        <name xmlns="http://www.pheme.eu/ontology/pheme#">Syria</name>
        <startnode xmlns="http://www.pheme.eu/ontology/pheme#">86</startnode>
        <rdf:type rdf:resource="http://www.pheme.eu/ontology/pheme#PhemeMention"/>
</rdf:Description>
</rdf:RDF>
```

8.2. RDF description of the UserAccount concept type

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://www.pheme.eu/ontology/pheme#user-1264291556">
        <description xmlns="http://rdfs.org/sioc/ns#">JJ. McCabe and Fallin'Angel..are signed to Flicknife
                Records UK.. Rock and Roll with a twist..</description>
        <twitterFollowersCount xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#long">3623</twitterFollowersCount>
        <twitterFriendsCount  xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#long">1279</twitterFriendsCount>
        <twitterStatusesCount xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#long">131709</twitterStatusesCount>
        <twitterUserAccountLocation xmlns="http://www.pheme.eu/ontology/pheme#">
                </twitterUserAccountLocation>
        <twitterUserVerified xmlns="http://www.pheme.eu/ontology/pheme#">false</twitterUserVerified>
        <rdf:type rdf:resource="http://rdfs.org/sioc/ns#UserAccount"/>
        <accountName xmlns="http://xmlns.com/foaf/0.1/">JJMCCABE2</accountName>
        <depiction xmlns="http://xmlns.com/foaf/0.1/">
        http://pbs.twimg.com/profile_images/800369865236353024/UhEGNqFM_normal.jpg</depiction>
        <name xmlns="http://xmlns.com/foaf/0.1/">JJMCCABE</name>
</rdf:Description>
</rdf:RDF>
```

8.3. RDF description of the Tweet concept type

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://www.pheme.eu/ontology/pheme#tweet-824542446562537472">
        <has_container xmlns="http://rdfs.org/sioc/ns#" rdf:resource=
                "http://www.pheme.eu/ontology/pheme#thread-824542446562537472"/>
        <has_creator xmlns="http://rdfs.org/sioc/ns#" rdf:resource=
                "http://www.pheme.eu/ontology/pheme#user-48662881"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/0ac2782215424cfb9477d4c784f44e74"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/0febafc697bc4ec9b9b33843133611e5"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/20a9086aec1e4f75a056c2c81a9b04bb"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/24ee9b20ae9b44e6a1027fd2d20cc371"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/3a11bde69258481bb7f145845a70cde9"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/6a0ff733a05d4c9cb203c10efd4b3e81"/>
        <containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
                "http://www.pheme.eu/resources/pheme/6c79be3c97224b4f8ed9f5c6ba3d6eb2"/>
```

<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/7c0005a91ed9404ca5aea24d5831ea01"/>
<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/7e09bba7ec7643c79f0be4418a5b3b1d"/>
<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/81bb1a466b0f48fcb8594c678abb20cb"/>
<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/9a3918155d9945c6833a6c68bb6f44af"/>
<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/bda8e5e112f34d9a93bc295b066fa3a4"/>
<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/c64aaf7b31914b7582f117ddd6186b71"/>
<containsMention xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/f06431dbf94a4212842bca8599255d91"/>
<createdAt xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
"http://www.w3.org/2001/XMLSchema#dateTime">2017-01-
26T11:00:13.000+02:00</createdAt>
<dataChannel xmlns="http://www.pheme.eu/ontology/pheme#">d6dacda2</dataChannel>
<eventClusterTitle xmlns="http://www.pheme.eu/ontology/pheme#">A1OQ: RT @ianbremmer: Trump
to
block visas to anyone from Iran Iraq Libya Somalia Sudan Syria Yemen 9/11 bombers from
Saudi Arabia 15 UAE 2 Egyp?</eventClusterTitle>
<eventId xmlns="http://www.pheme.eu/ontology/pheme#">2333059</eventId>
<hasCrossMediaLink xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#crossmedialink-824542446562537472"/>
<hasEvidentiality xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#no-evidence"/>
<isRumour xmlns="http://www.pheme.eu/ontology/pheme#">false</isRumour>
<langid xmlns="http://www.pheme.eu/ontology/pheme#">en</langid>
<langidProbability xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
"http://www.w3.org/2001/XMLSchema#double">0.9999999996409741</langidProbability>
<phemeEntity xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#phemeMention-
01c537cdf41143d3897aaacb21aaaebc"/>
<phemeEntity xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#phemeMention-
0420d05b47dd4e16ae05c09a717fa80c"/>
<phemeEntity xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#phemeMention-
37a46a55fe7743239939a06b60f8039e"/>
<phemeEntity xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#phemeMention-
5d6553d6e5ee4c5884a0365f83a6b2f0"/>
<phemeEntity xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/ontology/pheme#phemeMention-
c0ef70aa1e0e436a8cce43bf3f423d60"/>
<retweetOfId xmlns="http://www.pheme.eu/ontology/pheme#">824344479578681344</retweetOfId>
<rumourCoefficient xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
"http://www.w3.org/2001/XMLSchema#double">2.79787565845573E-
75</rumourCoefficient>
<sdq xmlns="http://www.pheme.eu/ontology/pheme#">support</sdq>
<sdqProbability xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
"http://www.w3.org/2001/XMLSchema#double">0.5203752982655573</sdqProbability>
<sourceReputation xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
"http://www.w3.org/2001/XMLSchema#double">0.2212410160548547</sourceReputation>
<sourceType xmlns="http://www.pheme.eu/ontology/pheme#">twitter</sourceType>
<userLocations xmlns="http://www.pheme.eu/ontology/pheme#" rdf:resource=
"http://www.pheme.eu/resources/pheme/cac04244c529431a8266c5554128b5f1"/>
<veracity xmlns="http://www.pheme.eu/ontology/pheme#">true</veracity>
<veracityScore xmlns="http://www.pheme.eu/ontology/pheme#" rdf:datatype=
"http://www.w3.org/2001/XMLSchema#double">1.0</veracityScore>
<version xmlns="http://www.pheme.eu/ontology/pheme#">v8</version>
<textualContent xmlns="http://www.semanticdesktop.org/ontologies/2011/10/05/dlpo#">RT
@ianbremmer:
Trump to block visas to anyone from
Iran

```
Iraq
Libya
Somalia
Sudan
Syria
Yemen

9/11 bombers from
Saudi Arabia 15
UAE 2
Egyp?</textualContent>
<rdf:type rdf:resource="http://www.pheme.eu/ontology/pheme#SourceTweet"/>
<rdf:type rdf:resource="http://www.pheme.eu/ontology/pheme#Tweet"/>
</rdf:Description>
</rdf:RDF>
```


8.4. RDF description of the CrossMediaLink concept type


```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://www.pheme.eu/ontology/pheme#crossmedialink-824542446562537472">
<crossCleanedTweetText xmlns="http://www.pheme.eu/ontology/pheme#" xml:lang="en">Donald
Trump Just
Offered Putin Exactly What He Wants URL</crossCleanedTweetText>
<crossKeywordCandidate xmlns="http://www.pheme.eu/ontology/pheme#" xml:lang="en">Lavrov:
We may
improved relations U.S. Trump</crossKeywordCandidate>
<crossKeywordCandidate xmlns="http://www.pheme.eu/ontology/pheme#" xml:lang="en">relations
U.S.</crossKeywordCandidate>
<crossLinkedArticleDomain xmlns=
"http://www.pheme.eu/ontology/pheme#">www.politico.com</crossLinkedArticleDomain>
<crossLinkedArticleHeadline xmlns="http://www.pheme.eu/ontology/pheme#"
xml:lang="en">Lavrov: We
may have improved relations with U.S. under Trump -
POLITICO</crossLinkedArticleHeadline>
<crossLinkedUrl xmlns=
"http://www.pheme.eu/ontology/pheme#">http://www.politico.com/story/2017/01/russia-us-
relations-
trump-kremlin-233673?utm_source=dlvr.it&amp;utm_medium=twitter</crossLinkedUrl>
<crossPresenceOfContradiction xmlns=
"http://www.pheme.eu/ontology/pheme#">False</crossPresenceOfContradiction>
<crossPresenceOfLinkedArticle xmlns=
"http://www.pheme.eu/ontology/pheme#">True</crossPresenceOfLinkedArticle>
<crossRelatedArticleHeadlines xmlns="http://www.pheme.eu/ontology/pheme#"
xml:lang="en">Donald
Trump promises post-Brexit Britain a 'fair' trade deal</crossRelatedArticleHeadlines>
<crossRelatedArticleHeadlines xmlns="http://www.pheme.eu/ontology/pheme#"
xml:lang="en">Lavrov says
Russia keen for dialogue with Trump</crossRelatedArticleHeadlines>
<crossRelatedArticleHeadlines xmlns="http://www.pheme.eu/ontology/pheme#"
xml:lang="en">Trump calls
NATO 'obsolete,' pitches Russia nuke deal, hits Merkel over refugee crisis | The Japan
Times</crossRelatedArticleHeadlines>
<crossRelatedArticleHeadlines xmlns="http://www.pheme.eu/ontology/pheme#"
xml:lang="en">Trump is
Putin's mouthpiece</crossRelatedArticleHeadlines>
<crossRelatedArticleHeadlines xmlns="http://www.pheme.eu/ontology/pheme#" xml:lang="en">Why
Europe
Is Worried About Donald Trump's Latest Remarks</crossRelatedArticleHeadlines>
<crossSummaryOfLinkedArticle xmlns="http://www.pheme.eu/ontology/pheme#"
xml:lang="en">Doubling
down on his goal to upend the established world order and remake it in his own image—one that
```

looks particularly like told two foreign newspapers over the weekend that he would consider lifting
sanctions on Russia and believes that the NATO alliance, put in place to check Russian influence in
the wake of WWII, is obsolete.</crossSummaryOfLinkedArticle>
<**crossTweetId** xmlns="http://www.pheme.eu/ontology/pheme#">821334259545690116</crossTweetId>
<**rdf:type** rdf:resource="http://www.pheme.eu/ontology/pheme#**CrossMediaLink**"/>
</rdf:Description>
</rdf:RDF>