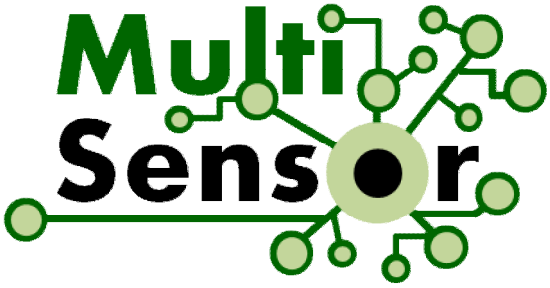

MULTISENSOR

Mining and Understanding of multilinguaL contenT for Intelligent Sentiment Enriched coNtext and Social Oriented inteRpretation

FP7-610411

D7.1

Roadmap towards the implementation of MULTISENSOR platform

Dissemination level:	Public
Contractual date of delivery:	Month 4, 28 February 2014
Actual date of delivery:	Month 4, 28 February 2014
Workpackage:	WP7 System development and integration
Task:	T7.1 MULTISENSOR architecture
Type:	Report
Approval Status:	Final Draft
Version:	1.1
Number of pages:	70
Filename:	D7.1_Roadmap_2014-02-28_v1.1.pdf

Abstract

The objective of this document is to define the outline of the time plan for the development of the MULTISENSOR platform. In this context the deliverable provides a high level view of MULTISENSOR Platform architecture and a specification of the infrastructure layout. In addition the deliverable includes an initial functional description of each of the technical modules, describing the supporting technologies, the increasing functionalities over time and the development timeline. Finally, the deliverable includes a summary of the use cases

and a global project timeline, describing the platform's scheduled iterations and the levels of functionality at the project milestones;

co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	01/02/2014	First version	A. Callabed (EVERIS)
0.2	10/02/2014	Contributions by partners	All partners
0.4	15/02/2014	First integrated draft	A. Callabed, A. Mas, E. Staromiejski (EVERIS)
0.5	21/02/2014	First internal review	S. Vrochidis (CERTH)
0.6	23/02/2014	Additional contribution by partners	All partners
0.7	26/02/2014	Second integrated draft addressing the review comments	EVERIS
0.8	27/02/2014	Final internal review	S. Vrochidis (CERTH)
1.0	28/02/2014	Final document	A. Callabed (EVERIS)
1.1	28/02/2014	Final updated document correcting minor errors	A. Callabed (EVERIS)

Author list

Organisation	Name	Contact Information
EVERIS	Axel Callabed	acallabe@everis.com
EVERIS	Alan Mas	alan.mas.soro@everis.com
EVERIS	Enric Staromiejski	enric.staromiejski.torregrosa@everis.com
CERTH	Anastasia Moumtzidou	moumtzid@iti.gr
CERTH	Theodora Tsikrika	theodora.tsikrika@iti.gr
CERTH	Ioannis Kompatsiaris	ikom@iti.gr
UPF	Gerard Casamayor	gerard.casamayour@upf.edu
LINGUATEC	Vera Aleksić	v.aleksic@linguatec.de
LINGUATEC	Boris Vaisman	b.vaisman@linguatec.de
LINGUATEC	Gregor Thurmair	g.thurmair@linguatec.de
LINGUATEC	Reinhard Busch	r.busch@linguatec.de
BM-Y!	Ioannis Arapakis	arapakis@yahoo-inc.com
BM-Y!	Nicola Barbieri	barbieri@yahoo-inc.com
BM-Y!	Iris Miliaraki	irismili@yahoo-inc.com
ONTOTEXT	Maurice Grinberg	maurice.grinberg@ontotext.com
ONTOTEXT	Vladimir Alexiev	vladimir.alexiev@ontotext.com
ONTOTEXT	Dimitar Manov	mitac@sirma.bg
ONTOTEXT	Kiril Simov	kiril.simov@ontotext.com
ONTOTEXT	Svetoslav Petrov	svetoslav.petrov@ontotext.com
pressrelations	Michael Jugov	michael.jugov@pressrelations.de

PIMEC	Teresa Forrellat	TForrellat@pimec.org
Deutsche Welle	Nicolaus Heise	nicolaus.heise@dw.de

Executive Summary

This deliverable presents the technological roadmap towards the implementation of the MULTISENSOR platform.

First, this roadmap presents a high-level view of the envisioned architecture for the MULTISENSOR platform, illustrating the conceptual architecture and a draft of the complete technical architecture. Then, the deliverable describes the modules that are developed in each Workpackage and comprise the MULTISENSOR platform. For each module a functional description is provided including a summary of its scientific and technical objectives and supporting technologies, input/output specifications, the increasing functionality over time, the development timeline, as well as the detailing schedule and resource allocation. In addition, D7.1 provides a summarised vision of the proposed use cases and their implementation strategy. Finally it presents a global project timeline, describing the platform's scheduled iterations and the levels of functionality at the project milestones.

Abbreviations and Acronyms

ASR	Automatic Speech Recognition
CI	Continuous Integration
DoW	Description Of Work
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LAN	Local Area Network
LOD	Linked Open Data
MT	Machine Translation
NER	Named Entities Recognition
OWL	Ontology Web Language
RDF	Resource Description Framework
REST	Representational State Transfer
SMT	Statistical-based machine translation
SOA	Service Oriented Architecture
SOAP	Simple Object Activation Protocol
SQL	Simple Query Language
SSL	Secure Sockets Layer
SVN	Subversion
WP	Work Package
XML	Extensible Mark-up Language

Graphic conventions

The following graphics/icons are used throughout this document, in diagrams and elsewhere:


Web application / Web page


Integration service


Business service


Repository

Business service, belongs to WP3

Table of Contents

1 INTRODUCTION

This document presents a roadmap on how the Consortium plans to develop the MULTISENSOR platform. The objective of the deliverable is to specify in detail: (i) the progressively increasing functionality (in particular at the time points of the milestones) of the individual modules as well as of the platform as a whole, (ii) the temporal and functional synchronisation of the individual modules to ensure this functionality; (iii) the resources that will be needed to achieve this functionality; (iv) the technical infrastructure specifications.

To this end, and based on a model of iterative development, the roadmap includes:

1. A high level view of the envisioned Platform architecture, illustrating the conceptual architecture and a draft of the complete technical architecture. A specification of the infrastructure layout and topology is also provided.
2. A functional description of each of the technical modules, describing:
    a. A summary of its scientific and technical objectives, and supporting technologies;
    b. Its progressively increasing functionality over time;
    c. Its timeline, detailing schedule and resource allocation;
3. A summarised vision of the proposed use cases and their implementation strategy;
4. A global project timeline, describing the platform's scheduled iterations and the levels of functionality at the project milestones;

Therefore this deliverable is structured as follows. Section 2 presents the architecture, while section 3 provides the functional description of the modules involved in alignment with the project workpackages (WPs). Then, section 4 presents the development cycle and briefly discusses the use cases and finally section 5 provides the overall project timeline and the milestones. Finally, section 6 concludes the deliverable.

2 ARCHITECTURE

This section includes the initial architecture of MULTISENSOR. It shows the evolvement from the high-level schema foreseen in the DoW to a 4-layer structured service-oriented architecture. The full Description of MULTISENSOR architecture will be provided in [D7.2].

2.1 Description

The MULTISENSOR architecture will drive the development process and organise the different components that make up the platform.

The architecture drafted for the DoW defines the initial high-level and work-package oriented structure of the platform, the processing flows and the connections between its components, as shown in Figure 1.

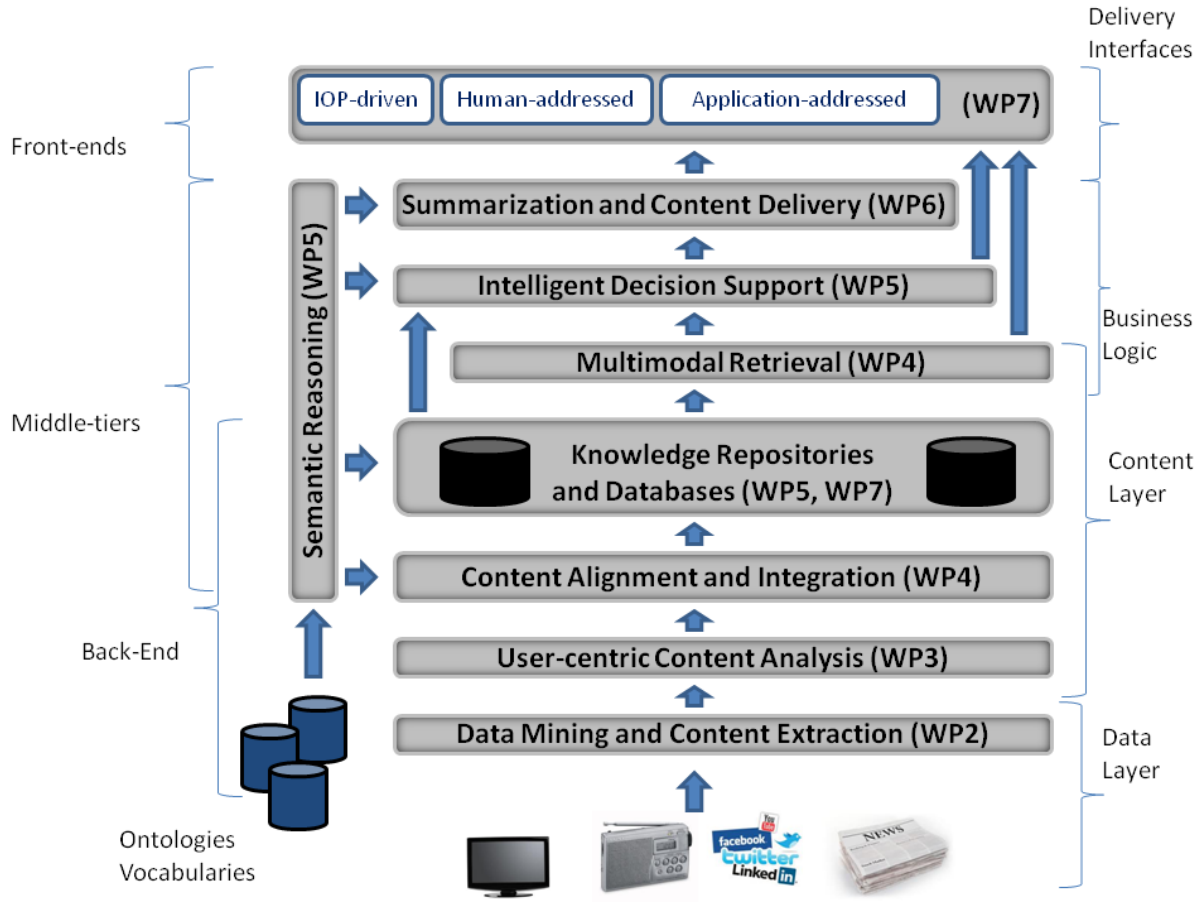

Figure 1: Preliminary MULTISENSOR architecture

The MULTISENSOR platform will be modelled as a **service-oriented architecture** (SOA[1]), which fits well with the distributed, loosely-coupled nature of the processes to integrate. The main processes will be modelled as orchestrations of services to achieve the desired functionality.

[1] See **http://www.opengroup.org/soa/source-book/soa/soa.htm** for a general definition of SOA and other SOA-related information.

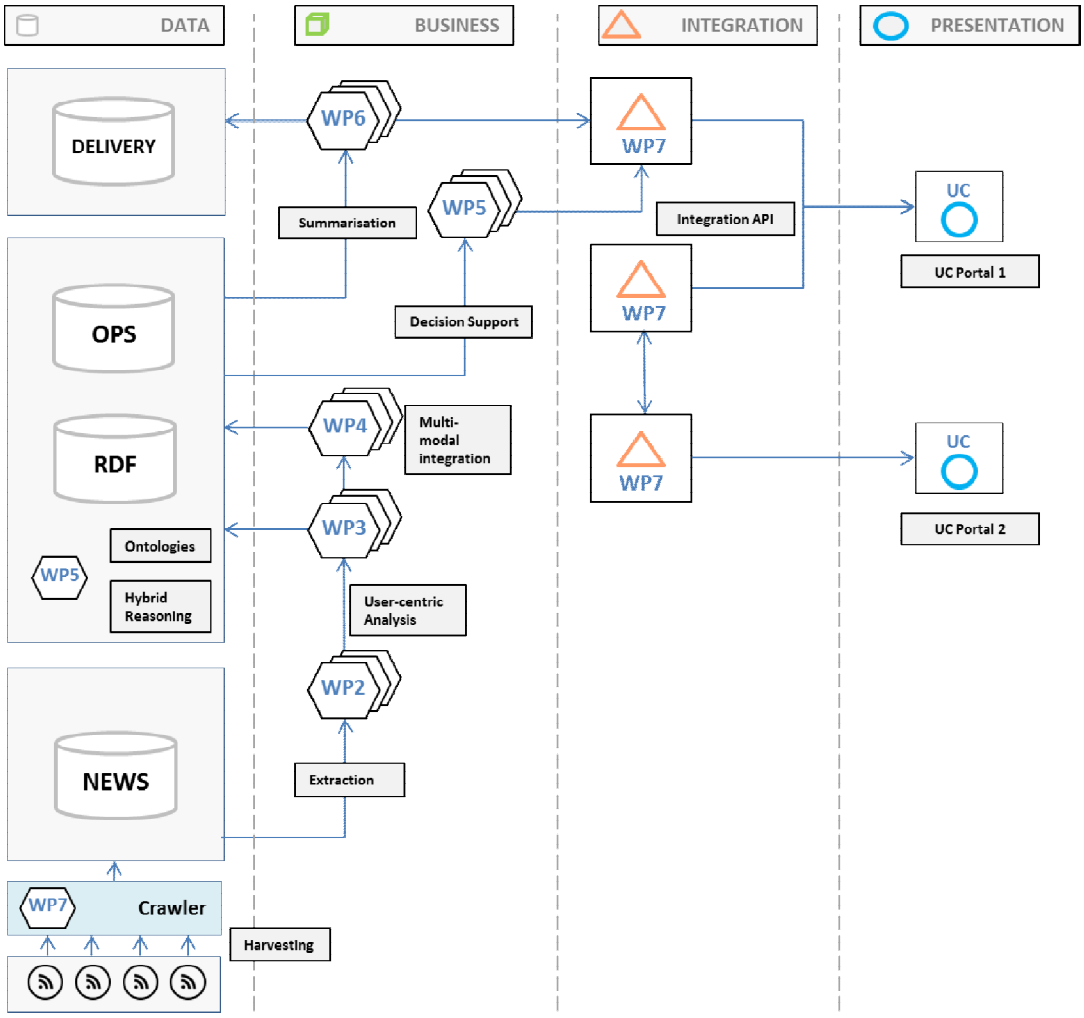

Figure 2: Draft architecture, high-level view

The **REST architectural pattern** will be used to model the services used by the different services. REST proposes a very lightweight, HTTP-based method of stateless operations between resources in the Web, as opposed to more complex, convoluted standards such as SOAP[2]. The platform as a whole will also be engineered as an N-tier architecture, made up of the following layers:

- A **data layer**, containing data repositories and crawlers, and services to query and store the data;
- A **business layer**, containing the services that perform the actual processing;
- An **integration layer**, containing services that provide an API to the portals to access the platform and deliver data in an optimised way to the presentation layer;
- A **presentation layer**, containing web applications that implement the Use Cases.

A high-level view of the global service-oriented architecture of MULTISENSOR platform that extends the initial work-package oriented view is displayed in Figure 2

[2] See **http://www.w3.org/2005/Talks/1115-hh-k-ecows/#(1)** for a discussion of the REST architectural pattern and comparison with other RPC methods.

In this architecture, a periodic process of **content harvesting** (see 3.6.3) is foreseen, which will retrieve source material for the Platform by crawling a set of sources for news, multimedia and social network content. The harvested content will be converted to a common JSON format and stored in a Central News Repository, to be queried and accessed via a service API.

The contents will be retrieved from the Central News Repository (CNR) and run through a distributed pipeline (see 3.1) for **extraction**. The pipeline will include machine translation of content, Named Entity Extraction, and Video and Audio feature detection.

The results from the extraction module will then undergo **user-centric content analysis** (see 3.2) to perform sentiment and context extraction and analysis of social interactions. The aggregated knowledge will then be clustered and classified in topics, **integrated** (see section 3.3) and stored in the Semantic Repository. The Semantic Repository will host all semantic information generated by processing of the source material, plus knowledge bases and ontologies, and provide **inference, reasoning and decision support capabilities** to assist in the implementation of the Use Cases (see 3.4).

Then, the information from the Semantic Repository will be **summarised** and formatted for delivery and stored in the Delivery Repository (see 3.5) according to the user preferences.

Finally, the **integration services** will provide an optimised and homogeneous API to the summarised information and the decision support subsystem for the Use Case demonstration Portals.

2.2 Technology stack

In order to keep the architecture coherent and manageable, and simplify hosting and maintenance, a controlled set of technologies will be used to implement the shared components of the platform (those running in the central infrastructure; see section 3.6.1).

As a general principle, open source technologies will be used to implement the platform ("open source" defined as software licensed under an OSI[3]-approved licence that is included in the list of EUPL-compatible licences[4]). Exceptions will be proprietary or non EUPL-compatible licences for software belonging to a partner in the MULTISENSOR consortium, or specific products, where no EUPL-compatible alternatives exist.

2.2.1 Programming languages

A preliminary list of the selected programming languages for module implementation is provided in Table 1.

Language/framework	Versions allowed	Notes
Java	Sun JVM, 1.7+	Default language for components.

[3] Open Source Initiative, see **http://opensource.org/**.

[4] See **https://joinup.ec.europa.eu/software/page/eupl/eupl-compatible-open-source-licences** for complete list of EUPL-compatible open source licences.

Python	2.7.x	For scripting, and lightweight services.
C/C++	ANSI-C 99 / ISO C++	For performance-critical services and processes. Restrictions: ▪ Code must be 64-bit safe ▪ Code must compile and run on Linux x64, using GCC 4.x and linking against GNU libc (glibc) 6.x
Javascript	Any	node.js 0.10.x
Bash	Any	For small UNIX shell scripts / utilities (Assume Linux x64 host)

Table 1: Programming languages

2.2.2 **Databases/Repositories**

A preliminary list of selected database/storage solutions is provided in Table 2.

Database	Versions allowed	Notes
PostgreSQL	9.x	For work repository: access via JDBC (Java)/node-postgres (Javascript)/psycopg2 (Python)
OWLIM-SE	5.x	For RDF repository: access via Sesame/HTTP+SPARQL
ElasticSearch	1.x	For Central News Repository: access via REST API
MongoDB	2.4.x	For fast, non-relational storage

Table 2: Database and repository packages

2.2.3 **Other technologies**

A preliminary list of miscellaneous technological packages (application servers, special purpose stacks, etc.) is provided in Table 3.

Product	Versions allowed	Notes
Nginx	1.4+	Webserver (fronting app servers, reverse proxy, compression, SSL, static content, etc).
Jetty	9.x	J2EE application server (preferred).
Tomcat	7.x	J2EE application server.
Node.js	0.10+	Javascript application server.
Hadoop	2.2.x	For distributed computing via Map/Reduce jobs.

Table 3: Other technologies

2.2.4 Exchange formats

UTF-8 encoding will be used and enforced throughout the platform to ensure correct and consistent handling of multilingual content.

The preferred exchange format for structured data will be **JSON**[5]. For tools and/or modules, which do not support JSON, or when deemed appropriate, **XML** will also be used.

For semantic data representation and serialisation, the **Turtle** RDF format[6] will be used. **JSON-LD**[7] will be used where appropriate to represent linked data.

For annotations of multimedia content, the MPEG-7 standard[8] will be used.

[5] See **http://json.org/** for formal description of the JSON serialisation format.

[6] See **http://www.w3.org/TeamSubmission/turtle/** for information on the W3C Turtle standard.

[7] See **http://json-ld.org/** for information on the JSON-LD specification.

[8] **See http://mpeg.chiariglione.org/standards/mpeg-7** for information on MPEG-7

3 FUNCTIONAL DESCRIPTION OF MODULES

3.1 Content Extraction Module (WP2)

The content (news and articles) stored in the Central News Repository will be sent through an analysis pipeline to extract and distil important information that can be used to populate the RDF repository. The extraction of content will start from articles in multiple languages and delivered as text, audio or video. Relevant Links to multimedia content in the articles will be followed and the multimedia files will also be sent through the pipeline. The output of the Content Extraction Module will consist of semantic data extracted from the input news and will be stored in the Semantic Repository.

The extraction pipeline will comprise different steps depending on the modality of the media to be analysed, as illustrated in Figure 3. Text sources will be processed by the following components in sequence: Named Entity recognition, dependency parsing and concept extraction. Audio will be converted to text by a Speech recognition component and the resulting text will processed by the aforementioned components. In the case of video, the audio will be separated in an audio stripping step and then treated as an audio source, while the video signal will be processed in a separate pipeline consisting of video segmentation, visual feature extraction and visual concept extraction components. In case an image is retrieved as part of an article this can still be processed by the Video analysis service (as a video of a single keyframe).

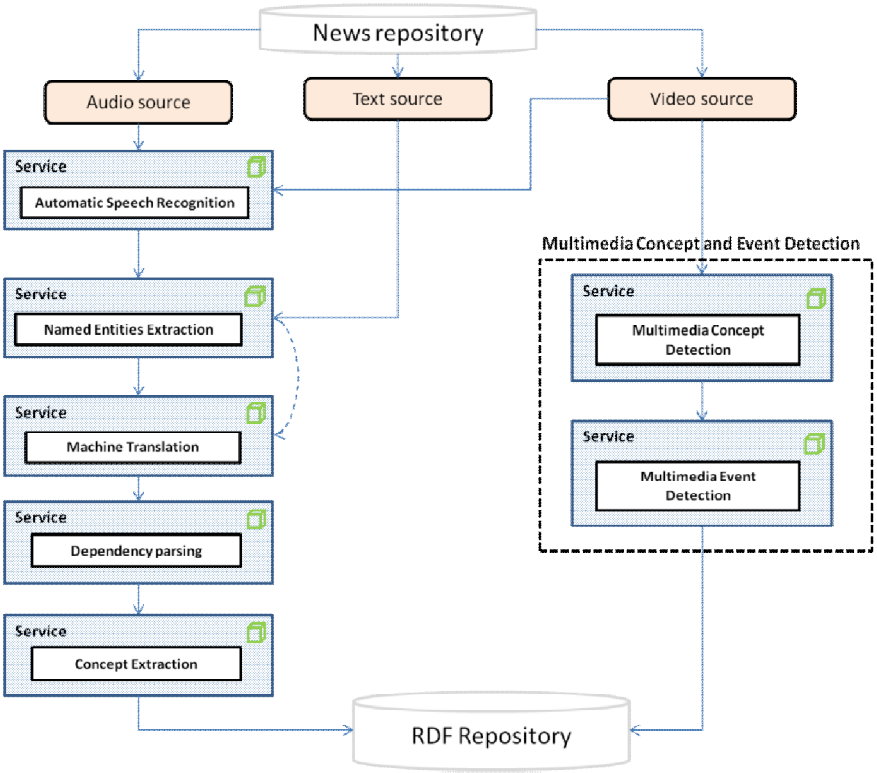

Figure 3: Content Extraction Pipeline

The development of the text processing pipeline foresees a baseline implementation, where texts in languages (other than English) will be translated to English before the dependency parsing and concept extraction steps. A machine translation component will be used for that

purpose. The final implementation of the content extraction module will support multilingual processing of texts in all its steps, making translation unnecessary at this stage.

The activities and components, which are relevant to the development of this module include:

a) The development of a named entity extraction.
b) The development of a component for dependency parsing, and concept and relations extraction from text.
c) Implementation of automatic speech recognition (ASR).
d) Implementation of a multimedia and event detection component.
e) The development of a machine translation (MT) component.

3.1.1 Named entities extraction component

Named Entities Extraction (A.2.1) is a component that recognises and marks some types of proper names (person names, geographical entities, company names etc.) in a text.

The technology is rule-based and it is implemented as a finite-state automaton. As resources it needs the finite state grammars, as well as monolingual lexicons and lists of entities and indicators (Mrs, Dr, ltd, GmbH …). There are special subgrammars for each type of NE, to feed analysis with the right information (semantic type, gender, onset, etc.). The component is language-dependent.

Special challenges for the NER module are the different spelling variants for the same names, and the recognition of partial entities (e.g., only surname). In the context of the MULTISENSOR project the NER results will be used as one of important resources for the improvement loops in the development of machine translation and speech recognition (correct usage and translation, correct pronunciation of named entities). This module will consider the following languages: English, German, Spanish, French, and Bulgarian.

Input	Text with no mark-ups, in plain text format and UTF-8 encoding
Output	Text with NEs marked up, in plain text format and UTF-8 encoding (format will be agreed upon by partners).
Programming languages	Java
Dependencies	N/A
Critical Factors	▪ Recall of the component: there is a risk of lower recognition rate of some very creative names, such as company names, or in cases that the proper names and common nouns cannot be disambiguated (Fischer = surname, Fischer = fisherman). ▪ Precision of the component: there is a risk of insufficient disambiguation of different NER types, if ambiguous (Bayern = region, Bayern = institution). ▪ For the development and testing phase the named entities extraction component will be hosted on Linguatec servers. Depending on future requirements for performance and response time, it may be necessary to move the component to the Amazon EC2 cloud administered by Everis.

Table 4: Named Entities Extraction Component summary

3.1.2 Dependency parsing component

Dependency parsing (A.2.2) consists in analysing texts in terms of the syntactic structure of sentences (i.e., grammatical relations between the different words of a sentence). In this project, we will use *deep* dependency parsing, a recently developed technique, which comprises two steps. First, a *shallow* dependency parsing indicates dependencies between all the words of a sentence. Then, the second step aims at excluding non-meaningful words from dependency structures, disambiguating meaningful words, and substituting grammatical relations by predicate-argument relations. This makes the analysis more "semantic", and hence more suitable for concept extraction. State-of-the-art statistical parsers are available; they are language-independent, and only need to be trained on annotated corpora.

Input	Multilingual texts annotated with the results of applying the Named Entity Recognition.
Output	Texts where each sentence is annotated with its deep dependency-based syntactic/shallow semantic analysis
Programming languages/tools	Mate Tools parser, TALN-UPF parser (Java)
Dependencies	▪ Machine translation component for the baseline implementation
Critical Factors	▪ The quality of a parser largely depends on the quality and size of the training corpus, and on the similarity of the genres of the training and working data. Shallow training data exist for the languages covered in the project, but deep training data will have to be compiled, or a rule-based module to be developed in order to perform the second step of the deep parsing.

Table 5: Dependency parsing component summary

3.1.3 Concept extraction component

Concept extraction (A.2.3) refers to the extraction of information from texts that can be expressed as facts in the semantic repository. It involves identifying bits of information communicated in the text that can be described using the ontologies of the semantic repository. More precisely, the extraction process will identify which entities can be mapped to ontology concepts and which relations between them can be extracted from the text. For the latter task, the concept extraction strategy will make use the results of the deep analysis provided by the dependency parsing.

Input	Multilingual texts annotated with the results of applying the Named Entity recognition and dependency parsing.
Output	A set of RDF facts extracted from the texts via prior dependency parsing and NER.
Programming languages/tools	Java
Dependencies	▪ Named Entity recognition ▪ Dependency parsing

	▪ Ontologies
Critical Factors	▪ Depending on the coverage of the entities recognised in the text, it may be necessary to add a co-reference resolution component to the pipeline. The successful extraction of content communicated in the text will depend on the coverage of the domain ontologies considered for the semantic repository. In other words, only information that can be described using the concepts defined by the ontologies will be extracted.

Table 6: Concept extraction component summary

3.1.4 **Automatic speech recognition component**

The speech recognition component (A.2.4) converts the spoken language from a given audio signal into the written text. This module receives audio files and returns the recognised text. To handle huge multimedia data (audio), this module is supported by file storage and a MySQL database. The development of this component includes first the development of a baseline system trained with general domain data and then the adapted system, which is trained on domain related data.

It is a server-based module with web-based interfaces and the following supporting sub-components, as described hereinafter.

Interface subcomponent. It implements the web-service interfaces in the REST syntax for the Client-Server and Server-Server based communication with other modules. This subcomponent will be based on the web servers Apache and Unicorn in the Rails framework and developed in Ruby.

Audio preprocessor. This subcomponent implements the audio normalisation steps. It is a script-based component that uses ffmpeg library for handling the audio data. The executable is implemented in C++.

Audio segmentation. This subcomponent is responsible for the segmentation of the normalised audio by acoustic features. The job of the audio segmentation step is to try to separate the spoken language from other (non-speech) noise and to isolate silences, as well as to try to identify speaker turns and to group speakers in separated clusters. This module is an executable developed in C++.

Recognition subcomponent. It implements the speech recognition of the audio segments, and the conversion into the written text. The Recognition subcomponent consists of a complex series of different parts, comprising the following main steps, executed by different programmes and scripts:

- First pass recognition – delivers the acoustic- and language-model based recognition of the spoken text from the specified segments. Executable, developed in C++.
- Speaker adaptation – adapts the acoustic features based on the results of the first pass recognition, applied on the identified speaker segments/clusters. Thus, a speaker-independent recognition is enabled (no speaker training necessary). Executable, developed in C++.

- Second pass recognition – delivers the re-computed text recognition for each recognised segment/cluster of segments, according to the speaker adaptation results. Executable, developed in C++.
- Coordination utilities – responsible for handling of audio segments and their representations and encodings between the different recognition steps. Bash and Python Scripts.

The recognition process consumes significant CPU power, and will be supported by a powerful web, storage and Blade server farm, also in order to enable parallel recognition of multiple audio segments with specific acoustic features at the same time.

Linguistic pre- and postprocessors. Different kinds of pre- and post-processors are required for both the training phase and the recognition phase. They implement several normalisation and mapping steps from the spoken language onto the written text during the training phase, and after the recognition. Numeric values, abbreviations, acronyms etc. are expanded for the LM training (e.g. "$125" = "one hundred twenty five dollars"), but then converted back after the recognition ("one hundred twenty five dollars" => "$125"). It deals also with different kind of orthography issues such as capitalisation, spelling variants etc. This component is based on hybrid rule-based and statistic-based linguistic methods. It is implemented in Java and uses different kinds of linguistic resources such as dictionaries, lists, grammars and data bases.

Input	Audio or video file, language parameter (German or English)
Output	The output is retrieved asynchronously: In the first step a file-ID 2) In the second step (by using the file-ID and the GET method): a) Either the recognised text as a plain utf-8 text or b) The CTM file with time steps for each recognised word
Programming languages/tools	C++, Bash, Python, Rails
Dependencies	▪ In order to be able to adapt the ASR for the use case domains, there will be a need to collect (monolingual) corpora for the defined domains. Thus, the ASR quality will depend on the quality and the use-case closeness of the data which will be used for the training.
Critical Factors	▪ The recognition quality might be low for the selected domains, if the training data differs from the testing data. ▪ For the development and testing phase the Automatic Speech Recognition component will be hosted on Linguatec servers. Depending on future requirements for performance and response time, it may be necessary to move the component to the Amazon EC2 cloud administered by Everis.

Table 7: Automatic speech recognition component summary

3.1.5 Multimedia concept and event detection component

This component (A.2.5) deals with the detection of concepts and events found in multimedia files (i.e. video and images) by considering the visual features that are extracted in the current component and the textual features derived from the speech recognition component and it can be further analysed to the following subcomponents:

- Concept detection
- Event detection

Concept detection

This subcomponent deals with the detection of a set of predefined concepts in multimedia files such as video and images and it consists of the following three steps that involve the development of a basic approach and more advanced one of the subcomponent.

Video decoding: this step is applied only in case that the input file is a video and it is responsible for extracting several frames from the video. The frames that will be extracted can be either predefined (i.e. specific number of frames per second), or representative for each shot or for each scene. The last two methods involve applying shot and scene segmentation techniques on the video. The first approach will be applied in the basic version of the component, while the other two will be tested in the advanced version, only in case they do not insert significant complexity (time and computational complexity) during the processing.

Feature extraction: this step refers to the extraction of descriptors that describe visually images by capturing either global or local information out of the images. To deal with global information, MPEG-7 (MPEG, 2014) and HSV (Novak and Shafer, 1992) descriptors are used, while for the local information SIFT (Lowe, 2004) and SURF (Bay, et al. 2006) features are extracted. It should be noted that our experiments conducted within TRECVID workshop revealed that increasing the number of extracted descriptors does not entail improvement in the performance of concept extraction module in total. In the sequel and as far as the descriptors that extract local information is concerned, two approaches are studied for aggregated the local information into a single fixed-length vector representation. The first and basic version of the system relies on the Bag of Words (BoW) representation method for constructing a visual codebook while the advanced version uses the most recent quantisation technique called VLAD (Jégou, et al. 2010). Finally, pyramidal decomposition (Lazebnik, et al. 2006) can be applied in both cases (i.e. BoW and VLAD) in order to further improve the system's performance.

Classification: this step is related to the development of models used for classifying images or video frames to the set of predefined concepts/ categories. The models are developed using supervised learning techniques such as Support Vector Machines (SVM) or Logistic Regression. A set of models will be developed, each one using one of the aforementioned visual descriptors for recognizing a specific concept. Therefore, in the end several models will be developed per concept and their scores (i.e. degree of confidence) should be aggregating in order to provide a single and final score. At this point, it should be noted that in the basic version this final score will result from averaging the scores of each single models, while in the advanced version other ways will be investigated as well. Moreover, at the final version apart from the visual features, models that combine textual information with visual information will be developed. The textual information will be available from the

textual concept extraction module. The methodology that will be used for fusing the different modalities (i.e. visual and textual) is late fusion.

Input	A video or image
Output	Degree of confidence of each concept for the input image or the selected video frames
Programming languages/tools	▪ libsvm (Chang and Lin, 2001) that supports SVM, which is an executable file implemented in C/C++. ▪ Weka (WEKA) which is a suite of machine learning algorithms developed in Java and can be used both as standalone application or JAR libraries. ▪ aceExtract for extracting MPEG-7 descriptors that is an executable file implemented in C/C++. ▪ ffmpeg library for handling multimedia data, which similarly to the previous libraries is also an executable file implemented in C/C++.
Dependencies	▪ Concept detection component uses a meaningful limited set of the classes or instances of the ontologies developed in WP5 as concepts for detection. ▪ Use of concept extraction component (see Section 3.1.4) of WP2 that produces textual concepts ▪ Use of polarity and sentiment (see Section 3.2.2) of WP3.
Critical Factors	▪ High scores in non-compatible concept/ events: The scores indicating the degrees of confidence of the selected set of concepts may be in certain cases contradictory and thus will not allow reaching conclusions on the real content of the image. In this case we will rely only on the best performing classifiers. ▪ Performance: The time-efficiency of the algorithms, which will be developed is another significant issue that might cause slow performance of the system. As a contingency plan we will perform only scene detection or concept extraction for certain keyframes. Depending on future requirements for performance and response time, it may be necessary to move the component to the Amazon EC2 cloud administered by Everis

Table 8: Multimedia concept detection component summary

Event detection

This subcomponent deals with the detection of a set of predefined events in multimedia files such as video and images. In case static visual information is used, event detection can be applied both to images and videos and relies on the results of the concept detection module. Specifically, the results of concept detectors are concatenated to a single feature vector for each frame and then a single feature vector is used for representing the whole video by concatenating (e.g. averaging) the model vectors along all keyframes. The produced feature vector is then forwarded to a model developed using supervised learning techniques like Support Vector Machines for retrieving the degree of confidence of each event for the input file. It should be noted that dimensionality reduction techniques are applied before using the model in order to discard noisy or non-relevant features. This pipeline of processing techniques is part of the basic version of the event detection subcomponent.

On the other hand, in the advanced version, motion information is introduced, which involves the extraction of new motion features that take into consideration the concept of time and can be applied only to videos.

Input	A video or image
Output	Degree of confidence of each event for the input file
Programming languages/tools	▪ libsvm (Chang and Lin, 2001) that supports SVM, which is an executable file implemented in C/C++. ▪ Weka (WEKA) which is a suite of machine learning algorithms developed in Java and can be used both as standalone application or JAR libraries. ▪ aceExtract for extracting MPEG-7 descriptors that is an executable file implemented in C/C++. ▪ ffmpeg library for handling multimedia data, which similarly to the previous libraries is also an executable file implemented in C/C++.
Dependencies	▪ Use of a meaningful limited set of the classes or instances of the ontologies developed in WP5 as set of events. ▪ Similarly to the concept detection module, in the process of detecting events using different modalities it makes use of textual concepts produced from WP2. ▪ Use of concept Extraction component (see Section 3.1.4) of WP2 that produces textual concepts ▪ Use of polarity and sentiment information (see Section 3.2.2) of WP3.
Critical Factors	▪ High scores in non-compatible concept/ events: The scores indicating the degrees of confidence of the selected set of event-related concepts may be in certain cases contradictory and thus will not allow reaching conclusions on the real content of the image. ▪ Performance: The time-efficiency of the algorithms, which will be developed is another significant issue that might cause slow performance of the system.

Table 9: Multimedia event detection component summary

3.1.6 **Machine translation component**

In MULTISENSOR, the machine translation component (A.2.6) will support the content extraction modules and the summarisation task to cope with the multilingual aspects.

It will adopt the statistical approach in automatic machine translation and thus be based on the open source SMT system Moses[9], developed in C++ for UNIX-based operation systems. For the real time translation approach, this module will be optimised and compiled on the runtime server machines. It has several dependencies to packages like boost, zlib, giza++, etc. For language modelling IRSTLM will be used.

[9] More information can be found on the official open source site: **https://github.com/moses-smt/mosesdecoder**

The MT module receives a text to be translated, with the specified translation direction (source and target languages), translates it into the specified target language and passes the translated text back. The development of this component includes first the development of a baseline system trained with general domain data and then the adapted system, which is trained on domain related data.

It is a server-based component with web-based interfaces, as described hereinafter.

Interface subcomponent. It implements the web-service interfaces in the REST syntax for the Client-Server and Server-Server network-based communication with other modules. This component will be based on the web servers Apache and Unicorn in the Rails framework and developed in Ruby.

Preprocessor subcomponent. This subcomponent is responsible for linguistic text preprocessing such as text filtering, input normalisation, sentence segmentation, tokenisation etc. It implements those important encoding and linguistic steps before passing the Translation Server through the XMLRPC-based interface. This component will be developed in Ruby, and follow the REST protocol style.

Distribution subcomponent. This subcomponent implements the communication with the Machine Translation Servers. It enables a fast ('real time') translation and implements the distributed architecture between several translation servers, assigned different translation directions and domains. This component will also be developed in Ruby.

All three modules run in a Ruby on Rails Framework based application and will bundle the following RubyGems: sqlite3, execjs, therubyracer, psych, unicode library, nokogiri, XMLRPC.

Language directions to be developed for MULTISENSOR are: English-German, English-French, English-Spanish, English-Bulgarian, German-English, French-English, Spanish-English, Bulgarian-English. By using English as a pivot language also all possible non-direct translation directions will be supported, e.g. French-(via English)-Spanish.

Input	Plain text in UTF-8 format, and the language-direction parameters (source language, target language).
Output	Plain text in UTF-8 format in the target language
Programming languages/tools	C++, Bash, Python, Rails
Dependencies	▪ In order to to adapt the MT for the use case domains, there will be a need to collect bilingual corpora for the defined domains. Thus, the MT quality will depend on the parallel (translated) data delivered by the project partners and on the amount and quality of the parallel corpora crawled for the project.
Critical Factors	▪ Translation quality: Quality might be low for the selected domains, if the training in-domain data are not available. ▪ For the development and testing phase the Machine Translation component will be hosted on Linguatec servers. Depending on future requirements for performance and response time, it may be necessary to move the component to the Amazon EC2 cloud administered by Everis.

Table 10: Machine Translation component summary

3.1.7 Activity table and workload

Activity	Subactivity	Description	Dependencies	PMs
Named Entities Extraction (A.2.1)	Creation of NER grammars and resources for de, en, fr and es.	Identify named entities to be extracted. Create resources for all languages.	Use case definition and user requirements (T8.1)	7,5
	Creation of resources for Bulgarian	Create resources for Bulgarian.	-	2
Dependency parsing (A.2.2)			Machine translation (baseline only).	4
Concept extraction (A.2.3)	Concept mapping	Map named entities to classes on ontologies.	Named Entity recognition, dependency parsing. WP5 ontologies.	3
	Relation extraction	Identify patterns in deep analysis of texts that can be mapped to relations between individuals in the semantic repository.		6
Automatic Speech Recognition (A.2.4)	Baseline systems	ASR trained on general-domain data	-	4
	Domain-adapted systems	ASR adapted for the use-case domains and tuned by in-domain data	T 7.2 Crawler T 8.1 Content provision	4
Multimedia concept and event detection (A.2.5)	Concept detection	This module will be capable of detecting concepts from video or images using visual information.	Ontologies – WP5 Textual concepts - WP2 Sentiment information - WP3	5
	Event detection	This module will support the detection of events video or images.	Ontologies – WP5 Textual concepts - WP2 Sentiment - WP3	6,5
Machine Translation (A.2.6)	Baseline systems	MT trained on freely available bilingual corpora, (e.g. Europarl, JRC) which are general-domain data	-	4
	Domain-adapted systems	MT adapted for the use-case domains and trained and tuned by in-domain data	T 7.2 Crawler T 8.1 content provision	4

Table 11: WP2 activity table

3.1.8 Timeline and dependency from other modules

ACTIVITY	Y1											Y2												Y3											
WP2						D2.1					D2.2											D2.3										D2.4				
A.2.1 Named entities Extraction																																			
Creation of NER grammars and resources for de, en, fr and es.																																			
Creation of resources for Bulgarian																																			
A.2.2 Dependency Parsing																																			
A.2.3 Concept extraction																																			
Concept mapping																																			
Relation extraction																																			
A.2.4 Automatic Speech recognition																																			
Baseline system																																			
Domain-adapted system																																			
A.2.5 Multimedia concept and event detection																																			
Concept detection																																			
Event detection																																			
A.2.6 Machine translation																																			
Baseline system																																			
Domain-adapted system																																			

Table 12: WP2 timeline

3.2 User and context-centric content analysis (WP3)

The user and context-centric content analysis component consists of three components, the context extraction and representation component, the sentiment extraction component, and the component for information propagation and social network analysis.

In Figure 4, we depict how the different components interact with each other. As shown, the sentiment extraction component can process textual content provided by the other components and return the appropriate sentiment annotations. Apart from the former interaction, the components are considered independent from each other.

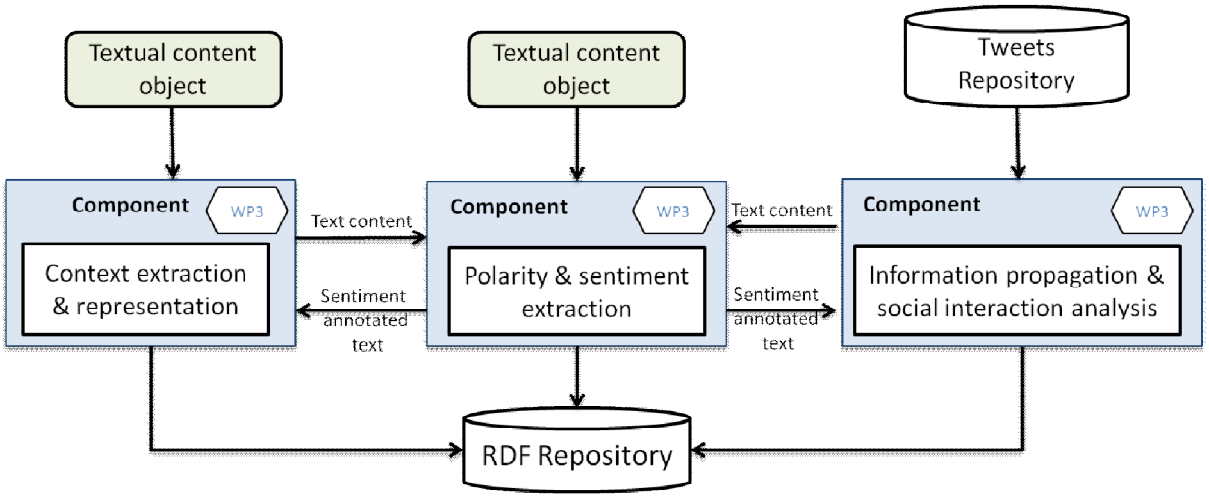

Figure 4: User and context-centric analysis modules

3.2.1 Context extraction and representation component

To effectively understand, interpret or even summarise textual content (e.g., a news article or a blog), one should extract its context and the various attributes that comprise it. We use the term *context* to refer to a set of features that characterise the input and we use these features for representation purposes. This representation will enable context-based search and can also be used for summarisation purposes.

The component will take as input textual content, corresponding for example to a news article or a blog, along with some metadata. This content will have first been processed by the content extraction component (see Section 2), potentially identifying named entities, relations between them, and also performing a deep dependency-based syntactic analysis. The goal of the context-modelling component (A.3.1) will be to gather, extract and represent the contextual features. The main focus will be on the news and finance domain. As a final step, for representation purposes, the contextual features will be incorporated into an ontology and stored in the RDF repository.

The methods employed will be language dependent focusing on German, French, English, Spanish, and Bulgarian. If a subset of these languages cannot be directly supported by the component, we will depend on the translations provided by the machine translation component (described in Sec. 4.1.6).

Next, we describe the main subcomponents performed by the context-modelling component.

Contextual feature discovery and extraction: This subcomponent will gather and extract contextual features. We first consider features that can be directly derived by the metadata associated with the input content. Such features include for example, the publication date, the country of origin or some author information. Secondly, we consider features that require additional resources to be learned like the importance of a news article or its topic. The importance of an article could be derived for example by the total occurrences of this article in some social network content (e.g., total tweets sharing the link of an article). Additional feature engineering will take place exploiting more advanced methods like text mining. For example, such features can include frequent phrases (n-grams or sequences of words) appearing in the news article (e.g., "financial crisis in Europe") and characterizing its topic in a fine-grained way. The final selection on the set of contextual features will depend also on the use case requirements from the partners. The properties of the subcomponent responsible for this task are as follows.

Input	A textual content item annotated with disambiguated named entities, a deep dependency-based syntactic analysis, and extracted concepts (provided by the content extraction module, see Section 3.1). We may also consider sentiment-related annotations of the input text if these have been already produced and are being considered as important contextual features. Any additional metadata associated with the input content will also be considered part of the input.
Output	The output consists of a set of contextual features and their values
Programming languages	Use of available NLP tools and implementation of additional functionalities in Java
Dependencies	▪ Content extraction component (NER, dependency parsing, concept extraction) ▪ Topic-based modelling component (this is a potential dependency) ▪ Sentiment extraction component (this is a potential dependency)
Critical Factors	N/A

Table 13: Contextual feature discovery and extraction subcomponent summary

Contextual assessment: After selecting our contextual features, we will employ a graph-based modelling approach for studying how they correlate with each other and assess also their importance. To assess whether a factor is important we will need to define a set of properties that a context factor should satisfy for being considered important. One such property may be its discriminative power allowing it to distinguish among the different contents. The properties of the subcomponent responsible for the assessment task are as follows.

Input	A set of contextual features and their values (this can include also previously processed content items or a test collection of content items).

Output	An assessment of the importance of the different contextual features and a graphical representation of how the different features are associated with each other.
Programming languages	Java
Dependencies	N/A
Critical Factors	▪ A possible risk has to do with the requirement of tests collections (i.e., the training sets from which we will learn the importance of contextual factors and how they correlate with each other). This can be mitigated both by the collections supplied by user partners and by the publicly available test collections.

Table 14: Contextual assessment subcomponent summary

Context representation: As a final step, we will incorporate the extracted contextual factors into an ontology in the form of a contextual taxonomy. This will allow for a standardised representation and facilitate context-based search. The properties of the subcomponent responsible for the representation task are as follows.

Input	A set of contextual features and their values
Output	An ontological representation of the contextual features. This instance will be represented using the RDF data model and stored in the RDF repository.
Programming languages	N/A
Dependencies	▪ Ontologies
Critical Factors	N/A

Table 15: Contextual representation subcomponent summary

3.2.2 Polarity and sentiment extraction

Sentiment analysis (A.3.2) involves two main subcomponents: sentimentality (also known as subjectivity) detection and polarity detection. The former task deals with classifying text as either subjective or objective. The latter task, also known as sentiment classification, tries to classify text as either having positive or negative sentiments. Our aim is to implement an accurate, learnt, sentiment analysis tool that will identify opinions and dependencies between opinion holders in news, multimedia, and social network content. Our sentiment analysis system will take as input a collection of web pages and a set of entities. As output it will produce a sentiment summary (numeric sentiment scores) for each entity in a given set of entities, as well as for the text itself at sentence- or document- level. The system will be built as a pipeline consisting of three components (see Figure 5) described below.

Text extractor: The text extraction subcomponent will read the parsed HTML content and split the extracted text into sentences. This step takes place during the crawling phase (see Section 3.6.2). Considering that web pages tend to contain a lot of clutter around the page's main content (e.g., navigational elements, templates, and advertisements), we plan to isolate the meaningful portions of text by removing this so-called boilerplate. To this end, we will use Boilerpipe (Kohlsütter 2010), a classifier that identifies HTML document blocks as

either content or boilerplate. The classifier is based on some simple shallow text features such as word counts in HTML blocks, and its performance matches more complex strategies. Considering the scale of our data, the use of Boilerpipe fits well to our purposes, as the required processing time is negligible. Another possibility is Goose[10], an article extractor written in Java that can extract the main body and meta-data of a news article or article type web page.

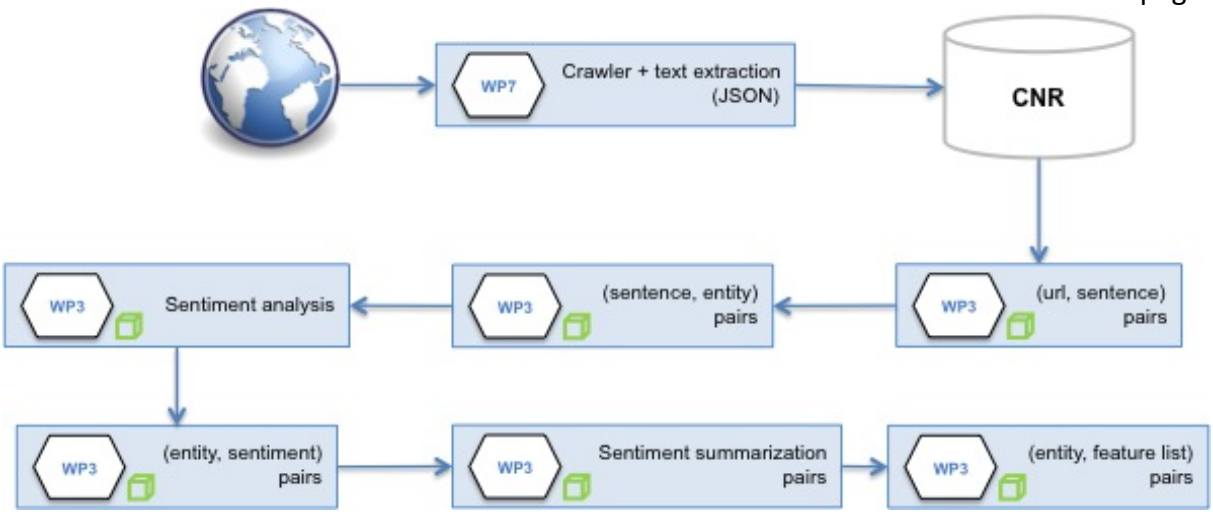

Figure 5: Sentiment analysis architecture

Sentiment analyzer: The sentiment analyzer will be responsible for detecting sentiments and their strengths in English text. For a given piece of short text, the SentiStrength tool will generate a positive sentiment score (integers in the range from +1 [neutral] to +5 [extremely positive]) and a negative sentiment score (integers in the range from -1 [neutral] to -5 [extremely negative]) for each word in the text. Finally, it will compute the sentimentality of the text by summing the absolute values of the positive and negative scores, and subtracting the value of 2 to scale the final score between 0 – 8 as follows:

$$|score_{pos}| + |score_{neg}| - 2 = score_{sent}$$

For the implementation of the sentiment analyzer we intend to explore the following classification approaches: (i) domain-specific classifier, (ii) all-in-one classifier, and (iii) hybrid classifier (e.g., 70% domain-specific, 30% all-in-one classifier ensemble). The framework and pipeline will be implemented in Java. For shallow linguistic processing (e.g., tokenisation) we will use Mallet[11], a Java-based package for statistical natural language processing, document classification, topic modeling, and information extraction. In addition, Mallet will be used for machine learning applications to text, using a range of supervised learning algorithms such as Naïve Bayes, Maximum Entropy, and Support Vector Machines (SVM).

Sentiment summariser: Finally, sentiment summaries will be generated per entity and per document, or for the whole collection. To this end, we will consider a wide range of features, such as the number of sentences that contain the entity, the number of sentences that contain no sentiment, or contain positive or negative polarity, word-level features, the

[10] https://github.com/GravityLabs/goose

[11] http://mallet.cs.umass.edu

number of words with positive or negative sentiments, and features that take into account the strength of sentiments associated with the words in the text (Table 2).

The basic properties of the sentiment extraction subcomponent are as follows.

Input	Text annotated with disambiguated named entities and a dependency-based syntactic analysis.
Output	Sentiments: (i) about entities mentioned in the web content, i.e. sentiment summaries, or (ii) the text itself on a sentence- or document-level, i.e., text labelled with sentiment-related features
Programming languages/tools	Java, Hadoop, Mallet
Dependencies	▪ Named entity recognition component ▪ Machine translation component ▪ Text extraction (html content parsing) at crawling time.
Critical Factors	▪ A potential risk concerns scarceness of data, which can be mitigated by the data sources that will be made available by the relevant partners. ▪ Because of the hardness of the sentiment extraction problem especially when this occurs at a fine-grained level there is a trade-off between the quality of the sentiment extraction methods and their efficiency. ▪ Dealing with large amounts of data may cause scalability problems.

Table 16: Sentiment extraction component summary

3.2.3 Information propagation and social interaction analysis

The spread of information in a society does not happen all at once, but it is rather a complex process that, by exploiting channels of communication among individuals, starts from few adopters and propagates gradually reaching eventually the mass market. The phenomenon of information propagation is strictly tied with the structure of the underlying network. In fact, if the network has a modular structure, with communities that are densely connected internally, and loosely connected externally, the propagation of information acts mainly locally inside each community.

The goal of this component is to analyse social interaction data and information propagation cascades (A.3.3) to detect relevant communities, interesting patterns in the propagation flow and influential users.

The component will retrieve social interaction data, in the form of tweets, from the **Tweets Repository**. Those tweets will be collected (see 3.6.3) by querying the Twitter public API[12] for hashtags/concepts/person of interest and stored in the standard JSON format and the central repository will be periodically fetched into the central repository. The collection of tweets constitutes the only input to this component and enables the subsequent analysis, which will be performed by the subcomponents described below and shown in Figure 6.

[12] See **https://dev.twitter.com/docs/streaming-apis**

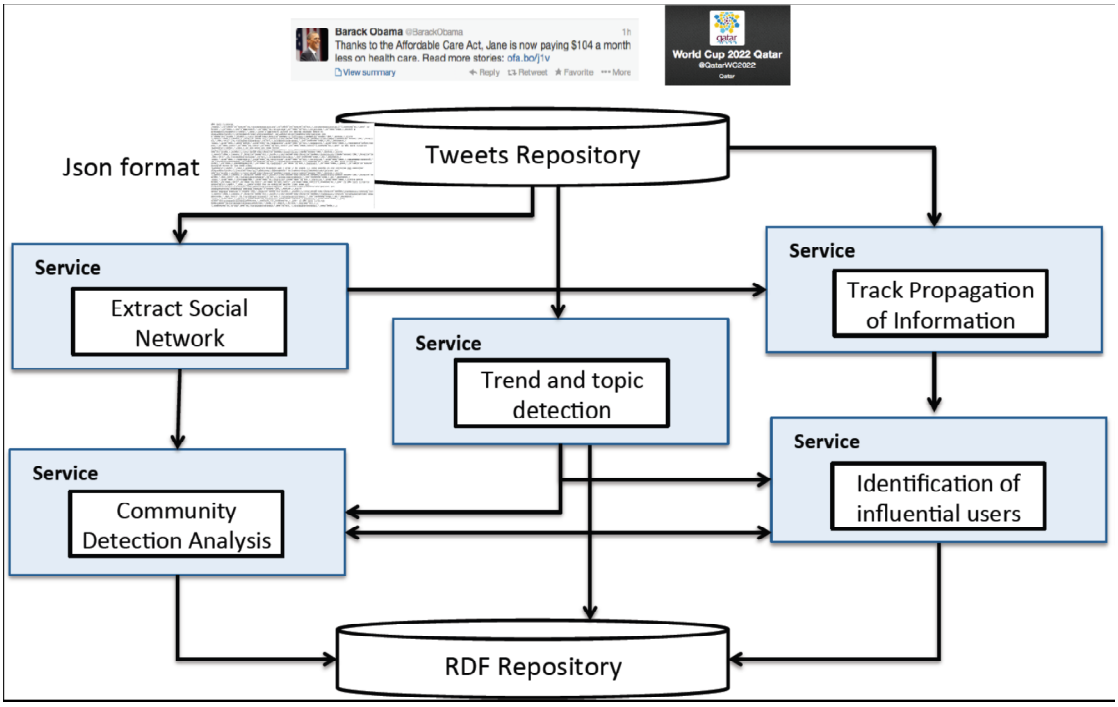

Figure 6: Information propagation and social interaction – subcomponents and dependencies

Social network extraction: Since the underlying social network is not directly available, this subcomponent will process tweets and reconstruct the network by using mentions (@) and retweets. This subcomponent will hence receive as input tweets and provide as output a directed weighted graph, where the direction of each link represents the directionality of the flow of information and the weight measures its strength. The properties of this subcomponent are as follows.

Input	Tweets
Output	Social network G=(V,A) and information about users (frequently used hashtags)
Programming languages/tools	Java
Dependencies	N/A
Critical Factors	▪ The network reconstruction step could produce a low quality approximation of the real network, which however is not available. Since the network is a fundamental input for all the other subcomponents, we will focus on measuring statistical properties (sparsity, number of connected components, clustering coefficient) of the reconstructed network and assess its suitability.

Table 17: Social network extraction subcomponent summary

Trend and topic detection: This subcomponent will analyse the textual stream of data provided by tweets and detect significant topics of conversations. Typically, those topics in Twitter can be represented as a small set of hashtags. Frequent pattern mining and

statistical topic detection techniques will be used to exploit co-occurrence of hashtags for the identification of topics. The properties of this subcomponent are as follows.

Input	Tweets
Output	Trending hashtags in a given period of time
Programming languages/tools	Java
Dependencies	N/A
Critical Factors	N/A

Table 18: Trend and topic detection subcomponent summary

Tracking of information propagation: In the context of tweet data, information is typically represented in the form of URLs. The goal of this subcomponent is to analyse the data, detect URLs and track their propagation on the network identified in the previous step. The output will be a time stamped log that records at which time each user consumed each item (URL).

Input	Tweets, social network
Output	Log (User, Information, Time)
Programming languages/tools	Java
Dependencies	N/A
Critical Factors	N/A

Table 19: Tracking of information propagation subcomponent summary

Identification of influential users: Cascade logs produced in the previous step will be further analysed to detect influential users and the strength of their influence on peers. The output of this subcomponent will be a directed weighted graph, where the direction is preserved with respect to the original network and each weight on the directed arc A→B measure the influence of A on B.

Input	Tweets, social network
Output	Tweets, Social Networks, Log (user, information, time)
Programming languages/tools	Java
Dependencies	N/A
Critical Factors	▪ Algorithms to detect influential users and the strength of peer wise influence relationships are typically compute-intensive. This may prohibit the application of these techniques to very large propagation logs.

Table 20: Influencer detection subcomponent summary

Community detection: This subcomponent will analyse social relationships and information cascade logs to identify relevant communities (either social or topical) and the role (authority/susceptible) of each user within these communities.

Input	Social Network
Output	Overlapping groups of users and their description
Programming languages/tools	Java
Dependencies	Sentiment detection and opinion mining tools could be useful in the task of community detection, where each community will be in this case represented by highly connected peers who tend to express the same sentiment on same topics.
Critical Factors	N/A

Table 21: Community detection subcomponent summary

3.2.4 Activity table and workload

Activity	Subactivity	Description	Dependencies	PMs
Context modelling and representation (A.3.1)	Contextual feature discovery and extraction	It gathers and extracts contextual features that can be directly derived by the metadata, or that require additional resources to be learned from a news article or its topic or that require more advanced methods like text mining.	2.3, 2.7	5
	Contextual assessment	It is responsible for assessing the importance and correlation of the discovered features discovered.		4
	Context representation	It incorporates the extracted contextual factors into an ontology in the form of a contextual taxonomy.	5.1	1
Polarity and sentiment extraction (A.3.2)	Text extractor	It involves splitting the text extracted during the crawling phase into sentences and isolating the meaningful portions of text.		1
	Sentiment analyser	It is responsible for detecting sentiments and their strengths in English text.	2.7	6
	Sentiment summariser	It involves the generation of sentiment summaries per entity and per document, or for the whole collection.	2.2	3
Information propagation and social interaction	Extract social network	It will process tweets and reconstruct the network by		1

analysis (A.3.3)	extractor	using mentions and retweets and produce a directed graph showing the direction of information.		
	Trend and topic detection	It will analyse the textual stream of data provided by tweets and detect significant topics of conversations.		2
	Tracking of information propagation	It is responsible for detecting URLs and tracking their propagation on the network.		3
	Identification of influential users	It will use the produced cascade logs to detect influential users and the strength of their influence on peers.		1
	Community detection	It will analyse social relationships and information cascade logs to identify relevant communities and the role of users inside these communities.		4

Table 22: WP3 activity table

3.2.5 Timeline and dependency from other modules

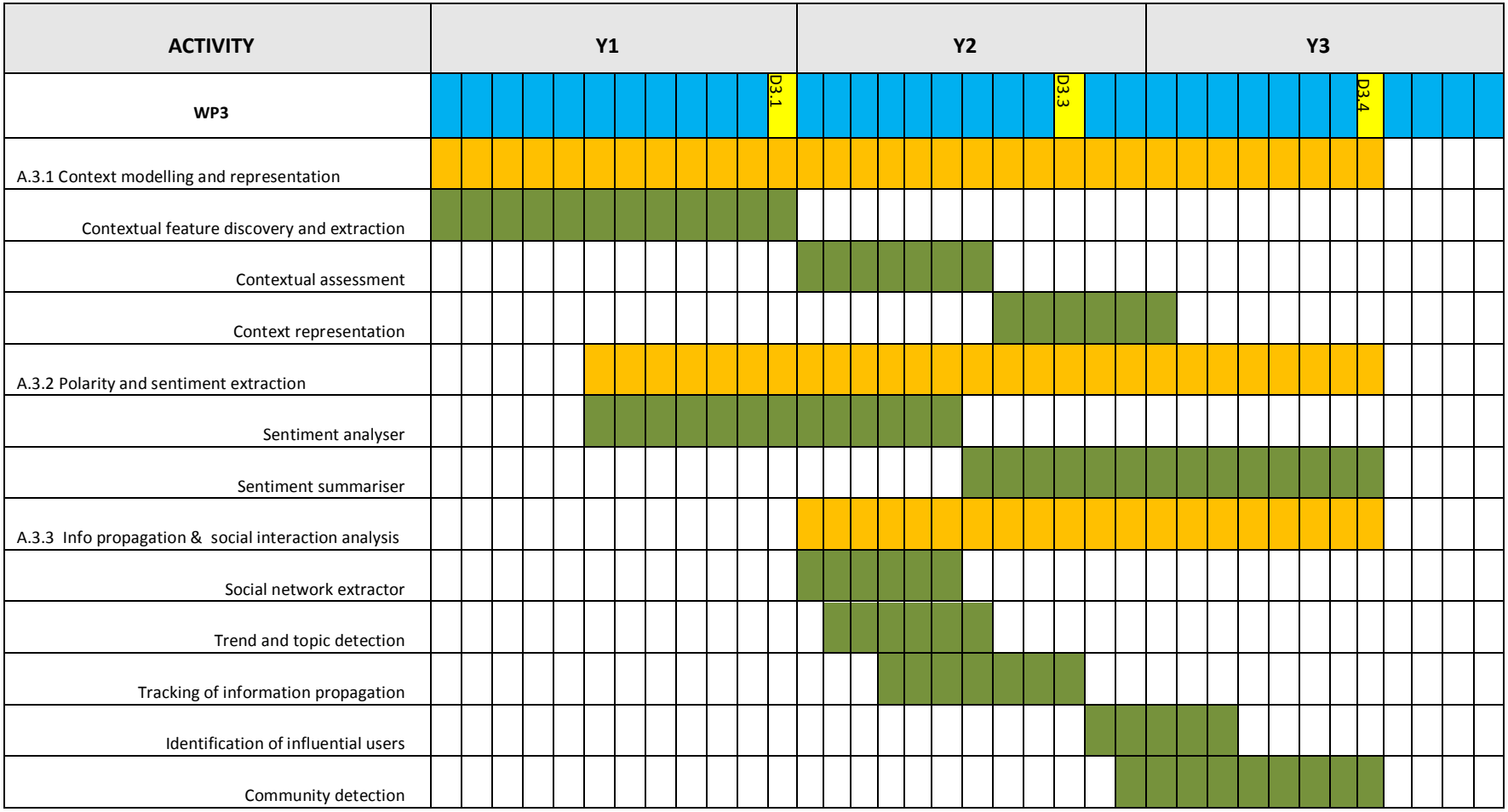

Table 23: WP3 timeline

3.3 Multidimensional content integration (WP4)

This module includes two different and independent sub-modules. The first (see Figure 7) is related to the development of an indexing structure for storing and searching over the dataset holding all the extracted multimodal information such as textual, visual and contextual. The same information is used for topic-based modelling that involves topic-based classification and topic event detection.

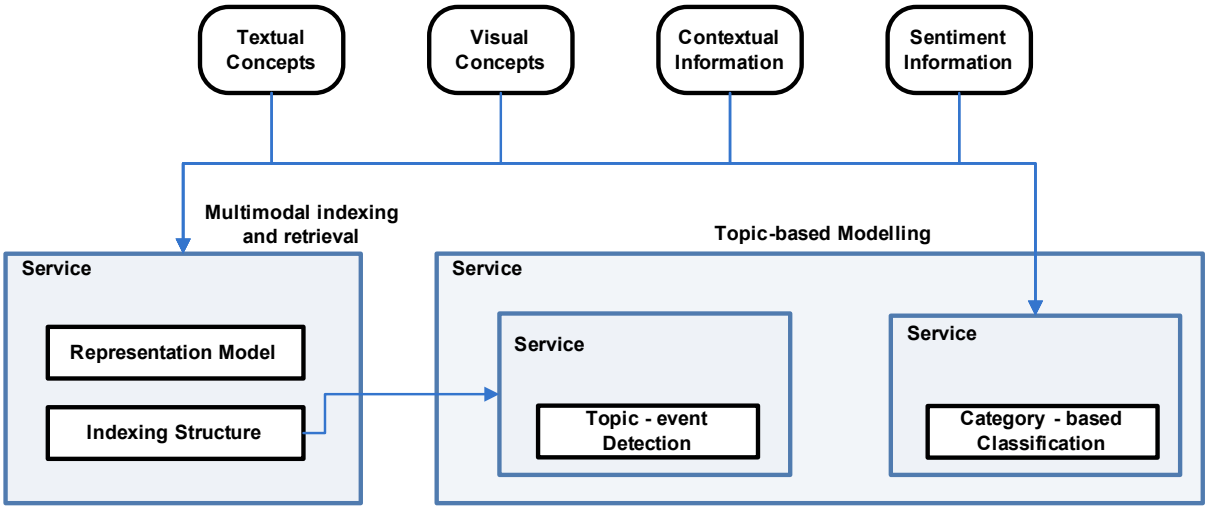

Figure 7: Multidimensional content integration

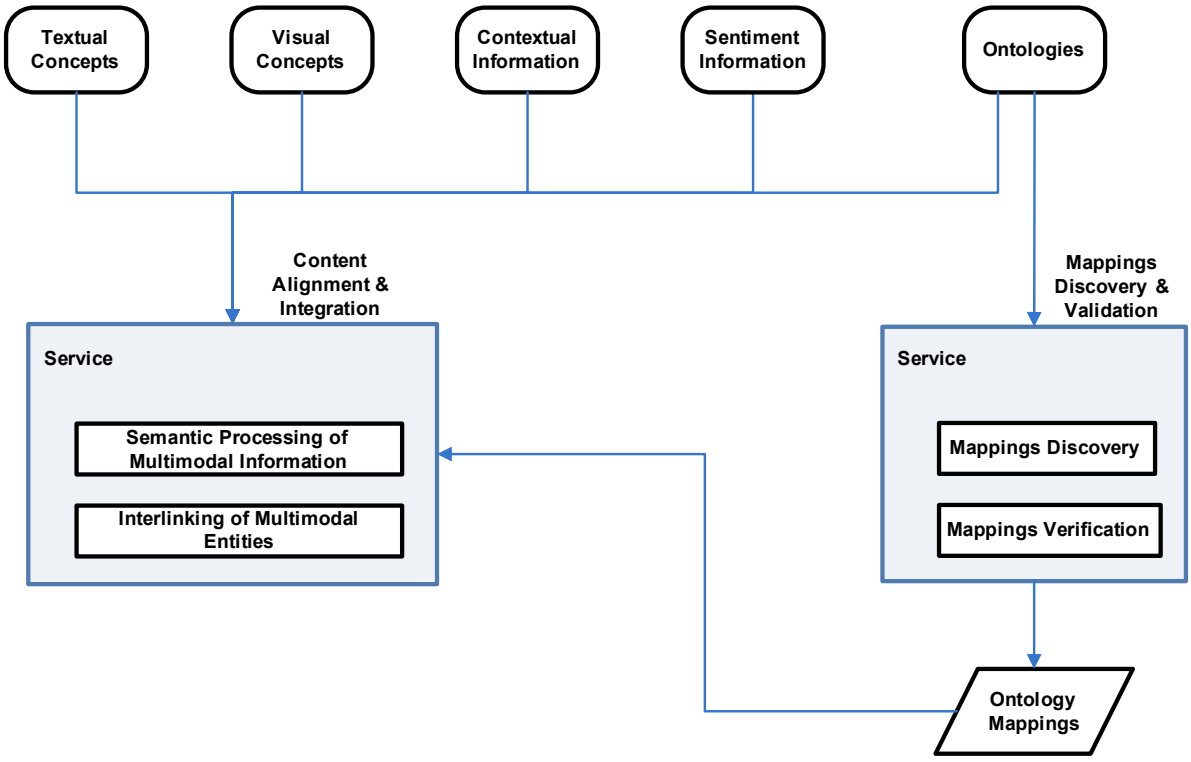

Figure 8: Content alignment process

The second sub-module (see Figure 8) involves content alignment and semantic content processing components. Content alignment refers to the discovery of ontology mappings from the ontologies and their verification, while content integration refers to the alignment between semantic entities.

The activities and components, which are relevant to the development of this module include:

a) The development of a multimodal indexing and retrieval component (T4.4)
b) The development of a topic-based modelling component (T4.1)
c) The implementation of a mapping discovery and validation component (T4.2)
d) The implementation of a content alignment and integration component (T4.3)

3.3.1 Multimodal indexing and retrieval component

This component (A.4.4) deals with the development of a structure that holds the multimodal information produced during the processing of the data found in the News Repository and it can be analysed to the following subcomponents:

Model development: In this subcomponent a representation for holding several dimensions (i.e. textual, visual, contextual, sentiment, location and time) of the multimedia information is specified. This representation model will be in the form of an XML file holding all the aforementioned multimodal data. The model that will be developed draws upon several existing models and combines them in order to achieve a more complete description of an object. The models that will be used as basis are: RUCoD (Daras, et al. 2010), WebLab Exchange Model (WebLab) and MPEG-7 model (MPEG7).

Indexing structure: An indexing structure for holding and retrieving efficiently the multimodal entities of multimedia information will be developed. It should be noted that each modality will be treated differently during the indexing and retrieval procedure, that is different retrieval methods will be used. Some techniques that can be used for indexing are: inverted indexes, hash indexes, r-trees and b-trees.

Input	Textual, visual concepts, name entities, sentiment, polarity and context information.
Output	A Representation model in xml format holding multimedia information and indexing module for retrieving and comparing efficiently the stored models
Programming languages/tools	▪ XML ▪ Open source database management system (e.g. MySQL) ▪ Existing representation models such as RUCoD, WebLab Exchange Model and MPEG-7 model ▪ Indexing structures such as inverted indexes, hash indexes, r-trees and b-trees
Dependencies	▪ Multimodal indexing and retrieval component uses the multimodal features created in WP2 (textual, visual concepts, name entities) ▪ Sentiment, polarity and context information produced in WP3
Critical Factors	▪ A potential risk concerns the development of a complex representation due to the capturing of several modalities (i.e.

	text, visual, contextual and sentiment information) which might eventually slow down the retrieval performance

Table 24: Multimodal indexing and retrieval component summary

3.3.2 Topic-based modelling component

This component (A.4.1) deals with the classification into general categories of the content retrieved by the News Repository by using the multimodal (i.e. textual and visual concepts, events, contextual and sentiment information) information produced by other components (WP2, WP3) and stored in the system and the detection of specific topics inside these categories. This component can be further analysed to the following subcomponents:

- Category-based classification
- Topic-event detection and tracking

Category-based classification

This subcomponent deals with the classification of News Items retrieved from the News Repository into generic categories and it involves the following steps:

- Topics identification: The general categories that will be handled, will be defined by the end-users experts but they will be rather generic, e.g. politics, economics, sports and social news
- Classification. This steps involves the following activities:
  - Construction of a training set: Gathering of multimodal data (i.e. video, images, text) that cover each one of the generic categories in order to form the training set for each category
  - Model Development/ Training: The model developed will use the multimodal features extracted in other WPs (WP2 and WP3) and the techniques that will be applied are supervised learning techniques such as Support Vector Machines (SVM) and Random Forests. It should be noted that the news items vectors will be able to store the semantic relations among the items or keywords in order to improve the classification results. Another type of classification that will be used is rule-based classification that constructs rules from the training set and based on them a decision is made on the relevance of new/ unseen news items.
  - Model Testing: During this activity, unseen data are classified to the specified categories using the models that were earlier developed.
- Fusion of the results of the multimodal and rule-based models into a new model (i.e. late fusion) for retrieving improved results.

Input	Multimodal features existing in the indexing structure developed in T4.4
Output	The degree of confidence of each category for each News Item
Programming languages/tools	▪ Java ▪ libsvm (Chang and Lin, 2001) that supports SVM, which is an executable file implemented in C/C++ ▪ Weka (WEKA), which is a suite of machine learning algorithms developed in java and can be used both as standalone

	application or jar libraries
Dependencies	▪ Multimodal features produced in WP2 (textual and visual concepts, name entities) and WP3 (sentiment, polarity and contextual information) ▪ Ontologies developed in WP5 for realizing the rule-based classification.
Critical Factors	▪ The diverse nature of the input data and content might create a problem during the topic detection procedure.

Table 25: Category-based classification subcomponent summary

Topic-event detection and tracking

This subcomponent deals with tracking topically related material in streams of data (e.g. broadcast news), as well as in capturing of events (Papadopoulos, et al. 2010) in social media streams and uses the features stored in the indexing structure of Task 4.4. Recent experiments revealed that applying topic event detection techniques (e.g. clustering) on large datasets is not considered very efficient (Petkos, et al. 2012) compared to applying in the beginning a first rough filtering. Therefore, the steps involved in the implementation of this subcomponent are the following:

- Category related data: it involves the collection of a set of news items that belong to the same generic category (output of previous subcomponent).
- Identification of stories or first filtering: it involves a first identification of the topics/stories (e.g. from RSS feeds) or a first filtering of the set of news items in order to retrieve a more coherent set of data
- Clustering: it involves the application of clustering techniques for finding very specific topics found inside news items.

At this point, it should be noted that all the algorithms developed in this activity will be designed on Hadoop and executed on the cloud for scalability reasons.

Input	Multimodal features existing in the indexing structure developed in T4.4
Output	Set of clusters containing similar News Items
Programming languages/tools	▪ Java ▪ Hadoop
Dependencies	▪ Multimodal features produced in WP2 (textual and visual concepts, name entities) and WP3 (sentiment, polarity and contextual information) ▪ Ontologies developed in WP5 for realizing the rule-based classification.
Critical Factors	▪ The diverse nature of the input data and content might create a problem during the topic detection procedure.

Table 26: Topic event detection and tracking subcomponent summary

3.3.3 Mapping discovery and validation component

Mapping discovery (A.4.2) refers to the methodology that will be developed in order to provide correspondences between ontological entities, such as classes and properties, between the different ontologies that will be used in MULTISENSOR for semantic content representation. Validation refers to the process of verifying that a specific mapping between two different ontologies and their concepts is true. This process will take into account the most probable matches generated by the mapping process and verify their validity. The mapping with the highest validation score will be selected. The state of the art in ontology mapping will be investigated and a number of existing ontology matching approaches will be benchmarked with MULTISENSOR ontologies. The most appropriate of those will be selected and extended in order to further enhance their performance according to the project's requirements.

The following subcomponents are identified for the completion of the component:

(i) Terminological and semantic analysis of ontological concepts subcomponent
(ii) Component for the creation of prioritised matching sets between ontologies
(iii) Automated matching set validation and selection subcomponent

Input	WP5 ontologies
Output	Domain independent ontology alignment framework
Programming languages/tools	▪ Java ▪ Ontology alignment frameworks, such as AlignmentAPI or S-Match
Dependencies	▪ Strong dependency on the ontologies developed in WP5, since e ontology mapping methods are in general dependent on the actual ontologies that they will be applied to ▪ Dependent on the reasoning infrastructure of WP5.
Critical Factors	▪ Incomplete ontology definitions: The ontologies that will be developed to semantically represent the UCs content should be defined with enough detail and depth so that the mapping process can perform a meaningful analysis.

Table 27: Mapping discovery and validation component summary

3.3.4 Content alignment and integration component

The content alignment and integration component (A.4.3) will provide an additional level of alignment between semantic entities by introducing semantic processing in the ontological mapping framework. The component will provide an additional layer of integration where it will be employed in the content level. Content alignment will be based on the ontologies and the discovered mappings between them and perform reasoning on the content itself to further refine and identify inconsistencies between entities and identify new linkage of related entities, especially between different modalities, and linkage based on content spatiotemporal features. The refinement will be based on developed knowledge structures where the appropriate alignment conditions will be defined.

The following subcomponents have been identified in this activity:

(i) Content alignment definitions repository
(ii) Identification of content alignment inconsistencies subcomponent
(iii) Identification of cross-modal linkage subcomponent
(iv) Spatiotemporal analysis and reasoning subcomponent

Input	WP5 ontologies, extracted content from WP3 and WP2
Output	The output involves identifying inconsistencies between entities as well as cross-modal linkage between entities. Add property relations between them
Programming languages/tools	▪ Java ▪ Spatial reasoning framework, such as the one offered by OWLIM ▪ Temporal reasoning support e.g. using an appropriate time ontology
Dependencies	▪ Depends on the ontologies and reasoning infrastructure developed in WP5 ▪ Depends on WP2, WP3 for content extraction
Critical Factors	▪ Insufficient knowledge for content alignment requirements and validation: The knowledge structures for content alignment depend on the use cases that will be defined. The use cases should be well formed in order to identify the requirements for content alignment but also they should allow for effective validation of the latter.

Table 28: Content alignment and integration component summary

3.3.5 **Activity table and workload**

Activity	Subactivity	Description	Dependencies	PM
Multimodal Indexing and Retrieval (A.4.4)	Model development	It supports the development of representation model for multimodal data	Visual and textual concepts - WP2 Sentiment and contextual information - WP3	4,5
	Indexing structure	It supports the development of indexing structure for storing and retrieving the multimodal data	Visual and textual concepts - WP2 Sentiment and contextual information - WP3	5,5
Topic-based Modelling (A.4.1)	Category-based Classification	It supports the classification of News Items into categories	Visual and textual concepts - WP2 Sentiment and contextual information - WP3 Indexing structure - WP4 (T4.4)	7,5
	Topic-event detection	It supports the detection of topics	Visual and textual concepts - WP2 Sentiment and contextual information - WP3	8,5

			Indexing structure - WP4 (T4.4)	
Mapping Discovery and Validation (A.4.2)	Analysis of ontological concepts	It involves the terminological and semantic analysis of ontological concepts	Ontologies and reasoning infrastructure - WP5	4,5
	Matching sets	It involves the creation of prioritised matching sets between ontologies	Ontologies and reasoning infrastructure - WP5	3
	Validation	It involves the automated matching set validation and selection	Ontologies and reasoning infrastructure - WP5	2,5
Content alignment and Integration (A.4.3)	Content alignment definitions	It involves the creation of a repository where the content alignment definitions will be stored	Ontologies – WP2 Content extraction – WP2 and WP3	1,5
	Inconsistencies identification	It supports the identification of the content alignment inconsistencies	Ontologies and reasoning infrastructure – WP2 Content extraction – WP2 and WP3	2,5
	Cross-modal linkage	It supports the identification of cross-modal linkage	Ontologies and reasoning infrastructure – WP2 Content extraction – WP2 and WP3	3
	Spatiotemporal analysis and reasoning	It supports Spatiotemporal analysis and reasoning submodule for identifying content similarities	Ontologies and reasoning infrastructure – WP2 Content extraction – WP2 and WP3	3

Table 29: WP4 activity table

3.3.6 Timeline and dependency from other modules

ACTIVITY	Y1											Y2											Y3											
WP4											D4.1							D4.2							D4.3				D4.4					
A.4.1 Topic-based modelling																																		
Category-based classification																																		
Topic-event detection																																		
A.4.2 Mapping discovery & validation																																		
Ontological concepts analysis																																		
Matching sets																																		
Validation																																		
A.4.3Content alignment & integration																																		
Content alignment definitions repository																																		
Inconsistencies identification																																		
Cross-modal linkage																																		
Spatiotemporal analysis and reasoning																																		
A.4.4Multimodal Indexing and Retrieval																																		
Model development																																		
Indexing Structure																																		

Table 30: WP4 timeline

3.4 Semantic reasoning and decision support (WP5)

The module developed within WP5 comprises three interlinked sub-modules: (1) Data Infrastructure Module (DIM): actual conceptual information in form of ontologies and factual world knowledge necessary to support the realisation of the use cases; (2) Semantic Representation Infrastructure Management System (SRIS): a system for management of the conceptual information of the project which will support the storage of RDF statements, reasoning over them and access to them via standard query languages like SPARQL; and (3) Decision Support System (DSS): it will provide the access of the user to the conceptual information stored in DIC.

3.4.1 Data infrastructure module

The data infrastructure module is the output of the knowledge modelling activity (A.5.1). It consists of a semantic repository and basic ontologies and datasets. It will serve as the storage layer for the (meta)data of the MULTISENSOR platform. The semantic representation infrastructure will be used for data storage, reasoning and linked data management. It will be based on open semantic standards, e.g. RDF, and will integrate knowledge from the LOD cloud with knowledge collected as a result of the technologies developed in WP2, WP3, and WP4. Thus DIM will be divided in two main components: *General Knowledge Component* (GKC) and *Use-case Knowledge Component* (UcKC). In the first component the main ontologies to support the structure of the world knowledge and the functionality of the system will be loaded together with datasets representing the background world knowledge. In the second component the use-case related ontologies will be presented together with the datasets containing the world statements extracted from the information sources for each use case.

The MULTISENSOR semantic representation infrastructure will have the ability to support continuous updates, and will provide a feature of sharing knowledge across repositories, developed by ONTO, called nested repositories, allowing to build a cascade of repositories sharing inferred knowledge. The MULTISENSOR semantic representation infrastructure will be stored in an OWLIM semantic repository, and will scale to tens of billions of retrievable statements. This task is language independent. The output of this task will be the semantic representation infrastructure.

General knowledge component

The general knowledge component will be based on the RENDER data layer. It is a collection of datasets in RDF loaded in OWLIM semantic repository. It consists of background knowledge, and data specific for the MULTISENSOR use cases will be added. The background knowledge is represented in a reason-able view of the web of data – FactForge, a service maintained by Ontotext. The datasets of FactForge are:

- DBpedia (http://dbpedia.org)
- Freebase (http://www.freebase.com)
- Geonames (http://www.geonames.org)
- New York Times (http://data.nytimes.com)
- MusicBrainz (http://www.musicbrainz.org)
- Wordnet (http://wordnet.princeton.edu/)

- Lingvoj (http://www.lingvoj.org/)
- Lexvo (http://www.lexvo.org/)
- CIA Factbook (https://www.cia.gov/library/publications/the-world-factbook/)

It is also supplied with a reference layer of a light weight upper level ontology, developed by Ontotext, which is interlinked with the ontologies of DBpedia, Geonames and Freebase – PROTON (http://www.ontotext.com/proton-ontology).

This reference layer allows for efficient access and navigation of the data by using the classes and predicates of PROTON and obtaining results from all FactForge datasets together.

Additional datasets/ontologies that are included in the MULTISENSOR data layer are:

- DMOZ topics (http://rdf.dmoz.org)
- KDO (http://render-project.eu/resources/kdo/, http://kdo.render-project.eu/kdo#)
- SIOC (http://sioc-project.org/, http://rdfs.org/sioc/ns#)

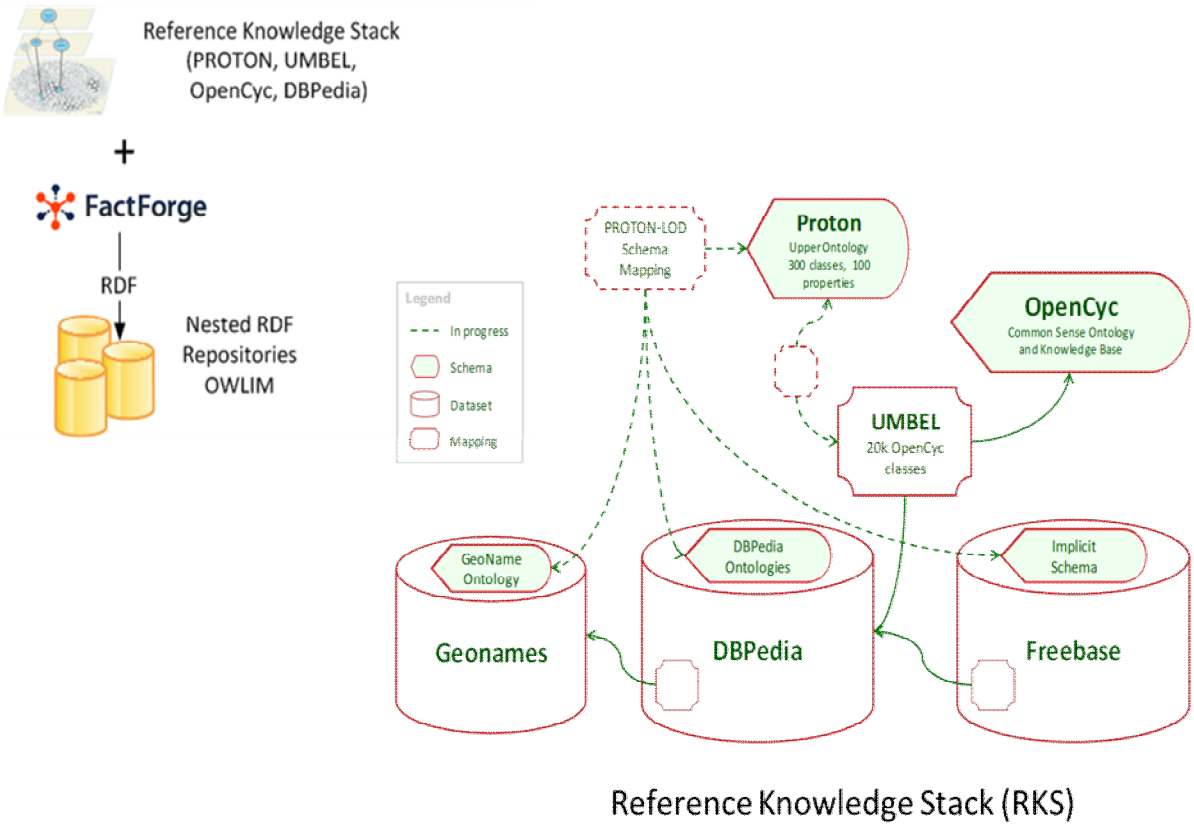

Figure 9: Reference Knowledge Stack

Additionally, a review of upper and core ontologies will be made and RKS will be updated with respect to the use cases' needs.

Use-case knowledge component (Nested Repositories)

The Use-case knowledge component will consist of nested repositories. Nested repositories are an experimental OWLIM feature, which allows for data sharing across repositories. It is set on top of the repository containing the basic data layer, one nested repository per

secondary dataset, e.g. one for the news articles. In MULTISENSOR, the nested repository functionality will be further developed and improved to meet the use case needs.

Processed data, based on the technologies developed in WP2, WP3, and WP4, and RDF-ised based on the MULTISENSOR data models, which include KDO, SIOC, DC, PROTON, and some MULTISENSOR specific properties and entities, will be loaded in the corresponding nested repositories, which will produce inferred statements as a result of the interaction between the data in the basic data layer and the secondary data loaded.

MULTISENSOR Data Infrastructure Module components will be accessible via a standard SPARQL end point RESTFul service for querying, loading and deleting, the later requiring security credentials. A human user friendly Forest interface will give the possibility to query the data with keywords, with SPARQL 1.1 queries, and relation discovery between two given entities.

The additional use case dependent ontologies and related properties and entities will be based on a review and analysis of existing ontologies relevant to the use case domains.

3.4.2 Semantic representation infrastructure management system module

This module comprises of the following components.

Hybrid reasoning component

The hybrid reasoning component (A.5.3) will provide the functionalities needed to perform forward- and backward-chaining concurrently to allow for efficient query-answering. It will also report on the developed multi-threaded reasoning allowing for efficient data loading and data updates. The component will also implement temporal reasoning techniques and the geo-spatial module extensions. The architecture of OWLIM will be updated as follows (see Figure 10):

- Backward chaining will be implemented as a new component, using Special Resultset Iterators;
- Geospatial plugin is an existing Owlim plugin; we simply need to make sure it works well with the rest of the extensions;
- Temporal reasoning will be implemented as a new plugin, using the Owlim Plugin API;
- Forward chaining reasoned (TRREE) will be expanded with multithreaded-capabilities.

Input	A set of RDF facts relevant to the query from the user.
Output	A graph of query relevant facts extracted from the Semantic Repository. Relevant information will be encoded in the graph.
Dependencies	▪ WP7 and WP2-4 – the use cases needs and requirements with respect to the data type and reasoning; ▪ WP5 - Decision Support system
Critical Factors	▪ The CLAS tool is sensitive to how rich the annotation of data is; ▪ The lack of specificity can lead to poor results with this component.

Table 31: Hybrid reasoning component summary

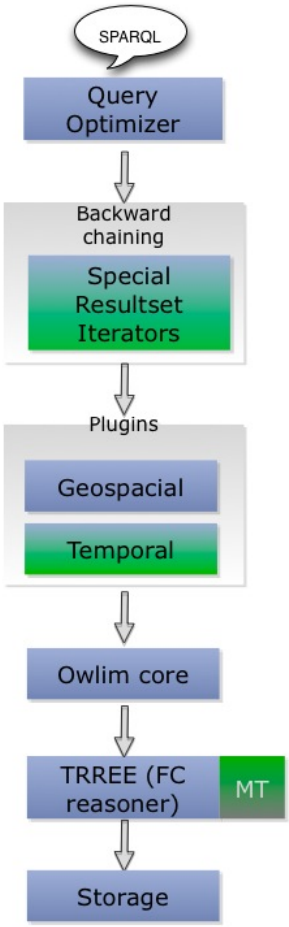

Figure 10: Hybrid reasoning component

Clustering and spreading activation (CLAS) component

CLAS is plug-in to OWLIM which has been designed to provide machine learning capabilities that complement OWLIM formal reasoning capabilities. It has been developed and exploited in the RENDER project and will be further developed for the purposes of MULTISENSOR as part of the decision support components. It can be accessed via a SPARQL endpoint and aims at combining the power of structured reasoning and statistical methods like spreading activation and clustering on graphs. The main principles of CLAS are extendibility and modularity. CLAS tools can be managed by a common SPARQL interface which allows the generation of sophisticated datasets extracted from large RDF triple stores. Such datasets can be further processed using the modules of CLAS and the results used for further data querying.

Input	A subgraph selected by the end user of DSS.
Output	Clusters where the subgraph elements belong to and ranking of retrieved information (i.e. news articles) based on typicality, sentiment, etc. The output can be made context dependent by using spreading activation.
Programming languages	Java, C/C++
Dependencies	▪ WP2, WP3 and WP4 knowledge extracted from different

	sources.
Critical Factors	▪ The CLAS tool is sensitive to how rich the annotation of data is; ▪ The lack of specificity can lead to poor results with this component.

Table 32: CLAS component summary

3.4.3 Decision support system module

The Decision Support System (DSS) module (A.5.4) will be actively used in the process of extraction of relevant information from the Semantic Repositories (SRs) of the Semantic Representation Infrastructure (SRI). The input to DDS will contain two parts: (1) query related information; and (2) user related information. The user related information will determine the interests of the user, the tasks to be solved, etc. It will predetermine the patterns for search of information within the SRs of SRI. The query information will consist in all the facts extracted from the user query in terms entities, relations, and goals. Depending on both kinds of information, the DSS will instantiate one or more search patterns and will execute them within SRI. The execution will exploit the hybrid reasoning of the SRI in order to determine the related facts in SRs. Besides the efficiency of the reasoning, in order to support the user in real time working, the DSS will use heavily the temporal and geo-spatial reasoning to determine the context of the events and objects extracted from the semantic repositories. Extracted facts will be grouped on the basis of clustering performed by DSS. This clustering will facilitate the comprehension of the relations between extracted facts supporting the decision making process. In some cases the system will support formulation of new user queries. After some user sessions with the system, some facts about the session will be stored in the user profile for later usage. This module consists of two components: Decision Support Component and Relevance Explanation Component.

Decision support component

This component will evaluate the user queries with respect to the repositories containing the use case world knowledge. The search is language independent and thus it will provide information for users using different interaction languages. The extracted information will be the basis for taking decisions, depending on the task.

Input	A set of RDF facts relevant to the query from the user.
Output	A graph of query relevant facts extracted from the Semantic Repository. Relevant information will be encoded in the graph.
Programming languages	Java
Dependencies	▪ WP6 interpretation of the query in terms of RDF facts; ▪ WP5 Semantic Representation Infrastructure.
Critical Factors	▪ The accuracy of this service is crucial to the quality of user supporting information. Extracting of irrelevant information could be distractive for the end user.

Table 33: Decision support component summary

Relevance explanation component

This component will extract additional facts from DIM repositories via SRI components. The additional facts will be selected in such a way to present to the user the mechanisms of inferences using for processing user query. The processing in this component is also language independent. This means that in many cases the results will be not easily understandable by the end users. Thus, we consider the results in form of RDF (sub)graphs as input for the generation component of WP2.

The activities needed for the implementation of the components of this module are related to the development of tools based on hybrid reasoning and CLAS which are use case specific and provide the required decision support functionality.

Input	A subgraph selected by the end user of Decision Support Component.
Output	A graph of relevant reasoning steps (spreading activation steps or feature similarities) that support the extraction of the facts in the input subgraph.
Programming languages	Java
Dependencies	▪ WP5 Semantic Representation Infrastructure.
Critical Factors	▪ The accuracy of this service is crucial for understanding of the extracted supportive information. The user needs to see the reasoning steps in an intuitive way.

Table 34: Relevance explanation component summary

3.4.4 Activity table and workload

Activity	Subactivity	Description	Dependencies	PM
Knowledge modelling (A.5.1)	Review of existing ontologies and choice of ontologies	Review of ontologies and choice of upper and core ontologies. Review and analysis of ontologies for the use cases' domains.	Use case needs - WP6, WP8 Content integration and retrieval and semantic annotation - WP2, WP3, and WP4	4
	Data and user models for MULTISENSOR	Design and creation of a domain specific ontology(ies)	Use case needs - WP6, WP8 Content integration and retrieval and semantic annotation - WP2, WP3, and WP4	4
Semantic representation infrastructure (A.5.2)	Initial data infrastructure – OWLIM, FactForge	Creation of a project OWLIM repository with FactForge, KDO and additional ontologies and datasets	Choice of upper and core ontologies – WP2, WP3, WP4	2
	Integration of knowledge for MS use	Storage of processed data for	Provision of data from the use	3

	cases based on e.g. semantic annotation, sentiment analysis, etc.	the use cases	cases – WP2, WP3, WP4, WP7	
	Implementation of nested repositories	The nested repositories are an experimental feature of OWLIM which will be deployed depending on the needs in MULTISENSOR	Use cases – WP7	3
	Data update and curation	Support of data storage, update, checks, error correction	Quantity and quality of data from the use cases – WP2, WP3, WP4, WP7	3
Hybrid reasoning (A.5.3)	Forward and backward chaining based reasoning	Implementation of the two types of reasoning and analysis of the cases when they can be used	Use-cases needs – WP7	4
	Multithreaded reasoning	Allows for parallel processing	Use-cases needs – WP7	2
	Temporal reasoning	Supports reasoning about events and time	Use-cases needs – WP7	3
	Geo-spatial reasoning	Supports spatial reasoning	Use-cases needs – WP7	2
Decision support (A.5.4)	Development of use-case specific hybrid reasoning tools	Tools for decision support based on hybrid reasoning	Use-cases needs and content extraction – WP2, WP3, WP4, WP7	7
	Development of use-case specific CLAS based information ranking and selection tools	Tools for decision support based on clustering and spreading activation	Use-cases needs and content extraction – WP2, WP3, WP4, WP7	7

Table 35: WP5 activity table

3.4.5 Timeline and dependency from other modules

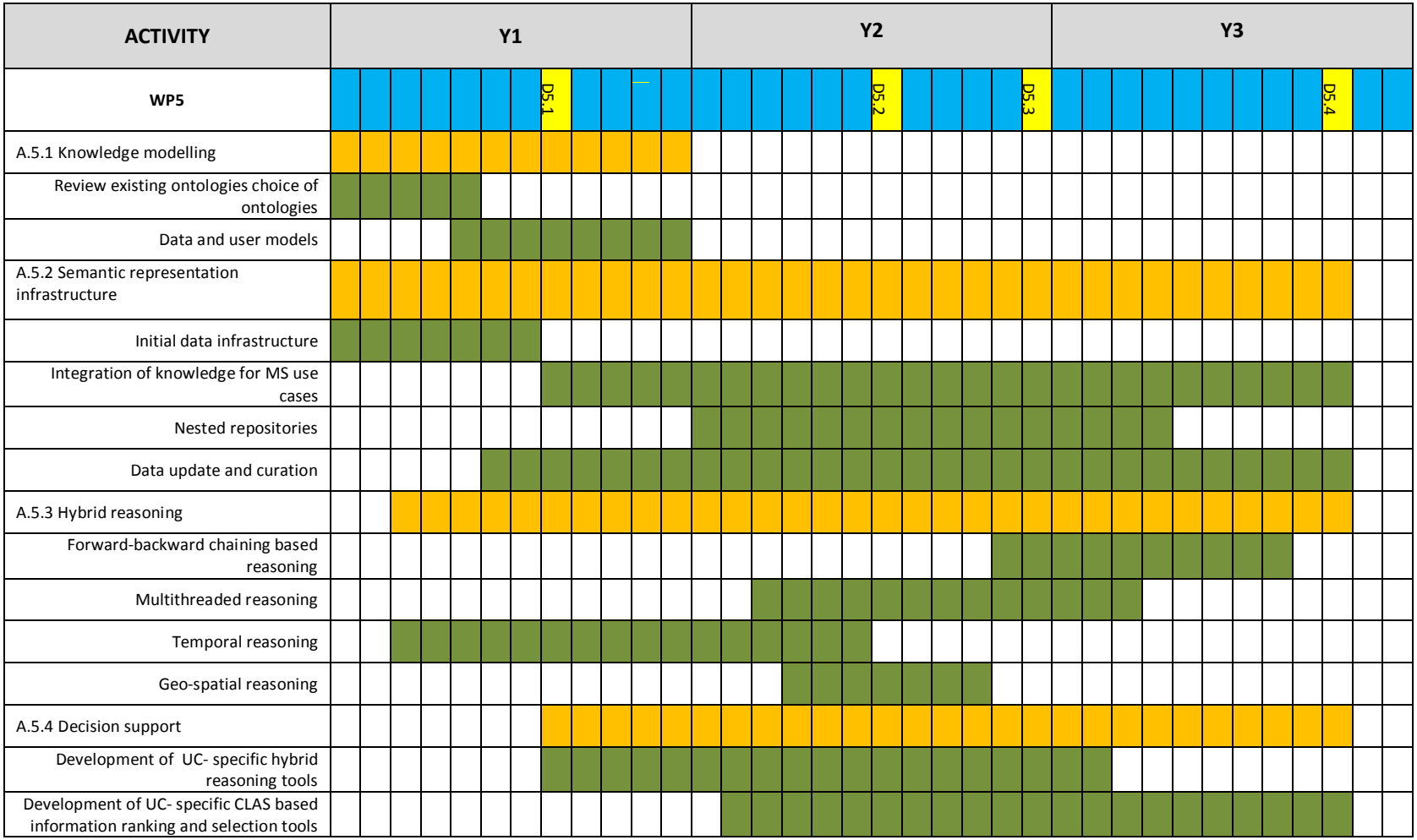

Table 36: WP5 timeline

3.5 Content summarisation and delivery (WP6)

In response to a given request, end users will receive a summary in natural language. The delivered summary will be based on the contents deemed as relevant by the decision support strategies of WP5. A first baseline implementation is foreseen (Tasks 6.1, 6.2 and 6.3) where extractive summarisation techniques will be applied to the original texts in the News Repository from which relevant contents where parsed. Fragments of these texts will be extracted and used to compose the summary. Machine translation methods will be used to support multilinguality in the input texts as well as in the delivered summary.

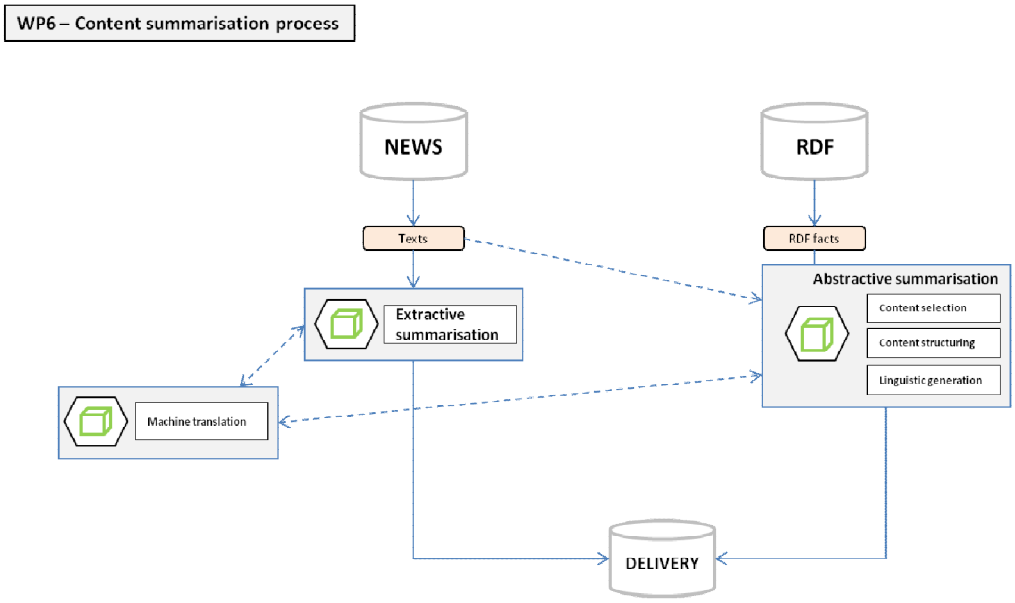

Figure 11: Content summarisation process

This first baseline will be replaced by a second baseline (Tasks 6.5 and 6.6) implementing an abstractive strategy where the production of summaries of news will start directly from the relevant contents stored in the Semantic Repository. Natural Language Generation methods will be used to produce the abstractive summary. In a first text planning step, the specific contents to include in the summary and their organisation in the text will be decided. This will be followed by the generation of the linguistic expressions that render the selected contents as a readable, coherent and grammatically correct text. As shown in Figure 11, the implementation of the abstractive strategy will consist of a pipeline of three components addressing the steps mentioned before:

a) Content selection
b) Content and discourse structuring for planning the text of the summary
c) Linguistic generation for the realisation of the final text in natural language (Task 6.4). Even though the Semantic Repository will constitute the main input to the abstractive summarisation, the original texts referenced from the contents will also be used, i.e. to extract metrics.

In a second baseline, the information production component will generate texts in English and an additional component will apply machine translation to produce summaries in other

languages. The final system will produce multilingual output without recourse to machine translation methods.

3.5.2 Extractive summarisation

The extractive summarisation component (A.6.1) will be based on SUMMA[13], a text summarisation toolkit which follows the architectural precepts of the GATE framework therefore providing much needed functionality in the form of language and processing resources for composition of practical summarisation applications. SUMMA provides resources for statistical text analysis, both single and multi-document and supporting multilingual processing.

The following subactivities have been identified for this task:

i. Extractive summarisation from a single document.

ii. Multidocument summarisation.

Input	Texts from the News Repository from which contents relevant to the user request were obtained.
Output	Texts assembled from relevant fragments extracted from the input texts.
Programming languages	Java
Dependencies	▪ Machine translation service ▪ WP5 decision service.
Critical Factors	▪ The coherence and readability of the generated summaries may suffer due to the fact that fragments come from different documents.

Table 37: Extractive summarisation component summary

3.5.3 Abstractive summarisation

Content selection

The assessment and selection of contents (A.6.2) will be confined to those contents in the Semantic Repository that the decision support strategies of WP4 have marked as relevant for the user request. In WP6, a fine-grained selection of contents at the level of RDF facts will determine exactly what information will be communicated in the final summary. For this purpose, the Content Selection component will employ heuristics obtained from the larger set of contents in the Semantic Repository, as well as the original texts in the News Repository, from which the contents were extracted (excluding sources with modes other than text, e.g. video or images).

The following subcomponents have been identified for this task:

i. Extraction of heuristics from texts for the evaluation of contents.

ii. Development of a statistical planning approach to content selection based on Reinforcement Learning, Automated Planning and Supervised Learning techniques.

[13] See **http://www.taln.upf.edu/pages/summa.upf/index.htm**

Input	A set of RDF facts relevant to the query from the Semantic Repository
Output	A subset of the input facts, marked as selected in the Semantic Repository.
Programming languages	Java
Dependencies	▪ WP4 ontologies ▪ WP5 decision service.
Critical Factors	▪ The accuracy of this service is crucial to the quality of the summaries, as in general readers do not tolerate well missing information.

Table 38: Content selection component summary

Content and discourse structuring

In order to generate a text that is both fluent and coherent, the set of RDF statements marked for inclusion in the text to be generated need to be organised in a way that guarantees a natural flow of information and is rhetorically-motivated. This organisation covers the ordering of individual statements according to local-coherence constraints (e.g. statements about the same topic or making reference to the same entities are placed in the summary in a way that reflects their relatedness, discourse relations holding between the statements, etc.), and also the rhetorical relations that relate larger sections of the text providing it with a global coherence. The rhetorical structure of the text will be determined in large part by the information available about the user and his communicative goals, possibly encoded in a user profile and the user request, and also by the results of opinion and sentiment analysis of WP3, to be found as part of the semantic contents themselves.

The following subactivities have been identified for this task (A.6.3):

i. Local coherence methods: research and implement mechanisms to find orderings of RDF facts that optimise local coherence.

ii. Global coherence strategy: development of a strategy based on the communicate goals of the user that determines the rhetorical structure of the text.

Input	A set of RDF facts relevant to the query from the Semantic Repository.
Output	A plan of the text to be generated, consisting of the ordered set of input facts, related to each other by coreference and other discourse relations. The plan will be asserted in the Semantic repository.
Programming languages	Java
Dependencies	▪ WP4 ontologies ▪ Content selection.
Critical Factors	The discourse relations between the contents in the output plan will be established according to various criteria, e.g. the user profile, the semantics of the contents, the analysis of target texts, etc. It may not always be possible to rhetorically relate contents to other contents in the plan, leading to summaries with a lesser degree of

	internal coherence.

Table 39: Content and discourse structuring component summary

Linguistic generation

The linguistic generation (A.6.4) will make use of the multilingual MATE Generator framework available at UPF. MATE is a graph transducer platform which is able to map an input graph into one or more output graphs. Both dictionaries and grammatical resources have to be compiled for this task. On the one hand, we will need three types of dictionaries: one that includes equivalences between the concepts of the RDF facts and the words of the different languages of the project (semantic dictionary), one that contains the description of the syntactic combinatorial of these words (lexical dictionary), and one that contains the description of their inflection patterns (morphological dictionary). On the other hand, we will need several transduction grammars, i.e. set of rules, which each perform a particular subtask of the linguistic generation pipeline. The following subtasks have been identified:

i. Communicative organisation of the RDF facts, i.e., defining what is said about what, based on the facts and the coreference and discourse relations holding between them;

ii. Mapping the RDF language-independent structures onto abstract language-specific structures (using the semantic dictionary);

iii. Defining the internal syntactic structure of each sentence and introduce all the words (using the lexical dictionary);

iv. Ordering and inflecting the words (using the morphological dictionary).

Input	A text plan asserted in the Semantic Repository.
Output	A summary in natural language.
Programming languages	Java, MATE Transduction Grammars.
Dependencies	▪ Machine translation service ▪ WP4 ontologies ▪ Content and discourse structuring
Critical Factors	N/A

Table 40: Linguistic generation component summary

3.5.4 Activity table and workload

Activity	Subactivity	Description	Dependencies	PM
Extractive summarisation (A.6.1)	Single document summarisation	A summary of a single summary is generated.	Machine translation service, WP5 decision service.	3
	Multi-document summarisation	A summary of multiple documents is generated.		3
Content selection (A.6.2)	Heuristics	Extract heuristics from texts.	WP4 ontologies, WP5 decision service.	5
	Statistical	Develop a statistical		10

	planning	planning system that employs the heuristics.		
Content / discourse structuring (A.6.3)	Local coherence	Find mechanisms to ensure correct ordering of contents.	WP4 ontologies, content selection.	5
	Global coherence	Develop strategy to rhetorically organise the contents.		7
Linguistic generation (A.6.4)	Development of grammars lexica; corpus annotation		Machine translation service, WP4 ontologies, content structuring.	12

Table 41: WP6 activity table

3.5.5 Timeline and dependency from other modules

ACTIVITY	Y1												Y2												Y3											
WP6										D6.1													D6.2										D6.3				
A.6.1Extractive summarisation																																				
Single document																																				
Multiple document																																				
A.6.2 Content selection																																				
Heuristics																																				
Statistical planning																																				
A.6.3 Content structuring																																				
Local coherence																																				
Global coherence																																				
A.6.4 Linguistic generation																																				

Table 42: WP6 timeline

3.6 System development and integration (WP7)

The list of the main objectives of system development and integration is as follows:

1. To plan the roadmap and design the architecture based on user requirements.
2. To design and implement the infrastructure required for hosting the MULTISENSOR platform.
3. To develop crawlers and data wrappers to collect data from selected content sources.
4. To design and implement prototype portals for each use case. The main goal of these portals is to demonstrate the developed technologies and allow the users to evaluate the system and the quality of the results.

3.6.1 MULTISENSOR architecture

This task (A.7.1) will involve the definition of the global architecture for the MULTISENSOR platform, from its high-level component design to the server layout definition. A roadmap will also be defined, to drive the process of implementation (this document).

Dependencies	User requirements, specification of pilot use cases and validation plan: the Use Cases will drive the processes and ultimately the architecture will be built to service those use cases, so a clear definition is essential.
Critical Factors	▪ Incomplete Use Case definition: UCs must be clearly defined and well understood to ensure the appropriate systems, processes and capabilities are put in place in the architecture. UCs must include comprehensive scenarios and validation plans to ensure the architecture is built correctly to specifications. ▪ Incomplete Business Service Definition: each of the technical development WP is responsible for defining the service API for components related to its research. Failure to provide the API specification in a timely and thorough manner can lead to delays or errors in definition of the architecture.

Table 43: Architecture definition summary

3.6.2 Technical infrastructure

Runtime infrastructure

This task (A.7.3) implies the provisioning and operation of the infrastructure, on which the MULTISENSOR platform will run. This includes the UC Portals, integration services, business services, and associated repositories.

Due to the distributed nature of the MULTISENSOR platform, and its location-independence, 2 infrastructure types have to be considered:

- **Central infrastructure**: servers/resources residing in a central (virtual) location, collocated together, running in the same LAN and connected through high speed network link. This infrastructure will be hosted in the Amazon EC2 cloud, and administered by EVERIS.

- **WP infrastructure**: servers/resources residing in a partner's premises or under a partner's control. Running in a separate network and administered by the owner.

Some subsystems of the MULTISENSOR platform cannot be deployed to the central infrastructure, for licensing reasons; others may be dependent on secondary systems that cannot be easily replicated in a different environment, or may require concentrated computing power that is provided by a highly specialised expert system in a partner's data centre. In those cases, proxy services will be built by partners and exposed to the rest of the platform. Proxy services will delegate work to the actual services and do any extra work as required in a transparent way.
Since they're accessed by URL, services can be moved around based on the platform's needs: e.g. a service that needs high network throughput is placed in the central infrastructure, and a service that needs to be close to a partner's proprietary database is placed in that partner's infrastructure and accessed through a public proxy.

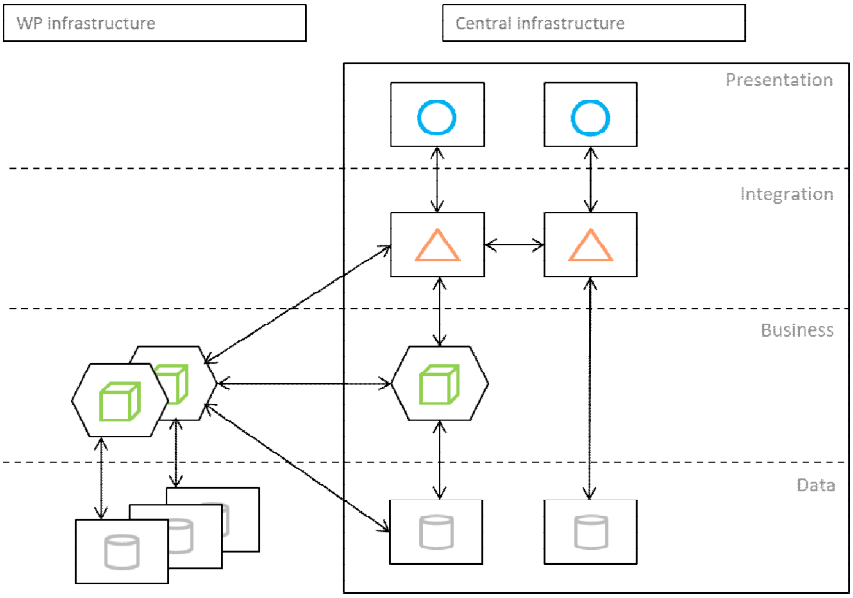

Figure 12: Distributed infrastructure model

The figure below displays the tentative model for physical distribution of the infrastructure elements in the segments.

- WP2 automated translation and NER processes require support from proprietary infrastructure owned by LT which cannot be moved; translation and NER services will be exposed via a service in their infrastructure that will delegate work to LT's internal system.
- The Semantic Repository will be provisioned by ONTO in their premises and access will be provided to services in the platform.
- PR, DW and BM-Y! will crawl/select/filter/curate content in their own specialised systems and provide raw data via feeds or via uploads to an agreed location in the Central Infrastructure for aggregation into the Central News Repository.
- The UC Portals, Integration Services, Business Services, Ops and Delivery Repository will run in the central infrastructure, to minimise latency and maximise runtime performance.

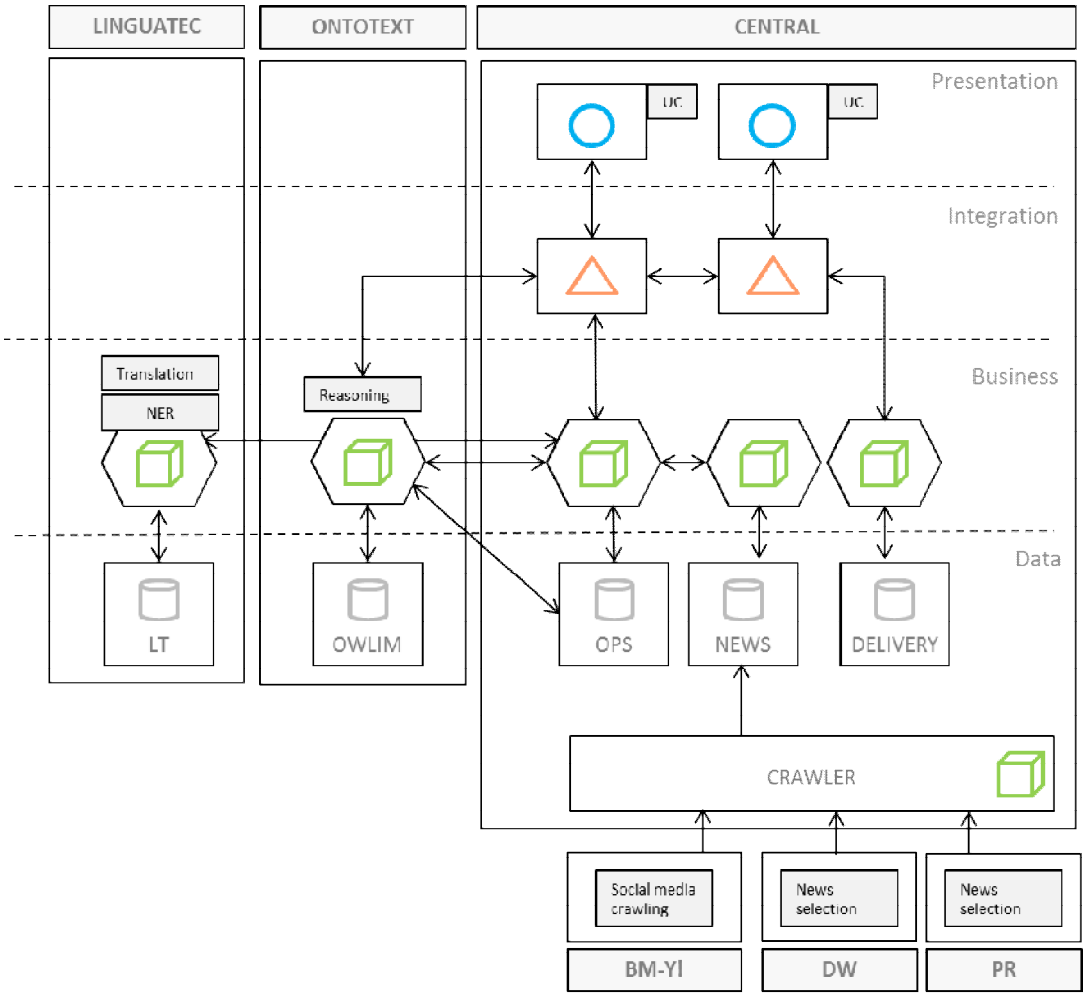

Figure 13: Draft infrastructure location map

Development infrastructure

This activity also foresees provisioning and setup of the development infrastructure for the project.

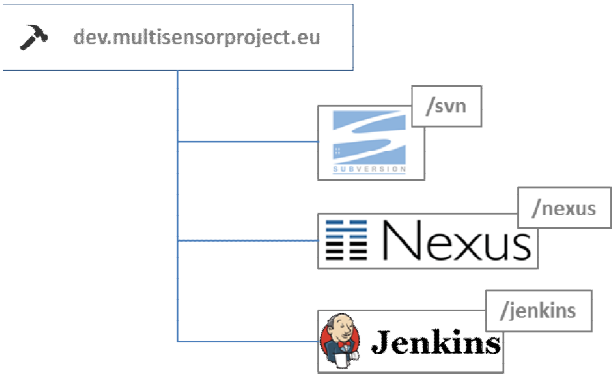

Figure 14: Infrastructure services

Source code control: All code and related assets will be centrally stored in a Subversion repository hosted in the Central Infrastructure network. Source code will be structured in sub-trees mapped to the different WPs that make up the project, for manageability.

Artefacts repository: All Java software packages (services, support programs, etc) will be built as Maven artefacts and they will be stored in a Nexus repository hosted in the Central Infrastructure network.

Continuous integration server: A Jenkins server will be used to automate builds and perform automated testing and code analysis.

File server: A file storage area will be available to partners to share arbitrary data. Files will be password-protected and accessible over HTTP.

Dependencies	System architecture: technical infrastructure will be provisioned to implement the system architecture.
Critical Factors	▪ Distributed infrastructure: parts of the infrastructure are foreseen to be distributed and include systems which are run in-house by some partners. Issues with firewalls, authentication, etc. are to be expected, and acted upon as needed. ▪ Performance: the scope and complexity of the MULTISENSOR processes and the size of the data to be handled is not yet known and is likely to evolve as research progresses. Performance problems may arise, and resizing and/or provision of extra capacity may be required. Encapsulation of the functionalities as services will greatly simplify infrastructure changes under the hood.

Table 44: Infrastructure definition summary

3.6.3 **Crawlers and data channels infrastructure**

Crawlers (A.7.2) will be responsible for sourcing content for the platform to process and use for analysis.

A scheduled process will regularly crawl a pre-defined pool of sources for news on specific subjects. Sources will include print and online news sites, social media, and audio and video sources (online radio, podcasts, tube sites, etc.). Raw data will be normalised into a common JSON format and stored in the Central News Repository.

Initially, two main content sources are planned: the PR crawler, and the Social Media crawler.

News crawler: a process to regularly collect and process data harvested by the available news sources (see Figure 15). The crawler will retrieve content and aggregate it into the Central News Repository for analysis (see 3.1 for description of news content analysis).

Social Media Crawler: a process to traverse social media networks (e.g. Twitter) and collect news, status updates, and perform social and sentiment analysis. The crawler will collect data and aggregate it into the Central News Repository for analysis (see 3.2.3 for description of social media analysis).

Input	News (text) and multimedia content (audio, video)
Output	JSON serialisation of harvested items
Dependencies	Use Cases & Content sources: selection of sources to be processed by the crawlers.
Critical Factors	Source selection: selection of sources for news/multimedia should be relevant to the problem domains defined in the UCs and be updated regularly. Outdated, incorrect or irrelevant data can compromise the quality of the results of the MULTISENSOR processes.

Table 45: Crawler component summary

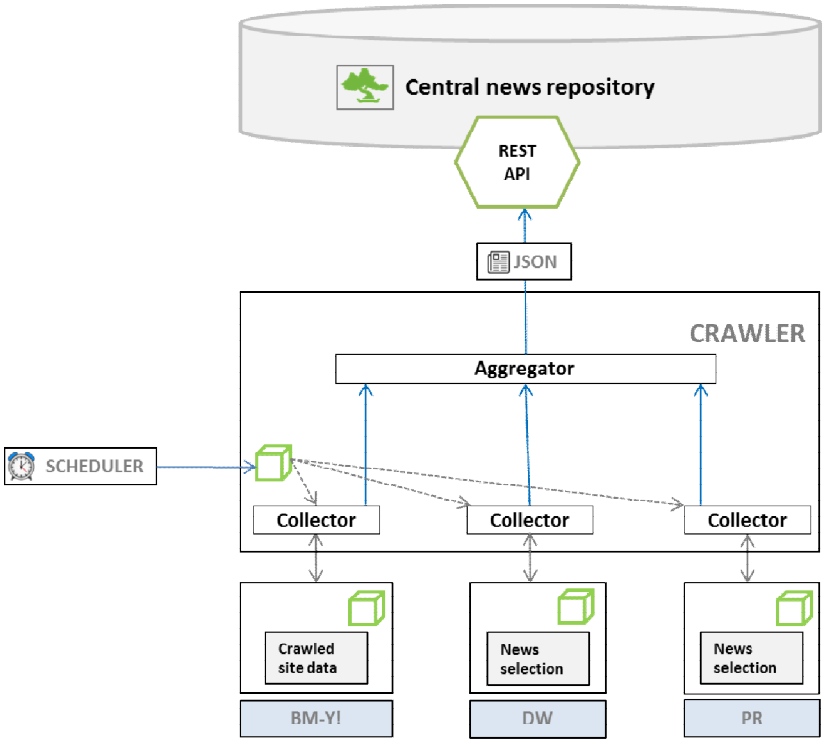

Figure 15: News harvesting process

3.6.4 System development

This task (A.7.4) is the umbrella task for all development activities related to technical implementation of the Use Cases and any other work required for completion of the objectives.

Development will be iterative, from an initial dummy prototype to the final feature-complete version, with the following cycle planned:

Operational prototype

- **Architecture**: Dummy end-to-end architecture; all service dummies in place, operating on test/mock data
- **Infrastructure**: dev. architecture, first system architecture in place
- **Crawlers**: first version (crawling dummy PR+DW data, dummy social media data)

- **Interface**: first usable version of interface

Prototype 1

- **Architecture:** using real services from WP2-WP6 Prototype 1 milestones
- **Crawlers**: second version
- **Interface**: improved usable version

Prototype 2

- **Architecture:** using real services from WP2-WP6 Prototype 2 milestones
- **Crawlers**: final version
- **Interface**: final usable version

Final prototype

- Complete MULTISENSOR platform

Dependencies	▪ All WP Business services ▪ WP8 Use Cases and Evaluation
Critical Factors	▪ Changing requirements: UCs should remain stable during the project. While minor changes are always to be expected, major breaking changes may cause throwing away work that has already been done and impact on delivery capabilities. ▪ Changing service API: the business services API should be defined early, and remain stable during the project. API changes impact on the rest of partners and can have negative consequences on delivery.

Table 46: System development summary

3.6.5 Activity table and workload

Activity	Subactivity	Description	Dependencies	PM
MULTISENSOR Architecture (A.7.1)	Roadmap	This document	N/A	4
	Technical requirements and architecture design	Description of technical requirements derived from UCs, and detailed design of platform architecture	Use case needs - WP8	4
	UI & Operational prototype	UI mock-ups, dummy-based platform	Requirements and architecture design – WP7	2
Crawlers and data channels infrastructure (A.7.2)	News crawler	Crawler for processing PR content	N/A	4
	Social media crawler	Crawler for processing Twitter / other systems	N/A	7
Technical	Development	Basic facilities for		2

infrastructure (A.7.3)	infrastructure	development		
	Runtime infrastructure	Platform hosting in cloud environment	Technical requirements & architecture design – WP7	13
System development (A.7.4)	Prototype 1	First iteration of MULTISENSOR platform	Runtime infrastructure – WP7 WPs 2-6	18
	Prototype 2	Second iteration of MULTISENSOR platform	Prototype 1 – WP7 WP 2-6	9
	Final system	Final version of MULTISENSOR platform	Prototype 2 – WP7 WP 2-6	4

Table 47: WP7 activity table

3.6.6 Timeline and dependencies from other modules

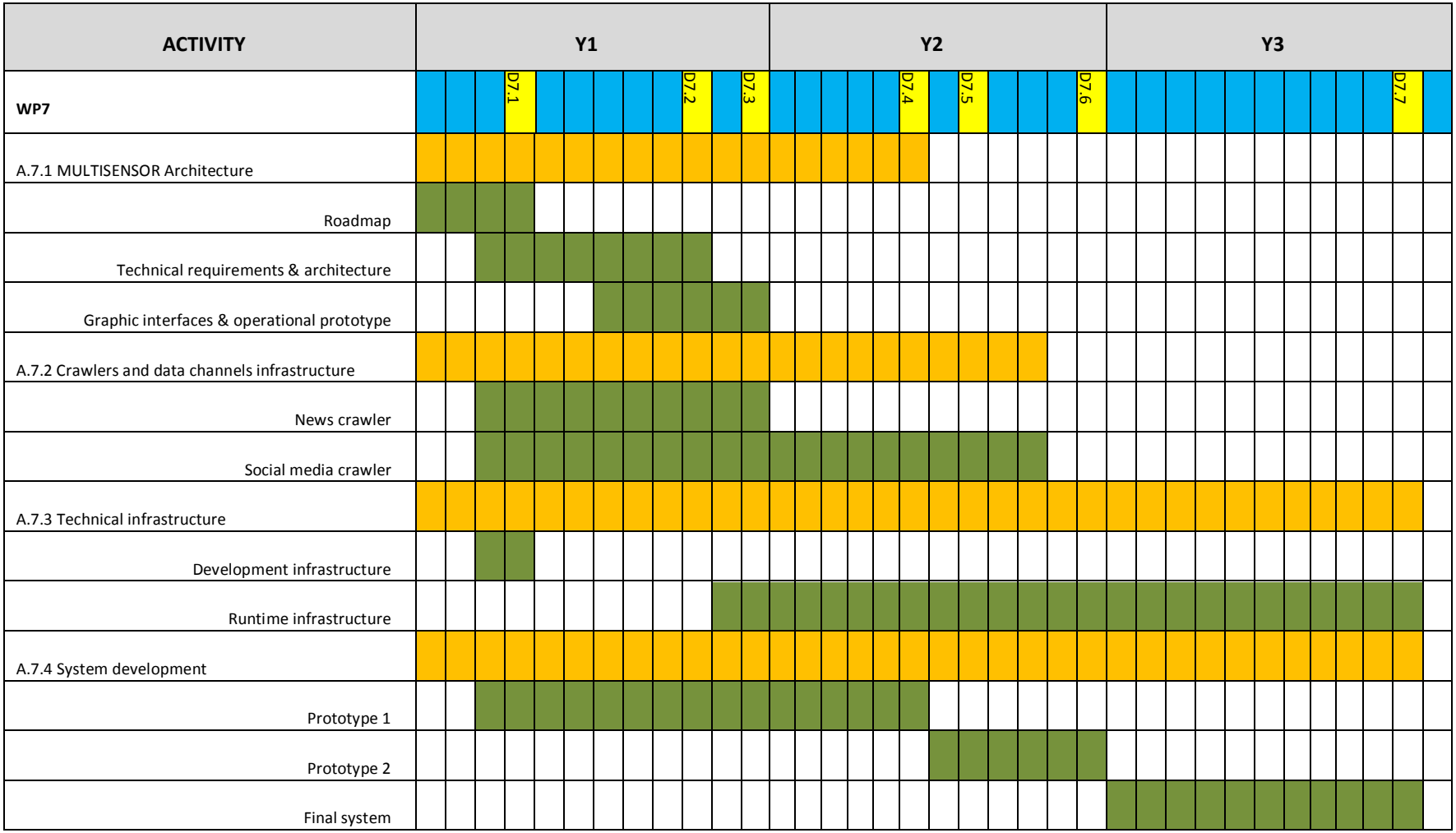

Table 48: WP7 timeline

4 DEVELOPMENT CYCLE AND USE CASES

The overall strategy for the development of the MULTISENSOR platform is to start with simple scenarios and proceed stepwise towards the final system. The detailed resource allocations and work plans of the individual modules presented in section 3 will follow the general cycle: after the completion of each prototype version (timing defined by Milestones) of the MULTISENSOR (discussed in the next section), the functioning of the prototype is assessed, the needs for further development are identified and agreed upon, a detailed work plan for the next prototype version is finalised and the development cycle towards the next prototype is initiated. One important predefined concept for the different prototypes is the definition of the practical user cases each prototype is serving.

Implementation of use case-related features will evolve with the successive prototypes:

- **Prototype 1**: wireframe like interfaces, rough design, not all functions available yet.
- **Prototype 2**: visual interface close to final version; most functions available.
- **Prototype 3**: final interface: all functionalities fully available as planned.

It should be noted that the Full Description of the use cases will be provided in D8.2.

4.1 Scenario 1: international media monitoring

Target group: Journalists, Media experts, media companies.

Objectives: MULTISENSOR should provide support:

- In understanding the specific meaning of a textual document independent of the language, tone and idioms in use.
- In summarising the content of very complex and heterogeneous texts and bringing out the most important information of the original document.
- In identifying related content and in combining textual data with other related data, especially multimedia items.
- In finding out more about the contributors of specific information as well as about how a story (a news event, a promotional campaign) evolves over time.
- In visualising social media conversations according to relevancy or other criteria of choice.

4.2 Scenario 2: SME internationalisation support

Target group: SME's initiating or interested in the process of internationalisation, with all kinds of products.

Objectives: MULTISENSOR should provide support:

- In understanding the specific meaning of a textual document independent of the language, tone and idioms in use.
- In identifying content (e.g. market data, legal texts) as well as institutions/persons that are relevant to their business goals.
- In summarising the content of very complex and heterogeneous texts and bringing out the most important information of the original document.
- In organising information about different markets according to specific criteria.

5 PROJECT TIMELINE AND MILESTONES

The platform will be built incrementally and iteratively, starting with a barebones system made up of dummy services for formal end-to-end validation, and delivering increasingly enriched functionality and further capabilities on each milestone[14].

The figure below illustrates the timeline for the technical milestones in the evolution of the platform.

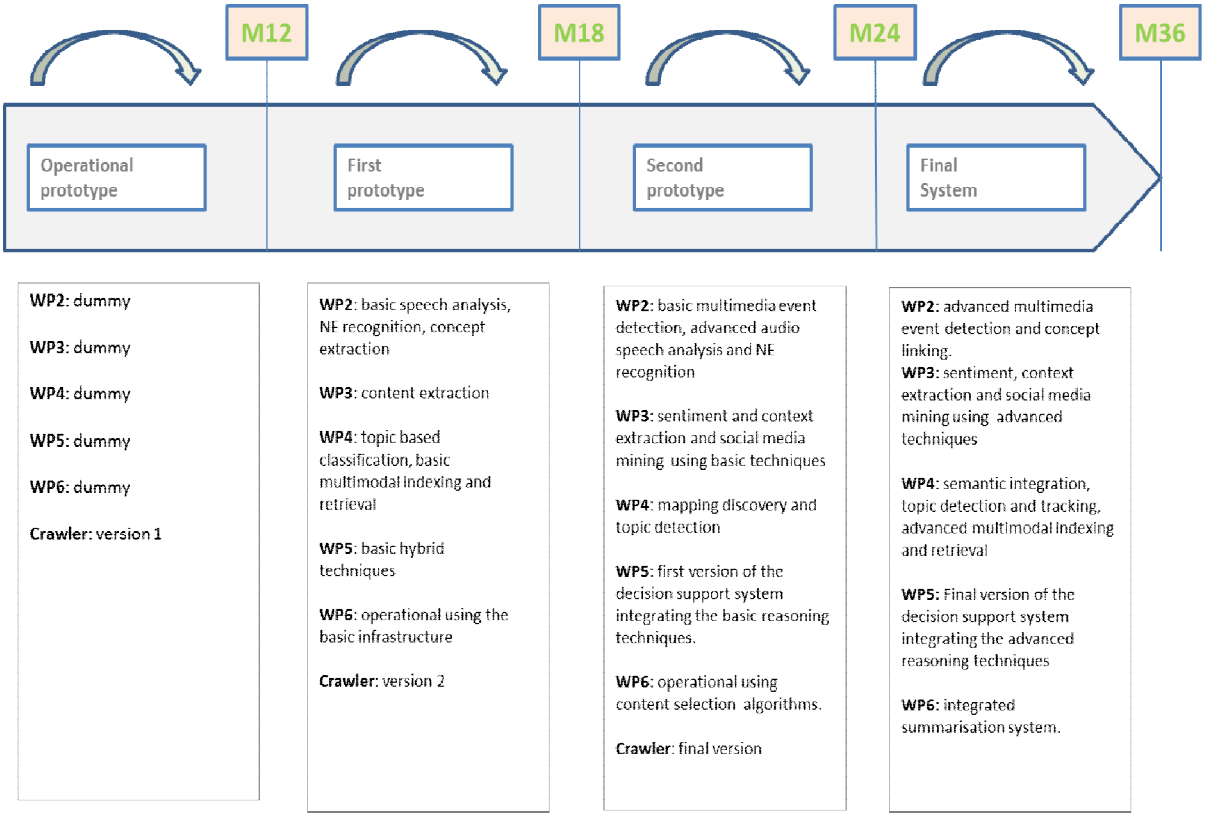

Figure 16: MULTISENSOR walking skeleton

[14] See **http://alistair.cockburn.us/Walking+skeleton** for a discussion on the "walking skeleton" strategy for architecture design and implementation.

6 SUMMARY

In this deliverable we presented the technical roadmap of MULTISENSOR platform. This document is very important, since it will guide the developments of the project with a view to attaining the scientific and the technical objectives envisaged.

However, since the project is following an iterative development procedure (use cases-requirements-development-evaluation), it is expected that small adaptations and deviations on the initial technical specifications and the module functionalities foreseen by this document will be needed to satisfy the final user requirements. Such changes/adaptations might be required at component or subcomponent level. The platform architecture together with the detailed specification of the modules (including any updates required) will be reported in D7.2 (Technical requirements and architecture design).

7 References

Bay, H., Tuytelaars, T., Gool, L.V. 2006. “SURF: Speeded Up Robust Features”, In: Proceedings of the ninth European Conference on Computer Vision.

Chang, C.C., Lin, C.J. 2011. “LIBSVM: a library for support vector machines”, ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27.

Daras, P., Axenopoulos, A., et al. 2011. “Introducing a Unified Framework for Content Object Description”, Int. Journal of Multimedia Intelligence and Security (IJMIS), 2, pp. 351–375.

Jégou, H., Douze, M., Schmid, C., Pérez, P. 2010. “Aggregating local descriptors into a compact image representation”, In: IEEE Conference on Computer Vision & Pattern Recognition, pp. 3304-3311.

Kohlschütter, C., Fankhauser, P., Nejdl, W. 2010. “Boilerplate detection using shallow text features”, In: Proceedings of the 3rd ACM International Conference on Web Search and Data Mining, pp. 441–450.

Lazebnik, S., Schmid, C., Ponce, J. 2006. “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories”, In: Proc. of IEEE conference on Computer Vision and Pattern Recognition (CVPR), pp. 2169-2178.

Lowe, D. G. 2004. “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, vol. 60(2), pp. 91-110.

Mansour, R., Refaei, N. and Gamon, M. and Abdul-Hamid, A. and Sami, K. 2013. “Revisiting the Old Kitchen Sink: Do we Need Sentiment Domain Adaptation?”, In Proceedings of the International Conference Recent Advances in Natural Language Processing, pp. 420-427.

Novak, C.L. and Shafer, S.A. 1992. “Anatomy of a color histogram”, In: Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 599 - 605.

Petkos G., Papadopoulos S., Kompatsiaris I. 2012. “Social event detection using multimodal clustering and integrating supervisory signals”, In: Proceedings of the 2nd ACM ICMR’12.

MPEG, The Moving Picture Experts Group: MPEG-7, http://mpeg.chiariglione.org/standards/mpeg-7, retrieved 5/1/2014

The University of Waikato: WEKA, http://www.cs.waikato.ac.nz/ml/weka/

WebLab Exchange Model, http://weblab-project.org/index.php?title=WebLab_1.2.2/WebLab_Exchange_Model