## D1.3 - Event, Weather and Multilingual Data Services Specification

<table>
<thead>
<tr>
<th>Deliverable n:</th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>21 December 2017</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Version:</td>
<td>1.2</td>
</tr>
<tr>
<td>Authors:</td>
<td>Aljaž Košmerlj (JSI), Flavio De Paoli (UNIMIB)</td>
</tr>
<tr>
<td>Contributors:</td>
<td>JSI, UNIMIB</td>
</tr>
<tr>
<td>Reviewers:</td>
<td>Olga Melnyk (ME), Matej Žvan (Browsetel), Matjaž Dolenc (CDE)</td>
</tr>
<tr>
<td>Distribution:</td>
<td>Public</td>
</tr>
</tbody>
</table>
# History of Changes

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>24/10/2017</td>
<td>Tentative Table of Contents</td>
<td>Aljaž Košmerlj</td>
</tr>
<tr>
<td>0.9</td>
<td>19/12/2017</td>
<td>All content except for Section 4.3</td>
<td>Aljaž Košmerlj</td>
</tr>
<tr>
<td>1.0</td>
<td>20/12/2017</td>
<td>Section 4.3 on ASIA</td>
<td>Flavio De Paoli</td>
</tr>
<tr>
<td>1.1</td>
<td>21/12/2017</td>
<td>Implemented the comments of internal reviewers</td>
<td>Aljaž Košmerlj</td>
</tr>
</tbody>
</table>
Executive Summary

In this deliverable we provide the specification for the event, weather and multilingual data services. These services supply contextual information to the business data provided by the project business partners as well as the cross-lingual linking of datasets in different languages. The requirements for the services based on business needs are presented and the APIs addressing them are outlined.

The deliverable is based on requirements collected in deliverable D1.1 and data format specifications in deliverable D1.2. The business requirements for the described services are based on business case descriptions from deliverable D4.1.
# Table of context

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>History of Changes</td>
<td>2</td>
</tr>
<tr>
<td>Executive Summary</td>
<td>3</td>
</tr>
<tr>
<td>List of figures</td>
<td>6</td>
</tr>
<tr>
<td>List of tables</td>
<td>7</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>8</td>
</tr>
<tr>
<td>1.1 Relationship to Other Deliverables</td>
<td>8</td>
</tr>
<tr>
<td>1.2 Abbreviations and Acronyms</td>
<td>8</td>
</tr>
<tr>
<td>1.3 Document Structure</td>
<td>10</td>
</tr>
<tr>
<td>Chapter 2 Event Data</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Data Source</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Project Requirements</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Event Data API</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 API</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 Data format</td>
<td>15</td>
</tr>
<tr>
<td>Chapter 3 Weather Data</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Data Source</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Project Requirements</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Weather Data API</td>
<td>16</td>
</tr>
<tr>
<td>3.3.1 API</td>
<td>17</td>
</tr>
<tr>
<td>3.3.2 Data Format</td>
<td>17</td>
</tr>
<tr>
<td>Chapter 4 Multilingual Data Linking Services</td>
<td>18</td>
</tr>
<tr>
<td>4.1 Project Requirements</td>
<td>18</td>
</tr>
<tr>
<td>4.2 Wikifier</td>
<td>19</td>
</tr>
<tr>
<td>4.3 ASIA</td>
<td>20</td>
</tr>
<tr>
<td>4.4 XLing</td>
<td>21</td>
</tr>
<tr>
<td>Chapter 5 Conclusion</td>
<td>22</td>
</tr>
<tr>
<td>References</td>
<td>22</td>
</tr>
<tr>
<td>Appendix</td>
<td>23</td>
</tr>
<tr>
<td>5.1 Event Registry Data model</td>
<td>23</td>
</tr>
</tbody>
</table>
5.2 Weather Data Attributes ........................................................................................................ 26
5.3 Wikifier Annotation Format ..................................................................................................... 27
List of figures

Figure 1: Event Registry (http://eventregistry.org/) online graphical user interface displaying the advanced search options. ........................................................................................................................................... 12
List of tables

TABLE 1. ABBREVIATIONS AND ACRONYMS ................................................................. 8
TABLE 2. SHORT REFERENCES FOR PROJECT PARTNERS ........................................... 9
TABLE 3: SUMMARY OF THE TOOLS IN THE EW-SHOPP PLATFORM. ...................... 10
TABLE 4: LIST OF WEATHER ATTRIBUTES SUPPORTED BY THE EW-SHOPP WEATHER API. ................................................................. 26
Chapter 1  Introduction

One of the main aims of the EW-Shopp project is to contextualize business data with social and environmental factors. These are represented as events reported in the global media and weather. These factors both intuitively and in the experience of project business partners have a strong influence on consumer behaviour. Analysing these influences for insights and using them to build predictive models can offer a competitive advantage on the market.

Modern business also operate on several different markets and commonly handle data in several different languages. To enable efficient management and integration of such multilingual data a set of tools is needed for cross-lingual linking and annotation.

This deliverable provides the specifications for accessing the contextual data sources. Event Registry, a platform for real-time global media monitoring and analysis, is used as a data source for event data and the data source for weather data is the operational archive of the European Centre for Medium-Range Weather Forecasts. Specification for the cross-lingual annotation services developed by project technical partners is also provided to address the challenges of managing multilingual datasets.

Whenever possible, the specifications are presented using existing state of the service APIs and examples of their use. Most of the services and APIs described in this document are still in active development and may evolve in the coming months. Development and deployment of project pilot services is especially likely to have an impact. All subsequent changes to these services will be reported in follow-up deliverables.

1.1 Relationship to Other Deliverables

This deliverable specifies data services used in the EW-Shopp project. The data formats used by the services are specified by deliverable D1.2 [2] and follow the interoperability requirements specified in deliverable D1.1 [1]. The functionalities of the described data services were designed following the business case requirements and pilot specifications from deliverable D4.1 [3].

1.2 Abbreviations and Acronyms

Abbreviations and acronyms used in the document are explained in Table 1.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
</tbody>
</table>
### Abbreviation Description

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td>Business Case</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Values</td>
</tr>
<tr>
<td>EAN</td>
<td>European Article Number</td>
</tr>
<tr>
<td>EC</td>
<td>European Commission</td>
</tr>
<tr>
<td>ECMWF</td>
<td>European Centre for Medium-Range Weather Forecast</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>SFTP</td>
<td>Secure File Transfer Protocol</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>JSON</td>
<td>Java Script Object Notation</td>
</tr>
<tr>
<td>KG</td>
<td>Knowledge Graph</td>
</tr>
<tr>
<td>LOD</td>
<td>Linked Open Data</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>REST</td>
<td>Representation State Transfer web services</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>

Table 2 shows the project partners along with their short references for easier mentions throughout the document.

#### Table 2. Short references for project partners

<table>
<thead>
<tr>
<th>No.</th>
<th>Beneficiary (partner) name as in [GA]</th>
<th>Short reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA</td>
<td>UNIMIB</td>
</tr>
<tr>
<td>2</td>
<td>CENEJE DRUZBA ZA TRGOVINO IN POSLOVNO SVETOVANJE DOO</td>
<td>CE</td>
</tr>
<tr>
<td>3</td>
<td>BROWSETEL (UK) LIMITED</td>
<td>BT</td>
</tr>
<tr>
<td>4</td>
<td>GfK EURisko SRL</td>
<td>GfK</td>
</tr>
<tr>
<td>5</td>
<td>BIG BANG, TRGOVINA IN STORITVE, DOO</td>
<td>BB</td>
</tr>
<tr>
<td>6</td>
<td>MEASURENCE LIMITED</td>
<td>ME</td>
</tr>
</tbody>
</table>
Finally Table 3 contains a summary of the tools and components which comprise the EW-Shopp platform.

**Table 3: Summary of the tools in the EW-Shopp platform.**

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataGraft</td>
<td>DataGraft is a cloud-based platform for data hosting and interactive data transformations. In the platform it has the role of the data wrangler component together with Grafterizer, its data transformation interface. It is developed and maintained by SINTEF.</td>
</tr>
<tr>
<td>ASIA</td>
<td>A tool for the semantic enrichment of data available in tabular formats. It is supported by ABSTAT, a tool to profile knowledge graphs represented in RDF based on linked data summarization mechanisms. It is included as a plugin in DataGraft and is developed and maintained by UNIMIB.</td>
</tr>
<tr>
<td>QMiner</td>
<td>QMiner is a data analytics platform for processing large-scale real-time streams containing structured and unstructured data. In the platform it has the role of the data analyser component. It is developed and maintained by JSI.</td>
</tr>
<tr>
<td>Knowage</td>
<td>Knowage is a business intelligence suite with strong support for producing high-quality reports of the transformed, enriched and analyzed information obtained from the platform. In the platform it has the role of the data reporting component. It is developed and maintained by ENG.</td>
</tr>
</tbody>
</table>

**1.3 Document Structure**

The document is organized as follows. Chapter 1 has introduced the topic of the deliverable, namely the specification of event, weather and multilingual data services and has placed this document among its related deliverables. In Chapter 2 event data service based on the Event Registry platform is specified. Chapter 3 describes the weather data service and the API developed for its use. Finally in Chapter 4 the three tools comprising the multilingual data service are outlined.
Chapter 2  Event Data

In order to measure the effect of social context on consumer behaviour, a reliable source of events from the entire world is needed. In EW-Shopp this role is fulfilled by Event Registry\(^1\), a platform for real-time monitoring of global news. This chapter describes Event Registry, outlines the event data requirements in the project and specifies the event data API for use within EW-Shopp.

2.1 Data Source

Event Registry is a platform for real-time monitoring and analysis of global news. Its news feed service collects on average 175,000 news articles daily from over 26,000 news sources. The collected articles come from all over the world and are in 15 different languages. The sources include major global news outlets such as CNN or BBC, international news agencies such as Reuters or Associated Press as well as smaller local news publishers. All the news items are collected from the publishers’ RSS feeds and are stored together in the Event Registry database. Event Registry has been running continuously since December 2013 and has amassed over 209 million articles and over 7 million events.

All articles are processed with a lexical and semantic analysis pipeline and are then clustered together according to their content. All articles in a cluster discuss the same real-world event. In this document we use the term ‘event’ interchangeably for the occurrence and the article cluster describing it.

Semantic processing for in Event Registry includes semantic annotation using the Wikifier service, which is described in Section 4.2. This means entity mentions in all articles are linked to their concepts denoted by the URLs of the Wikipedia pages describing them. All articles are also categorized into a three level category taxonomy which consists of the top three levels of the DMOZ taxonomy\(^2\). Concept data is aggregated for events and can be further aggregated for any set of given events during analysis.

For all events the date of the event and the location of the event are determined. Note that the date of the event is not necessarily the same as the date when the articles were published. Event Registry analyses the dates mentioned in article content and determines the most likely date when the event discussed in the article has occurred. Only if none of the mentions is reliably deemed to be correct is the median publishing date of the articles in the event taken. A similar approach with location mentions is taken when determining the event location with the difference that the location is left empty if it cannot be determined in the content. The location can be either a city and/or a country and is represented using its GeoNames\(^3\) URI. Details about the representation of location data can be found in deliverable D1.2 [2].

\(^1\)http://eventregistry.org/  
\(^2\)http://dmoztools.net/  
\(^3\)http://www.geonames.org/
Figure 1: Event Registry (http://eventregistry.org/) online graphical user interface displaying the advanced search options.

All the listed event information can be used to query for events and articles in Event Registry. This can be performed using the online graphical interface (shown in Figure 1) with a number of rich visualization options. An alternative is through the Event Registry official API for Python⁴ and JavaScript⁵. Both APIs support querying events and articles through any of their properties and obtaining the result in JSON format.

2.2 Project Requirements

Event data provides the social context for the business process we are analysing in the project. Examples of this based on experience from the project business partners are:

⁴ https://github.com/EventRegistry/event-registry-python
⁵ https://github.com/EventRegistry/event-registry-node-js
• After highly publicised product releases from major electronics companies (e.g. Apple, Samsung, Huawei) here is increased interest in a product released and the brand to which it belongs. There is a spike in sales once the product actually reaches the stores.
• Certain major sports events have a positive effect on sales of select items. For example, the world championship in football is accompanied by an increase in sales of television sets.
• On the other hand, popular sports events can have a negative effect on web traffic. High profile games attract people into stadiums or in front of TVs and away from active web browsing. This can have significant impact on the level of impressions and click-through rates of online marketing campaigns. This effect can be even more pronounced if the local team is playing.

Based on such experience the business partners have formed the business case requirements described in deliverable D4.1 [3]. From those we can design the event data requirements which can be summarized into the following list. The event data service needs to support:

1.) querying for events from a given time range;
2.) querying for events from a given geographical area;
3.) querying for events from a given category;
4.) querying for events related to a given concept or keyword;
5.) aggregation of event data for a set of events;
6.) any combination of previous requirements.

The exact mechanism of integrating event data with other datasets based on temporal and spatial information is covered in deliverable D1.2 [2]. Precise matching may not always be possible for both temporal and spatial matches. If nothing else, event time is only determined on the level of date and not the hour and event location is determined at most at city level. It is worthy to note that this is not an issue. Most relevant events, such as those from examples in the beginning of this section, have an effect with a wide scope. By this we mean their effect lasts for some significant time period and affects a wider geographical region. Thus it will be captured even with a more coarse matching approach.

This requirements list was built before the deployment of pilots and is based on understanding of the business cases by the business partners and the expertise of the technical partners. The requirements may evolve over time as this understanding deepens and the business hypotheses are tested in the pilots.

Finally, it is important to note that Event Registry only lists events reported in the media. There are types of events with influence on consumers that are not commonly covered by news outlets. Occurrences perhaps thought too banal to be newsworthy, such as the date when cars are legally obliged to switch to winter tires in countries with cold winters. Also not included are any kind of company internal events such as marketing campaigns, service updates, competitions for prizes etc. Should business partners have access to data regarding any such events, they have to be integrated through the platform as a separate data source. Efforts will be made during development of pilots to build a data model for such custom events that will cover the needs of all business cases.
2.3 Event Data API

After review of the requirements listed in Section 2.2, we conclude that the Event Registry API fulfils all of them and no major extension is needed at this point. It is likely some extension will become necessary during development of the pilots. Especially on streamlined filtering of events relevant to pilot domains. Since the extent and nature of those is unclear at this time, they will be reported in follow-up documentation. The details of the API are presented in this section.

2.3.1 API

Event Registry has an API for Python and JavaScript. Though the language is different, both APIs follow the same structure and in the background use the same REST API to perform all the online queries. It is also possible to use the REST API directly by crafting the HTTP requests themselves. However that would involve a lot more technical work and offer no advantage. Since the two APIs are so closely related, we use singular when referring to it in the rest of the text and the description holds for both.

The API uses JSON as the format for both the requests sent to the Event Registry service as well as the event data returned. The request object holds the query parameters such as the event time range or location, article language etc. The request object is then transmitted to the Event Registry server which computes the response and returns the data. A detailed overview of the API functions surpasses the scope of this deliverable and can be found online\(^6\). The main functionality can be summarized into the following operations:

- **Searching for events/articles** – In this operation Event Registry is queried for a list of events that fall within the bounds of the request parameters such as the time range, category or relevance to some concept. The data returned is the list of events or articles with a limited amount of basic data fields such as the event or article title. Besides these, aggregated information over all the events or articles can be requested such as an aggregation of relevant concepts or categories over all events or articles returned by the queries.

- **Obtaining information about events/articles** – This operation is used to obtain detailed information about a set of given Event Registry events or articles. This list most commonly comes from a previously executed search operation. All the detailed information that Event Registry holds about specific events or articles can be obtained using this operation.

- **Matching keywords to appropriate concepts** – Event Registry supports queries using normal keywords, similar to any web search engine. However searching using concept annotation is preferred, since it takes advantage of the entity disambiguation performed during annotation. This operation obtains the appropriate concepts (represented by their Wikipedia URIs) for given keywords.

Event Registry offers more functionality than covered in this overview, such as information about long-term trends in events or about the sharing of articles in social media. However we do not

\(^6\) https://github.com/EventRegistry/
foresee any direct use for this data at this point. Should they prove relevant in future, we will add them to the project data and report this in future documentation.

### 2.3.2 Data format

As mentioned before, the format of the returned data is JSON. The full data model is available in the Event Registry online documentation[7] and contains the data model for Event Registry representation of article, event, category, concept and news source. A subset of the data fields for a selection of the most relevant entities is presented in the appendix in Section 5.1.

### Chapter 3  Weather Data

While the events represent the social context of consumer behaviour, the weather represents the environmental context. To obtain high quality weather data the EW-Shopp project has an agreement with the European Centre for Medium-Range Weather Forecasts[8] (ECMWF). ECMWF is an independent intergovernmental organisation supported by 34 states and is both a research institute and a 24/7 operational weather analysis and forecast service. This chapter presents the ECMWF data services, outlines the weather data requirements in the project and specifies the weather data API developed for EW-Shopp.

#### 3.1 Data Source

ECMWF is one of the leading meteorological institutions in the world. It has been established in 1975 and today joins experts and resources from 34 supporting countries. Besides being a research institute it is an operational weather analysis and forecast service providing weather forecast data to its member states. This supports national weather services of member countries which use this data to prepare the weather forecasts their citizens follow in the media.

As a part of their operational activities, ECMWF uses the data of weather measurement from all the member states and partner institutions to compute the model of the full state of global weather twice per day with a forecast for the following ten days. The weather state and forecast data is stored in their Meteorological Archival and Retrieval System[9] (MARS) archive. EW-Shopp has obtained full access to this archive spanning back over 30 years.

#### 3.2 Project Requirements

---

[8] https://www.ecmwf.int/
[9] https://www.ecmwf.int/en/faq/what-mars
Weather has a strong influence over human behaviour. It can cause physical discomfort with great heat, limit our movement with a strong downpour or even damage our property with high winds or hail. Of course this extends to consumer behaviour as illustrated by the following examples based on the business partner experience:

- Warm sunny days mean a decrease in web traffic as more people are outdoors and not using their devices to browse the web.
- The first wet season in autumn triggers a surge in sales of clothes dryers, as hanging clothes out to dry becomes significantly less effective.
- Bad weather conditions such as strong rain and high winds result in less foot traffic in front of stores as people take refuge from the elements.

Similarly as for events data we can form requirements for weather data from the business case requirements described in deliverable D4.1 [3]. The weather data service needs to support:

1.) querying for both actual weather state or weather forecast made at some given time;
2.) querying for weather data from a given time range;
3.) querying for weather data from a given geographical area;
4.) aggregating weather data over some time range or geographical area; aggregation includes computing the minimum, maximum or average values.

As with event data requirements, weather data requirements may change during the course of the project as deeper insights into the effects of weather on consumers are discovered. All potential extensions will be documented in future deliverables.

### 3.3 Weather Data API

MARS can be accessed via REST API using a simple query language. For example:

```python
retrieve,
    class  = od,
    type   = an,
    expver = 1,
    date   = 19990215,
    time   = 12,
    param  = t,
    levtype = pressure level,
    levelist = 1000/850/700/500,
    target = "t.grb"
```

This request retrieves 1000, 850, 700 and 500 hPa temperatures from the 15th of February 1999 12:00 UTC Analysis. The request syntax is complex and requires a lot of knowledge about MARS internal data structure. An official Python wrapper around this API is available, which improves on this but is unfortunately still in development and of limited use. Both approaches also download the data in the GRIB (GRIdded Binary) format commonly used in meteorology. This format, though efficient with respect to disk space, is not easy to decode and needs to be transformed for practical use. To alleviate these issues JSI developed a dedicated wrapper for use in EW-Shopp.
3.3.1 API

The EW-Shopp weather API developed by JSI is freely available on the web\(^{10}\), but data can only be obtained with an appropriate API key provided by the ECMWF. JSI is in charge of distributing the API key among the project partners. The API is written in Python and uses a combination of the official Python API and direct compilation of MARS requests in the background.

Since the API is dedicated for use in the EW-Shopp project it has been focused to only return weather attributes relevant to the project. The MARS archive has a rich selection of over 100 meteorological attributes to choose from, however most of them are relevant for analysis of meteorological phenomena and not relevant to the project aims. 15 relevant attributes were manually selected for inclusion into the project weather data. The full list with descriptions is included in the appendix in Section 5.2.

The API is designed around two main objects: `WeatherApi` and `WeatherExtractor`. `WeatherApi` handles data acquisition from the MARS archive. It builds a request, transmits it to the MARS data service and downloads the requested data. The request can specify the parameters listed below.

- **date (range):** The date or the start/end dates of the dataset.
- **time:** Selection of the time of day when the data and forecasts are computed. Either 00:00:00 or 12:00:00.
- **step:** Time(s) in hours from the time of computation (value of the ‘time’ parameter) for which forecast data is returned. Values can be in \([0,1,2,...,89]\) u \([90,93,96,...,141]\) u \([144,150,156,...,240]\).
- **area:** The latitude/longitude boundaries of the area for which the data is returned.
- **resolution:** The data is computed over a grid. This parameter specifies the latitude/longitude resolution for which the data is returned. The values are in degrees, for example: \([0.125, 0.125]\) for roughly 15 km x 15 km grid.

The data is downloaded in GRIB format and stored into a file on disk. Extraction and management of the data in the GRIB file is handled by the `WeatherExtractor` object, which is described in the next section.

3.3.2 Data Format

As described in the previous section, weather data is downloaded by the API to disk in the GRIB format. The `WeatherExtractor` object loads the data from the GRIB file, transforms it into a pandas\(^{11}\) DataFrame which serves as internal representation of the data and performs filtering and aggregation operations. Pandas is an open-source Python data analysis library and DataFrame is its main data structure. Due to its generality, this format is well suited for integration and use by the

---

\(^{10}\) https://github.com/JozefStefanInstitute/weather-data

\(^{11}\) https://pandas.pydata.org/
project business partners. From it the data can easily be transformed into a number of other data formats such as JSON, SQL, csv etc.

Several aggregation operations are also available, where aggregation can be computation of minimum, maximum or average value. The data can be aggregated over time and geographical area. For time an hour, day or week range can be specified. For area an area with specific latitude and longitude boundaries can be specified or a set of target points can be given. In the latter case aggregated values are computed for the given target points. Each grid data point is aggregated only to the target point closest to it. The results of all aggregation operations are again internally represented as a pandas DataFrame and can be transformed into some other format for use or output.

Note that business partners may be aware of weather-related events that may have special meaning. For example the first snow of winter commonly causes a much larger disruption than when it snows later. They may also have records of extreme weather events such as strong hail or frost which impacted their services significantly. Any such records are a complementary data source to the weather data service which holds measurements, not discrete events. These special weather events will be integrated via the same custom event data model mentioned in the last paragraph of Chapter 2.

Chapter 4  Multilingual Data Linking Services

Project business partners operate over diverse international geographical areas. As consequence, the data they handle is multilingual. In order to enable interoperability of such data and enable the use of insights from areas with one language in areas with another, a set of tools for cross-lingual data linking must be provided in the project. This section presents the requirements for these services in the project and three tools providing the needed functionality – the Wikifier, ASIA and XLing.

4.1 Project Requirements

Textual data in the project comes in two main forms:

- **free text** – such as news articles, product descriptions and documentation, marketing materials;
- **tabular data** – most data provided by business partners is in tables containing textual pieces of data such as the product name, brand, manufacturer, colour and other non-numerical properties.
For both forms, most of the data coming into the project is unprocessed and unlinked. Such raw text is not well suited for machine processing. Though direct string matching techniques can be used to support certain functionality, of course, these do not work in a cross-lingual setting.

In discussion with business partners, the three main functionalities of the multilingual data linking services listed below were identified.

- **annotation of concept mentions in free text** – In text data analysis this task is known as *semantic annotation*. The purpose is to find mentions of concepts, disambiguate them and link them to some reference knowledge base. In the scope of the EW-Shopp this, for example, means identifying mentions of manufacturers, brands and technologies in free text product descriptions or mentions of products in news article text.

- **linking tabular data to concepts** – Most business data (perhaps most of all data) is stored in tabular form. Record ids or some other identifying values are typically used to link tables of data between each other (keys in relational databases are an example of this). This approach is highly efficient but depends on the existence of such matching values. In cases when they are unavailable such an approach is impossible. It is common to have reference data (e.g. a curated repository of product data) and a dataset we want to link to this reference data (e.g. a catalogue of products available on a web store) that share no identifiers. We need a method that is able to link two records from these two datasets based on the semantics of their values. An example of this would be to match products from the Ceneje web store catalogue to GfK product reference data based on product properties since they do not share a common set of product ids.

- **measuring semantic relatedness of free texts** – A large amount of information is still stored in the form of unstructured text, such as product descriptions, marketing materials and news items. Humans easily parse such information and can identify the same type of product (e.g. televisions or refrigerators) or identical products (e.g. the very same model of toaster) from different descriptions. Ceneje, as a comparison shopping engine, regularly need to solve this problem when matching product descriptions from different stores on their website. Therefore, a method for automatically measuring such semantic relatedness between pieces of text is needed.

All three functions must work in a cross-lingual setting. The following sections present tools each answering one of the three listed above requirements, correspondingly.

### 4.2 Wikifier

The Wikifier\(^\text{12}\) is a semantic annotation service developed and maintained by JSI. It performs a subtype of semantic annotation called *wikification*. This name stems from its use of Wikipedia as a reference database and the source of concept identifiers. In this setting, the Wikipedia is treated as a large and fairly general-purpose ontology: each page is thought of as representing a concept, while the relations between concepts are represented by internal hyperlinks between different Wikipedia

\(^{12}\) [http://wikifier.org/](http://wikifier.org/)

pages, as well as by Wikipedia’s category memberships and cross-language links. Wikifier is the semantic annotation tool used to compute article and event concepts in the Event Registry.

The task of performing wikification on an input document can be broken down into several closely interrelated subtasks:

1.) identify phrases (or words) in the input document that may refer to a concept;
2.) perform disambiguation - determine which concept is the one that a phrase is referring to;
3.) determine which concepts are relevant enough to the entire document, so that they should be included in the output of the system.

The Wikifier uses an approach similar to the one described in [4] based on the PageRank algorithm [5] to perform steps 2. and 3. A detailed technical description is outside the scope of this document. In layman’s terms it selects those concepts from the annotation candidates that are “close” to each other in Wikipedia (i.e. following internal hyperlinks) and are relevant to the general document content. The result is a list of all concepts mentioned in the document along with the exact location of their mentions. Each annotation also has an estimated relevance score which can be used to control the level of annotation uncertainty.

The Wikifier service is maintained by JSI and is available via a REST API. Full documentation is available online13. It can annotate documents in any of the top 100 languages sorted by the size of their respective Wikipedia corpora. A document can be submitted via a HTTP GET o POST request and a JSON response is returned with the annotated result. Some examples of the Wikifier response structure are presented in the Appendix in Section 5.3.

4.3 ASIA

ASIA is a tool for the semantic enrichment of data available in tabular formats, developed by UNIMIB. Joining tabular data that does not use the same record ids or some other identifying values is not straightforward, due to the unavailability of direct joining points (i.e. equal values). Addressing this issue requires to create links from table values to a shared system of identifiers, which allows to bridge the gap between data.

ASIA aims to help users in creating these links, by implementing semantic reconciliation algorithms to perform the entity linking on tables, that is linking table values to some external reference data. Particularly, the entity linking is performed at two different levels:

- Schema-level linking: linking table schema values (i.e. the header of a table) to shared vocabularies and ontologies;
- Instance-level linking: linking data values to shared systems of identifiers.

Even if the instance-level linking might be enough in order to enable the enrichment, also the schema-level linking is expected to be considered, since it is helpful in addressing some issues.

---

13 http://wikifier.org/info.html
related to the instance-level linking (e.g. schema-level linking results are used to apply blocking techniques while performing the instance-level linking).

Schema-level and instance-level links are created by ASIA as annotations for the table. Users can create schema-level annotation through the ASIA interface, by validating ASIA suggestions about classes and properties – defined in several ontologies and vocabularies – to be used. If a user specifies a different class (or property), ASIA suggests classes (or properties) that syntactically match the user input (autocomplete functionality). Otherwise, the instance-level annotations are expected to be created by ASIA automatically, because of the higher dimension of the data values, which cannot be validated one-by-one.

Table annotations enable two different functionalities of ASIA:

- Generation of knowledge graphs (KGs) from a tabular data set: the schema-level annotations are transformed into executable data transformation to publish tabular data as a KG; data values will be used to create new instances and populate the graph.

- Enrichment of tabular data with third-party data: instance-level annotations (along with schema-level annotations, when needed), which link table values to reference KGs, are used to facilitate enrichment of business data with data from these reference KGs (e.g., the link to a product in the products datasets allows to retrieve also the product brand, stored in the same dataset) or from third-party data by using links as bridges (e.g., the link to a GeoNames location can be used to retrieve the GeoNames location identifier, that is required for retrieving events from the Event Registry). ASIA supports this key enrichment process by providing data enrichment widgets, which exploit links to reference KGs to ease the extraction of additional data from third-party sources and their fusion into the original tabular data.

The ASIA interface is developed as component of Grafterizer, a tool for tabular data cleaning and transformation, developed at SINTEF. The ASIA Backend is developed and maintained by UNIMIB. All linking services implemented in the ASIA Backend are made accessible via REST APIs.

The current version of ASIA provides only the schema-level linking suggestions, based on ABSTAT. This service can be invoked via a HTTP GET request, passing some parameters (string to be autocompleted, type of requested suggestion – class or property, number of results to be returned, and so on). The service returns a JSON with all classes or properties that are syntactically similar to the string passed as parameter.

### 4.4 XLing

XLing\(^{14}\) is a cross-lingual document semantic similarity measuring service and is one of the inputs used for article clustering in Event Registry. It receives two documents as input and computes a similarity score as a number in the \([0, 1]\) range. As Wikifier it works on the top 100 languages sorted by the size of their respective Wikipedia corpora.

\(^{14}\) [http://xling.ijs.si](http://xling.ijs.si)
The approach used to compute the similarity score is based on canonical correlation analysis. It avoids direct translation of the documents and rather uses a statistical hub language to connect them. The hub language is computed using Wikipedia articles as an aligned multilingual text corpus. Details of the approach can be found in [6].

The service is available via a simple REST API. The two documents are submitted in a post request with their respective languages (i.e. their ISO 693-1 codes). The similarity score is returned in the HTTP response.

Chapter 5  Conclusion

This document presented the specification of event, weather and multilingual data services for the EW-Shopp platform. Data sources and tools for each data type were presented with their requirements and APIs. The requirements were based on descriptions of business partners’ workflows and specifications of their pilot services in deliverable D4.1. Extensions to the tools were planned and developed where necessary, however other issues and needs will likely be discovered during deployment of the pilots.

One such important factor is the workload resilience and responsiveness of the services. The services were developed with performance in mind, however once the platform is tested on real data streams, more optimizations might prove to be necessary. Caching of queried data and improved support for bulk data retrieval (i.e. multiple queries per request) are examples of techniques that could be used to address such issues if they arise.

API security is also an open issue. Most of the services listed in the document have been developed in an academic setting and are very lenient in that respect. As they get plugged into a commercial platform the dangers of attacks and exploits will likely rise severely. There are ongoing discussions in the project on addressing this satisfactorily.

It is clear the services will most assuredly continue to evolve during the project. All subsequent changes will be reported in future deliverables.

References

[2] D1.2: Spatial, temporal and product data format specification
Appendix

5.1 Event Registry Data model

We present selected elements from the Event Registry Data model using example data. Further details can be studied in the Event Registry online documentation\textsuperscript{15}.

Article

```
{
  // article's URI (newsfeed id)
  "uri": "143701955",
  // web url
  // article's title
  "title": "Desperate Obama Tries to Reset Agenda with New Staff",
  // article's full body
  "body": "Highlight the link and press CTRL/Command + C to copy the link to your clipboard. In As Phil Schiliro arrived at his first meeting last ...",
  // date and time of publishing
  "date": "2013-12-18",
  "time": "11:40:00",
  "datetime": "2013-12-18T11:40:00Z",
  // event URI to which the article is assigned to (if any)
  "eventUri": "20588",
  "source": {
    // details about the news source (see Source data model)
  },
  "categories": [
    // list of categories (see Category data model)
  ],
  "concepts": [
    // list of concepts (see Concept data model)
  ],
  // dates that were extracted from the article
  "extractedDates": [
    {
      "amb": false,          // ambiguous?
      "date": "2013-12-03",  // normalized date
      "dateEnd": "2013-12-08",
      "detectedDate": "Dec. 3-8", // detected string
      "imp": true,           // was the year value imputed?
      "posInText": 6164,     // location in text
      "textSnippet": "ublican attacks. A Dec. 3-8 poll of 86 competit"
    }
  ]
}
```

\textsuperscript{15}https://github.com/EventRegistry/event-registry-python/wiki/Data-models
Event

{
  // event URI
  "uri": "3403979",
  // total articles reporting about the event
  "totalArticleCount": 100,
  // articles per language
  "articleCounts": {
    "deu": 82,
    "eng": 18
  },
  // list of concepts (see Concept data model)
  "concepts": [
  ],
  // list of categories (see Category data model)
  "categories": [
  ],
  // event title in available languages
  "title": {
    "deu": "Obama kommt zur Eröffnung der Hannover Messe",
    "eng": "White House says Obama will make 5th visit to Germany, take in trade show"
  },
  // event summary in available languages
  "summary": {
    "deu": "Hannover (dpa) US-Präsident Obama kommt 2016 wieder nach Deutschland: In Hannover eröffnet er die weltgrößte Industrie-Messe. Die Sicherheitsmassnahmen werden schärfer sein als 2013. ...",
    "eng": "HONOLULU, Hawaii - The White House says President Barack Obama will travel to Germany in late April to attend the world’s largest trade show for industrial technology ..."
  },
  // which dates have been frequently found in articles about this event
  "commonDates": [
    {"date": "2016-04-24", "freq": 11}
  ],
  // when the event happened
  "eventDate": "2016-04-24",
  // how much impact on social media did articles about the event get
  "socialScore": 91.4,
  // where did the event happen
  "location": {
    "area": 357021,
    "code2": "DE",
    "code3": "DEU",
    "continent": "Europe",
    "country": "Germany",
    "geoNamesId": "2921044",
    "label": {
      "eng": "Germany",
      "spa": "Alemania"
    }
  }
}
"lat": 51.5,
"long": 10.5,
"type": "country",
"wikiUri": "http://en.wikipedia.org/wiki/Germany"
},
"geoNamesId": "2910831",
"label": {
  "eng": "Hanover",
  "spa": "Hannover"
},
"lat": 52.37052,
"long": 9.73322,
"population": 515140,
"type": "place",
"wikiUri": "http://en.wikipedia.org/wiki/Hanover"
},
// if event is provided as a result of a query, wgt represent relevance to the query (in range 0-100)
"wgt": 98
}

Category
{
  // category's URI
  "uri": "dmoz/Society/Issues/Warfare_and_Conflict",
  // URI of the parent category
  "parentUri": "dmoz/Society/Issues",
  // category label
  "label": "Society/Issues/Warfare_and_Conflict",
  // URIs of children categories
  "childrenUris": [],
  // how much was the category trending in the last days
  "trendingHistory": {
    "latestArticleTimestamp": "2016-03-17 03:44:00",
    "news": [
      5867,  // 29 days ago
      6818,
      5927,
      ...
      3371,
      5957,
      5782,  // 2 days ago
      5139,  // yesterday
      646   // today
    ]
  },
  // internal ER id - do not use!
  "id": 283
}

Concept
{
  // concept's URI
  // concept type - person, loc, org or wiki
  "type": "loc",
  // concept labels in requested languages
  "label": {
    "eng": "United States",
    "spa": "Estados Unidos"
  },
  // what classes does the concept belong to
}
5.2 Weather Data Attributes

The API downloads the weather attributes listed in Table 4.

Table 4: List of weather attributes supported by the EW-Shopp weather API.

<table>
<thead>
<tr>
<th>name</th>
<th>unit</th>
<th>short name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud base height</td>
<td>m</td>
<td>cbh</td>
<td>Searching from the 2nd lowest model level upwards, the height of the level where cloud fraction becomes &gt;1% and condensate content &gt;1.E-6.</td>
</tr>
<tr>
<td>Maximum temperature at 2 meters in the last 6 hours</td>
<td>K</td>
<td>mx2t6</td>
<td>Maximum temperature at 2 metres in the last 6 hours</td>
</tr>
<tr>
<td>Minimum temperature at 2 meters in the last 6 hours</td>
<td>K</td>
<td>mn2t6</td>
<td>Minimum temperature at 2 metres in the last 6 hours</td>
</tr>
<tr>
<td>10 metre wind gust in the last 6 hours</td>
<td>m/s</td>
<td>10fg6</td>
<td>10 metre wind gust in the last 6 hours</td>
</tr>
<tr>
<td>Surface pressure</td>
<td>Pa</td>
<td>sp</td>
<td>Air pressure at ground level</td>
</tr>
<tr>
<td>Total column water vapour</td>
<td>kg/m²</td>
<td>tcwv</td>
<td>Vertically integrated water vapour</td>
</tr>
<tr>
<td>Snow depth</td>
<td>m of water</td>
<td>sd</td>
<td>Depth of snow coverage</td>
</tr>
<tr>
<td>Snowfall</td>
<td>m of water</td>
<td>sf</td>
<td>Convective + stratiform snowfall. Accumulated</td>
</tr>
<tr>
<td>name</td>
<td>unit</td>
<td>short name</td>
<td>description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>equivalent equivalent field.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total cloud cover</td>
<td>(0 - 1)</td>
<td>tcc</td>
<td>Total cloud cover derived from model levels using the model's overlap assumption</td>
</tr>
<tr>
<td>2 metre temperature</td>
<td>K</td>
<td>2t</td>
<td>Temperature 2 m above the ground</td>
</tr>
<tr>
<td>Total precipitation</td>
<td>m</td>
<td>tp</td>
<td>Height of water in one m^2 from precipitation. Convective precipitation + stratiform precipitation (CP +LSP). Accumulated field.</td>
</tr>
<tr>
<td>Precipitation type</td>
<td>code table (see description)</td>
<td>ptype</td>
<td>Describes the type of precipitation at the surface at the validity time. A precipitation type is assigned wherever there is a non-zero value of precipitation in the model output field (however small). The precipitation type should be used together with the precipitation rate to provide, for example, indication of potential freezing rain events. Precipitation type (0-8) uses WMO Code Table 4.201 Values of ptype defined in the IFS: 0 = No precipitation, 1 = Rain, 3 = Freezing rain (i.e. supercooled), 5 = Snow, 6 = Wet snow (i.e. starting to melt), 7 = Mixture of rain and snow, 8 = Ice pellets</td>
</tr>
<tr>
<td>Visibility</td>
<td>m</td>
<td>vis</td>
<td>Visibility in metres.</td>
</tr>
<tr>
<td>Maximum total precipitation rate in the last 6 hours</td>
<td>kg m^{-2} s^{-1}</td>
<td>mxtpr6</td>
<td>The total precipitation is calculated from the combined large-scale and convective rainfall and snowfall rates every time step and the maximum is kept since the last 6 hours.</td>
</tr>
<tr>
<td>Minimum total precipitation rate in the last 6 hours</td>
<td>kg m^{-2} s^{-1}</td>
<td>mntpr6</td>
<td>The total precipitation is calculated from the combined large-scale and convective rainfall and snowfall rates every time step and the minimum is kept since the last 6 hours.</td>
</tr>
</tbody>
</table>

### 5.3 Wikifier Annotation Format

The Wikifier returns a JSON response of the following form:

```json
{
  "annotations": [ ... ],
  "spaces": ["", " "],
  "words": ["New", "York", "City"],
  "ranges": [ ... ]
}
```
Where the annotations field contains a list of annotation objects; the words and spaces fields contain the tokenized document text and can be used to reconstruct it; and the ranges field lists the concept mention candidates. Each annotation has the following structure:

```json
{
  "title": "New York City",
  "url": "http:\/\en.wikipedia.org\wiki\New_York_City",
  "lang": "en",
  "pageRank": 0.102831, "cosine": 0.662925,
  "secLang": "en",
  "secTitle": "New York City",
  "secUrl": "http:\/\en.wikipedia.org\wiki\New_York_City",
  "wikiDataClasses": [
    {"itemId": "Q515", "enLabel": "city"},
    {"itemId": "Q1549591", "enLabel": "big city"},
    ...
  ],
  "wikiDataClassIds": ["Q515", "Q1549591", ...],
  "dbPediaTypes": ["City", "Settlement", "PopulatedPlace", ...],
  "dbPediaIri": "http:\/\dbpedia.org\resource\New_York_City",
  "supportLen": 2.000000,
  "support": [
    ... a list of ranges in the text that are mentions of this concept ...
  ]
}
```

A detailed description of all annotation fields is available in the Wikifier online documentation.