Networked Media Search Engine, Visualisation and User feedback v1

Deliverable D3.2

TOSCA-MP identifier:	TOSCAMP-D3.2-VRT-NetworkedSearchEnginev1-v06.docx
Deliverable number:	D3.2
Author(s) and company:	M. Matton and B. Vandenbroucke (VRT) O. Lopez, G. Álvaro and C. Ruiz (PLY) A. Messina and M. Montagnuolo (RAI) S. Gerke and E. Muki (HHI)
Internal reviewers:	Michael Weber (DTO)
Work package / task:	WP3
Document status:	Final
Confidentiality:	Public

Version	Date	Reason of change
01	2012-06-28	document created (e.g. structure proposed, initial input…)
02	2012-08-31	Early draft version including contributions from RAI, HHI, VRT
03	2012-09-05	Initial input from PLY
04	2012-09-21	Final input from all partners
05	2012-09-24	Version for internal review
06	2012-10-04	Final version

Acknowledgement: The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287532.

Table of Contents

Table of Contents iii

List of Figures v

List of Tables vi

1 Executive Summary 7

2 Introduction 8
- 2.1 Purpose of this Document 8
- 2.2 Scope of this Document 8
- 2.3 Status of this Document 8
- 2.4 Related Documents 8

3 Metadata Sources 9
- 3.1 Essence and basic Metadata (mammie) 9
  - *3.1.1 Information page 9*
  - *3.1.2 Object identifiers 9*
  - *3.1.3 Token management 9*
  - *3.1.4 Thumbnail Generation 10*
  - *3.1.5 Datastreams 11*
  - *3.1.6 Search by SOLR query 11*
- 3.2 Concept detection 11
- 3.3 Quality analysis 12
- 3.4 Full text search and retrieval on multimodal news aggregations 12
  - *3.4.1 Index field types and properties 13*
  - *3.4.2 News broadcasting full text index 13*
  - *3.4.3 Multimodal news topics full text index 14*
  - *3.4.4 Named Entities Full Text Index 16*
  - *3.4.5 Querying the indexes 17*
- 3.5 Automatic Semantic Content Annotation 18
  - *3.5.1 Overview of the architecture 19*
  - *3.5.2 Components 20*
  - *3.5.2.1 Playence Ontology Manager 20*
  - *3.5.2.2 Named entity recognition by gazetteer 21*
  - *3.5.2.3 Named entity recognition by multiple ontologies 22*
  - *3.5.2.4 Indexation of annotation 23*

4 Search Engine 25
- 4.1 Core search engine 25
- 4.2 Query interface 26
  - *4.2.1 Parameters 26*
- 4.3 Integration with other systems – Mammie + Other Metadata Services 27
- 4.4 Query example 29
- 4.5 Initial user interface 30

5 Conclusions 31

6 References 32

7 Glossary..33

List of Figures

Figure 1 - Overview of the annotation architecture 20
Figure 2 - Playence Ontology Manager........ 21
Figure 3 - Named entity recognition by gazetteers 22
Figure 4 - Named entity recognition by ontologies........ 23
Figure 5 - Indexation of annotations........ 24
Figure 6 – Integration of the search engine, the mammie system, and other metadata services 28
Figure 7 - Initial user interface for the networked media search engine 30

List of Tables

Table 1. News broadcasting index: available fields for anagraphic metadata 14
Table 2. News broadcasting index: available fields for categorical metadata 14
Table 3. News broadcasting index: available fields for content metadata 14
Table 4. Multimodal news topics index: available fields for anagraphic metadata 15
Table 5. Multimodal news topics index: available fields for categorical metadata 15
Table 6. Multimodal news topics index: available fields for content metadata 15
Table 7. Named Entities full text index: available fields. 17

1 Executive Summary

This deliverable contains a description of the different components of the first version of the networked media search engine.

For this first iteration, the focus of the work is on having an early running prototype as back-end connecting the available individual components with an existing visualization. Thus, it does not yet include a connection to the TOSCA-MP MPMF. Instead, the different components are integrated point-to-point where necessary.

The source for essence and basic metadata is the mammie platform (D4.2.1). In this document, the REST API of mammie is described.

Two more metadata sources are coupled with the networked media search engine, namely concept detection and quality analysis. The last information source used for this version comes from multimodal news aggregations, where natural language techniques are used on speech transcriptions and RSS feeds in order to extract named entities and other relevant keywords and topics.

The central component in this deliverable is the semantic search engine. This component provides the platform with search functionalities on textual or multimedia. It offers guiding and content filtering mechanisms to contextualize any search and retrieve more accurate results minimizing the time spent finding the right query.

2 Introduction

2.1 Purpose of this Document

This document describes the first version of the networked media search engine, which is delivered together with this deliverable as an early prototype for Y1.

2.2 Scope of this Document

Provision of a service-based semantically driven search engine for AV content beyond text, definition of new paradigm for results presentation, provision and validation in user studies and field trials, and definition of implicit and explicit feedback mechanism and its integration into the system.

2.3 Status of this Document

Final version of the deliverable.

2.4 Related Documents

This document refers to several other related deliverables of the TOSCA-MP project. The referred documents are the document on Speech/visual metadata extraction, semantic enrichment and linking and genre adaptive processing v1 (D.2.1), and Test material including ground truth v1: Initial annotated data sets, describing the mammie platform and its contents (D.4.2.1).

3 Metadata Sources

Description of the different metadata sources coupled with the search engine and documentation of the API they provide for integration.

This chapter is organised as follows: in section 3.1, the REST API of the mammie platform providing content and basic metadata is described. Section 3.2 describes the input and output paramaters for a module on concept detection. In section 3.3, a module for quality analysis is described. Next, section 3.4 describes a system for search and retrieval on multimodal news aggregations. Finally, section 3.5 presents a system for automatic semantic content annotation.

3.1 Essence and basic Metadata (mammie)

For a basic overview of the mammie platform and its contents, we refer to D4.2.1.

This section gives an overview of the current REST-endpoints that can be used as an interface with MAMMIE. The provided URL's are relative and should be appended after the base url for the REST-services. When certain parameters in-between slashes start with a colon ':', for example ':token', this means the parameter must be replaced by its corresponding value in the URL.

The REST base url is http://tosca-mp.lab.vrt.be/mammie_services/

3.1.1 Information page

/ (just slash, append nothing to the base url)

When only specifying the base url for the REST-services, you obtain a page describing the endpoints in the REST API. It is a slightly outdated version however.

3.1.2 Object identifiers

/id/create

Creates a new universally unique identifier (UUID) for an object, for example 81dd5cd0-2a0f-012d-f109-000c2977b342. An item with associated data- and metadatastreams is identified by putting 'tcmp:' in front of this UUID. An example of an **item identifier** is: tcmp:81dd5cd0-2a0f-012d-f109-000c2977b342. The 'tcmp' part is called the 'app_id' (application identifier).

3.1.3 Token management

MAMMIE uses tokens to enable access to media and REST functionality, as an authentication mechanism. An example token is token_36a3bf10-2a09-012d-f108-000c2977b342. For each use of a MAMMIE REST service, a token must be appended at the end of the URL as follows:

?token=token_36a3bf10-2a09-012d-f108-000c2977b342

When no token is appended to the URL, the service returns 'no token found'. When an appended token is not valid, this service returns 'invalid token'.

Some REST-services do not require an authentication token appended to the URL, for example the services to get a new token or to invalidate an existing token.

/tokens/new/:app_id

This REST URL allows to get a new token for the application with prefix app_id, in this case app_id = tcmp for TOSCA-MP. This becomes /tokens/new/tcmp (other prefixes than tcmp are for other internal VRT applications or projects).

After a valid certificate is present on the user's computer, new valid tokens can be requested from the MAMMIE-server using this service.

Obviously, for this REST-service no authentication token needs to be appended at the end of the URL.

/tokens/:token/app_id

For a given token, this service returns the prefix of the application this token belongs to. This service is only important for VRT, since for TOSCA-MP tokens, this service will always return 'tcmp'. An example with an example token: /tokens/token_36a3bf10-2a09-012d-f108-000c2977b342/app_id

Obviously, for this REST-service no authentication token needs to be appended at the end of the URL.

/tokens/:token/invalidate

This service invalidates and removes an existing token, so it can no longer be used. Example: /tokens/token_36a3bf10-2a09-012d-f108-000c2977b342/invalidate

This service returns 'true' if the token existed and was successfully invalidated, and 'false' for any other tokens.

Obviously, for this REST-service no authentication token needs to be appended at the end of the URL.

/tokens/:token/check

This service checks is the presented token is still valid. The service returns true when the token is valid and false otherwise.

3.1.4 Thumbnail Generation

/thumbs/:id/

This URL generates a thumbnail of the item with identifier ':id' at the position corresponding to frame number 3000 (which corresponds to 2 minutes at a frame rate of 25 frames per second, since $\frac{25\,frames}{s} \times \frac{60\,s}{min} \times 2\,\mathrm{min} = 3000\,frames$). An example use is:

/thumbs/tcmp:81dd5cd0-2a0f-012d-f109-000c2977b342?token=token_36a3bf10-2a09-012d-f108-000c2977b342

The first time it is called, an icon will be returned (generating.png), representing that transcoding using MPlayer is in progress. Once the image is generated, following requests to this URL will return the generated image. From then on, this image can also be requested at:

/Downloader/Download/:id/datastreams/MAM_thumb?token=:token

(Note: the base path for this URL is not the same as for the REST-services since this is not a REST-service, the full URL can also be found through the 'search' REST-service below)

where :id should be replaced by the item identifier and :token should be replaced by a valid token.

Example: **DIFFERENT BASEPATH** /Downloader/Download/tcmp:81dd5cd0-2a0f-012d-f109-000c2977b342/datastreams/MAM_thumb?token=token_36a3bf10-2a09-012d-f108-000c2977b342

/thumbs/:id/frames/:frame

This service works more or less the same as the service /thumbs/:id, except a specific frame number can be provided in the ':frame' parameter instead of the standard frame number 3000.

An example use for frame number 6000 is:

/thumbs/tcmp:81dd5cd0-2a0f-012d-f109-000c2977b342/frames/6000?token=token_36a3bf10-2a09-012d-f108-000c2977b342

Once the image is generated, it can also be requested at:

/Downloader/Download/:id/datastreams/MAM_thumb_:frame?token=:token

(Note: the base path for this URL is not the same as for the REST-services since this is not a REST-service, the full URL can also be found through the 'search' REST-service below)

where :id should be replaced by the item identifier, :frame should be replaced by the frame number that was used for thumbnail generation and :token should be replaced by a valid token. (Note for one exception: when frame number 3000 is explicitly used, the thumbnail datastream will be called MAM_thumb instead of MAM_thumb_3000 and the downloader URL for the standard thumbnail should be used instead of this one.)

Example for frame number 6000: **DIFFERENT BASEPATH** /Downloader/Download/tcmp:81dd5cd0-2a0f-012d-f109-000c2977b342/datastreams/MAM_thumb_6000?token=token_36a3bf10-2a09-012d-f108-000c2977b342

3.1.5 Datastreams

/items/:id/datastreams

This service returns the available datastreams for an item with identifier ':id'.

Example use: /items/tcmp:81dd5cd0-2a0f-012d-f109-000c2977b342/datastreams

At this moment, no token is required yet for this service.

3.1.6 Search by SOLR query

/search/:query/:rows_returned

This service allows to search in SOLR index for items that correspond to a certain query in SOLR format. The SOLR query can be provided with the :query parameter. An example query could be i*. The other parameter :rows_returned allows to specify the number of requested results.

An example use for the query i* with 4 results required is:

/search/i*/4?token=token_36a3bf10-2a09-012d-f108-000c2977b342

The results are represented in the JSON-format (JavaScript Object Notation) such that the results can easily be converted to objects when they are evaluated by a JavaScript-application. The results contain Dublin Core metadata such as the title (only the Dublin Core fields that have a value), information from the underlying database and URL's for downloading the available datastreams (original media, low resolution versions, standard thumbnails with frame number 3000) and metadatastreams (Dublin Core, Technical Metadata, RDF relationships, subtitles,…). Datastreams for thumbnails with a user-specified frame number (different from 3000) are not returned here but their presence can be checked by using the REST-service /items/:id/datastreams for a specific item.

In a future version of this service, the results will appear in descending order by creation date, where the most recent items will appear first. Also, a parameter may allow the user to search only for items that were created after a specified date (instead of the current ':rows_returned' parameter).

3.2 Concept detection

With reference to D2.1 for concrete details on the module.

Description

The concept detection module will output strings (e.g. "Person", "Vehicle") of the names of concepts detected in a video keyframe, or a sequence of video keyframes. The concepts previewed thus far are: **Studio, Indoor, Outdoor, Person, Interview, Male, Female and Anchor.**

API

Input	*Output*
video_id: *(compulsory)* ***keyframe_list****: list of frame numbers indicating keyframes of the video on which to perform concept detection* *(Input encoded in MPEG-7 or AVDP.)*	*Lists of Concepts for each keyframe.* *(Output encoded in MPEG-7 or AVDP .)*

3.3 Quality analysis

With reference to D2.1 for concrete details on the module.

Description

The quality analysis module will output one of the strings “excellent”, “good”, “fair”, “poor” and “bad”, or a numerical value in the range [0,1] to represent the overall quality (based on no-reference metrics) for each keyframe of a video.

API

Input	*Output*
video_id: *(compulsory)* ***keyframe_list****: list of frame numbers indicating keyframes of the video on which to perform quality analysis*	*List of strings to depict overall quality predictions for each keyframe.*
video_id: *(compulsory)* ***keyframe_list****: list of frame numbers indicating keyframes of the video on which to perform quality analysis* *(Input encoded in MPEG-7 or AVDP.)*	*List of numerical values to depict overall quality predictions for each keyframe.* *(Output encoded in MPEG-7 or AVDP.)*

3.4 Full text search and retrieval on multimodal news aggregations

This section describes the facilities available to allow full text search and retrieval on multimodal news aggregations. The challenge is to collect, connect and present data streams from different media, such as television and the Internet, and made of heterogeneous content types such as speech, text and video. In particular, the content from RSS feeds is automatically processed by natural language processing (NLP) tools in order to extract information about named entities (i.e. persons, locations and organisations), journalistic categories (e.g. Politics, Sports, Current Events) and other relevant keywords and topics. Also, speech transcriptions from news broadcasting are matched against RSS feed contents in order to provide aggregations of different information assets around the same topic [Messina 2009]. The information collected about each topic is indexed by the Apache Solr search engine[1], in order to provide a unified search and browse interface to users and applications, as described in the following subsections.

[1] http://lucene.apache.org/solr/

3.4.1 Index field types and properties

The provided indexes are built according to the Solr schema configuration file[2], an XML document containing all the details about which fields are contained in the indexed documents, and how these fields should be treated when adding new documents or querying for the existing ones. Each of the defined index fields can belong to one of the following classes:

- **String:** subtype of *solr.StrField*. String fields are literal character strings with all punctuation, spaces and case preservation;
- **Tags:** subtype of *solr.TextField.* Tags fields are similar to String fields but without case preservation (e.g. "TG1" is equivalent to "tg1");
- **Int:** subtype of *solr.TrieIntField*. Default for integer numeric values;
- **Long:** subtype of *solr.LongField.* Default for long integer numeric values;
- **Date:** subtype of *solr.TrieDateField.* This is a field type to represent any date and time information with millisecond precision. The format derives from the ISO 8601 standard[3] and it is expressed in the form YYYY-MM-DDThh:mm:ssZ at UTC timezone.
- **Text:** subtype of *solr.TextField.* Character strings with stop word removal, word tokenisation and lower case conversion;
- **alphaSort:** subtype of *solr.TextField*. Similar to the 'String' field with the addition of a filter (solr.TrimFilterFactory) for removing any leading or trailing whitespace.

Several options can be set for each field. The more commons are:[4]

- **Indexed** (true/false): true if the field has to be indexed, false otherwise;
- **Stored** (true/false): true if the field has to be retrievable at query time, false otherwise.
- **Multivalued** (true/false): true if the field can appear multiple times in a document (i.e. it can contain more than single value), false otherwise.

3.4.2 News broadcasting full text index

The news broadcasting full text index is made up of a set of three types of metadata, including anagraphic information (i.e. programme start/end date and time, programme title and edition, broadcast channel, and news story identifier), categorical information (i.e. journalistic category to which a given news story belongs to) and content information (i.e. speech transcription and extracted tag clouds), Each metadata item is stored and indexed, so that users can performs queries on each of them. Tables 1 to 3 describes all the included fields for each of the metadata types previously introduced.

[2] http://wiki.apache.org/solr/SchemaXml

[3] http://www.w3.org/TR/NOTE-datetime

[4] http://wiki.apache.org/solr/SchemaXml#Common_field_options

Table 1. News broadcasting index: available fields for anagraphic metadata

Field Name	Field Type	Indexed	Stored	Multivalued	Description
id	String	True	True	False	Univocal identifier of the news story
title	Tags	True	True	False	News programme title (e.g. TG1, TG2,...)
edition	String	False	True	False	News programme edition (e.g. 13:30, 17:00, ...)
channel	Tags	True	True	False	News programme channel (e.g. Rai1, Rai2, ...)
news_id	Int	False	True	False	Sequential number that identifies the position of the news story in the programme (e.g. 1, 2, ...)
start_datetime	Date	True	True	False	Start date and time of the news story
end_datetime	Date	True	True	False	End date and time of the news story

Table 2. News broadcasting index: available fields for categorical metadata

Field Name	Field Type	Indexed	Stored	Multivalued	Description
Category	Tags	True	True	False	Set of journalistic categories to which the news story belongs to (e.g. Sports, Politics, Economy and Finance,...)

Table 3. News broadcasting index: available fields for content metadata

Field Name	Field Type	Indexed	Stored	Multivalued	Description
Transcription	Text	True	True	False	Speech transcription of the news story
Keyword	Tags	True	True	True	Set of keywords extracted from the speech transcription (when required)
Image	String	False	True	False	URL for representative keyframe (when required)

3.4.3 Multimodal news topics full text index

The multimodal news topics full text index allows to search for detected aggregations of TV news stories and RSS feed items. Each document (topic) is described by an XML file (see Figure 1) and indexed by the fields reported in Tables 4 to 6.

Table 4. Multimodal news topics index: available fields for anagraphic metadata

Field Name	Field Type	Indexed	Stored	Multivalued	Description
id	long	True	True	False	Univocal identifier of the topic
title	text	True	True	False	Title of the topic, as detected by the aggregation algorithm
pubdate	Date	True	True	False	Date of the topic
channel	Tags	True	True	True	List of the TV channels for the included news stories
provider	Tags	True	True	True	List of the RSS feed providers for the included RSS items
programme	Tags	True	True	True	List of the titles for the included news stories

Table 5. Multimodal news topics index: available fields for categorical metadata

Field Name	Field Type	Indexed	Stored	Multivalued	Description
Category	Tags	True	True	True	Set of journalistic categories to which the topic belongs to (e.g. Sports, Politics, Economy and Finance,...). Each topic can have multiple categories with different membership degrees.

Table 6. Multimodal news topics index: available fields for content metadata

Field Name	Field Type	Indexed	Stored	Multivalued	Description
Text	Text	True	True	False	Set of the speech transcriptions and item descriptions for the included news stories and RSS items
totalTV	Int	False	True	False	Total number of included TV news stories
totalRSS	Int	False	True	False	Total number of included RSS items
Image	String	False	True	False	URL for representative Web page thumbnail
Keyword	Tags	True	True	True	Set of keywords extracted from the RSS items (i.e. nouns and proper nouns)

```
<daysubjects day="2010-08-05">
        < subject id="187308" rootItem="124226">
             <webshotgrabber>
                 <uuid>4315e1b3-4c79-e12b-ea00-575aa10c0670</uuid>
                 <shoturl>http://10.58.78.56/webshotsgrabber/shot_view.php?uuid=4315e1b3-4c79-e12b-ea00-
575aa10c0670</shoturl>
                 <thumburl>http://10.58.78.56/webshotsgrabber/shot_view.php?thumbnail=true&uuid=4315e1b3-
4c79-e12b-ea00-575aa10c0670</thumburl>
              </webshotgrabber>
             <lastUpdate>2010-08-05 19:25:56+0200</lastUpdate>
             <categories>
                   <category score="12.477439262386676">Giustizia Criminalita Sicurezza</category>
                   <category score="81.15300620942811">Politica Partiti Istituzioni Sindacati</category>
                   <category score="0.2012063208702231">Cronaca</category>
                   <category score="6.168348207314977">Economia Credito Finanza</category>
             </categories>
             <tagclouds>
                    <keyword relevance="0.9301960784313725">berlusconi</keyword>
                    <keyword relevance="0.22431372549019607">bersani</keyword>
                    <keyword relevance="0.18980392156862744">bossi</keyword>
             </tagclouds>
            <items>
                  <item id="1098428"/>
                  <item id="1116748"/>
            </items>
            <videonews>
                   <video id="1234"/>
                   <video id="5678"/>
            </videonews>
      </subject>
</daysubjects>
```

Figure 1. Example of the XML format used for the description of the multimodal news topics.

3.4.4 Named Entities Full Text Index

Named entities are identified through lexical-syntactic patterns (e.g. a continuous succession of proper nouns or proper nouns spaced out by prepositions) combined with external knowledge repositories such as Wikipedia and other ground-truth data sources (e.g. legacy or geospatial databases). This allows us to recognise as many entities as possible (thus augmenting recall) while minimising the number of false positive results (and thus augmenting precision). The named entities are extracted from the content of the RSS feed items, and include the following categories: organisations, locations, persons as defined in the EVALITA Named Entity Recognition task.[5] The named entity full text index includes for each detected entity its category and the publication date from which it was extracted. This allows to build a map of the most important entities in a given period of time and according to their occurrence in that period. Table 7 summarises the types and formats of the index.

[5] http://www.evalita.it/2011/tasks/NER

Table 7. Named Entities full text index: available fields.

Field Name	Field Type	Indexed	Stored	Multivalued	Description
id	long	True	True	False	Univocal identifier of the entity
title	string	true	True	False	Title of the RSS item from which the entity was extracted
day	Date	True	True	False	Publication date of the RSS item from which the entity was extracted
persons	alphaSort	True	False	True	List of person entities extracted from the RSS item
locations	alphaSort	True	False	True	List of location entities extracted from the RSS item
organisations	alphaSort	True	False	True	List of organisation entities extracted from the RSS item

3.4.5 Querying the indexes

As previously introduced, all the described indexes are built with the Apache Solr search engine. Solr supports multiple query syntaxes, as described in the Solr syntax Wki page.[6] Query results can be presented in different formats, among them XML/XSLT, JSON, php, ruby and javabin. An example of a snippet of the XML produced for the named entities index is illustrated in Figure 2. The example shows a query for all the entities extracted with no time limitation (<str name="q">*:*</str>), grouped by their category and ordered according to their occurrence.

[6] http://wiki.apache.org/solr/SolrQuerySyntax

```
<response>
        <lst name="responseHeader">
                <int name="status">0</int>
                <int name="QTime">100</int>
                <lst name="params">
                        <str name="version">2.2</str>
                        <str name="indent">on</str>
                        <str name="rows">10</str>
                        <str name="start">0</str>
                        <str name="q">*:*</str>
                </lst>
        </lst>
        <result name="response" numFound="2295476" start="0" maxScore="1.0"></result>
        <lst name="facet_counts">
                <lst name="facet_queries"/>
                        <lst name="facet_fields">
                                <lst name="persons">
                                        <int name="Mario Monti">7904</int>
                                        <int name="Fornero">3718</int>
                                        <int name="Passera">2126</int>
                                        <int name="Giulio Tremonti">2032</int>
                                </lst>
                                <lst name="organisations">
                                        <int name="Inter">2764</int>
                                        <int name="Guardia di Finanza">1972</int>
                                        <int name="Inps">1651</int>
                                        <int name="Bce">1551</int>
                                </lst>
                                <lst name="locations">
                                        <int name="Toscana">3529</int>
                                        <int name="Libia">3124</int>
                                        <int name="Washington">2779</int>
                                        <int name="Mosca">2301</int>
                                </lst>
…
```

Figure 1. Example of the XML Solr response for the named entities index.

3.5 Automatic Semantic Content Annotation

This section describes the process of automatic annotation of content, making special focus on the use of semantic technologies in the process. It is worth noticing that this process is tightly related to the indexation one, and therefore, we refer to it several times.

Traditional systems are only based on plain textual indexation of content and keyword annotation. However, exploiting the semantics of the user query and its relation with the domain can offer results more accuracy to the results. Such semantic mechanisms need to be incorporated into the content annotation process as well.

Prior to any indexation and search, the content goes through a number of phases where it is normalized and metadata from different vocabularies is added to enrich the information contained in each document. Among other things, the system will add information about entities appearing in the document: these entities usually are domain-independent: people, organizations, locations, dates, numbers, etc. However, it might be the case that some of those entities are ambiguous and more than one can fit into a term (e.g. “Apple” might refer to the technology company or the music record label). In order to perform this task, a linguistic procedure that prepares textual information from documents, applies lemmatization to each recognised word, identifies entities from existing gazetteers, and adds grammatical category information for each word is applied.

The following sections describe the architecture and the main components.

3.5.1 Overview of the architecture

The annotation architecture (Figure 1) is composed by a number of components with the aim of providing Natural Language Processing (NLP) functionalities, namely named entity recognition and content annotation. Such an architecture is designed and implemented on the GATE (General Architecture for Text Engineering) framework. GATE includes a number of basic components to support the processing of textual information (tokenizers, stemmers, POS tagging, etc.) as well as other linguistic resources to deal with several languages.

In addition to those linguistic resources, a number of additional modules have been designed and implemented to support semantic-based named entity recognition for annotation. Furthermore, these elements have been connected to the indexing architecture based on Solr to exploit advanced linguistics and semantic annotation at indexing time. Thus, a set of documents is handled and processed by the annotation components extracting all needed metadata for a proper indexing.

The following sections describe the architecture and the steps and components used by the automatic semantic content annotation system:

- Playence Ontology Manager (POM): providing all basic operations needed to deal with ontologies, namely object creation, update, deletion, and query.
- Named entity recognition by gazetteers module
- Named entity recognition by ontologies module
- Indexing module for indexing documents with annotations (linguistics and semantic).

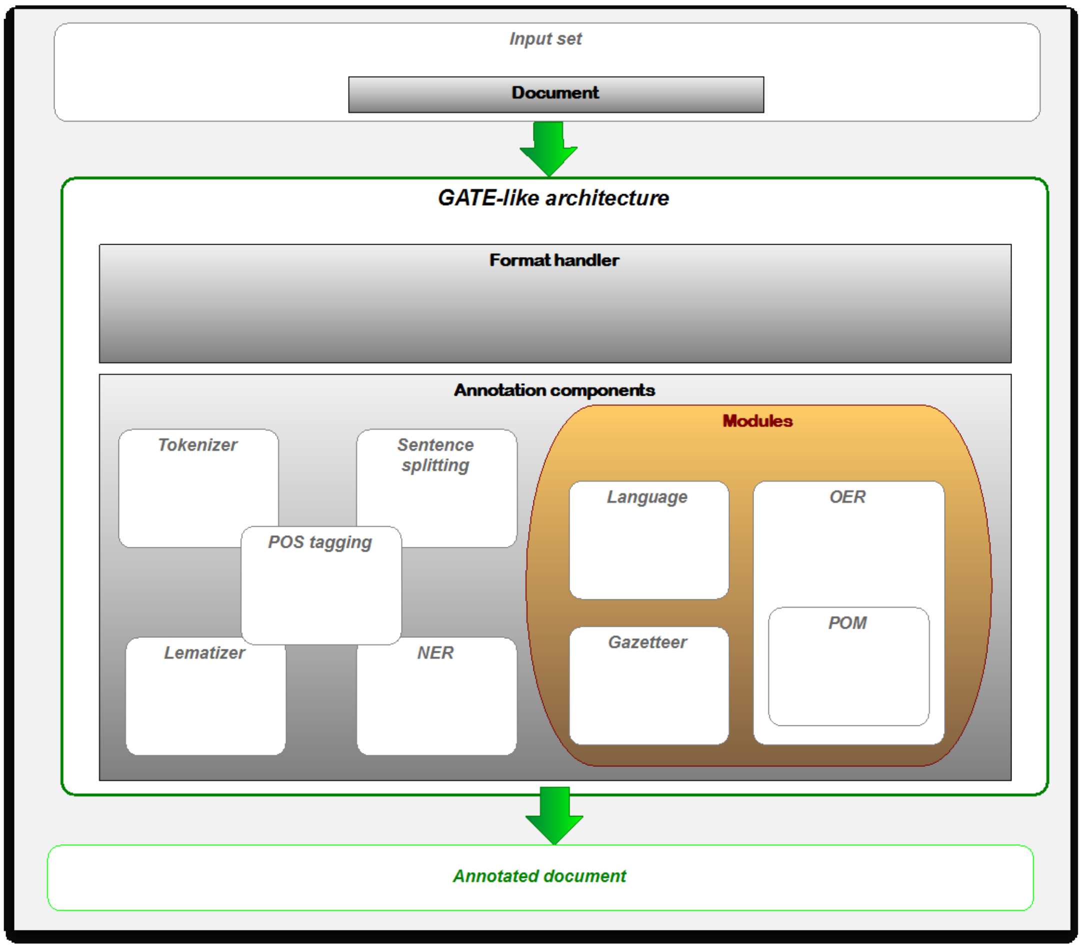


Figure 1 - Overview of the annotation architecture

3.5.2 Components

3.5.2.1 Playence Ontology Manager

The Playence Ontology Manager (Figure 2) is the component providing all basic operations needed to deal with ontologies, namely object creation, update, deletion, and query. This component is designed to work with ontologies in RDF and OWL. The Playence Ontology Manager can access ontologies store in local repositories (file-based ontologies) or external repositories offering access through SPARQL. For the latter, the component implements the Sesame SPARQL API.

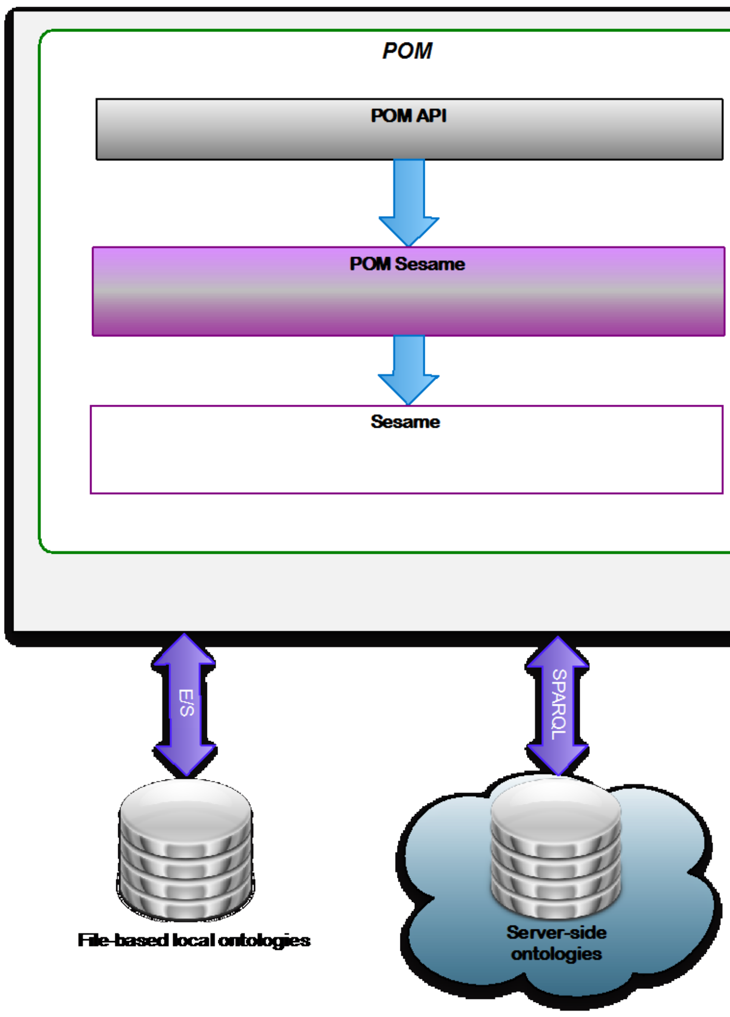


Figure 2 - Playence Ontology Manager

3.5.2.2 Named entity recognition by gazetteer

This component is an extension of the module provided by GATE, where there are some similar or related modules. However, the component was implemented to reduce memory consumption and allowing modifications at runtime (e.g. new non-recognized entities can be fed into the system without stopping the execution). Figure 3 shows the different steps involved.

This component runs in two steps: firstly, a configuration phase to process and index all terms available in the gazetters to ensure performance efficiency; secondly, documents are processed to extract entities and annotating lemmas, tokens, and POS tagging.

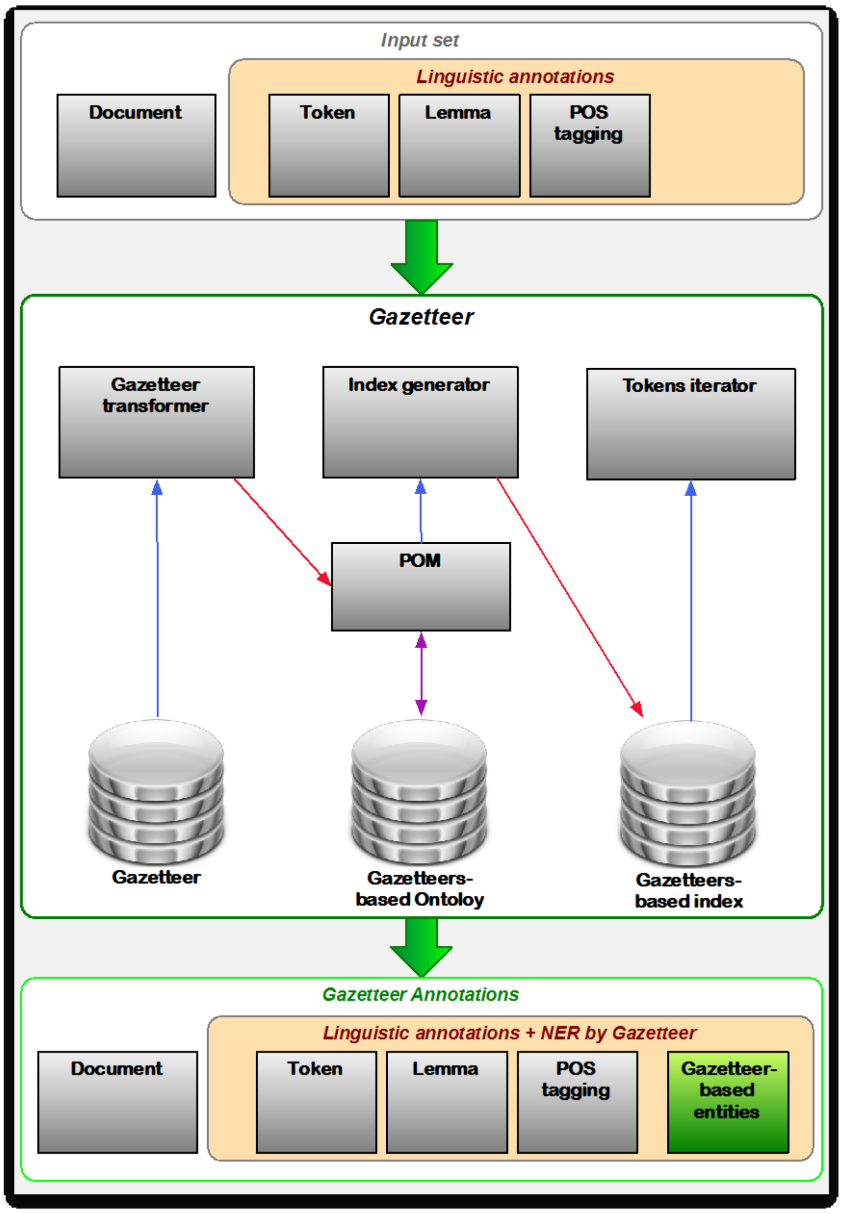


Figure 3 - Named entity recognition by gazetteers

3.5.2.3 Named entity recognition by multiple ontologies

This component is in charge of the Ontology Entity Recognition (OER): it finds and annotates terms within documents corresponding to entities in the domain ontologies (Figure 4).

Similar to the previous named entity recognition by gazetteers, this component requires two steps: firstly, it requires having an index with all the available normalized entities within the ontology; secondly, documents are processed to extract tokens corresponding to entities in the ontology

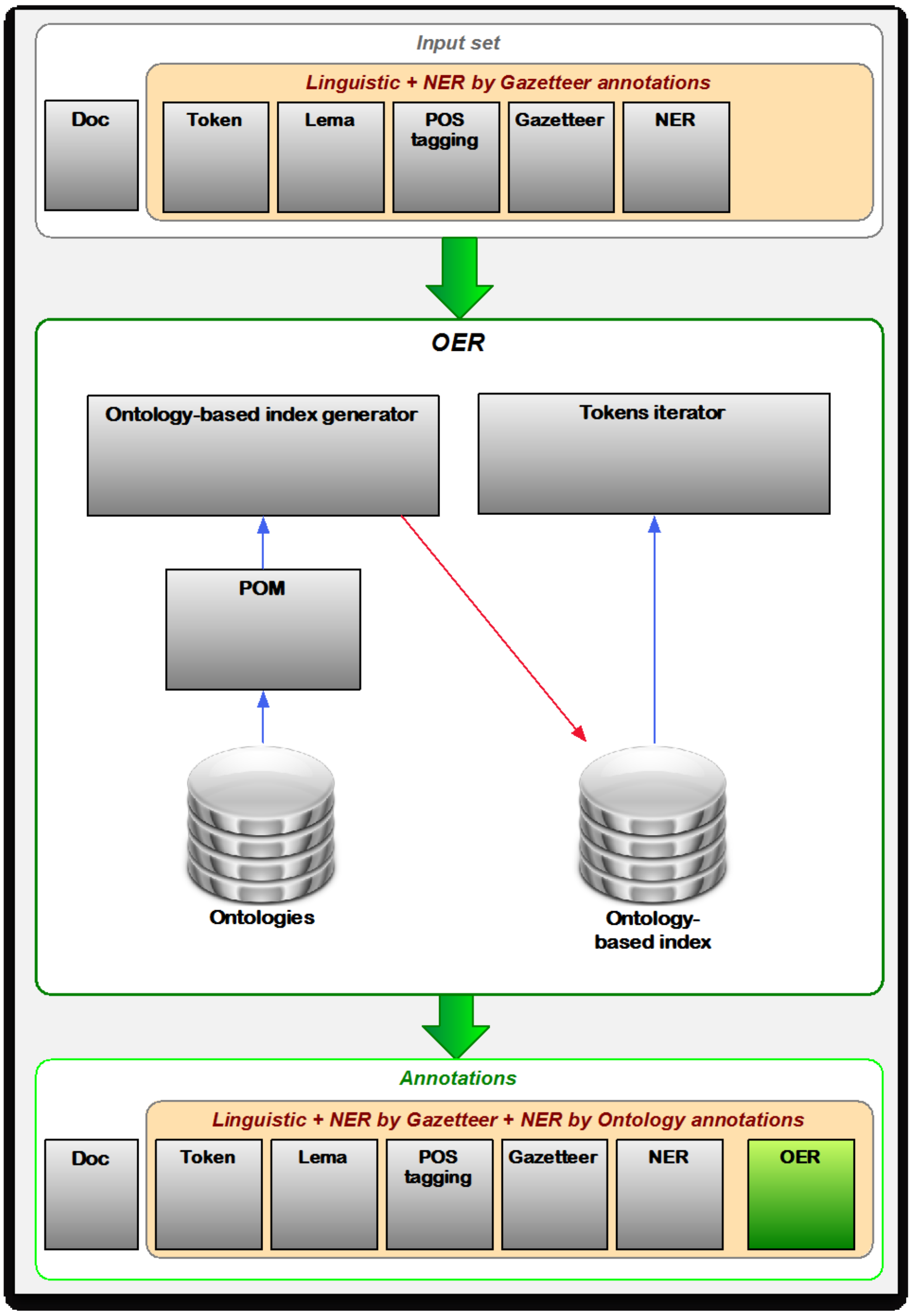


Figure 4 - Named entity recognition by ontologies

3.5.2.4 Indexation of annotation

The indexation process takes all the available annotations extracted from documents in previous steps and creates an index for later retrieval. Technology speaking, it uses Solr as search engine which offers good performance results for huge volumes of data. It supports several formats: rich documents (PDF, Microsoft Office…), connection with databases, Java objects and XMLs (with a structure equal to the index).

As part of the indexation process, we designed an index to contain all available types of annotations described (linguistics, and semantics) including other relevant annotations for retrieval (e.g. relevance)

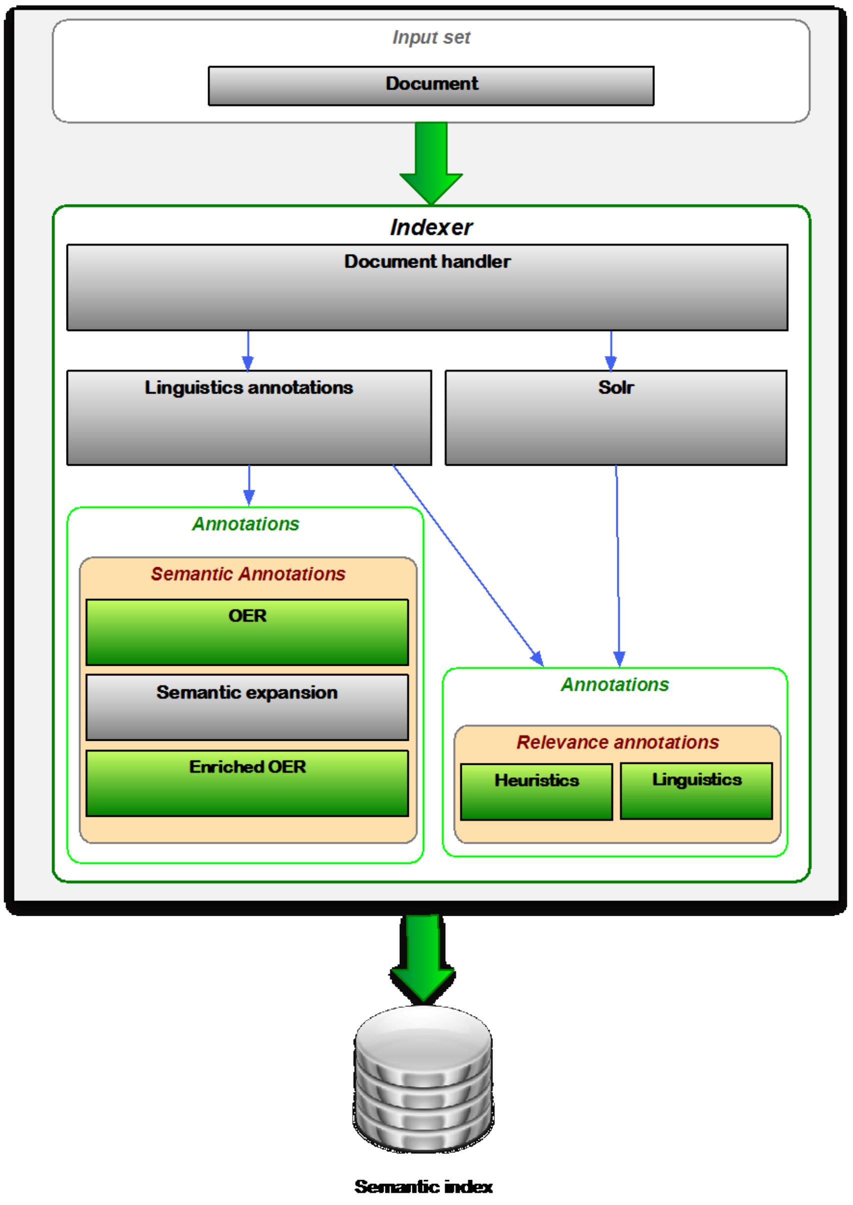


Figure 5 - Indexation of annotations

4 Search Engine

This section describes the building blocks that compose the architecture of the Networked Semantic Search Engine. Among them we can find the following:

- Annotation components
- Index server
- Semantic Search engine

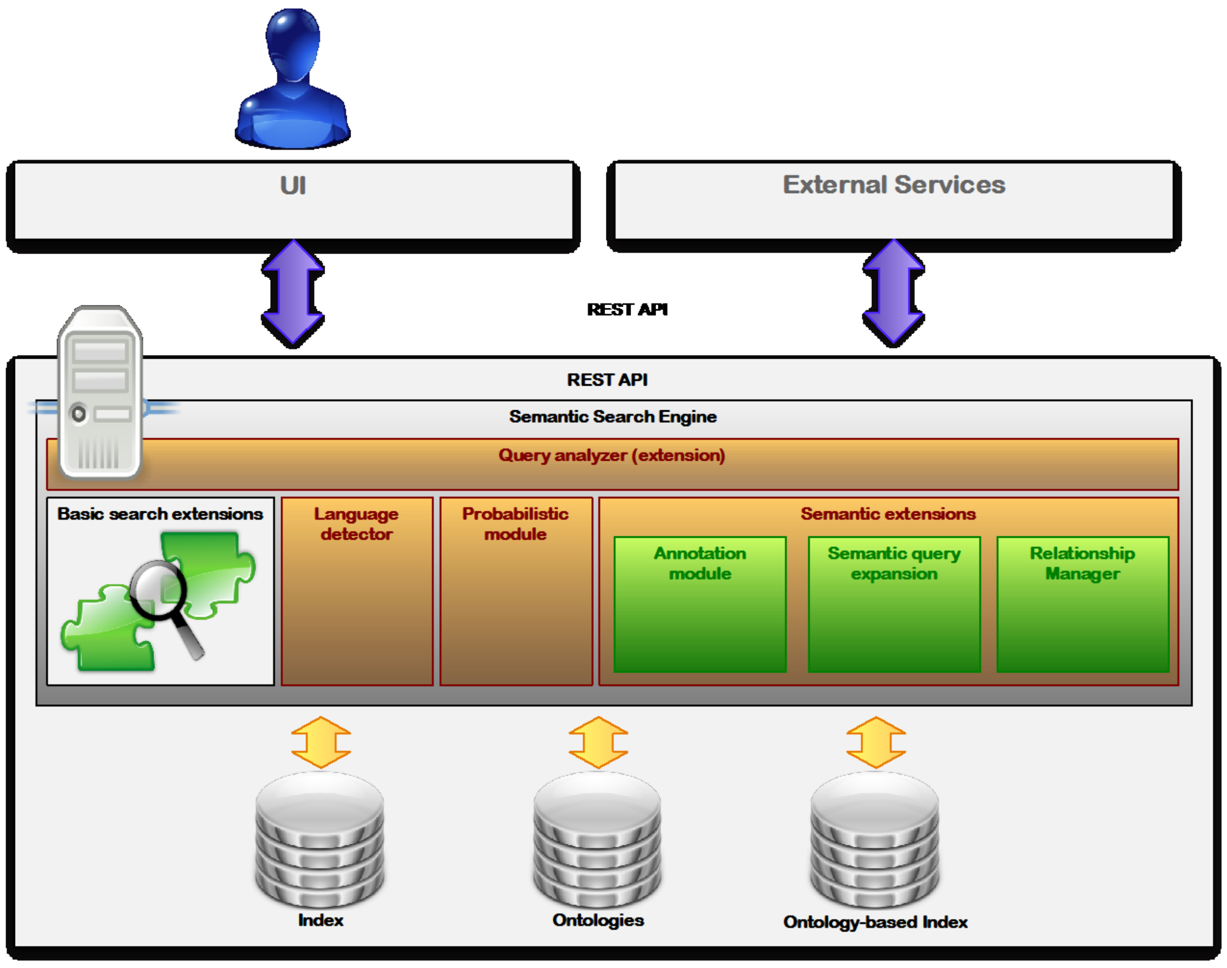
Those components are used at different steps (namely, pre-processing of the documents, the indexation, the search and the further filtering). To perform precise and efficient searches, documents and media resources that are subject to a possible search need to be previously processed, annotated and indexed. Therefore, as any other information retrieval system, the life cycle of the Networked Semantic Search Engine covers the following steps:

- Annotation and Indexation. During this process, all metadata and resources are gathered and semantic information concerning the document is stored to ease its further retrieval. For example, as part of the Semantic Search Engine we also consider the Automated Content Annotation component explained in Section 3.5.
- Search. Processing the user's query we are obtaining semantic metadata and other kind of annotations that allow for searches over the index that gathers related information with the parsed query.

4.1 Core search engine

The Semantic Search Engine is composed of several modules, which apply different mechanisms to analyse and improve user queries by exploiting concepts and relationships within ontologies. The implemented modules are:

- Query Analyzer: this module is responsible to analyse the free text-based query and find which terms are likely to be semantic elements (entities and relationships) within the ontology. By exploiting such information, the semantic query expansion engine can apply different heuristics to expand the query at hand with other related entities.
- Semantic query expansion: this module is in charge of expanding free-text based queries enhanced by the query analyser with other related entities to improve the recall, and in most of the cases, the accuracy of the query retreival.
- Language detector: this module is responsible for detecting the query language to apply the proper linguistic resources of each language
- Relationship Manager: this module provides a visual metaphor of the document content as a relationship graph. It allows both navigation and filtering based on the concepts and relationships.
- Probabilistic Module: this module provides different mechanisms to show summaries of a result set and apply filtering (e.g. an overview of tags within a result set)



The combination of these elements makes it possible to perform document summaries, document classification, syntactic annotation, and identification of named entities and other features. Relevance algorithms are used in the retrieval of related documents based on term appearance frequency and the distance of the terms of the query in these documents. Moreover, within the semantic annotation and indexing, semantic information is analysed and exploited through the use of ontologies and the explicit and implicit information inferred with the use of reasoning engines. This process allows capturing and extract meaning from the user query and thus, the retrieval of documents can be much more precise.

4.2 Query interface

The search engine has a REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. In order to search, the semantic search engine can be invoked via HTTP GET on a selected URL with the query string in the parameter *q*. A number of optional request parameters can be included to control what information is returned. The query is specified using Solr-like query language.

http://<server>:<port>/search-engine/select?q=<<query>>&[OPTIONAL_FIELDS]

4.2.1 Parameters

- Parameter q (required). This parameter is used to express the query to be searched in the set of indexed documents. It is the only mandatory query parameter, and MUST be specified in Lucene query syntax[7].

[7] http://lucene.apache.org/java/3_2_0/queryparsersyntax.html

- Parameter field (optional). It specifies the field in the schema in which the search engine has to look for a matching result.
- Parameter suggestField (optional). Developers may want to use this field to retrieve information from the index to make arithmetic operations in order to return the probabilistic tag cloud.
- Parameter ontologySeed (optional). Text used to calculate data for ontology relationship viewer and the time occurrences.
- Parameter q.op (optional). It specifies the default operator for query expressions, overriding the default operator. Possible values are "AND" or "OR". Default operator is, by default, “OR”.
- Parameter start (optional). This parameter is used to paginate results from a query. When specified, it indicates the offset in the complete result set for the queries where the set of returned documents should begin. The default value is "0".
- Parameter rows (optional). This parameter is used to paginate results from a query. When specified, it indicates the maximum number of documents from the complete result set to return to the client for every request. (You can consider it as the maximum number of result appear in the page) The default value is "10".
- Parameter fq (optional). The "fq" parameter stands for Filter Query. This parameter can be used to specify a query that can be used to restrict the super set of documents that can be returned.
- Parameter fl (optional). This parameter can be used to specify a set of fields to return, limiting the amount of information in the response. When returning the results to the client, only fields in this list will be included. The set of fields returned can be specified as a space (or comma) separated list of field names. The string "score" can be used to indicate that the score of each document for the particular query should be returned as a field, and the string "*" can be used to indicate all stored fields the document has.
- Parameter debugQuery (optional). If this parameter is present (regardless of its value) then additional debugging information will be included in the response, including "explain" info for each of the documents returned. This debugging info is meant for human consumption. The default behaviour is not to include debugging info.
- Parameter debug (optional). Clients may also specify control over individual parts of debugging output by specifying debug= with one of four options:
  - Timing -- Provide debug information about timing of components, etc. only
  - Query -- Provide debug information about the query only
  - Results -- Provide debug information about the results (currently explains)
  - True -- If true, this is the equivalent of &debugQuery=true
- Parameter timeAllowed (optional). The time allowed for a search to finish. This value only applies to the search and not to requests in general. Time is in milliseconds. Values <= 0 mean no time restriction. Partial results may be returned (if there are any).
- Parameter omitHeader (optional). Exclude the header from the returned results. The header contains information about the request, such as the time it took to complete. The default is false.

4.3 Integration with other systems – Mammie + Other Metadata Services

The search engine works in combination with the Mammie platform and other Metadata services offering advanced search functionalities.

In brief, the mammie platform is a web interface to a lightweight media asset management system (MAM). It contains a few basic MAM functions, such as access control, ingest, update and download of media assets and the corresponding metadata streams. All available media will be fed into Mammie that analyses and extracts some metadata in an offline mode. When metadata is ready, Mammie will invoke a specific service available in the REST API to ingest that metadata in the search engine, enrich with other metadata, and recreate the index.

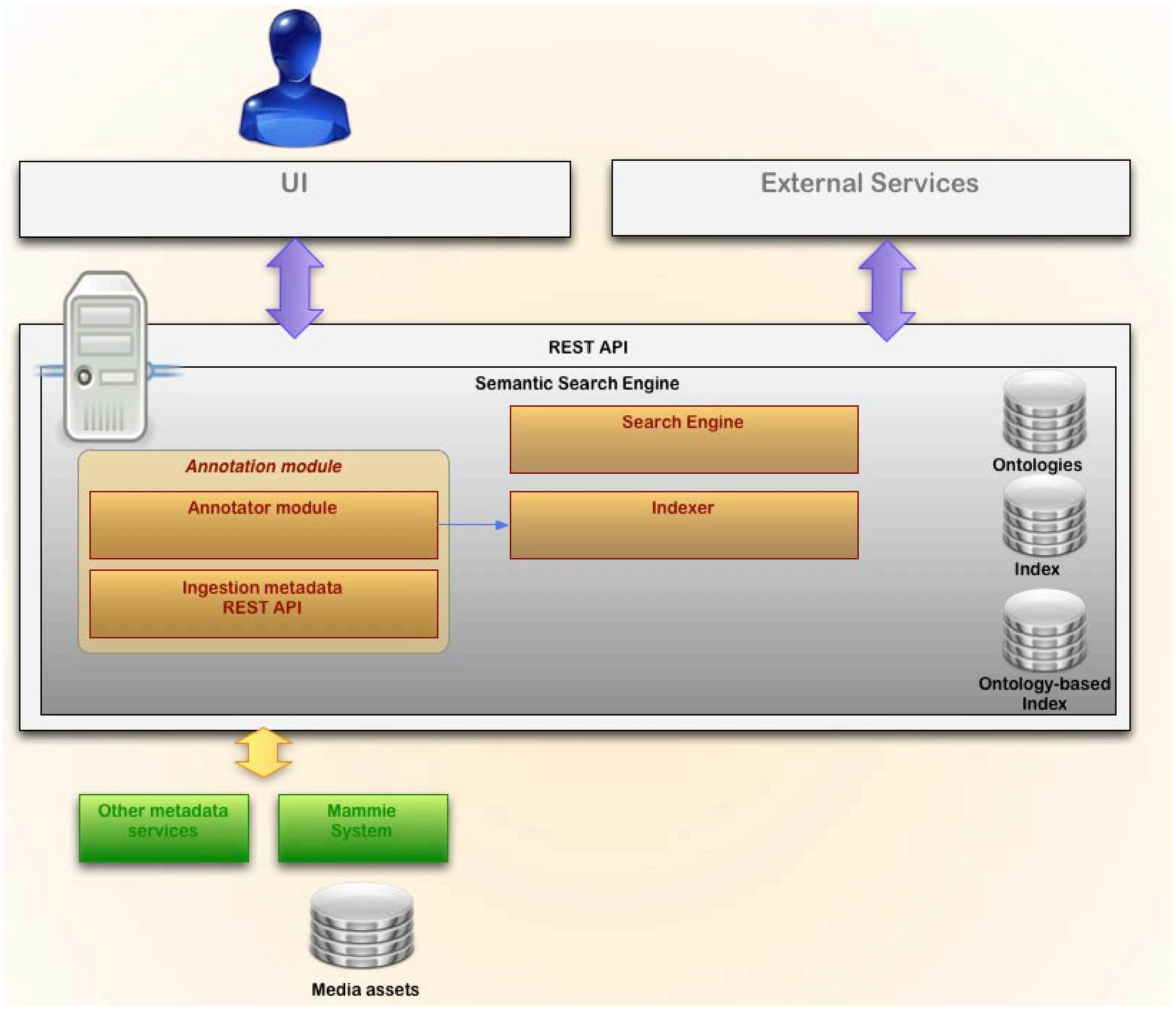


Figure 6 – Integration of the search engine, the mammie system, and other metadata services

On the other hand, there is a number of metadata services already available, namely concept detection, quality analysis, and the semantic content annotator, which offer concrete annotations functionalities and whose annotations are indexed for later retrieval.

4.4 Query example

The following shows an example the output format:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">138</int>
  <lst name="params">
    <str name="indent">on</str>
    <str name="rows">20</str>
    <str name="start">0</str>
    <str name="q">text:(thousand)</str>
    <str name="suggestField">contentOriginal</str>
  </lst>
</lst>
<result name="response" numFound="4" start="0">
  <doc>
    <str name="assetDescription">RCMP North District Federal Drug Enforcement has arrested four people on
drug and weapons charges, including a 43-year-old man who is believed to be at the centre of cocaine
trafficking in Northern BC. On the afternoon of April 12, 2011, David George Massey and two other men were
arrested at a strip mall in downtown Quesnel.   Two kilograms of cocaine and a very large sum of cash were
seized from two vehicles.  Along with Massey, 36-year-old James Darren Peacock of Chilliwack and 34-year-
old Kelly Edward Champagne of Abbotsford were also arrested. Immediately following the arrests, RCMP Drug
Enforcement officers obtained warrants on a home and four businesses in Quesnel.  Several rifles, two
handguns, a large cache of ammunition, body armour, two tasers and another large sum of cash were seized.
...</str>
    <str name="documentType">video</str>
    <str name="fileName">Major  quesnel  cocaine  trafficker  arrrested  16  9  </str>
    <long name="fileSize">28672</long>
    <str name="filetype">WEBM</str>
    <str name="imageSize">720x480</str>
    <str name="internalURI">data/videos/Major_Quesnel_Cocaine_Trafficker_arrrested_16_9.webm</str>
    <str name="lang">en</str>
    <str name="mimetype">video/webm</str>
    <str name="sourceFile">data/videos/Major_Quesnel_Cocaine_Trafficker_arrrested_16_9.webm</str>
    <str name="text">In the the the the trees the also with the same this the eyes it's an amazing variety of
stolen items worth hundreds of thousands of dollars we've got a large five and flat deck truck a trailer with an
excavator in contact or more than forty thousand dollars here alone and in the truck that was stolen back in two
thousand four that we've recovered also very valuable it's a truck that we seized as offense related property at
the time of the initial arrest a cornell man was driving this vehicle inside was about two kilograms of cocaine
also sees at the time of the arrest was a silver volkswagen jetta inside was a large sum of got just a sample of
the exhibits that been taken in this file over here we've got twelve this ammunition two tasers two kilograms of
cocaine and roughly two hundred and fifty thousand dollars the to seized for this file are several prohibited and
restricted weapons two of these long guns are stolen these handguns over here four of them are actually
prohibited on top of that you've got these cartridges these magazines they can take far more than the legally
allowed limit we see here is the cocaine and this at the ... </str>
    <str name="title_feature">Major Quesnel Cocaine Trafficker Arrested With Guns, Drugs, Cash</str>
    <str name="transcription">data/transcriptions/ENCAMajor_QuesnelCocaine_Traffickerarrrested_16-
9mpeg.xml</str>
  </doc>
   . . .
</result>
</response>
```

4.5 Initial user interface

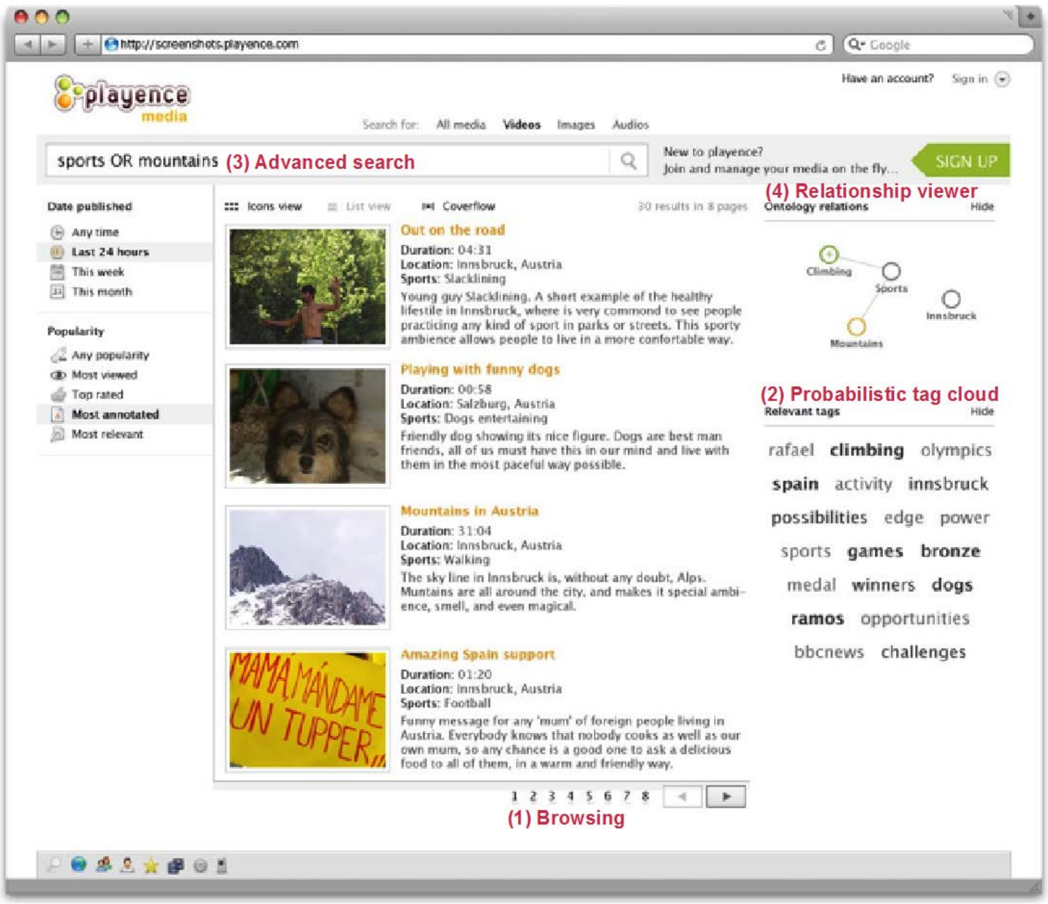
As a presentation layer, the engine initially uses an existing user interface as starting point for visualization. The user interface includes the following functionalities:

(1) Browsing: This enables users to navigate through the results and directly refine the result set by indicating whether terms and concepts should be added or removed from the result set just like in the case of the tag cloud.

(2) Probabilistic tag cloud: This control expands the traditional concept of the tag cloud in two ways. Firstly, tags are calculated using only the results of each individual user query. This provides significantly more accurate tags than when calculated based on the contents of the whole repository. Secondly, query-specific tags can be used to filter the result set, by including or excluding particular tags from the set of results.

(3) Advanced search: In addition to several common advanced search features, it offers the ability to define complex Boolean relations directly applying the concepts and instances in the target repositories and ontologies.

(4) Relationship viewer: The relationship viewer provides a very intuitive, powerful and visual way to filter the result set. Taking as starting point the main concept introduced in the user query, the control presents all related concepts one degree away according to the current information map. The user can then navigate back and forth, exploring the different relationships among concepts and filtering the result set at will until the desired assets are found.



Out on the road
Duration: 04:31
Location: Innsbruck, Austria
Sports: Slacklining
Young guy Slacklining. A short example of the healthy lifestile in Innsbruck, where is very commond to see people practicing any kind of sport in parks or streets. This sporty ambience allows people to live in a more confortable way.

Playing with funny dogs
Duration: 00:58
Location: Salzburg, Austria
Sports: Dogs entertaining
Friendly dog showing its nice figure. Dogs are best man friends, all of us must have this in our mind and live with them in the most paceful way possible.

Mountains in Austria
Duration: 31:04
Location: Innsbruck, Austria
Sports: Walking
The sky line in Innsbruck is, without any doubt, Alps. Muntains are all around the city, and makes it special ambience, smell, and even magical.

Amazing Spain support
Duration: 01:20
Location: Innsbruck, Austria
Sports: Football
Funny message for any 'mum' of foreign people living in Austria. Everybody knows that nobody cooks as well as our own mum, so any chance is a good one to ask a delicious food to all of them, in a warm and friendly way.

Figure 7 - Initial user interface for the networked media search engine

5 Conclusions

This document describes the first version of the networked media search engine and the connection with some of its main components.

For this initial version, the media asset management system (mammie) and the metadata services are directly linked with the semantic search engine to provide an early prototype of the Networked Media Search Engine. As a presentation layer, the engine initially exploits an existing user interface as starting point for visualization

For Y2, the work will focus on two aspects: firstly, integrating other involved components of the TOSCA-MP project, mainly the distributed repository framework and metadata production management framework, when available; secondly, extending the current visualization functionalities with new paradigms for result presentation leading to novel presentation approaches.

6 References

[Messina 2009] A. Messina and M. Montagnuolo. A generalized cross-modal clustering method applied to multimedia news semantic indexing and retrieval. In *Proceedings ofthe 18th international conference on World wide web*,WWW ’09, pages 321–330, 2009.

7 Glossary

Partner Acronyms

DTO	Technicolor, DE
EBU	European Broadcasting Union, CH
FBK	Fondazione Bruno Kessler, FBK
HHI	Heinrich Hertz Institut, Fraunhofer Gesellschaft zur Förderung der Angewandten Forschung e.V., DE
IRT	Institut für Rundfunktechnik GmbH, DE
KULeuven	Katholieke Universiteit Leuven, BE
JRS	JOANNEUM RESEARCH Forschungsgesellschaft mbH, AT
PLY	Playence KG, AT
RAI	Radiotelevisione Italiana S.p.a., IT
VRT	De Vlaamse Radio en Televisieomroeporganisatie NV, BE

Acknowledgement: The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287532.