Deliverable D2.5 Specification of the Linked Media Layer

Jan Bouchner, Lukáš Pikora, Barbora Červenková, Tomáš Kliegr, Jaroslav Kuchař, Ivo Lašek/ UEP
Mathilde Sahuguet, Benoit Huet, José-Luis Redondo Garcia, Raphaël Troncy/ EURECOM
Jan Thomsen, Ali Sarioglu / CONDAT
Lyndon Nixon / STI
Evlampios Apostolidis, Vasileios Mezaris / CERTH
Daniel Stein, Stefan Eickeler / Fraunhofer IAIS

11/10/2013

Work Package 2: Linking hypervideo to Web content

LinkedTV
Television Linked To The Web
Integrated Project (IP)
FP7-ICT-2011-7. Information and Communication Technologies
Grant Agreement Number 287911

Dissemination level	PU
Contractual date of delivery	30/09/2013
Actual date of delivery	11/10/2013
Deliverable number	D2.5
Deliverable name	Specification of the Linked Media Layer
File	`D2.5_Specification*of*the*Linked*Media*Layer.tex`
Nature	Report
Status & version	Reviewed & v1.0
Number of pages	51
WP contributing to the deliverable	2
Task responsible	UEP
Other contributors	EURECOM, CONDAT, STI, CERTH, FRAUNHOFER IAIS
Author(s)	Jan Bouchner, Lukáš Pikora, Barbora Červenková, Tomáš Kliegr, Jaroslav Kuchař, Ivo Lašek/ UEP Mathilde Sahuguet, Benoit Huet, José-Luis Redondo Garcia, Raphaël Troncy/ EURECOM Jan Thomsen, Ali Sarioglu / CONDAT Lyndon Nixon / STI Evlampios Apostolidis, Vasileios Mezaris / CERTH Daniel Stein, Stefan Eickeler / Fraunhofer IAIS
Reviewer	Dorothea Tsatsou
EC Project Officer	Thomas Kuepper
Keywords	Video Enrichment, Linked Media, Crawling, Information Retrieval
Abstract (for dissemination)	This deliverable describes the process of providing a set of prospective enrichment resources for seed video content in the LinkedTV project. Additionally, we participate in the Search and Hyperlinking task of the MediaEval 2013 benchmarking initiative and we report the approaches and results we obtained.

History

Table 1: History of the document

Date	Version	Name	Comment
09/07/2013	v0.1	Kliegr, UEP	deliverable structure
22/08/2013	v0.2	Kliegr, UEP	unstructured search chapter
27/08/2013	v0.3	Sahuguet, EURECOM	MediaCollector chapter
03/09/2013	v0.4	Thomsen, CONDAT	Linked Media Architecture chapter
19/09/2013	v0.5	Sahuguet, EURECOM	MediaEval chapter
26/09/2013	v0.6	Nixon, STI	LSI chapter and introduction
26/09/2013	v0.65	Kliegr, UEP	revise introduction
27/09/2013	v0.7	Kliegr, UEP	add statistics of the index
08/10/2013	v0.8	Kliegr, UEP	address QA comments
09/10/2013	v0.9	Troncy, EURECOM	complete MediaCollector chapter
09/10/2013	v0.95	Troncy, EURECOM	general proof read
10/10/2013	v0.97	Troncy, EURECOM	address second QA comments
11/10/2013	v1.0	Kliegr, UEP	address comments of the scientific coordinator

0 Table of Content

1 Introduction

The vision of "Linked Media" in the LinkedTV project is a vision of a potentially Web scale layer of structured and semantic metadata about media items, connected to the Web of Data, so that agents can determine significant links between different online media resources.

Today's Web is increasingly non-textual in content, yet Web systems are still widely tuned to the processing of textual information from Web pages, even when considering the widely used search engines or in fact the hyperlinking mechanism that so fundamentally underlies the concept of the Web. Integrating all of this, rapidly expanding in scale, non-textual content into the existing Web infrastructure is a significant challenge which is still being addressed in work on media analysis, annotation, management, search and retrieval.

As Television and the Web converge, this disjunction between the traditional end user consumption of audiovisual streams and the interaction with online media resources within web sites becomes very clear. LinkedTV aims to seamlessly integrate content from both sources into a single interactive experience, which requires that cross-network media shares a common description and an agreed means to access and process that description, and that a LinkedTV system is able to compare media item descriptions in order to generate links between media that can be used in enriching a TV program or enabling end users to browse within a collection of media items from different sources.

We have proposed a set of principles to underlie Linked Media [19]:

1. Web media descriptions need a common representation of media structure
2. Web media descriptions need a common representation of media content
3. Web media descriptions need to use a media ontology which supports description of both the structure and content of media
4. The descriptions of media in terms of common representations of structure and content are the basis for deriving links across media on the Web

In the same presentation, a set of specifications are proposed that can fulfil the requirements of each principle[1]:

1. Media structure can be represented using W3C Media Fragment URIs
2. Media content can be represented using the Linked Data URI concept space
3. Media descriptions can be based on the W3C Ontology for Media Resources, the W3C Open Annotation Model (core) and some extensions
4. Media hyperlinking can be derived from the semantic analysis of media descriptions, e.g. concept similarity or named entity detection

LinkedTV contributes to the vision of Linked Media, since the generation of links between media resources is a fundamental part of the LinkedTV experience of enriched TV programming. It has taken up the use of the specifications mentioned above, and proposes a new media ontology based on re-use of existing media ontologies where appropriate and extended with the requisite additional information needed for linking media (the LinkedTV Core ontology). Finally, it is researching how media hyperlinks can be derived on the basis of these shared representations of media structure and content, part of which is to generate relevant semantic descriptions of existing online media resources (whether from online media platforms such as YouTube, social networks such as Twitter and Facebook, or white-listed web sites from the scenarios themselves as specified in [2]) to permit their comparison with the seed media fragments.

While previous deliverables have addressed the common representations of media structure and content for LinkedTV and introduced the LinkedTV ontology, this deliverable focuses on the work done to build upon this and provide Web services for exposing online media resources to the "Linked Media layer"[2] and systems for processing the media metadata in that layer, in order to derive representative links to relevant and conceptually related media for enriching a seed media fragment.

[1] Slides available at http://www.slideshare.net/linkedtv/www-linked-media-keynote

[2] We define the "Linked Media Layer" as a decentralised web-based platform based on the distribution of web media annotations and associated conceptual models over the web. It extends the Linked Data model of web-scale interlinking of concepts and metadata about concepts, with media annotations which tie online media with Linked Data concepts. See also `http://www.linkedtv.eu/development/linked-media-layer/`.

According to the specification provided in the preceding deliverable D2.3[1], a set of technologies for retrieving the enrichment content from the web was developed and integrated via the LinkedTV Linked Media Layer, which is covered in (Section 2). The two fundamental resources for obtaining this content are regular web sites and a web service access to popular social networks and media sharing platforms such as Flickr, Instagram, Twitter, Facebook, Google+ or YouTube. To exploit these resources, LinkedTV developed a so-called **Unstructured Search Module** presented in Section 3 which crawls a list of web sites provided by the LinkedTV content partners and creates a full-text index. This system was crafted to include also multimedia content (images, podcasts, video) available on those web sites.

Social networks and media sharing platforms that expose web APIs are harnessed through the so-called **Media Collector** described in Section 4 and the complementary **Linked Services Infrastructure** described in Section 5. Since supporting a new web site or a web API requires some amount of customization and consumes computational and storage resources, the set of resources covered was prioritized to include those relevant to the two LinkedTV scenarios – RBB and Sound & Vision. To this end, the respective chapters include also illustrative examples from these domains. The scientifically relevant algorithms developed for or already incorporated into the LinkedTV services have been put to test within the MediaEval benchmarking initiative (Section 6). The span of this section overreaches WP2 as also the approaches to video and speech processing developed within WP1 are discussed. A brief summary of the overall state of the WP2 integration activities is provided in the conclusion.

2 Linked Media Layer Architecture

The Linked Media Layer is an integrated collection of components, metadata and services for managing, adding, updating and querying of enriched annotations of media resources and media fragments. This chapter describes the architecture, which underlies the realization of the Linked Media Layer as the central component of the LinkedTV Platform. The Deliverable D5.4 [5] having a “Confidential” dissemination level, the overview presented in this chapter provides the minimal description between the Linked Media Layer, the individual “enrichment services” and the rest of the LinkedTV workflow.

2.1 Linked Media Workflow

The Linked Media Layer has to support the Linked Media Workflow as defined and realized in LinkedTV. In itself, it does not contain any components which generate data, but it rather provides the means to store, access, search, link, update or manage that data. Figure 1 shows the basic Linked Media Workflow from ingestion, analysis, to serialization and enrichment. This workflow generates and aggregates metadata upon which personalization can be performed. The playout filters and adapts the linked media data to context and personal profiles. Each of these steps consists in itself of multiple sub-processes. The Linked Media Layer also allows interaction with these sub-processes on an individual basis. A more detailed description of the integration of these different process steps within the Linked Media Layer is provided in the Deliverable D5.4 [5].

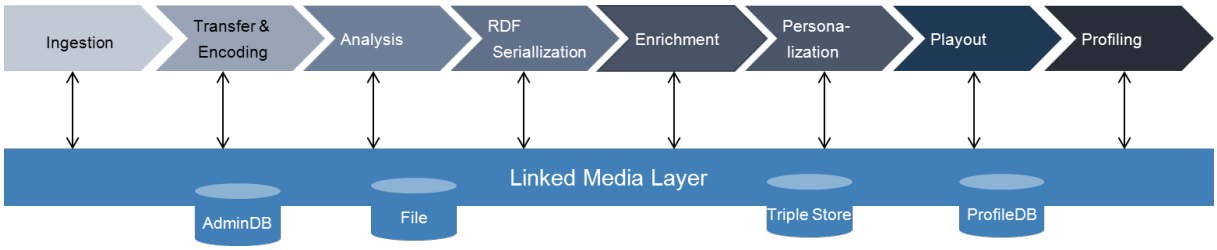

Figure 1: Linked Media Workflow

2.2 Linked Media Layer Architecture

The Linked Media Layer provides all services for accessing the persistency components such as storage for video content, annotations, ontologies and user data as well as functions for retrieval, data access and management. It also provides the central integration communication infrastructure. Thus, the Linked Media Layer consists itself of three main sub-layers (Figure 1):

- **Repository Layer**: includes all persistency storage facilities for videos, metadata, ontologies, personalization and context data, as well as for application specific data.
- **Integration Layer**: includes the central integration and communication component called the *Linked Media Service Bus*.
- **Service Layer**: provides API services for retrieval, accessing and manipulating data stored in the repositories.

All layers are conceptual layers but not physical layers. The different components can be (and actually are) distributed over various different locations, hosted by different partners. They could also be hosted by cloud services.

2.2.1 Repository Layer

The Repository Layer is usually located on the server side of the platform. However, some smaller data sets, such as for the user profile or history can also be stored on the client side. For the Metadata Repository, the ontologies, user and social data, an RDF repository (or Triple Store) is provided which also offers other types of storage (SQL, XML, Documents, Free Text). The triple store used in LinkedTV is OpenLink Virtuoso[3].

[3] http://virtuoso.openlinksw.com/

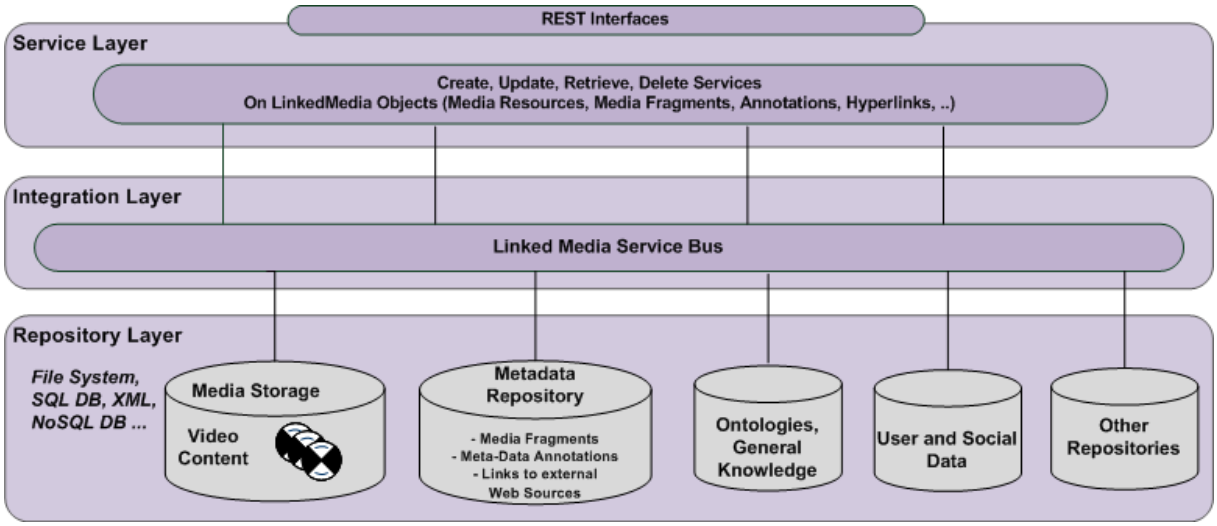

Figure 2: Linked Media Layer

2.2.2 Media Storage

In general, the Linked Media Layer in LinkedTV does not require that media resources which are being analyzed and annotated are stored within the platform itself, but can rather be accessed either through URIs over the Internet or by connecting to media archives (e.g. of a broadcaster). However, the LinkedTV platform does support the storage of media resources which is needed in particular for the analysis part.

2.2.3 Metadata Repository

The Metadata Repository is the central part of the LinkedTV platform, as it contains the metadata aggregated through the different processes for analysis, fragment detection and annotation. At present, it contains the following data:

- **General Metadata**: Metadata which applies to the base resource as a whole, such as title, duration, owner, author, format, genre, etc. The General Metadata is a subset of the EBUCore Metadata Set[4]. Table 2.2.3 lists the attributes used. This metadata is not in RDF format as it is collected before the RDF-ization steps. In the LinkedTV platform, it is stored within an SQL Database, however NoSQL Databases would be also appropriate. Access to the general metadata is provided through `http://api.linkedtv.eu/mediaresource/`.
- **Annotated Media Fragments Metadata**: The annotated and enriched Media Fragments are all those data which are generated by the Metadata Conversion Service (tv2rdf) [4] and the Enrichment Service (TVEnrichment) (Section 4 of this deliverable). They are provided in RDF format and stored within the RDF Repository. Additionally, annotations can be added and managed via a so-called Editor Tool [3]. The RDF Repository is accessible at `http://data.linkedtv.eu` while the SPARQL interface is at `http://data.linkedtv.eu/sparql`.

2.2.4 Integration Layer

The Integration Layer provides the components and services for the integration of the different LinkedTV components and the communication between them. The main purpose is to ensure an open, extendable, distributed scalable integration backbone for LinkedTV in order to support the Linked Media Workflow but also different workflows and business models. Since the analysis process, data provision and applications can be located at different physical places, it is also possible to flexibly integrate third party content or service providers.

[4] `http://tech.ebu.ch/docs/tech/tech3293v1_4.pdf`

Attribute	Description
Identifier	The unique id (key) of the Media Resource (this is a UUID)
TitleName	String. The title of the Media Resource (if not available, by default the file-name)
Language	URI | String
Locator	The logical address at which the resource can be accessed (e.g. a URL or a DVB URI).
DateInserted	DateTime of import into the media repository
Collection	The collection where the media resource belongs to (in our case the original ftp directory)
Publisher	URI, String (e.g. 'rbb', http://www.rbb-online.de, 'dbpedia:rbb')
FrameSizeWidth	Decimal (e.g. 720)
FrameSizeHeight	Decimal (e.g. 480)
Compression	URI, String (e.g. ogg, mpeg4)
Format	URI, String; MIME Type
Duration	Decimal (in seconds)
hasRelations	MediaResourceRelation. Handles the relations to other related resources, like EXMARaLDA file, playout versions in different formats or metadata files
LinkedTVStatus	Not part of Ontology for Media Resource, used to describe the workflow status of this media resource in LinkedTV (LINKEDTV_STATUS)

Table 2: Object Model of the AdminDB of the LinkedTV Platform

The main component of the Integration Layer is the Linked Media Service Bus (LMSB). The Linked Media Service Bus is based on general Enterprise Service Bus (ESB) technology and provides the following features, to name just the most relevant ones in the Linked Media context:

- support of Enterprise Integration Patterns (EIP)[5]
- message routing
- publish/subscribe messaging
- automatic type conversion (custom converters could be added)
- numerous different types of endpoints (e.g. file system, ftp, email, RSS, REST services)
- logging and monitoring

The LinkedTV LMSB is realized on top of the Apache Camel[6].

2.2.5 Service Layer

The Service Layer of the Linked Media Layer includes the general LinkedTV services which are provided for the different components of the LinkedTV Platform for analysis, annotation, enrichment, or personalization, as well as external clients such as the Editor Tool or the LinkedTV Player. The Service Layer is completely realized through a set of REST services.

The two main repositories, the Administration Database and the RDF Repository, are exposed via two different REST Base URLs at `http://api.linkedtv.eu` and `http://data.linkedtv.eu`. The first one gives access only to complete media resources, whereas the second one gives access to the RDF data of complete media resources as well as media fragments, annotations, and more. Both REST APIs ensure that the LinkedTV Platform is an open platform which can be used by other clients as well as integrated with other IPTV Environments.

2.2.6 Administration REST API

The base URL for the REST API URIs is `http://api.linkedtv.eu`. Table 3 lists the REST API functions for getting and setting complete media resources. This also includes the management of files related to a media resource, such as analysis result files, RDF files, etc.

[5] `http://www.enterpriseintegrationpatterns.com/`
[6] http://camel.apache.org/

Operation	HTTP Method	URI Pattern
Get all media resources, Filter by status, Filter by publisher	GET	/mediaresource ?status ?publisher
Create a media resource	POST	/mediaresource
Get a media resource	GET	/mediaresource/{id}
Replace a media resource	PUT	/mediaresource/{id}
Delete a media resource	DELETE	/mediaresource/{id}
Get all relations for a media resource	GET	/mediaresource/{id}/relation
Create a media resource relation	POST	/mediaresource/{id}/relation
Get a media resource relation	GET	/mediaresource/{id}/relation/{id}
Replace a media resource relation	PUT	/mediaresource/{id}/relation/{id}
Delete a media resource relation	DELETE	/mediaresource/{id}/relation/{id}

Table 3: Administration REST API functions of the Linked Media Layer

A JSON representation of the response can be retrieved by adding the extension .json to the request (e.g. `http://api.linkedtv.eu/mediaresource.json`). The deliverable D5.4 provides a detailed description of the REST API functions [5].

2.2.7 RDF Repository REST API

The base URL for the REST API URIs is `http://data.linkedtv.eu`. Table 4 lists the collection of REST API calls to access the RDF Repository.

Operation	HTTP Method	URI Pattern
Get all media resources	GET	/mediaresource
Get a specific media resource	GET	/mediaresource/{id}
Get all media fragments of a media resource	GET	/mediaresource/{id}/mediafragment OR /mediafragment?isFragmentOf={mediaresourceURI}]
Get all media fragments of a media resource for a specific time interval	GET	http://data.linkedtv.eu/mediafragment?isFragmentOf={mediaresourceURI}&min-temporalStart={startTime}&max-temporalEnd={endTime}
Get all shots of a media resource	GET	http://data.linkedtv.eu/mediaresource/{ID}/shot
Get all annotations of media resource Full view	GET	/mediaresource/{id}/annotation?_view=full
Get all media fragments	GET	/mediafragment
Get all shots	GET	/shot
Get all annotations	GET	/annotation
Get spatial objects	GET	/spatialobject
Add an annotation to a media fragment	PUT	/mediaresource/{id}/annotation

Table 4: Triple Store REST API functions of the Linked Media Layer

Besides these specific REST calls (more will be added in future releases) the general SPARQL endpoint at `http://data.linkedtv.eu/sparql` allows to retrieve arbitrary data from the Triple Store via SPARQL queries. The response can be requested in various formats through indicating the content type. The following content types are supported:

- text/javascript, application/javascript: JSONP representations (on some browsers)
- text/plain: Plain text representation
- application/rdf+xml, text/turtle: RDF/XML or Turtle representation
- application/json: JSON representation

– text/xml, application/xml: XML representation

– text/html: model rendered by stylesheet applied to XML rendering

The REST API for the RDF triple graph is implemented with Elda[7], a JAVA implementation of the Linked Data API specification[8]. The Linked Data API provides a configurable way to access RDF data using simple RESTful URLs that are translated into queries to the SPARQL endpoint. For a more detailed description of the RDF Repository API, see the deliverable D5.4 [5].

[7] http://www.epimorphics.com/web/tools/elda.html
[8] http://code.google.com/p/linked-data-api/

3 Unstructured Search Module

This section describes the progress made on the "Unstructured Search Module", which serves within the LinkedTV project for retrieval of additional content from a curated list of web sites. The functionality, dependencies, and the main technologies used for the implementation described here were specified in the Deliverable D2.3 [1] (Section 3.1). The Unstructured search module performs crawling, a methodical, automated manner to collect data from Web resources. It also provides indexing of the crawled data, so that these can be quickly retrieved in response to a query. The query interface is exposed via REST web services.

This section is organized as follows: section 3.1 focuses on a detailed description of the technological background. The interconnection of the individual components into a complete workflow is covered in section 3.2. The extensions and new modules developed within LinkedTV are described in section 3.3. The index structure is described in section 3.4, and the API of the Unstructured Search Module in section 3.5. We present some benchmark results in section 3.6 and future work in section 3.7.

3.1 Frameworks Used

The Unstructured Search Module is developed on top of the Apache Nutch, Apache Solr and Apache HBase frameworks. These frameworks are used to gather information from the white-listed web sites (Apache Nutch), store them (Apache HBase) and index them for fast retrieval (Apache Solr). All these frameworks are developed and maintained by the non-profit *Apache Foundation* and released under the open-source Apache License[9].

The selected frameworks represent the de-facto standard in the open-source community, which provides a rich set of features as well as the availability of free support via the mailing lists. Nevertheless, it was necessary to perform some customization to cater for the specific LinkedTV needs, mainly related to the indexing of multimedia content (videos, podcasts). The implemented extensions are described in section 3.3 and the foreseen extensions in section 3.7.

3.1.1 Apache Nutch

Within the Unstructured Search Module, Apache Nutch (`nutch.apache.org`) is used as a highly extensible and scalable web crawler. Apache Nutch can run on a single machine, but it can also run in a cluster (Apache Hadoop). Apache Hadoop allows for the distributed processing of large data sets across clusters of computers. Hadoop, Nutch core and all their modules and plugins are written in Java. Plugin is a set of extensions which correspond to the extension points defined by Nutch. Some of extension points are e.g.:

- IndexingFilter
- Parser
- HtmlParseFilter
- URLFilter

Apache Nutch (in version 2.x) implements an abstract storage layer, which enables to use a data store of choice. This abstraction is handled by Apache Gora (`gora.apache.org`), a new source open source framework providing an in-memory data model and persistence for big data.

3.1.2 Apache Solr

Apache Solr (`solr.apache.org`) is an advanced search platform based on Apache Lucene (`lucene.apache.org`). Major features are powerful full-text search, faceted search, near real-time indexing, database integration and more. Additionally, Solr provides comprehensive web-based administration interfaces.

Apache Lucene (`lucene.apache.org`) is a high-performance text search engine. It is provided as a library written in Java. Lucene is used by Solr at its core for full-text indexing and search. Solr is also written in Java and runs as a standalone full-text search server within a servlet container, for example in Apache Tomcat (`tomcat.apache.org`) .

Solr provides a REST API, which can be called over HTTP from any programming language or platform.

[9] http://www.apache.org/licenses/LICENSE-2.0.html

3.1.3 Apache HBase

HBase (`http://hbase.apache.org/`) is non-relational, distributed, column-oriented database for big data running on top of HDFS[10] and providing BigTable-like capabilities for Hadoop. It is useful for random, realtime read/write access to Big Data because BigTable can be used both as an input source and as an output target for MapReduce jobs.

3.2 Workflow

Section 3.1 described the frameworks on which the Unstructured Search Module is based on. This section gives a step-by-step account of the crawling process.

3.2.1 White-list

To start indexing data, Nutch requires a seed file. This *seed* file contains the urls of web pages which should be crawled. The seed is an ordinary text file with a list of webpages, one url per line, e.g.:

```
http://www.kulturradio.de/
http://www.fritz.de/
http://www.funkhauseuropa.de/
```

After the seed file has been specified, it is possible to restrict the crawling to some domains, or more generally, to url patterns. This is done by defining regular expressions in a configuration file `regex-urlfilter.txt`. For example, if crawling is limited to the domains defined above, the specification is as follows:

```
+^http://([a-z0-9]*.)*kulturradio.de/
+^http://([a-z0-9]*.)*fritz.de/
+^http://([a-z0-9]*.)*funkhauseuropa.de/
```

The lines above will instruct Nutch to include all url in these three domains (respecting `robots.txt` file). Example of `robots.txt` disallowing Nutch to crawl the website:

```
User-agent: Nutch
Disallow: /
```

3.2.2 Crawling

Once the *seed* and *regex-urlfilter* configuration files have been specified, the crawling can start. All of the steps described below (see schema 3) are provided by Java classes within the packages:

- org.apache.nutch.crawl
- org.apache.nutch.fetcher
- org.apache.nutch.parse

The workflow runs within Nutch crawl script called *crawl* with four input parameters

```
crawl <seedDir> <crawlID> <solrURL> <numberOfRounds>
```

where:

- **seedDir** path to the directory with seed files,
- **crawlID** the id to prefix storage schemas to operate on,
- **solrURL** url where Solr index is located,
- **numberOfRounds** iterations of *generate - fetch - parse - update* cycle (earlier versions used *depth* parameter).

Example:

```
crawl /opt/ir/nutch/urls webpage http://127.0.0.1/solr/ 10
```

[10] Hadoop Distributed Filesystem

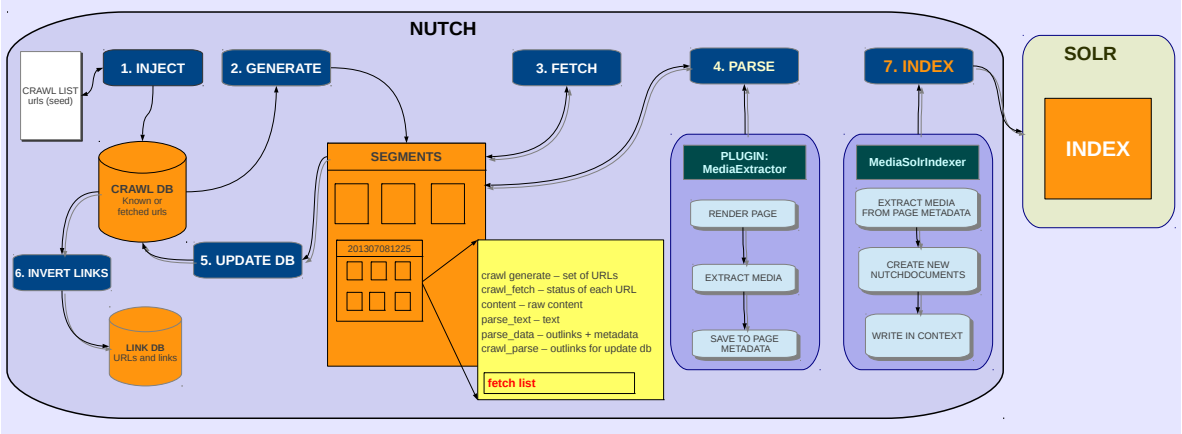

Figure 3: Crawling workflow.

3.2.2.1 Inject This step takes the urls from the seed file and adds them to a *crawlDB* (backend datastore). In the Unstructured Search Module, an HBase table is used as the backend datastore. These urls are stored with information about fetch schedule, fetch status and metadata.

3.2.2.2 Generate This step generates a subset of data from a crawlDB, which should be fetched. The main purpose is to generate a *segment* which tells Nutch which and how to fetch the urls.

Segment is a set of web pages that are fetched and indexed as a unit. It holds all information used for generate/fetch/update crawling cycle (for example `fetchlist` which is a file that lists the set of pages to be fetched). There is a number of parameters that can be specified for this step. Empirically, one of the most important parameters is the fetch list size – number of top urls to be selected. The result is a fetchlist (list of urls to be fetched) which is stored in the segment directory.

3.2.2.3 Fetch This step fetches the urls generated in the previous step. Nutch fetches the resources using a defined protocol (for example http) and places the data back to a segment directory under the timestamp when the url was created. One of the key parameters of this step is the number of fetching threads per task. This step usually is the most time-consuming one.

3.2.2.4 Parse The fetched data are parsed within this step. The parser processes the textual content of each web page and extracts the links. The output of the parser is the parsed text, metadata, and the newly acquired urls. This step was extended with our own solution for crawling multimedia content (section 3.3)

3.2.2.5 Updatedb Within this step, the list of urls to be fetched is updated with the newly acquired urls (outlinks) during the parsing step. After this, the crawling process continues to cycle through generate, fetch, parse and updatedb steps. A pre-specified number of iterations can be made.

3.2.2.6 Solrindex At the end of the workflow described above, the Solrindex step is initiated to save crawled data to an index. Nutch delegates all data created in the parsing step to the IndexingFilter extension, which generates the data to be indexed. The output of the IndexingFilter extension is a NutchDocument. This document is again delegated to Nutch.

Nutch then decides which data should be indexed based on the *mapping file*. The mapping file (see section 3.3.4) defines which NutchDocument fields will be mapped to which SolrDocument fields. The implementation of the mapping file in Nutch was too restricted for LinkedTV purposes, leading to the need to develop an extension/replacement for this step as described in section 3.3.4.

3.2.3 Refreshing index

One of the main requirements in the LinkedTV scenarios is keeping the index in sync with the white-listed web sites. For this purpose, Nutch implements the *Adaptive Fetch Schedule*. This feature allows

to adapt the revisit frequency for a particular page to the rhythm of changes in the page content.

When a new url is added to the crawlDB, it is initially set to be re-fetched at the default interval. The next time the page is visited, the Adaptive Fetch Schedule will increase the interval if the page has not changed and decrease it if the page has changed. The maximum and a minimum revisit interval are configuration parameters. Over the time, the pages that change often will tend to be re-indexed more frequently than the ones that do not change frequently.

3.3 Developed Extensions

3.3.1 Multimedia Indexing Plugin for Nutch

Nutch, by default, does not offer features for multimedia parsing. To the best of our knowledge, there is no plugin that would allow an easy integration and retrieval of multimedia content. As a consequence, the corresponding extension had to be designed for LinkedTV purposes. This lead to substantial changes in the index structure, because Nutch follows a single web page - single index entry paradigm, which no longer holds if images, podcasts, and videos featured on a page become objects of standalone importance.

The main class of the developed plugin is *MediaParseFilter*. This class overrides the default Nutch extension point *ParseFilter*. This can be considered as entry point for every web page to be parsed. In its overridden method called “filter”, this class searches for multimedia on the webpage. Additionally, it provides additional meta data about the web page.

At the beginning of every parse step for a single web page, lists of Document Object Model (DOM) managers and taggers are created. DOM manager is implementing a common interface that we dubbed as *DomManager*. The interface has only one method *matchAndParseMetadata()* declared. Its purpose is to:

- parse a web page,
- search for multimedia objects,
- identify additional metadata.

After the list is initialized, every instance of the DOM manager from the list is run. The input parameters of the *matchAndParseMetadata()* method are a representation of HTML page, url of the page and the set of known ids (id is an url of the media found) to prevent from storing twice the same multimedia item from one page by more dom managers[11]. The page content is represented by the W3C Document Object Model, which allows the managers to use XPath to query for elements and attributes. Regular expressions are also frequently used.

The result of the method is a set of multimedia objects. Each of these objects is saved as a separate document to the index (during indexing phase), and assigned with a document representing the HTML page within which the document was located. This results in every *occurrence* of a multimedia object having a unique index document (as opposed to every unique object). This behaviour is required to create a custom solution for the Solrindex step (see section 3.2.2.6). The metadata and the time of parsing are also saved.

This plugin performs also a first filtering of some multimedia objects such as logos and banners. The current implementation uses a blacklist of names, which, if present in the multimedia object’s file name, cause the object to be skipped. This simple, yet effective technique can be extended in multiple ways:

1. blacklist based on object dimensions,
2. number of occurrences of the object on the web site,
3. content-based classification.

3.3.2 Rendering-Based Multimedia Extraction - HtmlUnit and Dom managers

HtmlUnit (`http://htmlunit.sourceforge.net/`) is an open source Java library. It creates HTTP calls which imitate the browser functionality from a code. It proved to be useful to use this library to access and process the rendered source code, in addition to using the unprocessed source code as directly

[11] This could happen, for example, with videos represented by both *<object>* and *<embed>* html elements, which are often repeated within the same page multiple times (with small syntactical variations), but referencing the same multimedia object.

returned from the server over the HTTP protocol. The motivation for using the rendered code is the fact that many pages from the whitelist specified by the content partners contain multimedia objects embedded in such a way that they cannot be found in the unprocessed source code.

Illustrative example. Consider the web page `http://www.rbb-online.de/brandenburgaktuell/index.html`:

Multimedia representation before rendering by HtmlUnit:

```
function setupLb_586015052(){
jwplayer("lb_586015052").setup({
'modes': [
{type: 'flash',
src: '/cqbase/jwplayer/5_9/player.swf',
config: {
'file':
'/rbb/aktuell/dossier/aktuell_20130418_Zeitzeugentrailer_17._Juni_m_16_9_512x288.mp4',
'provider': 'rtmp',
streamer': 'rtmp://ondemand.rbb-online.de/ondemand'
```

With HtmlUnit:

```
<object type="application/x-shockwave-flash" data="/cqbase/jwplayer/5_9/player.swf"
width="100%" height="100%" bgcolor="#000000" id="1817906585" name="1817906585"
tabindex="0">
<param name="allowfullscreen" value="true">
<param name="allowscriptaccess" value="always">
<param name="seamlesstabbing" value="true">
<param name="wmode" value="transparent">
<param name="flashvars" value="netstreambasepath=http%3A%2F%2Fwww.rbb-online.de%2F
brandenburgaktuell%2Findex.html&id=1817906585&image=%2Fetc%2Fmedialib%2Frbb%2Frbb%2F
aktuell%2Ffans_fiebern_championsleage.file.512.288.jpg&skin=%2Fstatic%2Frbb%2F
jwplayer%2Fskins%2Fclassic%2Fclassic.xml&screencolor=0x000000&bufferlength=5
&smoothing=true&idlehide=false&stretching=uniform&wmode=transparent&bgcolor=0x000000&
autostart=false&file=%2Frbb%2Faktuell%2Faktuell_20130525_fan_m_16_9_512x288.mp4&
provider=rtmp&streamer=rtmp%3A%2F%2Fondemand.rbb-online.de%2Fondemand&
controlbar.position=bottom&display.icons=true&dock=false">
</object>
```

The object element above is generated by JavaScript. But additionally, the script checks whether Flash plugin is installed before rendering the object. Therefore, custom BrowserVersion (objects of this class represent one specific version of a given browser) that supports Flash should be used. After that, the web page is represented in the code like an HtmlPage object.

HtmlPage is a class that represents a single web page along with the client data (HTML, JavaScript, CSS etc.) and provides different methods to access the page content (html elements) like *getElementById* or *getByXPath*. In order to increase the number of identified videos per web page on the whitelist, a survey of coding practices for embedding multimedia objects[12] into a web page was performed. Consequently, several classes called *dom managers* were designed (non-exhaustive list):

- ObjectTagDomManagerImpl
- VideoTagDomManagerImpl
- JavascriptContentDomManagerImpl
- IFrameDomManagerImpl
- YoutubeObjectTagDomManagerImpl

[12] Videos and podcasts, images do not generally pose a processing problem.

3.3.3 Nutch Annnotated Crawling List Extension

For the purposes of keeping urls from the two whitelists (RBB and Sound&Vision) apart within one physical index, it was necessary to create an additional configuration file for Nutch. This apparent simple task required a greater amount of changes because Nutch does not support custom configuration files. Nutch source code was extended with one class called *DomainTaggerImpl*.

This class is used to load the *annotated filter* configuration file, which contains a flag indicating to which whitelist the url belongs. After detection of a domain source, the url is assigned to the appropriate field called domain_source. To make Nutch aware of this configuration file, a new property to configuration file `nutch-default.xml` needs to be added:

```
<property>
 <name>urlfilter.whitelist.file</name>
 <value>whitelist-urlfilter.txt</value>
 <description>
  Name of a file on CLASSPATH containing urls of whitelisted pages
  used by media-extractor (DomainTaggerImpl) plugin
 </description>
</property>
```

A format of the configuration file (whitelist-urlfilter.txt):

```
# LinkedTV configuration file for white-listed urls.
# RBB White List - http://linkedtv.eu/wiki/index.php/RBB_White_List
# SV White List - http://linkedtv.eu/wiki/index.php/Sound_and_Vision_White_List
#
# If you want to add resource you can do it in this way:
#
# Format:
# Set prefix "R_" for urls which you want to have indexed as "rbb" in the Solr field
# called "domain_source"
# or set prefix "S_" for urls which you want to have indexed as "sv" in the Solr
# field called "domain_source"
#
# then add url (without http or www)
# example: R_rbb-online.de

R_antennebrandenburg.de/
R_inforadio.de/
S_amsterdammuseum.nl/
S_imdb.com/
S_linkedmdb.org
```

3.3.4 Nutch-Solr bridge

The development of extensions for handling multimedia objects described in sections 3.3.1 and 3.3.3 triggered the need to implement a replacement for the *SolrIndex* class (section 3.2.2.6), which is unable to work with the resulting custom data structure.

Mapping of fields created by Nutch to fields defined (and expected) by Solr (*schema.xml*) is set in a configuration file called *solrindex-mapping.xml*. This file contains the setting used by the Solrindex step at the end of the script crawling process. For the LinkedTV purposes, this file cannot be used because it does not offer the flexibility required to deal with the customizations resulting from multimedia processing. Instead, the process of placing fields in the index is performed by the new class *MediaSolrIndexerJob*. This class extends the default Solr IndexerJob class and contains two inner classes:

```
SolrMediaIndexerJobMapper
SolrMediaIndexerJobReducer
```

These classes contain overriden methods to map, reduce, load and setup configuration (in addition to other methods). The main purpose of the map method is to create Nutch documents for all media items found in a web page including the web page itself. Nutch takes all the created Nutch documents, maps

them on Solr documents and writes them to the Solr index. After the documents are written, they are available in Solr for querying.

3.4 Index structure

One document in the index can be represented as an XML element *<doc>*. The children elements of *<doc>* correspond to fields. There are four types of documents in the index: *webpage*, *video*, *podcast* and *image*.

Illustrative Example. Document of *image* type.

```
<doc>
  <str name="media_type">image</str>
  <str name="media_addition"/>
  <str name="media_url">
   http://www.zdf.de/ZDFmediathek/contentblob/1938364/timg298x168blob/8171749
   </str>
  <str name="digest">5fc5d474b5b6383079780d955ddc6432</str>
  <date name="tstamp">2013-08-07T22:06:14.237Z</date>
  <str name="media_title">Bild zeigt den Berliner Reichstag bei Nacht</str>
  <str name="url">
   http://www.zdf.de/Berlin-direkt/Team-Berlin-direkt-22674890.html
   </str>
  <str name="source_code_rendered">
   http://www.zdf.de/Berlin-direkt/Team-Berlin-direkt-22674890.html
   </str>
  <str name="domain_source">rbb</str>
  <str name="id">
   http://www.zdf.de/ZDFmediathek/contentblob/1938364/timg298x168blob/8171749
   </str>
  <str name="source_code">
   http://www.zdf.de/Berlin-direkt/Team-Berlin-direkt-22674890.html
   </str>
  <str name="content">
   HERE IS A CONTENT OF A HTML PAGE WHERE THE IMAGE IS LOCATED BUT WITHOUT HTML
   ELEMENTS. JUST TEXT REPRESENTATION. IT IS OMITTED HERE FOR SHORTHAND.
   </str>
  <str name="media_solver_class">ImgMetaManagerImpl</str>
  <str name="title"/>
  <long name="parse_time_unix_timestamp">1374718537</long>
  <date name="parse_time">2013-07-25T04:15:37.99Z</date>
  <str name="media_description">Bild zeigt den Berliner Reichstag bei Nacht</str>
  <long name="_version_">1441505267507265536</long></doc>
<doc>
```

This entry shows how the document is represented in Solr. Below, we will briefly describe some of the fields:

- **media_type:** the document type (e.g. an image)
- **media_url:** the location of the source document (e.g. the URL of the image)
- **id:** a unique key for every document in the index, media_url is used for the key

The semantics of fields *source_code* and *source_code_rendered* differs depending on the media_type. For multimedia objects (media_type different from web page), the source code contains the id of the document (web page) containing the source code:

```
<str name="source_code">
http://www.zdf.de/Berlin-direkt/Team-Berlin-direkt-22674890.html
</str>
```

With this information, it is possible to retrieve the original document, which will be a web page:

```
<str name="media_type">webpage</str>
```

The fields source_code and source_code_rendered (see section 3.3.2) store complete source code of a web page with html elements and javascript.

```
 <str name="source_code_rendered"><?xml version="1.0" encoding="UTF-8"?>
 <html lang="de" class=" js n
..... etc.
```

```
<str name="source_code"><!DOCTYPE html><!--[if lt IE 7 ]>
<html lang="de" class="no-js ie6
..... etc.
```

The difference between these two fields are:

- **media_url**: It represents the url of a media where it can be retrieved (webpage, image, mp3 stream, mp4 stream etc.)
- **media_type**: Type of a media (webpage, video, image, podcast)
- **parse_time**: A time of parsing (in readable format yyyy-MM-dd'T'HH:mm:ss.S'Z'). Stored as *date* in Solr, which allows to perform range queries
- **parse_time_unix_timestamp**: A time of parsing (in Unix timestamp). Stored as *long*, which allows to perform range queries
- **domain_source**: Information about media origin – if it comes from RBB or S&V whitelist
- **media_title**: Media title (if found). For the *image* type, it corresponds to the `img` title attribute
- **media_description**: Media description (if found). For the *image* type, it corresponds to the `img` alt attribute
- **media_addition**: Reserved field for some additional information
- **source_code_rendered**: For webpages, source code rendered with HtmlUnit. For other media (image, video, podcast), it stores the id of the associated web page. May not be always available.
- **source_code**: For webpages, original source code. For other media (image, video, podcast), it stores id of the associated web page. Always available.

3.5 API Description

A simple HTTP-based API has been developed for remote data access. This API is superimposed over the Solr API. The query syntax corresponds to the Solr/Lucene query syntax. The response format is in JSON according to the format specified by the MediaCollector service (see section 4). The URI scheme for calling the API is composed of:

- query argument *q* for Lucene query,
- query argument *row* defining a number of items to be retrieved (by default 10).

```
Resource: https://ir.lmcloud.vse.cz/irapi/media-server/
HTTP method: GET
Response format: application/json;charset=utf-8
Parameters:
     q  --> query
         Example: q=media_url:*
     row --> number of items to retrive
         Default: 10
         Example: row=100
```

3.5.1 Syntax

Wildcards and Operators. Wildcards *?* and * replace one or more characters. The common boolean operators *AND*, *OR*, *NOT* are also available.

Range Queries. It is possible to use *[from TO to]* syntax to form range queries. The symbol * may be used for either or both endpoints to specify an open-ended range query.

- *field:[* TO 100]* matches all field values less than or equal to 100
- *field:[100 TO *]* matches all field values greater than or equal to 100
- *field:[* TO *]* matches all documents with the field present

In the LinkedTV setup, range queries are intended for the timestamp* fields. The complete *ISO 8601* date syntax that fields support, or the *DateMath* Syntax to get relative dates. For example to get all documents crawled since August 22, 2013:

```
parse_time:[2013-08-22T07:33:21.699Z TO *]
```

Alternatively, the range query can be executed against the parse_time_unix_timestamp field, which is stored as data type *long*:

```
parse_time_unix_timestamp:[1377149601 TO *]
```

Proximity. Words should be within a specific distance away.

- *content:"Berlin parliament"~2* finds documents, where words “Berlin” and “parliament” are within 2 words of each other

Fuzzy Search. Finding similar terms (Levenshtein[13] or edit distance) in spelling.

- *media_title:meine~0.7* finds documents with title similar to “meine” with similarity 0.7

Priorities (boost). It is possible to set priorities for individual fields. Default priority is 1. Higher number means higher priority.

- *(media_title:*Berlin*)^5 (media_description:*Parlament*)^0.5*

3.5.2 API response

The mapping relationship between the values of the Solr fields (section 3.4) and the response fields (section 4) is as follows:

- **url** -> micropostUrl
- **parse_time_unix_timestamp** -> timestamp
- **parse_time** -> publicationDate
- **media_type** -> type
  query: page | image (also photo is supported) | video | podcast (also audio is supported) | webpage
  response: photo | video | audio | webpage
- **media_url** -> mediaUrl
  if media_type equals "image" then media_url -> posterUrl
- **media_description** -> micropost.html
- **media_title** -> micropost.plainText

The content of the *publicationDate* field is set to the timestamp when the page was parsed by Nutch, i.e. it corresponds to the retrieved date, rather than to a publication date as for example contained in meta name="DC.Date" from a source code of a page.

Document uri authority (DNS host name) is used as source id (e.g. `www.ard.de`). Source id is an unique identifier containing items with properties like mediaUrl, type, publicationDate.

[13] `http://en.wikipedia.org/wiki/Levenshtein_distance`

```
www.ard.de: [
{
       micropostUrl: "http://www.ard.de/",
       micropost: {
              html: "Woller und Schwester Hanna (Bild: ARD/Barbara Bauriedl)",
              plainText: "Woller und Schwester Hanna (Bild: ARD/Barbara Bauriedl)"
       },
       mediaUrl:
        "http://static.daserste.de/cmspix/tvtipp/thumb_128_72_05022013187955.jpg",
       type: "photo",
       timestamp: 1398772232014,
       publicationDate: "2013-08-29T13:50:32Z"
}
]
```

3.6 Evaluation

To demonstrate the functionality of the Unstructured Search Module, the section 3.6.1 presents several sample queries on the LinkedTV content and section 3.6.4 presents some statistics about the current index.

3.6.1 Sample Queries

This subsection lists several queries based on the LinkedTV scenarios. The queries were executed against the index populated with web pages indexed from web sites on the whitelist provided by the LinkedTV content partners. The queries were manually constructed based on entities appearing in the LinkedTV scenarios. All queries returned result as listed at the time when they were designed. Due to changes on the web sites (especially RBB content is made unavailable after a certain time period), the listed response of the queries is valid only as of time of writing.

The search was performed mainly over the *content*, *title* and *media_title* fields, which hold the plain text content of the web page (or web page holding the multimedia object), value of HTML `title` tag and value of the `img` tag's title attribute. Higher precision can be obtained by involving the foreseen Metadata Extraction Service (MES) service (section 3.7.2). The list of result for individual queries listed below are abridged to contain only the most relevant results (unless noted otherwise).

3.6.2 RBB scenarios

The following queries address some of the information requirements related to a person called Nina from the RBB scenarios [6].

At the beginning of the scenario, Nina starts to watch TV news. The news are mainly about the US president *Barack Obama* and his visit to Berlin, but there are also other news items. Below is a demonstration of results returned by several specific queries.

Query RBB 1. The news starts with the president's departure. The prominent entities shown include *Air Force One*[14] or *Airport Tegel*, where the aircraft landed.

- **Object:** Air Force One
- **Query to API:** q=title:"Air Force One" AND (media_type:video OR media_type:webpage) AND domain_source:rbb
- **Relevant media content returned from the API:**
  - `http://www.ardmediathek.de/rbb-fernsehen/theodor/am-steuerknueppel-von-honeckers-air-force-one?documentId=16752626`
  - `http://www.whitehousemuseum.org/special/AF1/`

[14] The official air traffic control call sign of a United States Air Force aircraft carrying the President of the United States

Figure 4: Airport Tegel. Source: http://www.rbb-online.de/politik/thema/Flughafen-BER/BER-Aktuelles/hintergrund/beitraege/txl/_jcr_content/image.img.jpg/rendition=original/size=708x398.jpg

This query found relevant information on the entity *Air Force One*. The first one is a video about Air Force One pilots from media library and the second result is a web page about history of the aircraft with a lot of photos and other information.

- **Object:** Airport Tegel
- **Query to API:** q=(title:"Tegel" OR media_title:*Tegel*) AND domain_source:"rbb" AND (media_type: "podcast" OR media_type:"image")
- **Relevant media content returned from the API:**
  - http://www.rbb-online.de/politik/thema/Flughafen-BER/BER-Aktuelles/hintergrund/beitraege/berliner_flughaefen/_jcr_content/pictureList/13748/image.img.jpg/rendition=original/size=708x398.jpg
  - http://www.rbb-online.de/politik/thema/Flughafen-BER/BER-Aktuelles/hintergrund/beitraege/txl/_jcr_content/image.img.jpg/rendition=original/size=708x398.jpg
  - www.inforadio.de//cqbase/flash/jwplay.swf?streamer=rtmp://stream5.rbb-online.de/inf&autostart=false&image=/etc/medialib/rbb/inf/bild/politiker__prominente/mathias_koehne__spd.file.220.165.jpg&file=/programm/boe/201306/28/NP079362.mp3
  - www.inforadio.de//cqbase/flash/jwplay.swf?streamer=rtmp://stream5.rbb-online.de/inf&autostart=false&image=/etc/medialib/rbb/inf/bild/verkehr/luftverkehr/flughafen_berlin_tegel0.file.220.165.jpg&file=/programm/boe/201212/10/NP071961.mp3

This query found relevant information about the entity *Airport Tegel*. The results include images (see Fig. 4) of the airport and podcasts with content relevant to the query entity.

Query RBB 2. According to the scenario, Nina is curious about the meeting at the *Brandenburg Gate*. She wants to hear what Obama said. She also wants to see the image gallery of this big event.

- **Object:** Barack Obama at Brandenburg Gate
- **Query to API:** q=(title:"Barack Obama Brandenburg Gate"~5 OR (title:"Barack Obama Brandenburger Tor"~5)) OR (content: "Barack Obama Brandenburg Gate"~5 OR (content:"Barack Obama Brandenburger Tor"~5)) AND domain_source:"rbb"
- **Relevant media content returned from the API:**
  - http://www.radioeins.de/cqbase/jwplayer/5_9/player.swf?file=http%3A%2F%2Fdownload.radioeins.de%2Fmp3%2F_programm%2F8%2F20130619_1620_obama.mp3&autostart=true

Figure 5: Barack Obama at Brandenburg Gate. Source: http://www.radioeins.de/etc/medialib/rbb/rad/bilder0/201306/us_praesident_brack.file.66656.460.259.jpg

- http://www.dw.de//js/jwplayer/player.swf?streamer=rtmp://tv-od.dw.de/flash/&provider=rtmp&DWQualityPlugin.isold=false&DWQualityPlugin.initialbandwidth=&DWQualityPlugin.pluginmode=FLASH&controlbar.position=over&controlbar.idlehide=true&file=je/je20130619_obamaspeech_sd_avc.mp4
- http://dw.de/us-president-barack-obama-speaks-at-brandenburg-gate/a-16894277
- http://www.radioeins.de/etc/medialib/rbb/rad/bilder0/201306/us_praesident_brack.file.66656.460.259.jpg
- http://fritz.de/etc/medialib/rbb/fri/bilder/beitraege/neues_wort/aktuell/2013/130618_obama_berlin.file.jpg

This query found relevant information on Obama's speech at the Brandenburg Gate including images (consider e.g. Figure 5), a podcast, a video and a webpage.

Query RBB 3. Nina is interested in the speeches of *Klaus Wowereit* (Governing Mayor of Berlin).

- **Object:** Klaus Wowereit and his speech at Brandenburg Gate item **Query to API:** q=(title:"Wowereit Obamas"~50) OR (content:"Wowereit Obama"~50) AND domain_source: "rbb" AND (media_type:"podcast" OR media_type:"video" OR media_type:"webpage")
- **Relevant media content returned from the API:**
  - http://www.berlin.de/landespressestelle/archiv/20130619.1340.386279.html

  This query found relevant information about the text of the Governing Mayor's speech (Klaus Wowereit) during Obama's visit.

Query RBB 4. The voice in the news mentions Barack Obama's half-sister Auma. Nina would like to get some additional information about *Auma Obama*.

- **Object:** Auma Obama
- **Query to API:** q=title:"Auma Obama" AND domain_source:"rbb"
- **Relevant media content returned from the API:**
  - http://de.wikipedia.org/wiki/Auma_Obama
  - http://mp3-download.swr.de/swr1/bw/leute/auma-obama-entwicklungshelferin-in-kenia.6444m.mp3
  - http://www.dw.de/die-kenianische-menschenrechtlerin-auma-obama/a-16703706

  This query found relevant information about Auma Obama including a podcast.

Figure 6: Tim Raue, the chef at Schloss Charlottenburg. Source: http://www.radioeins.de/etc/medialib/rbb/rad/personen/r/tim_raue___dpa_bildfunk.file.12367.460.259.jpg

Query RBB 5. The next chapter of news is about the official dinner at *Schloss Charlottenburg*. Nina is very interested in it and tries to find out about the chef in charge, *Tim Raue* (see Figure 6).

- **Object:** Schloss Charlottenburg
- **Query to API:** q=(title: "Schloss Charlottenburg" OR media_title:*Charlottenburg*) AND domain_source:"rbb"
- **Relevant media content returned from the API:**
  - http://www.berlin.de/aktuelles/obama-besuch-2013/3099566-3095135.gallery.html?page=1
- **Object:** Tim Raue
- **Query to API:** q=(title:"Tim Raue" OR media_title:*Raue*) AND domain_source:"rbb"
- **Relevant media content returned from the API:**
  - http://www.radioeins.de/etc/medialib/rbb/rad/personen/r/tim_raue___dpa_bildfunk.file.12367.460.259.jpg
  - http://www.radioeins.de/cqbase/jwplayer/5_9/player.swf?file=http%3A%2F%2Fdownload.radioeins.de%2Fmp3%2F_programm%2F8%2F20121215_arena_raue.mp3&autostart=true
  - http://www.berlin.de/restaurants/1640924-1622830-restaurant-tim-raue.html

Query RBB 6. The last spot in the video is about politicians from Brandenburg. One of them is *Matthias Platzeck*. Nina wants to know more about him. She wants to see his profile and get more information.

- **Object:** Matthias Platzeck
- **Query to API:** q=(title:"Matthias Platzek" OR media_title:*Platzeck*) AND domain_source:"rbb"
- **Relevant media content returned from the API:**
  - http://www.antennebrandenburg.de/programm/dossiers_programm/2013/matthias_platzeck.html
  - http://www.antennebrandenburg.de/etc/medialib/rbb/ant/bilder/politiker/matthias_platzeck0.file.200.150.jpg
  - http://www.inforadio.de/etc/medialib/rbb/inf/bild/politiker__prominente/matthias_platzeck22.file.28953.402.171.jpg
  - http://www.berlin.de/aktuelles/berlin/3137562-958092-platzeck-wird-nach-auszeit-in-seiem-buer.html

- http://www.inforadio.de/programm/schema/sendungen/int/201301/16/platzeck_und_wowereit.html

The results presents a wealth of information about the query entity. Nina can view Platzeck's profile, relevant articles. The results also include pictures featuring Platzeck.

3.6.3 S&V scenarios

The queries below address some of the information requirements defined for the two selected S&V scenarios [6]. There is a person called Rita defined in the first scenario, and persons Bert and Anne in the second scenario.

Query S&V 1. Rita watches the show and she is interested in finding out more about the show's host *Nelleke van der Krogt*.

- **Object:** Nelleke van der Krogt
- **Query to API:** q=(title:"Nelleke van der Krogt" OR content:"Nelleke van der Krogt") AND domain_source:"sv"
- **Relevant media content returned from the API:**
  - http://www.rmo.nl/actueel/nieuws/2011/tussen-kunst-en-kitsch
  - http://cultuurgids.avro.nl/front/archiefkunst.html?trefwoord=NellekevanderKrogt

Query S&V 2. Rita also sees that the show has been recorded in the *Hermitage Museum* in Amsterdam. She has always wanted to visit the museum, and to find out what the link is between the *Amsterdam Hermitage* and the *Hermitage in St. Petersburg*.

- **Object 1:** Hermitage Museum in Amsterdam
- **Object 2:** Hermitage in St. Petersburg
- **Query to API 1:** q=(title:"Hermitage" AND content:"Hermitage Amsterdam history"~5) AND domain_source:"sv"
- **Query to API 2:** q=(title:"Hermitage" AND content:"Hermitage Petersburg"~5) AND domain_source:"sv"
- **Relevant media content returned from the API:**
  - http://www.hermitage.nl/en/hermitage_amsterdam/
  - http://www.artcyclopedia.com/featuredmuseums-feb99.html
  - http://www.hermitage.nl/en/st-petersburg_en_rusland/de_stad_st-petersburg.htm
  - http://www.the-athenaeum.org/sources/detail.php?id=141

  Results include web pages with articles about both Hermitages.

Query S&V 3. The show is coming to an end and the final person on the show has brought in a painting that has the signature *Jan Sluijters*. It is a famous Dutch painter. Rita wants to learn more about this painter (see Figure 7) and the art styles he represents.

- **Object:** Jan Sluijters
- **Query to API:** q=title:"Jan Sluijters" OR content:"Jan Sluijters"
- **Relevant media content returned from the API:**
  - http://de.wikipedia.org/wiki/Jan_Sluijters
  - http://www.artcyclopedia.com/artists/sluijters_jan.html
  - http://www.avro.nl/Images/jan_sluijters_tcm8-261885.jpg
  - http://staticextern.avro.nl/gfx/museumtv/D01280_1_tcm4-65412.jpg

  Results include images with paintings and also web pages about the painter and his life.

Figure 7: Jan Sluijters's painting. Source: `http://staticextern.avro.nl/gfx/museumtv/D01280_1_tcm4-65412.jpg`

Query S&V 4. The next scenario is about Bert and Anne. They also watch TV. Anne sees a chapter about *Greek mythology* (gods *Hebe* and *Karitas*) and she wants to brush up on her knowledge about these gods and the Greek mythology in general.

- **Object 1:** Hebe
- **Object 1:** Karitas
- **Query to API 1:** q=(title:"Hebe" AND content:"Griechische Gottheit")
- **Query to API 2:** q=title:"Karitas"
- **Query to API 3:** q=title:"mythologie" AND content:*Griech*
- **Relevant media content returned from the API:**
  - `http://www.the-athenaeum.org/art/detail.php?ID=92990`
  - `http://www.the-athenaeum.org/art/detail.php?ID=87341`
  - `http://de.wikipedia.org/wiki/Karitas`
  - `http://de.wikipedia.org/wiki/Dia_(Mythologie)`
  - `http://de.wikipedia.org/wiki/Griechische_Mythologie`

  The results contain multiple relevant web pages related to the queries. The first two items are paintings with titles containing the word "Hebe". However, these are false positives.

Query S&V 5. There is also one chapter in the programme about a gold watch by the famous watch maker *Breguet*. Anne likes the watch and she notices that a clock system is made like a *Jacquemart*[15].

- **Object 1:** Famous watch maker Breguet
- **Object 1:** Jacquemart
- **Query to API 1:** q=title:"Breguet" AND (content:"watch" OR content:"clock" OR content:"Uhr")
- **Query to API 2:** q=title:"Jacquemart"
- **Relevant media content returned from the API:**
  - `http://www.getty.edu/news/press/center/breguet.html`
  - `http://de.wikipedia.org/wiki/Abraham_Louis_Breguet`
  - `http://de.wikipedia.org/wiki/Jacquemart_(Uhr)`

  Those results could satisfy Anne's information need.

[15] System that produces sound by having one or more human figures strike the bells.

3.6.4 Index statistics

The statistics listed below illustrate the growth of the index of the Unstructured Search Module from an empty initial state (new software version released) over the course of two weeks and one week respectively.

Old statistics (31/05/2013 - 14/06/2013).

- **Video**: 3917
- **Podcast**: 30
- **Image**: 14904
- **Webpage:** web pages were not stored in the index

Current statistics (21/09/2013 - 30/09/2013).

- **Video**: 5584
- **Podcast**: 2693
- **Image**: 79996
- **Webpage**: 73251

The increase in the number of crawled resources shows increasing effectiveness of the Unstructured Search Module in extracting multimedia objects from the *whitelists*, the lists of web sites provided by the LinkedTV content partners[16].

3.7 Extensions

We already foresee a number of extensions to the Unstructured Search Module. A dashboard will be contributed in year 3 of the project. For the remaining two extensions (Metadata Extraction Service and Entity enrichment), a feasibility study will be performed, and based on its results the extension or a proof-of-concept will be developed.

3.7.1 Dashboard

Dashboard a web application that will serve index statistics through a friendly graphical interface. The dashboard will complement the existing reporting facilities in Solr, which e.g. does not distinguish between the types of documents stored (webpage, image, video, podcast).

3.7.2 Metadata Extraction Service

The purpose of the Metadata Extraction Service is to identify text associated with the multimedia object. The extracted text will serve as metadata about the object, and will be stored into separate fields of the index, allowing for higher precision queries.

3.7.3 Entity enrichment

A requirement of WP4 is to have enrichment content annotated with entity analysis. A foreseen extension of the current indexing process is to use an entity classification service to annotate the indexed textual content and store the result into the index. Due to high time-complexity of the entity-classification process, it is questionable whether such processing is feasible for all documents in the index.

[16] It should be noted that the list of web sites on these whitelists also grew in the respective period.

4 Media Collector

The widespread availability of mobile phones with higher resolution cameras has transformed citizens into media publishers and witnesses who are used to comment and share event-related media on social networks. Some examples with global impact include the shootings in Utøya, which first appeared on Twitter, the capture and arrest of Muammar Gaddafi, which first appeared on YouTube, or the emergency ditching of a plane in the Hudson river, which first appeared on Twitpic. Some news agencies[17] have even specialized in aggregating and brokering this user-generated content. In this section, we propose a new approach for retrieving all those event-related media items that are being published by users on several social networks [22, 17, 32]. They will provide a new source of LinkedTV enrichments that will prove to be particularly relevant in the context of a news program enrichment [6].

We have developed a so-called **Media Collector** composed of media item extractors for all the media sharing networks listed in Table 5. Furthermore, the Media Collector also integrates the Unstructured Search Module described in the section 3 via the Search API that this module exposes. Consequently, the Media Collector is the central module for the enrichment process that provides content coming from social networks and white-listed web sites, and that can be later on validated by an editor and/or further filtered by the LinkedTV personalization component.

The remainder of this section is organized as follows: in Section 4.1, we provide a classification of social networks according to the media support they provide. In Section 4.2, we detail the process of extraction and reconciliation of media items from different sources while we describe the query and response formats of the Media Collector in section 4.3. Finally, we provide examples of results given by the Media Collector for the LinkedTV scenarios in the section 4.4.

4.1 Social Networks

A social network is an online service or media platform that focuses on building and reflecting social relationships among people who share interests and/or activities. The boundary between social networks and media platforms is rather blurry. Several media sharing platforms, such as YouTube, enable people to upload content and optionally allow other people to react to this content in the form of comments, likes or dislikes. On other social networks (*e.g.,* Facebook), users can update their status, post links to stories, upload media content and also give readers the option to react. Finally, there are hybrid clients (*e.g.,* TweetDeck for Twitter using Twitpic) where social networks integrate with media platforms typically via third party applications. Therefore, we consider three types of support of media items with social networks:

- *First-order support*: The social network is centered on media items and posting requires the inclusion of a media item (*e.g.* YouTube, Flickr);
- *Second-order support*: The social network lets users upload media items but it is also possible to post only textual messages (*e.g.* Facebook);
- *Third-order support*: The social network has no direct support for media items but relies on third party application to host media items, which are linked to the status update (*e.g.* Twitter before the introduction of native photo support).

We consider 12 different social networks that all have powerful and stable APIs and, together, represent the majority of the market. The criteria for including media sharing platforms follow a study performed by the company Sysomos, specialized in social media monitoring and analytics [12]. Table 5 lists these platforms according to the categorization defined above.

4.2 Implementation

Twitter and its ecosystem (TwitPic, TwitterNative, MobyPicture, Lockerz or yfrog), GooglePlus and YouTube, Facebook and Instagram, Flickr and FlickrVideos, MySpace, all offer search APIs over the content they host. Those search functions, however, provide results that vary according to the time the query has been triggered, covering a window of time that ranges from only the recent past to many years ago. In addition, they offer different parameters that enable to customize search queries (e.g. filtering by location). The Media Collector is composed of media item extractors for these 12 media sharing platforms.

[17] *e.g.* Citizenside (`http://www.citizenside.com`)

Social Network	URL	Category	Comment
Google+	http://google.com/+	second-order	Links to media items are returned via the Google+ API.
MySpace	http://myspace.com	second-order	Links to media items are returned via the MySpace API.
Facebook	http://facebook.com	second-order	Links to media items are returned via the Facebook API.
Twitter	http://twitter.com	second-/third-order	In second order mode, links to media items are returned via the Twitter API. In third order mode, Web scraping or media platform API usage are necessary to retrieve links to media items. Many people use Twitter in third order mode with other media platforms.
Instagram	http://instagram.com	first-order	Links to media items are returned via the Instagram API.
YouTube	http://youtube.com	first-order	Links to media items are returned via the YouTube API.
Flickr	http://flickr.com	first-order	Links to media items are returned via the Flickr API.
MobyPicture	http://mobypicture.com	first-order	Media platform for Twitter. Links to media items are returned via the MobyPicture API.
Twitpic	http://twitpic.com	first-order	Media platform for Twitter. Links to media items must be retrieved via Web scraping.
img.ly	http://img.ly	first-order	Media platform for Twitter. Links to media items must be retrieved via Web scraping.
Lockerz	https://lockerz.com/	first-order	Media platform for Twitter. Links to media items must be retrieved via Web scraping.
yfrog	http://yfrog.com	first-order	Media platform for Twitter. Links to media items must be retrieved via Web scraping.

Table 5: Social networks with different support levels for media items and techniques needed to retrieve them

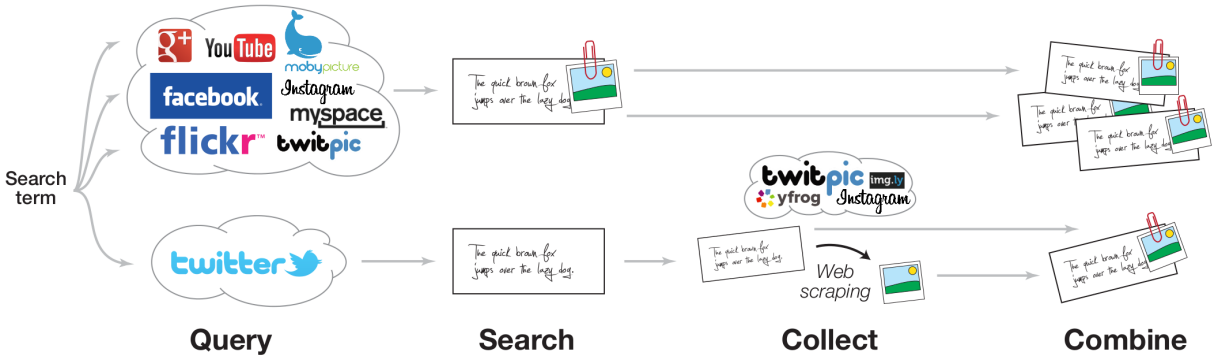

Figure 8: The media collector architecture: it proposes a hybrid approach for the media item extraction process using a combination of API access and Web scraping.

It takes as input a search term and a parallel key-search is then performed to these social networks. Each platform has a 30 second timeout window to deliver its results. When the timeout has expired, or when all social networks have responded, a unified output is delivered [22, 17, 32]. Figure 8 depicts the overall architecture of the media collector.

The metadata attached to the microposts retrieved vary in terms of schemas, data types and serialization formats. We harmonize these results and project them to a common schema (section 4.3). This component performs also a cleansing process, discarding items which are older than seven days ago, in order to keep only fresh media items. Besides this abstraction layer on top of the native data formats of those social networks, we perform a similar task for the social interactions (Table 6 [32]).

Likes	Shares	Comments	Views
Facebook Like	Facebook Share	Facebook Comments	YouTube Views
Google+ +1	Google+ Share	Google+ Comments	Flickr Views
Instagram Like	Twitter native ReTweet	Instagram Comments	Twitpic Views
Flickr Favorite		Twitter manual RT, @Replies	MobyPicture Views
YouTube Like		Twitpic Comments	
YouTube Favorite		MobyPicture Comments	
Twitter Favorite		Flickr Comments	

Table 6: Abstract social network interaction paradigms and their underlying native counterparts [32]

Media Collector provides not only a way to capture a snapshot at a particular instant of what has been shared in social media platforms, but enables also to monitor the results of a search query over a longer period of time, by automatically re-issuing the same query at a regular frequency and by cumulating the results [16].

The Media Collector was originally developed by Thomas Steiner at `https://github.com/tomayac/media-server` and forked twice for the purpose of LinkedTV at `https://github.com/vuknje/media-server` (in order to enable temporal search across social media platforms) and later one at `https://github.com/MathildeS/media-server` (in order to integrate the Unstructured Search Module). It is based on NodeJS[18].

4.3 Query and Response Format

Different settings are possible depending on the type of search performed:

- a S&V or RBB focused search can be made based on the white lists they provided, mainly using the unstructured search module (section 3);
- fresh media items can be searched on social networks using their provided APIs.

The query pattern is therefore: `http://linkedtv.eurecom.fr/api/mediacollector/search/TYPE/TERM` where:

[18] `http://nodejs.org/`

- TYPE is one of [RBB, SV, freshMedia, combined] and
- TERM is the search term.

The resulting behavior is:

- **RBB**: returns results from the RBB white list. The unstructured search module provides media items crawled from numerous white-listed web sites, while we also query some APIs offering white-listed content namely the YouTube API (on specific channels) and the Arte replay programs API (arte+7).
- **SV**: returns results from the Sound and Vision white list. Similarly as RBB, we query the unstructured search module and YouTube white-listed channels.
- **freshMedia**: returns fresh media items from the 12 social platforms.
- **combined**: combines results from all possible sources.

The Media Collector produces results into a unified JSON structure that first contains *sources* and then *items* for each source:

```
{
  www.ard.de: [],
  Arte+7: [],
  TwitterNative: [],
  Instagram: [
     {...},
     {...},
     {...},
     {...},
     {...},
     {...}
     {...},
  ],
  YouTube: [],
  FlickrVideos: [],
  Flickr: [],
  MobyPicture: [],
  TwitPic: [],
  Lockerz: [ ],
}
```

An *item* is described using the following attributes:

- **mediaUrl**: deep link to the media item itself.
- **posterUrl**: URL of a thumbnail (miniature) of the media item.
- **micropostUrl**: URL of the web document where the item is embedded (permalink), e.g. a YouTube page.
- **micropost**:
  - **html**: text description of the micropost in the original marked-up format.
  - **plainText**: cleaned text description of the micropost where the markup and some characters are removed.
- **userProfileUrl**: URL of the user that has published the micropost,
- **type**: type of the media item (photo or video).
- **timestamp**: reference time when the micropost was authored or the media item was uploaded (date in UNIX timestamp format).
- **publicationDate**: publication date using human-friendly syntax ("yyyy-MM-dd'T'HH:mm:ss'Z'").

Figure 9: Screenshot of the web page at `http://www.s-bahn-berlin.de/unternehmen/firmenprofil/historie.htm`, relevant to the named entity `S-Bahn` in the RBB scenario

- **socialInteractions**:
  - **likes**: total number of like, +1, heart for this media item.
  - **shares**: total number of re-share, re-post of this media item.
  - **comments**: total number of comments made about this media item.
  - **views**: number of times that this item has been viewed

4.4 Examples

We present in this section some example queries and content returned by the MediaCollector. Similar to the section 3.6.1, those queries are based on the LinkedTV scenarios. The search terms used correspond to the most frequent entities detected in media fragment annotations.

We observe that results coming from white-listed web sites are really relevant for the Sound and Vision scenario dedicated to cultural heritage material while recent media shared on social networks make more sense in the RBB scenario focused on news content.

4.4.1 RBB scenario.

- **Named entity**: S-Bahn
- **Query**: `http://linkedtv.eurecom.fr/api/mediacollector/search/RBB/S-Bahn`
- **Media content returned from API**:
  - source: YouTube
    type: video
    url : `http://www.youtube.com/watch?v=jHdpP2X9tE8`
  - source: www.s-bahn-berlin.de
    type: webpage
    url: `http://www.s-bahn-berlin.de/unternehmen/firmenprofil/historie.html`
  - source: www.mdr.de
    type: photo
    url: `http://www.mdr.de/sachsen-anhalt/halle/halleipzig100_v-standard43_zc-698fff06.jpg?version=54569`

Figure 10: Painting of Johan and Mies Drabbe by Jan Toorop, 1898 (collection H.F. Elout), relevant to the named entity `Toorop` in the Sound and Vision scenario. Source: `http://www.geschiedeniszeeland.nl/topics/kunstindomburg_22.jpg`

4.4.2 Sound & Vision scenario.

- **Named entity**: Toorop
- **Query**: `http://linkedtv.eurecom.fr/api/mediacollector/search/SV/Toorop`
- **Media content returned from API**:
  - source: YouTube
    type: video
    url : `http://www.youtube.com/watch?v=csVdymOUyNA`
  - source: www.geschiedeniszeeland.nl
    type: photo
    url: `http://www.geschiedeniszeeland.nl/topics/kunstindomburg_18.jpg`
  - source: www.geschiedeniszeeland.nl
    type: photo
    url: `http://www.geschiedeniszeeland.nl/topics/kunstindomburg_22.jpg`
  - source: www.boijmans.nl
    type: photo
    url: `http://www.boijmans.nl/images/pages/slides/72/2090_MK_De_Theems_bij_Londen_van_Jan_Toorop_web.jpg`

5 Searching Media Resources with LSI

The Linked Services Infrastructure (LSI) is another component, provided as a REST service, which can be integrated into the Media Collector and that provide relevant media resources (images and video) from online media repositories for a given Linked Data (DBPedia) concept.

In the case of LSI, we focus on making use of web APIs for online media platforms such as Flickr or YouTube, defining mapping rules between the semantic query (in terms of a Linked Data resource) and the structural API query to the non-semantic Web API. The process of transforming a semantic query to a structural query is called "lowering" while the transformation of the structured resource (usually JSON or XML) to a semantic result (RDF) is called "lifting". The use of mapping rules means that - provided the Web API does not change - media can be retrieved from that source repeatedly with the actual mapping only needing to be defined once. Media resource matches from different APIs is collected in parallel, while a local store of metadata of media resources relevant to a known, expected concept has been added to improve retrieval speed. LSI returns a list of matching media resources in RDF, with additional metadata for each media resource which could be used in subsequent ranking and selection of "most relevant" matches.

The rest of this section provides more details on the LSI implementation and usage. A standalone LSI instance for testing can be found at `http://production.sti2.org/lsi`.

5.1 Technology used

Linked Services Infrastructure makes use of Semantic Web Service technology developed in the Open University. The iServe[19] platform acts a repository for the descriptions of the Web APIs. These descriptions are themselves semantic, using the specifications hRESTs and MSM[20] to generate a RDF based machine processable description of the Web APIs inputs and outputs. Lifting and lowering rulesets (in XSLT extended by SPARQL based inline queries) define how a Linked Data URI may be mapped into a request to the Web API, and how the Web API response can be mapped to the LSI's RDF model (based on the W3C Ontology for Media Resources) for a consistent and aggregated response of relevant media resource descriptions to the client.

The REST API allows a single service to be queried (e.g. based on its description, such as only services who have content for a particular type of concept), a concept to be mapped to media resources by a single service call, or a concept to be mapped to media resources aggregated from all services called (based on their relevance to the concept in the query). Parameters allow that only image or video content is returned, a limit placed on the amount of media contents in the response, and whether Named Entity Recognition is used in the media resource titles and descriptions (this can reduce the latency in waiting on a response from LSI). Named Entity Recognition is used with an optional additional "context" parameter: the idea is that aggregating media content on the basis of a single concept can be very general (e.g. "Berlin"), while the provision of an additional *contextual* concept (e.g. "Obama") enables more focused media content responses from the LSI.

To determine sets of concepts related to a media item, we extract named entities as Linked Data URIs from the free text metadata of those media items such as their titles, descriptions, or user-provided tags. We work on using the Linked Data graph to map contextual concepts to related concept sets, for example "US Presidents" as the context to a query on "Berlin" should return images and videos related to US presidents and Berlin, e.g. a video annotated with the concepts "Berlin + Obama" will match a LSI query for "Berlin + US Presidents", based on the semantic closeness of Obama to the category of US presidents within the DBPedia graph.

5.2 Use of the REST API

The LSI API currently supports the invocation of the following APIs:

- Foursquare Venues API (`https://developer.foursquare.com/overview/venues`)
- Ookaboo API (`about.ookaboo.com/a/ookaboo_api`)
- Flickrwrappr API (`wifo5-03.informatik.uni-mannheim.de/flickrwrappr`)
- YouTube data API (`https://developers.google.com/youtube`)

[19] `http://iserve.kmi.open.ac.uk/`

[20] `http://iserve.kmi.open.ac.uk/wiki/index.php/IServe_vocabulary`

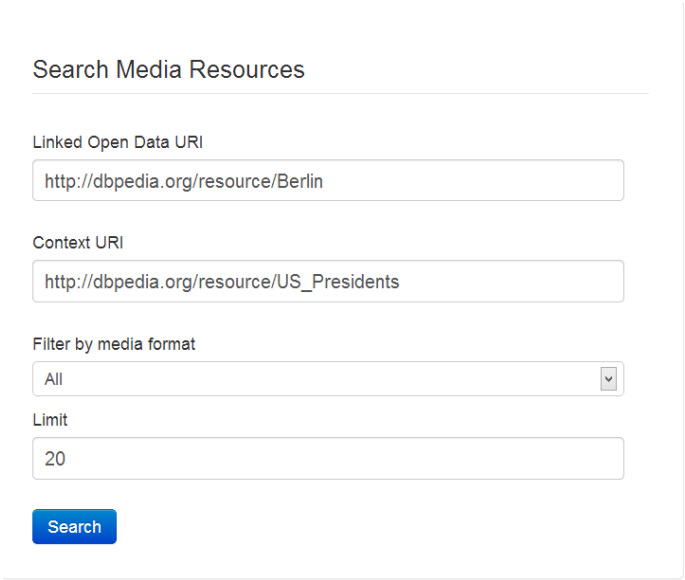

Figure 11: Search interface for 'Berlin + US Presidents'

– Instagram API (`instagram.com/developer/`)

– Flickr APIs (`www.flickr.com/services/api/`)

The Linked Service Infrastructure (LSI) provides a simple RESTful API for search and invocation of Web services. The primary REST method usable for extracting online media resources for a Linked Data URI is "Find Media Resources".

5.2.1 Find Media Resources.

Request URL : `http://production.sti2.org/lsi/api/invoke`

HTTP Method : GET or POST

HTTP Request Header : Use the Accept header to specify the required response format. Adding "Accept: application/rdf+xml" to the header of the request yields responses in the RDF/XML format; Adding "Accept: text/html" to the header of the request yields responses in the format of HTML snippets.

Name	Type	Example
lod	URI	http://dbpedia.org/resource/Berlin
mediaType	String	all, video, image
limit	Integer	10
ner	Boolean	yes or no
context	String	http://dbpedia.org/resource/US_Presidents

Table 7: Example usage of LSI for enriching content based on a search term and a context

Supported LOD URIs : The current implementation is tuned to DBPedia concepts while it will make a best effort with any Linked Data URI (any URI from which RDF based metadata can be found and used).

– Instances of `dbp:Place` use geolocation information. Two properties, i.e. latitude (`wgs84:lat`) and longitude (`wgs84:long`), will be used to assemble the actual requests to be sent to the endpoints of services

– All the instances of `rdf:resource` can be used in a service call using its value of `rdfs:label`. Note: for multi-lingual labels, LSI will choose the one in English.

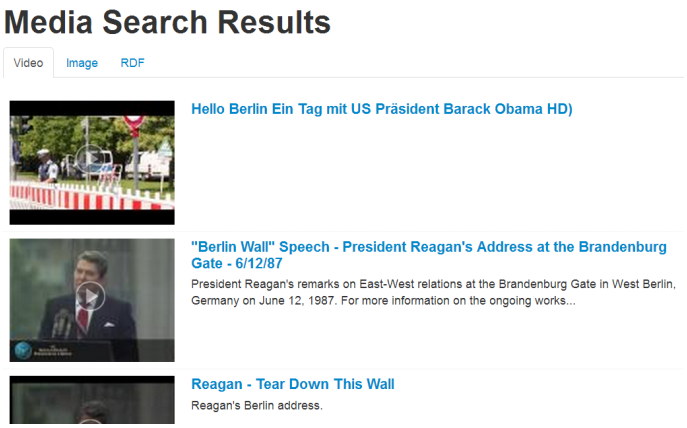

Figure 12: Video results for the query “Berlin + US Presidents”

Example : Here is an example for invoking the LSI API via HTTP GET.
Request:

```
http://production.sti2.org/lsi/api/invoke?lod=http://dbpedia.org/resource/Berlin&mediaType=all&li
mit=10&ner=yes&context=http://dbpedia.org/resource/US_Presidents
```

Response:
The response will be in the format of RDF and conform to the W3C Ontology for Media Resource. It contains a set of Media Resource instances with some properties and values according to what can be extracted from the original Web API. There is a mix of media characteristics (duration of video, dimensions of images) and conceptual relationships via the property hasKeyword (generated from textual titles and descriptions processed via Named Entity Recognition tools). For example, the RDF description of a YouTube video can look like this:

```
<rdf:Description rdf:about="http://production.sti2.org/lsi/media/2684e43caae442ce87e009f6c09214f2">
   <ma:hasKeyword rdf:resource="http://dbpedia.org/resource/Ronald_Reagan"/>
   <ma:hasKeyword rdf:resource="http://dbpedia.org/resource/Robert_Tear"/>
   <ma:hasKeyword rdf:resource="http://rdf.freebase.com/ns/en/berlin"/>
   <ma:hasKeyword rdf:resource="http://dbpedia.org/resource/State_Opening_of_Parliament"/>
   <ma:hasKeyword rdf:resource="http://dbpedia.org/resource/Berlin"/>
   <ma:hasKeyword rdf:resource="http://rdf.freebase.com/ns/en/ronald_reagan"/>
   <ma:hasKeyword rdf:resource="http://dbpedia.org/resource/Ronald_Reagan_Presidential_Library"/>
   <ma:title>Reagan - Tear Down This Wall</ma:title>
   <ma:publisher rdf:resource="http://www.youtube.com"/>
   <ma:locator rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
      https://www.youtube.com/watch?v=WjWDrTXMgF8&amp;feature=youtube_gdata_player</ma:locator>
   <rdf:type rdf:resource="http://www.w3.org/ns/ma-ont#MediaResource"/>
   <ma:duration rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">240</ma:duration>
   <ma:releaseDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
      2007-07-04T03:17:00</ma:releaseDate>
   <ma:description>Reagan’s Berlin address.</ma:description>
   <foaf:thumbnail rdf:resource="https://i.ytimg.com/vi/WjWDrTXMgF8/default.jpg"/>
   <ma:contributor rdf:resource="https://gdata.youtube.com/feeds/api/users/JohnJ2427"/>
 </rdf:Description>
```

5.3 Use in LinkedTV Media Layer

LSI will be added as an additional content source for the LinkedTV Media Collector (section 4) providing image and video resources matching the Linked Data concept with which the seed video fragment has been annotated. A first step in this integration will be the adaptation of the LSI response format to be conform with the JSON syntax expected by the Media Collector, which can be extended to call additionally the LSI REST API with the media annotation concept.

In due course, we want to explore which additional media resource properties are useful for enrichment selection and can be provided with the needed consistency and correctness (Web APIs vary in what information they provide about the media resources and how trustworthy those values are for automated processing).

Finally, further research needs to consider how contextual factors can be increasingly used to retrieve, in the first instance, the most relevant media resources from large scale media collections like Flickr or YouTube, and how these contextual factors can be brought into the initial query construction to LSI / Media Collector (e.g. Enrichments for a concept like “Chinese restaurants” could be geo-located to be close to the viewer’s current location).

6 MediaEval task

MediaEval is a benchmarking initiative for multimedia evaluation. It offers several tasks, among which the Search and Hyperlinking task we participated in. The MediaEval 2013 workshop will take place in Barcelona, Catalunya, Spain, on Friday-Saturday 18-19 October 2013[21], just before ACM Multimedia 2013[22].

In this section, we will describe what the goals of this challenge are and how it connects to LinkedTV goals, especially to the task of linking hypervideos to Web content. The following partners contributed to the LinkedTV submission to the challenge: CERTH, EURECOM, Fraunhofer and UEP (in alphabetical order).

6.1 The challenge

The MediaEval Search and Hyperlinking task tackles the issue of information seeking in a video dataset [7]. The scenario proposed is that of a user looking for a known video segment, and who could be interested in watching related content proposed by the system. Hence, such a task is at the heart of LinkedTV: it is a way to test the LinkedTV framework for hyperlinking videos together on a TV dataset, in real-world conditions. After evaluation, such a work can be extended to hyperlinking of videos to web content.

The dataset offered for this task contains 2323 videos from the BBC, amounting to 1697 hours of television content of all sort: news shows, talk shows, series, documentaries, etc. The collection contains not only the videos and audio tracks, but also some additional information:

- subtitles (manually transcribed)
- two types of ASR transcripts (LIMSI [11] and LIUM/Vocapia [28])
- metadata giving show title, description, date of airing, format, etc
- additional metadata: synopsis and cast from the BBC web site)
- shot boundaries and keyframes
- face detection and face similarity information
- concept detection

6.1.1 Search task

The goal of the Search task is to search for a known video segment in the dataset, using a textual query provided by a user. The query is constituted of two parts: the first part gives information for a text search while the other part gives cues on visual information in the searched segments, using words. 29 users defined 50 search queries related to video segments watched among this dataset.

Here are two examples of query:

```
<top>
  <itemId>item_1</itemId>
  <queryText>Little Britian Fat Fighters problem of gypsies in the area</queryText>
  <visualQueues>fat club comedfy</visualQueues>
</top>
<top>
  <itemId>item_2</itemId>
  <queryText>(Two pints of lager) Sandwich Maker Dirty Shirt My girlfriend is a
     housekeeper </queryText>
  <visualQueues>The guy with the dirty shirt In a Pub Having a Chat and making fun of
     his girlfriend </visualQueues>
</top>
```

The evaluation of the search task is based on the following measures:

[21] http://www.multimediaeval.org/mediaeval2013/
[22] http://acmmm13.org/

- the Mean Reciprocal Rank (MRR) assesses the ranks of the relevant segment returned for the queries. It averages the multiplicative inverse of the ranks of the correct answers (within a given time windows, here 60s).
- the Mean Generalized Average Precision (mGAP) is a variation of the previous that takes into account the distance to the actual relevant jump-in point. Hence, this measure also takes into account the start time of the segment returned.
- the Mean Average Segment Precision (MASP) assesses of the search in term of both precision of the retrieved segments and the length of the segments that should be watched before reaching the relevant content [9]. It takes into account the length of overlap between the returned segments and the relevant segment. It hence favors segments whose boundaries are close to the expected ones.

A window size of 10, 30 and 60 seconds was originally planned to report results, but in the final results delivered, only the 60s time window was used.

6.1.2 Hyperlinking task

The Hyperlinking task aimed at offering to the viewer content from the dataset related to what (s)he is watching. The user defines an anchor (a video segment, identified by the video name, the begin time and end time): this anchor is the basis for seeking related content in the collection. Condition LA requires to use only this segment for the hyperlinking, while condition LC also allowed to use the context of this segment (i.e. a broader segment from the video).

Examples of hyperlinking queries:

```
<anchor>
  <anchorId>anchor_1</anchorId>
  <startTime>13.07</startTime>
  <endTime>13.22</endTime>
  <item>
   <fileName>v20080511_203000_bbcthree_little_britain</fileName>
   <startTime>13.07</startTime>
   <endTime>14.03</endTime>
  </item>
</anchor>
<anchor>
  <anchorId>anchor_2</anchorId>
  <startTime>2.22</startTime>
  <endTime>2.37</endTime>
  <item>
   <fileName>v20080510_002000_bbcthree_two_pints_of_lager_and</fileName>
   <startTime>2.07</startTime>
   <endTime>3.27</endTime>
  </item>
</anchor>
```

A manual evaluation by users will follow by performing crowd-sourcing evaluation through Mechanical Turk. It was carried out for the top 10 ranks of the runs. Mean Average Precision will be reported, as well as precision at different ranks (i.e. how many relevant targets were retrieved in the top n ranks).

6.2 Pre-processing step

First, we worked on further processing the dataset, in order to have as much information as possible. We summarized the pre-processing performances in table 8: the processing time is fairly important due to the size of the dataset (1697 hours of video). The data extracted is described in the next paragraphs.

6.2.1 Concepts Detection

For visual concept detection, we follow the approach presented in [29], using a sub-set of 10 base (key-frame) detectors. The algorithm is applied on the key-frames of the video, aiming to detect 151 semantic

Table 8: Performances of the different analysis techniques on the whole dataset

Concepts Detection	20 days on 25 4-cores computers
Scene Segmentation	2 days on 6 cores
Keywords Extraction	5 hours
OCR	1 days on 10 cores
Face Detection and Tracking	4 days on 40 4-cores computers
Named Entities Extraction	4 days

concepts, both static and dynamic ones, selected from the list of concepts defined in the TRECVID 2012 SIN task [21]. The 10 used classification modules are derived from different combinations of the employed interest point detector, descriptor and visual word assignment method. Specifically, the considered interest point detectors are the Harris-Laplace corner detector [10] and a dense pixel sampling strategy, while the employed descriptors are the well known SIFT [15] and two colored variations of it, named RGB-SIFT and Opponent-SIFT [34]. Then, the low-level descriptors are assigned to visual words from two vocabularies that were created off-line through K-means clustering, employing hard- and soft-assignment [35], respectively.

For each one of the employed classification modules, one vector per key-frame is finally extracted and used as the actual input to the utilized SVM classifier. In order to increase the computational efficiency, linear SVM classifiers are employed instead of kernel SVMs that are typically used for this task, while another boost in time performance is obtained by using only one range file for the classification of the overall set of concepts (not one range file per concept, as when the algorithm runs for a small collection of videos). The latter results in a slightly lower detection accuracy, however reducing by 145 times the needed processing time, which is crucial when the algorithm is applied on large collections of videos, such as the MediaEval dataset. The output of each of the employed classifiers is a Degree of Confidence (DoC) score for the corresponding concept, which expresses the classifier's confidence in this concept being suitable for annotating the current shot. This process is iterated for each considered concept and for all used modules, and the extracted DoC scores are averaged to generate the final concept detection score. Finally, a vector of such scores, where each element of the vector corresponds to a different concept, is the system's output.

6.2.2 Scene Segmentation

The employed technique for scene segmentation is based on the algorithm introduced in [30]. This method groups the shots of the video (either automatically detected, or predefined as in the MediaEval 2013 Search and Hyperlinking task) into sets that correspond to individual scenes of the video, based on the visual similarity and the temporal consistency among them. Specifically, one representative key-frame is extracted from each shot of the video and the visual similarity between pairs of key-frames is estimated via HSV histogram comparison. The grouping of shots into scenes is then performed, by utilizing the two proposed extensions of the well-known Scene Transition Graph (STG) method [38], which clusters shots into scenes by examining whether a link, between two shots, exists.

The first extension, called Fast STG, reduces the computational cost of shot grouping, by considering shot linking transitivity and the fact that scenes are by definition convex sets of shots, thus limiting the number of shot pairs whose possible linking needs to be evaluated. The latter allows for faster detection of the scene boundaries, while maintaining the same performance with the original STG algorithm. The second extension, called Generalized STG, builds on the former in order to construct a probabilistic framework, towards multiple STGs combination, alleviating the need for manual STG parameter selection. As described in [30], this probabilistic framework can also be used for the realization of a multi-modal approach for scene segmentation, allowing the fusion of STGs built by considering different forms of information extracted from the video, such as low level audio or visual features, visual concepts and audio events.

6.2.3 Keywords Extraction

Keyword extraction is based on the algorithm as presented at [33]. It is mainly based on the inverse document frequency (TF-IDF, see [27]) paradigm and employs Snowball[23] as stemmer. The stop-word list was manually curated based on the subtitles.

[23] http://snowball.tartarus.org/

6.2.4 Optical Character Recognition (OCR)

For text localization, we employ the algorithm presented at [31]. The detection is based on color segmentation, using statistical region merging [20]. For a refined text separation, a Gaussian model is computed based on uniform colored connected components. For Optical Character Recognition (OCR), we employ the widely used tesseract[24] engine.

6.2.5 Face Detection and Tracking

Face analysis starts with face detection, where all frames of the videos are processed to extract faces. For this task, we use the well-known Viola and Jones' cascade face detector [36], or more precisely its implementation in the C++ openCV library as improved by Lienhart and Maydt [13]. Detection is combined with a skin color detector [18] for filtering out detections that are not likely to be faces.

The tested framework performs well on images. However, we had to adapt it to videos and to create face tracks. We use spatio-temporal information in order to smooth the results. We link detected faces through shots using a spatio-temporal matching of faces: if two faces in adjacent frames are in a similar position, we assume we can match them. Linear interpolation of missing faces also relies on matching similar bounding boxes in close but none adjacent frames through a shot. This process enables to smooth the tracking results and to reject some false positive (when a track is too short, it is considered as a false alarm).

6.2.6 Named Entities Extraction

For every video subtitle in the collection, we perform a named-entity recognition process using the NERD framework [23, 24, 26, 25]. Due to the huge size of the corpora, the extraction operation has rely exclusively in the service *Textrazor*[25], which provides low response times and a bigger processing quota than other available alternatives (thanks to a special academic agreement). For TV shows with long duration, the transcripts have been divided into chunks and submitted separately in order to fit with the maximum number of bytes supported by Textrazor. The output result is a collection of entities per subtitle file, aligned to the NERD Ontology v0.5[26]. In most of the cases, entities are disambiguated to resources in the Web of Data where extra information about them can be retrieved. This information was aimed at enriching the data. In order to use the detected concepts, we needed to be able to add semantics on top of them.

6.3 From visual cues to detected concepts

Linking visual concepts to semantics was a key challenge of this work. Taking as input the visual queues per search query, we have run keyword extraction operations using Alchemy API[27]. We have then aligned every spotted keyword with some LSCOM concepts (currently, from a subset of more than one hundred items currently considered in our automatic concept detection algorithms) by using a semantic word distance based on Wordnet synsets [14]. The output includes two indexes: the confidence score, which helps to decide how close a visual concept is related to a keyword, and the relevance, which gives an idea of the importance of a certain concept inside the scope of the text submitted. Manual evaluations lead us to discard concepts when the confidence score was below 0.7.

In the next section, we will see the next steps of this work: designing a framework for the search and hyperlinking tasks.

6.4 Lucene indexing

We have used the search platform Solr (see section 3.1.2). We indexed all available data in a Lucene index at different granularities: video level, scene level, shot level, subtitle block level, speech segments from transcripts. Hence, searches can be performed at a chosen granularity, the next step being to design an appropriate query.

Different indexes were created with different information, as described below. Documents were represented by both textual fields (for a text search) and floating point fields (with a float value). Float

[24] http://code.google.com/p/tesseract-ocr/
[25] http://www.textrazor.com/
[26] http://nerd.eurecom.fr/ontology/nerd-v0.5.n3
[27] http://www.alchemyapi.com/

fields are used to represent concepts: search will be performed using a range query: e.g., query for documents where the concept `Animal` is between 0.63 and 1.

6.4.1 Video index

Videos were defined using their name. Different text fields enabled to store and index diverse information taken from the metadata: title of the series the show is part of, episode title, channel the video was aired on, synopsis, short synopsis, description, cast. We also indexed subtitle and keywords extracted, plus the number of shot in each video(for the purpose of some search algorithm described in the next section).

6.4.2 Scene index

Scenes (created using scene segmentation, see 6.2.2) were defined using the id of the video they are part of, with begin time and end time. Each scene was aligned with subtitles. Hence, we indexed each scene with associated textual information: subtitle and various metadata (title, cast, synopsis, etc). Field representing concepts were also introduced. For each concept, we stored the value of the highest score of the concept across shots that constitute the scene.

6.4.3 Shot index

Similarly to scenes, shots were defined using the id of the video they are part of, with begin time and end time. We indexed subtitles (aligned to each shot) as well as OCR when detected. The number of faces per shot was also indexed but not used in the search and hyperlinking algorithms.

6.4.4 Speech segments index

We create speech segments by merging adjacent speech segments when they were uttered by the same speaker. Speech segments don't overlap and are connected to each other. It was prepared only for LIMSI transcripts using information about speakers. Hence, speech segment are defined using video id, begin time and end time. The indexed text was the corresponding transcript. We also made one attempt to use synonyms at index time.

This algorithm produces short video segments, whose length can vary from 1 to 45 seconds. This algorithm is not suitable for videos with one person speaking, because it will produce an extreme situation with a single long segment. An improvement that can be done is to take into account dialog/trialog/multidialog based segments or more speech segment from different speakers in one video segment. In the final runs, we didn't submit any results based on speech segments because of poor results.

6.4.5 Sliding windows index

Sliding window algorithm [8] was used with different parameters and different schema in Apache Solr. The index was created as described below.

First, each document is divided into sentence segments. Because of different format of data, the sentence segments have to be prepared differently. For LIUM transcripts, a fix length "sentence" of about 17 words was used, as it is considered as the average length of English sentence [37]. For LIMSI transcripts, splitting is done on punctuation. Finally, for subtitles, each line in `<span>` tag was extracted as one sentence.

Sentences are processed using POS (part-of-speech) tagging (for each sentence segment) to obtain information about the number of open words they contain. Open words are those words that carry the content or the meaning of a sentence - verbs, nouns, adjectives, adverbs and interjections. For open class words recognition, MaxtenTagger[28] was used.

Last, we index sentences as follows: we insert each sentence into sliding window and count open words for the whole actual window (a window can contain many sentences). When the sliding window is full, i.e. when the count of open class words is bigger than the parameter specified at startup, we merge sentences into one text and index it as one segment (start time is start time of first sentence and end time is end time of last sentence). We then remove the first sentence and repeat this process until we reach the end of the document. The last segment in the document can have a smaller count of open class words than other. This algorithm produces many overlapping segments and the density of coverage document is high.

[28] http://nlp.stanford.edu/software/tagger.shtml

6.5 Search task

The search task required to divide videos into smaller segments, between 1 and 15 minutes. We submitted 9 different runs by adopting two different strategies: we either used pre-constructed segments that were indexed in the Lucene engine, or we performed queries creating segments on the fly, by merging video segments based on their score.

For most of the search, we concatenated textual and visual query (content of *queryText* and *visualQueues*) to perform the text query, as it yields better results than using the text from the textual query only (when otherwise using the same settings).

After indexing the entire collection of videos, we found out that performing a text query on the video index returned the accurate video in the top of the list. Hence, for some runs, we could restrict the pool of videos that are going to be searched to a small number. The query then has to be made in two steps: first, we query the overall video index, extracting the 20 first videos, and then we perform additional queries for smaller segments restricted to this smaller dataset.

Search with visual cues. Text search is straightforward with Lucene/Solr: the search engine provides a default text search based on TF-IDF values. For this reason, incorporating visual information in the search task requires to design a new query that combines the provided visual cues and the output of the visual concept detection algorithm. Therefore, we initially created a mapping between keywords (i.e. terms) extracted from the visual cues query via text analysis 6.3, and the visual concepts that are considered by the applied concept detection algorithm. Based on this mapping, we were able to integrate the output of this algorithm (i.e. presence of visual concepts in the searched segment) into the search task, resulting in an enriched query that includes both textual and visual information.

For each one of the considered visual concepts we defined two values: its highest score across the entire group of key-frames/shots of the MediaEval data-set and a “valid detection” rate, calculated by examining its presence/absence within the key-frames/ shots of the MediaEval dataset that correspond to the top-100 highest scores for this concept. Built on this, we performed a filtering step using the “valid detection” rate, that occurred from the top-100 highest scores, as a new confidence score about this concept and we then discarded the concepts with a rate lower than 0.5. Afterwards, we linearly normalized these rates between 0 and 1, by mapping the highest score among them to 1.

Finally, when a visual concept was detected from the visual cues query and if it hadn’t been rejected by the previous filtering, we also added a range query to the Lucene query, beside the textual query: in order for the concept to be present, its normalized score had to lie above a threshold value, i.e. between this threshold and 1. We empirically set the threshold at 0.7. Hence, we considered that all the normalized scores above this threshold truly contain the given concept. The range query was as follows:

```
Animal:[0.7 TO 1]
```

We submitted 9 runs in total:

- *scenes-C*: Scene search using textual and visual cues. No filtering by video.
- *scenes-noC*: Scene search using textual cues only. For comparison purposes, this run has the same settings as the previous one, except that it makes no use of the visual concepts.
- *part-scenes-C*: Partial scenes search from shot boundary using textual and visual cues. This search was made in three steps: first, we filtered the list of videos as explained earlier; then we queried for shots inside each video and ordered them by score. As a shot is a unit that is too small to be returned to a viewer, we completed the segment using the scenes boundaries: for each shot, we created a segment that begins with the shot and ends at the end of the scene this shot is part of.
- *part-scenes-noC*: Partial scenes search from shot boundary using textual cues only. Same settings as the previous except from the visual concepts.
- *clustering10-C*: Temporal clustering shots within a video using text and visual cues. The clustering was done as following: First, we filtered out the set of videos to search as explained earlier. Second, we computed scores of every shot in the video, and clustered together shots that were closer than 10 seconds apart. We added the score of shots together to form the score of the segment.

- *clustering10-noC*: Temporal clustering shots within a video using text search only. Same settings as the previous except from the visual concepts.
- *scenes-S or scenes-U or scenes-I*: Scene search using only textual cue from transcript (scenes-I for LIMSI, scenes-U for LIUM) or subtitle (scenes-S). No metadata was used.
- *slidingWindows-60-I or slidingWindows-60-S*: Search over segments created by the sliding window algorithm (section 6.4.5) for LIMSI transcripts (slidingWindows-60-I) and subtitles (slidingWindows-60-S). The size of the sliding windows is 60.
- *slidingWindows-40-U*: The same as above for LIUM transcript with sliding window size of 40.

6.6 Hyperlinking task

For the hyperlinking task, we performed two styles of query, either using the search component or using a feature of the Solr component.

6.6.1 Hyperlinking using the search component

Our first approach amounts to use the search component to perform the hyperlinking task. The challenge was to create the query from the given anchor. For the first condition (LA), the text query was made by extracting keywords from the subtitle (aligned at start time and end time), using Alchemy API[29] default keyword extraction. The visual cues were extracted from the shots contained in the anchor. If the anchor was constituted by more than one shot, we took for each concepts the highest score over all shots. Indeed, we considered that if a concepts appeared in at least one shot, it should be taken into account. Then, we performed queries using the scene approach and the shot clustering approach.

For the second setting (LC), we used the following: for the textual query, we use both keywords extracted from subtitle aligned to the anchor and to the context, and gave a higher weight to words coming from the anchor than to words from the context. For the visual query, we took the highest score of all concepts across the anchor and its context.

6.6.2 MoreLikeThis

Our second approach amount to use the MoreLikeThis feature of the Solr component combined with THD (Targeted Hypernym Discovery) annotation. It is based on the premise that similar video segments are those that have the same topic. If a user is watching a video, e.g. on the theme “drugs and legalisation of cannabis”, we suppose that he is also interested in other videos with this subject. We experimented with THD and MoreLikeThis components.

THD : this is an unsupervised entity discovery and classification system. Recognized entities are enriched with links from Wikipedia (resp. DBpedia) and types from DBpedia and YAGO knowledge bases providing high semantic interoperability[30]. For example, for the text “Vltava” spotted in one video segment, THD returns annotations such as “Vltava, Rivers of the Central Bohemian Region, River, BodyOfWater, Place,” etc.

MoreLikeThis : this feature from Apache Solr is aimed at finding documents similar to the provided one. It compares documents based on specific fields. More information about the functionality of MoreLikeThis can be found in the official documentation of Solr[31].

The hyperlinking framework has 5 main steps:

- make segments from the LIMSI transcript with sliding window algorithm
- annotate text with THD annotations and index them
- create, annotate and index temporary document from the anchor for query
- ask with MorelikeThis for similar documents
- delete temporary document (to exclude contamination of index)

[29] http://www.alchemyapi.com/
[30] https://ner.vse.cz/thd/about/
[31] http://wiki.apache.org/solr/MoreLikeThis

6.7 Results

In this section, we report the results of our approaches given the measures described in 6.1.1 and 6.1.2. We do not have a baseline to compare to, nor the evaluation of other participants' runs. Those results will be disclosed during the day of the MediaEval workshop, which takes place in Barcelona, Catalunya, Spain, on Friday-Saturday 18-19 October 2013. Note that all these results are still being validated by the organizers of the benchmarking task. Bugs in the evaluation script have been found at the time of writing this deliverable and it is likely that all scores reported in this section will change. We expect that the updated results will be soon available at the challenge web site at `http://www.multimediaeval.org/mediaeval2013/hyper2013/`.

6.7.1 Search task

The results of the search task are listed in table 9. Overall, runs using scenes and the sliding window with the size 60 approach have the best performances. First, we notice than under the same conditions, subtitles perform significantly better than any of the transcripts. This was expected since subtitles are human-generated and are thus more accurate than transcripts. Hence, the runs that got the best results for each measure were generated using subtitles. If the subtitles were not available, the best option would be to use the scenes with LIMSI transcripts (scenes-l) approach as it yields the higher results over all measures when using transcripts.

It is also interesting to note that using the visual concepts in the query slightly increases the results for all measures (e.g., clustering10-C vs clustering10-noC). This could even be improved when more visual concepts will be taken into account (we used a subset of 151 concepts).

Table 9: Results of the Search task

Run	MRR	mGAP	MASP
scenes-C	**0.30946346604**	**0.176992668643**	0.195100503655
scenes-noC	0.309135738072	0.176714986483	0.194691217552
scenes-S	**0.315224269655**	0.163526098095	**0.202065496585**
scenes-l	0.261337609154	0.144401090929	0.158159589898
scenes-U	0.245763345432	0.134440892192	0.152835325636
part-scenes-C	0.22839916764	0.12414594745	0.102360579399
part-scenes-noC	0.228115789255	0.12399354915	0.102092742224
clustering10-C	0.292888150402	0.152518978931	0.181357372131
clustering10-noC	0.284924316767	0.147870395447	0.171263250592
slidingWindows-60-S	0.283311232393	**0.192486941209**	**0.202706179669**
slidingWindows-60-l	0.196479921556	0.120561880191	0.120417180025
slidingWindows-40-U	0.236816510313	0.134195367345	0.150069623509

In the following, we further look into the measures and compare the three runs that have the best performances: scenes-C, scenes-S and slidingWindows-60-S. MRR favors the approaches using scenes, scenes-S having slightly higher results. Hence, retrieving segments as scenes returns a better list of results in terms of "corresponding" segments, within a time window of 60s. When looking at mGAP, the sliding window approach is better: as this measure takes into account the distance of the jump-in point to the actual start of the video, it shows that using sliding windows is more precise when it comes to a more fine-grained analysis. Retrieved segments start points are closer to the actual ones, to an extent that enables them to catch up with other approaches with better MRR ranking. Last, MASP measures are very similar for scenes-S and slidingWindows approaches, the latter being higher by very little. This measure takes into account the boundaries of the segment watched: once again, such results hint that the sliding windows approach produces segments closer to the expected ones in term of start time and end time. A next step of this work could then be to refine the scenes in order to preserve the ranking but redefine the start and end times.

We can question to which extent it is possible to retrieve a segment within a very small time window, let's say 10 seconds. To this effect, it would be interesting to compare the degree of precision (in term of segment boundaries) offered by the sliding windows approach with what could be produced by a human.

Table 10: Results of the Hyperlinking task

Run	MAP	P-5	P-10	P-20
LA clustering10	0.2764	0.3733	0.2967	0.2200
LA THDMoreLikeThis	0.4760	0.4667	0.4533	0.3533
LA scenes	**0.5848**	**0.7467**	**0.6633**	**0.5233**
LC clustering10	0.3392	0.4467	0.3733	0.2417
LC THDMoreLikeThis	0.5457	0.5733	0.5433	0.4483
LC scenes	**0.7042**	**0.8533**	**0.8000**	**0.6683**

6.7.2 Hyperlinking task

For the hyperlinking task, there were originally 98 anchors. Due to time and financial constraints, the organizers have chosen a representative subset of 30 anchors to be evaluated via crowdsourcing. The results can be seen in table 10: MAP is the mean average precision, while P-5, P-10 and P-20 are the precisions at rank 5, 10 and 20.

For both LA and LC conditions, runs using scenes outperform other runs for all metrics. The THD/-MoreLikeThis approach comes second, and the least performant run is the one using the clustering search method. As expected, using the context increases the precision when hyperlinking video segments. It is also notable that the precision at rank n decreases with n. From this observation, we can conclude that those approaches will suit a scenario of a search engine, where mainly the top results will raise the user's attention.

6.7.3 Some figures on visual concepts

As we described earlier in this document, keywords spotted in the text query through a mapping were linked to visual concepts. We performed a filtering over concepts at two levels: first, we discarded visual concepts where the "valid detection" score among the top 100 scores was under 0.5. Second, we also applied a threshold of 0.7 on the mapping between text and concepts on the 50 search queries.

Table 11: Number of concepts at various stages

Filtering level	# concepts before filtering	# concepts after filtering	# concepts final
visual level	151	47	41
mapping level	976	144	over 50 queries

As can be seen in table 11, our filtering stages drastically reduces the number of concepts that are actually taken into account in the queries. 976 concepts were discovered in the 50 queries, and we eventually only take 41 of them into account in the final queries: those 41 concepts are spread over 23 queries, the remaining 27 queries having no visual concept attached after filtering. However, search results indicate that using concepts globally provides an improvement over textual query. This finding is very promising and should lead to greater research work in that direction.

6.8 Lessons learned and future work

This challenge enabled us to test our approaches with an entire dataset of diverse shows of a TV provider, which is very close to the LinkedTV conditions. Real users evaluated the results in the hyperlinking task, which is of great interest for our work.

The results evidence that approaches using scenes outperform others. In the hyperlinking task, using scenes along with context of the anchor returns results with a high precision (0.85 and 0.80) at rank 5 and 10: hence, proposing the first items to a viewer is very likely to reach the expected goal. Hence, improving on scene segmentation, by using features such as speaker clustering, faces or semantics may be a future direction to study.

We also observe that using visual concepts in the queries improve the results: we filtered both concepts detected in the videos and concepts mapped from the query, and still got better results than without using those visual cues. Those outcomes show that there is room for improvement and the use of visual concepts is another direction that should be better explored. We see two main processes that worth being looked at: first, at the text query level, the mapping between text and concepts; second, at the video analysis level, the confidence in visual concepts from the video.

7 Conclusion

In this deliverable, we already described the first software results for retrieval of enrichment for seed media fragments. The consortium has developed a diverse set of tools for acquiring enrichment content from various web sources including common media web services. The Unstructured search module provides a consortium-hosted search engine, which crawls content on a set of white-listed web sites and makes it available via full-text search exposed via a REST API. The Media Collector provides a single point for placing a query over a range of popular web resources for media content, including Flickr and YouTube. At the time of writing, these services have been already integrated with the LinkedTV platform, which provides a central point for orchestrating the analysis of seed content and its enrichment. The LSI service, which provides access to a list of API's complementary to those interfaced by the Media Collector, is yet to be integrated. This deliverable gave a concise specification of the APIs exposed by the platform[32].

The validity of the approach taken has been underpinned by the participation of the LinkedTV consortium in the search and hyperlinking tasks of the MediaEval contest. This task is particularly relevant to the LinkedTV use case, since it tested the ability of the tools and techniques available within the consortium to handle the problem of retrieving multimedia content relevant to a textual query and to another video segment. This deliverable gave a brief account of the approaches taken and the results obtained.

[32] A more detailed description of the platform is also available in the Deliverable D5.4 [5]

References

[1] LinkedTV consortium. Deliverable 2.3: Specification of web mining process for hypervideo concept identification, 2012.

[2] LinkedTV consortium. Deliverable 6.1: Scenario descriptions, 2012.

[3] LinkedTV consortium. Deliverable 1.3: Linkedtv annotation tool - first release, 2013.

[4] LinkedTV consortium. Deliverable 2.4: Annotation and retrieval module of media fragments, 2013.

[5] LinkedTV consortium. Deliverable 5.4: Final linkedtv integrating platform, 2013.

[6] LinkedTV consortium. Deliverable 6.2: Scenario demonstrators, 2013.

[7] M. Eskevich, J Gareth J.F., S. Chen, R. Aly, and R. Ordelman. The Search and Hyperlinking Task at MediaEval 2013. In *MediaEval 2013 Workshop*, Barcelona, Spain, October 18-19 2013.

[8] M. Eskevich, G.J.F. Jones, C. Wartena, M. Larson, R. Aly, T. Verschoor, and R. Ordelman. Comparing retrieval effectiveness of alternative content segmentation methods for Internet video search. In *Content-Based Multimedia Indexing (CBMI), 2012 10th International Workshop on*, pages 1–6, 2012.

[9] Maria Eskevich, Walid Magdy, and Gareth J. F. Jones. New metrics for meaningful evaluation of informally structured speech retrieval. In *Proceedings of the 34th European conference on Advances in Information Retrieval*, ECIR'12, pages 170–181, Berlin, Heidelberg, 2012. Springer-Verlag.

[10] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of 4th Alvey Vision Conference*, pages 147–151, 1988.

[11] Lori Lamel and Jean-Luc Gauvain. Speech processing for audio indexing. In *Advances in Natural Language Processing (LNCS 5221)*, pages 4–15. Springer, 2008.

[12] Sheldon Levine. How People Currently Share Pictures On Twitter, 2011. `http://blog.sysomos.com/2011/06/02/`.

[13] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900 – I–903 vol.1, 2002.

[14] Dekang Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[15] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[16] Vuk Milicic, José Luis Redondo García, Giuseppe Rizzo, and Raphaël Troncy. Tracking and Analyzing The 2013 Italian Election. In *10^{th} Extended Semantic Web Conference (ESWC'13), Demo Session*, Montpellier, France, 2013.

[17] Vuk Milicic, Giuseppe Rizzo, José Luis Redondo García, Raphaël Troncy, and Thomas Steiner. Live Topic Generation from Event Streams. In *22^{nd} World Wide Web Conference (WWW'13), Demo Session*, Rio de Janeiro, Brazil, 2013.

[18] J. See N. A. Abdul Rahim, C. W. Kit. RGB-H-CbCr Skin Colour Model for Human Face Detection. In *MMU International Symposium on Information & Communications Technologies (M2USIC)*, Petaling Jaya, Malaysia, 2006.

[19] Lyndon Nixon. The importance of Linked Media to the Future Web: a proposal for the linked media research agenda. In *22^{nd} International World Wide Web Conference companion (WWW'13)*, pages 455–456, Rio de Janeiro, Brazil, 2013.

[20] R. Nock and F. Nielsen. Statistical Region Merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(11):1452–1458, November 2004.

[21] Paul Over, George Awad, Martial Michel, Jonathan Fiscus, Greg Sanders, Barbara Shaw, Wessel Kraaij, Alan F. Smeaton, and Georges Quéenot. TRECVID 2012 – An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In *Proceedings of TRECVID 2012*. NIST, USA, 2012.

[22] Giuseppe Rizzo, Thomas Steiner, Raphaël Troncy, Ruben Verborgh, José Luis Redondo García, and Rik Van de Walle. What Fresh Media Are You Looking For? Retrieving Media Items from Multiple Social Networks. In *International Workshop on Socially-aware multimedia (SAM'12)*, Nara, Japan, 2012.

[23] Giuseppe Rizzo and Raphaël Troncy. NERD: A Framework for Evaluating Named Entity Recognition Tools in the Web of Data. In *10th International Semantic Web Conference (ISWC'11), Demo Session*, pages 1–4, Bonn, Germany, 2011.

[24] Giuseppe Rizzo and Raphaël Troncy. NERD: Evaluating Named Entity Recognition Tools in the Web of Data. In *Workshop on Web Scale Knowledge Extraction (WEKEX'11)*, pages 1–16, Bonn, Germany, 2011.

[25] Giuseppe Rizzo and Raphaël Troncy. NERD: A Framework for Unifying Named Entity Recognition and Disambiguation Extraction Tools. In *13th Conference of the European Chapter of the Association for computational Linguistics (EACL'12)*, 2012.

[26] Giuseppe Rizzo, Raphaël Troncy, Sebastian Hellmann, and Martin Bruemmer. NERD meets NIF: Lifting NLP Extraction Results to the Linked Data Cloud. In *5th International Workshop on Linked Data on the Web (LDOW'12)*, Lyon, France, 2012.

[27] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[28] A. Rousseau, F. Bougares, P. Deléglise, H. Schwenk, and Y. Estève. LIUM's systems for the IWSLT 2011 Speech Translation Tasks. In *Proceedings of IWSLT 2011*, San Francisco, USA, 2011.

[29] Panagiotis Sidiropoulos, Vasileios Mezaris, and Ioannis Kompatsiaris. Enhancing Video concept detection with the use of tomographs. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2013.

[30] Panagiotis Sidiropoulos, Vasileios Mezaris, Ioannis Kompatsiaris, Hugo Meinedo, Miguel Bugalho, and Isabel Trancoso. Temporal Video Segmentation to Scenes Using High-Level Audiovisual Features. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(8):1163–1177, August 2011.

[31] Daniel Stein, Stefan Eickeler, Rolf Bardeli, Evlampios Apostolidis, Vasileios Mezaris, and Meinard Müller. Think Before You Link – Meeting Content Constraints when Linking Television to the Web. In *Proc. NEM Summit*, Nantes, France, October 2013. to appear.

[32] Thomas Steiner. A Meteoroid on Steroids: Ranking Media Items Stemming from Multiple Social Networks. In *22nd World Wide Web Conference (WWW'13), Demo Session*, Rio de Janeiro, Brazil, 2013.

[33] S. Tschöpel and D. Schneider. A lightweight keyword and tag-cloud retrieval algorithm for automatic speech recognition transcripts. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association ISCA (INTERSPEECH)*, 2010.

[34] Koen van de Sande, Theo Gevers, and Cees Snoek. Evaluating Color Descriptors for Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, September 2010.

[35] Jan C. van Gemert, Cor J. Veenman, Arnold W. M. Smeulders, and Jan-Mark Geusebroek. Visual Word Ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, July 2010.

[36] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001.

[37] G.C. Whitworth. *Indian English: an examination of the errors of idiom made by Indians in writing English*. Bahri Publications, 1982.

[38] Minerva Yeung, Boon-Lock Yeo, and Bede Liu. Segmentation of video by clustering and graph analysis. *Computer Vision and Image Understanding*, 71(1):94–109, July 1998.