Deliverable D2.4 Annotation and retrieval module of media fragments

José Luis Redondo García, Raphaël Troncy (EURECOM)
Dorothea Tsatsou, Dimitrios Panagiotou (CERTH)

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	D2.4
Deliverable name	Annotation and retrieval module of media fragments
File	`D2.4.tex`
Nature	Report
Status & version	Final & v1.0
Number of pages	45
WP contributing to the deliverable	2
Task responsible	EURECOM
Other contributors	CERTH
Author(s)	José Luis Redondo García, Raphaël Troncy (EURECOM) Dorothea Tsatsou, Dimitrios Panagiotou (CERTH)
Reviewer	Michiel Hildebrand (CWI)
EC Project Officer	Thomas Kuepper
Keywords	Metadata models, Ontologies, Metadata Converters, Hyperlinking, Enrichment, Media Fragments, Media Annotations, LUMO, RDF
Abstract (for dissemination)	This deliverable presents an update of the LinkedTV metadata model as part of the WP2 of the LinkedTV project. It includes a set of mappings with other ontologies such as LUMO used in WP4. Second, we describe the converter module named TV2RDF, implemented as a REST service, that populates the LinkedTV triple store with RDF data resulting from the automatic conversion of legacy metadata provided by the content provider, of automatic analysis results generated by WP1, and of named entity recognition and disambiguation applied on subtitles provided with the content. Third, we describe the model and the service that aims to provide enrichments for particular media fragments of a seed video content. Finally, we present a number of useful SPARQL queries that are typically needed by the LinkedTV player or other clients that wish to reuse this semantic dataset.

History

Table 1: History of the document

Date	Version	Name	Comment
05/07/2013	v0.1	Troncy, EURECOM	Initial Version
02/09/2013	v0.2	Troncy, EURECOM	Add Chapter 2
04/09/2013	v0.3	Redondo, EURECOM	Add Chapter 4
27/09/2013	v0.4	Tsatsou, CERTH	Add Chapter 3
02/10/2013	v0.5	Redondo, EURECOM	Add Chapter 5 and 6
09/10/2013	v0.6	Tsatsou, CERTH	Address first QA comments
10/10/2013	v0.7	Troncy, EURECOM	general proof read

0 Table of Content

1 Introduction

Multimedia systems typically contain digital documents of mixed media types, which are indexed on the basis of strongly divergent metadata standards. This severely hampers the inter-operation of such systems. Therefore, machine understanding of metadata coming from different applications is a basic requirement for the inter-operation of distributed multimedia systems. In the context of LinkedTV, we have to deal with metadata standards that come from both the broadcast industry and the web community. Furthermore, the content will be processed by automatic multimedia analysis tools which have their own formats for exchanging their results. One of the main goal of LinkedTV is to enrich seed video content with additional content, personalized to a particular user, that come from diverse sources including broadcast archives, web media, news and photo stock agencies or social networks.

In the Deliverable D2.2, we have presented a state-of-art and requirements analysis that has led to the first version of the LinkedTV Core Ontology [1]. We have further developed and published this ontology (Section 2). This ontology must be then aligned with the The LinkedTV User Model Ontology (LUMO) in order to enable personalization. We present this alignment in the Section 3. We describe then the two main tools developed for annotating a seed video program (Section 4) and for generating enrichments for this seed video program (Section 5). The first tool is implemented as a REST service called TV2RDF. It takes as input an abstract identifier of a media resource stored by the LinkedTV platform, some legacy metadata, subtitles and results of multimedia analysis generated by WP1, and it is responsible for generating an RDF description of all metadata following the LinkedTV core ontology. The second tool is implemented as a REST service called TVEnricher. It takes as input an annotated seed video program and it will compute for each media fragments a set of enrichments. An essential component for this enrichment is the so-called Media Collector describe in the Deliverable D2.5 [6]. For both services, the Open Annotation is used to serialize the results. The motivation for this annotation differs: in the former case, the motivation is to *annotate* the seed video program while in the latter, the motivation is to *link* to other media items. We provide a set of useful SPARQL queries that enables to retrieve annotations and enrichments as they are stored in the LinkedTV platform (Section 6). Finally, we conclude this deliverable and outline future work in Section 7.

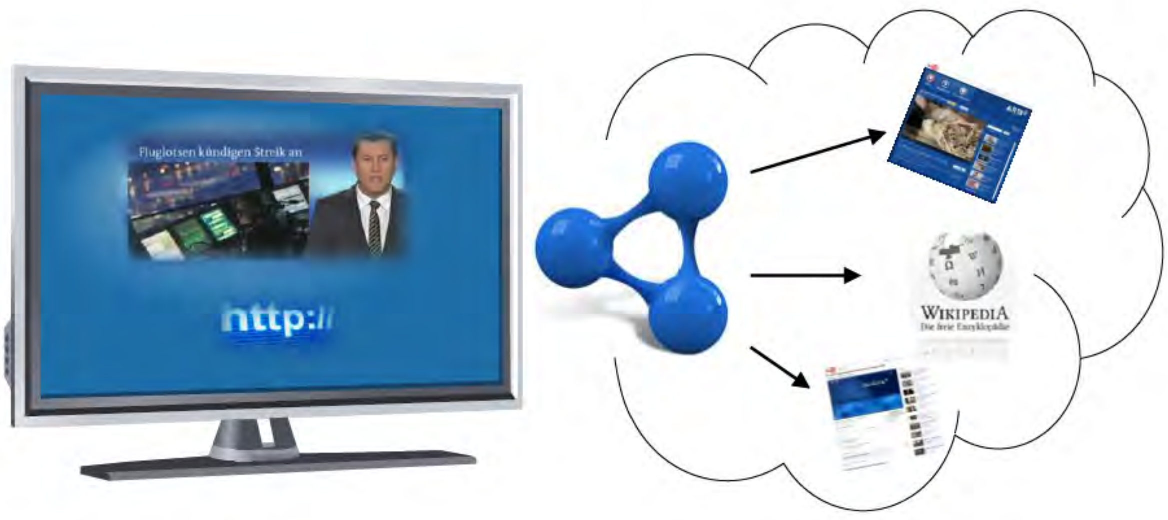


Figure 1: LinkedTV vision

2 LinkedTV Core Ontology

The LinkedTV Core ontology is available at `http://data.linkedtv.eu/ontologies/core`.

2.1 Metamodel architecture

The following vocabularies have been selected as a basis for the LinkedTV Core ontology:

- BBC Program ontology for representing broadcast related metadata: series, episodes, brands, categories, subtitles, physical channel, audio format, video compression, etc.
- Ontology for Media Resources for representing general properties about the content itself such as the title, description, format, license, etc. Also, it contains the classes for representing media items and fragments of media items (*ma:MediaResource* and *ma:MediaFragment*).
- Ninsuna ontology for describing explicitly the media fragments boundaries.
- Open Annotation ontology for linking the analysis results from WP1 (spatiotemporal segments, scene segmentation, shot segmentation, asr, etc.) with media fragments URI. It could also be used for representing additional information such as ratings or user preferences. Finally, it offers support for representing annotations of various types and simple tagging.
- NERD ontology for representing the general types of the named entities recognized by a Named Entity extractor.
- LSCOM ontology for representing the semantics of the visual concepts detected by multimedia analysis processes.
- FOAF ontology for representing the people recognized in video frames.
- PROV-O ontology for representing provenance information.
- LODE Ontology for representing events.

These ontologies are interlinked and import sometimes each other. The BBC Programmes ontology uses FOAF for the descriptions of actors and makes use of the Event ontology for modeling a broadcast as an event. The PROV-O ontology is used by the Core Annotation ontology to describe who has created an annotation and when this annotation has been generated.

The following vocabularies and datasets have also been selected for providing stable URIs of entities and concepts detected by automatic analysis tool. There will be used as values in LinkedTV annotations:

- LSCOM: `http://www.lscom.org/ontology/index.html`
- DBpedia ontology: `http://dbpedia.org/ontology/`
- WordNet 3.0: `http://semanticweb.cs.vu.nl/lod/wn30/`

In the following sections, we will show how the metadata required by the LinkedTV scenarios can be represented using this Linked TV ontology. This includes: the legacy metadata that comes with the seed video content, the metadata resulting from automatic multimedia analysis and the metadata resulting from performing named entity recognition on texts associated with the seed video content (generally the program subtitles).

When converting metadata in RDF, one needs to re-use or generate new identifiers for the first class objects of the model. According to the linked data principles, those identifiers are dereferencable URIs. We follow the best practices of the linked data community and mint new URIs when necessary in the `http://data.linkedtv.eu` domain. Then, the first class objects in LinkedTV are dereferencable using the following scheme:

- `http://data.linkedtv.eu/episode/UUID` for the resources of type `po:Episode`
- `http://data.linkedtv.eu/brand/UUID` for the resources of type `po:Brand`
- `http://data.linkedtv.eu/broadcast/UUID` for the resources of type `po:Broadcast`
- `http://data.linkedtv.eu/version/UUID` for the resources of type `po:Version`

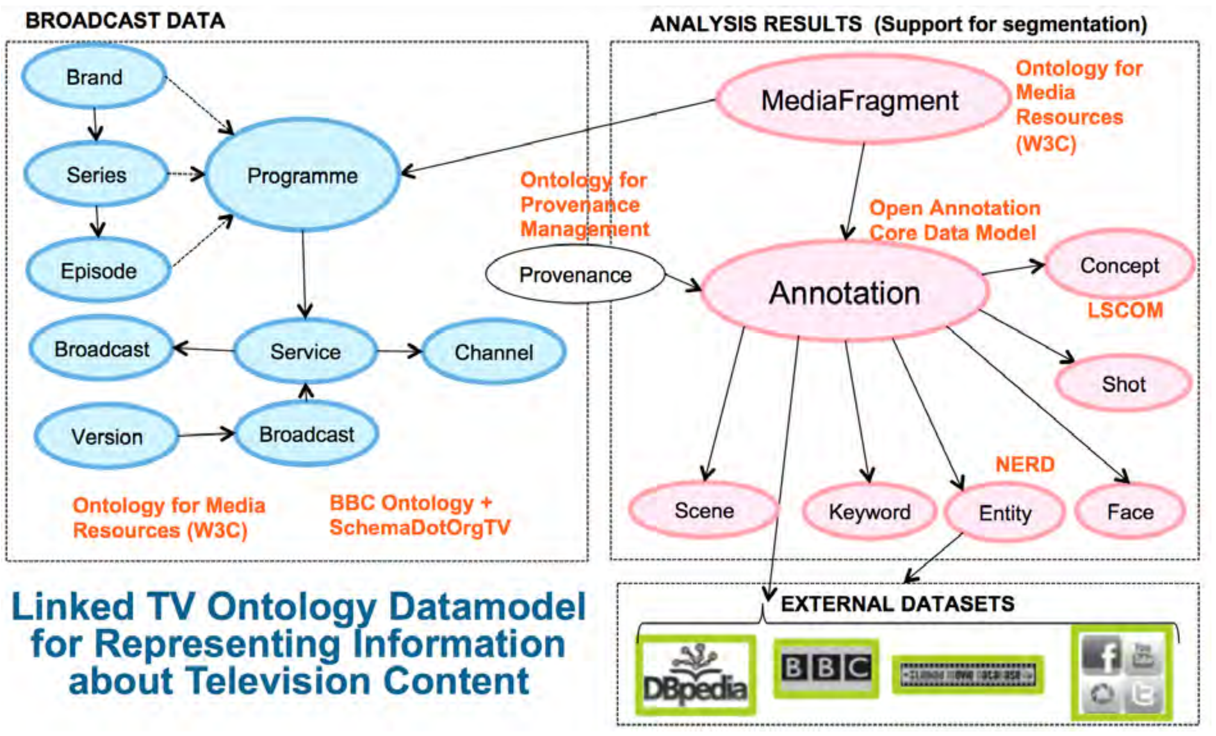

BROADCAST DATA
Brand
Series
Episode
Programme
Broadcast
Service
Channel
Version
Broadcast
Ontology for Media Resources (W3C)
BBC Ontology + SchemaDotOrgTV
Linked TV Ontology Datamodel for Representing Information about Television Content
ANALYSIS RESULTS (Support for segmentation)
MediaFragment
Ontology for Media Resources (W3C)
Ontology for Provenance Management
Provenance
Open Annotation Core Data Model
Annotation
Concept
LSCOM
Shot
NERD
Scene
Keyword
Entity
Face
EXTERNAL DATASETS
DBpedia
BBC

Figure 2: General LinkedTV metadata model

– `http://data.linkedtv.eu/media/UUID` for the resources of type `ma:MediaResource`
– `http://data.linkedtv.eu/annotation/UUID` for the resources of type `oa:Annotation`
– `http://data.linkedtv.eu/organization/UUID` for the resources of type `foaf:Organization`
– `http://data.linkedtv.eu/shot/UUID` for the resources of type `linked:Shot`
– `http://data.linkedtv.eu/person/UUID` for the resources of type `foaf:Person`
– `http://data.linkedtv.eu/entity/UUID` for the resources of type `linkedtv:Concept` or `nerd:Concept`
– `http://data.linkedtv.eu/asr/UUID` for the resources of type `linkedtv:ASR`

In the examples we give below, we generate artificially simple human readable identifiers for all primary objects in the LinkedTV model. However, the TV2RDF converter generates real UUID for identifying those objects (Section 4).

2.2 Annotating RBB Content

RBB has chosen as its seed video content a number of episodes of its daily local news program “RBB Aktuell”. The show is broadcast four times a day but for the project the late broadcast (at 21:45) is the most suitable as it is enhanced with subtitles which help to improve the results of the video analysis. On the one hand, RBB provides legacy metadata in the form of TV-Anytime like metadata. On the other hand, WP1 has processed the RBB videos in order to generate various EXMaRALDA files.

2.2.1 Legacy Metadata

The legacy metadata from RBB are expressed in a TV-Anytime like format. The translation to the BBC Program Ontology is straightforward. The instances created are:

– One instance of the class `po:Episode` that stores the title, the synopsis, the related subjects, and other basic attributes for the current material. Also, this individual has references to the different versions of the episode through the use of the `po:version` property.

```
 <http://data.linkedtv.eu/episode/e311e875-e31e-47d9-b564-88803ae616bc>
  a po:Episode ;
  dc:title "rbb AKTUELL vom 19.06.2013 21:45 Uhr" ;
  po:id "crid://rbb-online.de/rbbaktuell/e311e875-e31e-47d9-b564-88803ae616bc" ;
  po:long_synopsis "+++ Straffes Programm für den US-Präsidenten +++ Berlin schmilzt in der Sonne
      +++ Europa in Berlin +++ Moderation: Andrea Vannahme" ;
  po:microsites <crid://ard.de/bewertbar> ,
                   <crid://rbb-online.de/rbbaktuell/222ea8ad-3bf8-af4b-a9bd-c45dbdaf40af> ;
  po:short_synopsis "+++ Straffes Programm für den US-Präsidenten +++ Berlin schmilzt in der Sonne
      +++ Europa in Berlin +++ Moderation: Andrea Vannahme" ;
  po:subject "Politik" , "Regionales" , "Nachrichten" ;
  po:version <http://data.linkedtv.eu/version/3b2424f1-0943-4a4f-a6c3-49ada9bf4b22> ,
             <http://data.linkedtv.eu/version/ec39aec2-87fc-4780-8a8e-2656810f1b0d> ,
             <http://data.linkedtv.eu/version/11a207de-97c8-45b1-a1b8-51b1947b1869> ,
             <http://data.linkedtv.eu/version/d3d9d936-9223-40b6-b852-ecafaf3d243f> ,
             <http://data.linkedtv.eu/version/7b14881d-21c7-4c1c-aacc-49ab9a365e58> ,
             <http://data.linkedtv.eu/version/b3e7f8d6-03c8-4145-b156-419275544414> .
```

– One instance of the class `po:Brand` that stores information about the brand this episode belongs to.

```
<http://data.linkedtv.eu/brand/222ea8ad-3bf8-af4b-a9bd-c45dbdaf40af>
  a po:Brand ;
  dc:title "rbb AKTUELL" ;
  po:episode <http://data.linkedtv.eu/episode/e311e875-e31e-47d9-b564-88803ae616bc> ;
  po:id "crid://rbb-online.de/rbbaktuell/222ea8ad-3bf8-af4b-a9bd-c45dbdaf40af" ;
  po:microsites <crid://ard.de/sendung> ,
                   <crid://rbb-online.de/rbbaktuell> .
```

– Instances of the class `po:Broadcast` which establish a relationship between a particular version of a program and the `po:Service` instance where this version is broadcasted on.

```
<http://data.linkedtv.eu/broadcast/ace67658-60de-4ef1-a6f3-3bb4de3077ef>
  a po:Broadcast ;
  po:broadcast_of <http://data.linkedtv.eu/version/3b2424f1-0943-4a4f-a6c3-49ada9bf4b22> ;
  po:broadcast_on <rtmp://ondemand.rbb-online.de/ondemand/mp4:rbb/rbbaktuell/rbbaktuell_2145uhr/
      rbbaktuell_20130619_sdg_s_16_9_256x144.mp4> .
```

– Instances of the class `po:Service` which represent the television channel where a particular program is broadcasted.

```
<ftp://linkedtv@ftp.condat.de/rbb/rbbaktuell/>
  a po:Service .

<rtmp://ondemand.rbb-online.de/ondemand/mp4>
  a po:Service .
```

– Instances of the class `po:Version` which represent the appearance of a program at a particular date and hour and in a particular format.

```
<http://data.linkedtv.eu/version/3b2424f1-0943-4a4f-a6c3-49ada9bf4b22>
  a po:Version ;
  po:time
    [ a event:Interval ;
      event:start "2013-06-19 19:45:00"^^xsd:dateTime .
            event:end "2013-09-06 14:35:24.069" ^^xsd:dateTime .
    ];
  linkedtv:hasMediaResource <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd> .

<http://data.linkedtv.eu/version/d3d9d936-9223-40b6-b852-ecafaf3d243f>
  a po:Version ;
  po:aspect_ratio "urn:ard:tva:metadata:cs:ARDFormatCS:2008:90.3" ;
  po:time
    [ a event:Interval ;
      event:start "2013-06-19 21:45:00"^^xsd:dateTime
    ];
  linkedtv:hasMediaResource <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd> .
```

2.2.2 Multimedia Analysis Metadata

“RBB Aktuell” programs are completely processed by the WP1 multimedia analysis tool chain, yielding numerous metadata results serialized in the Exmaralda file. In the following, we show how each layer composing the Exmaralda file are converted in RDF using the LinkedTV core ontology.

– First, we create one instance of the class `ma:MediaResource` that represents the particular media item and links it with its physical location.

```
<http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd>
  a ma:MediaResource ;
  ma:locator <http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/rbb/video/249/>
      .
```

– Instances of the class `ma:MediaFragment` represent the different spatio-temporal fragments that belong to a particular media resource. These media fragments could be related also to other media fragments in a containment relationship. Keywords are also stored, when the media fragment correspond to a shot.

```
<http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t=1290,1293>
  a ma:MediaFragment ;
  ma:hasKeyword
    [ a linkedtv:keyword ;
      rdf:label "Herzberg"
    ];
  ma:isFragmentOf <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd> ;
  ma:isFragmentOf <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t=543,1432> .
```

– Instances of the class `oa:Annotation` describe the analysis result obtained from the different automatic processing tools. In this example, we can see an annotation that corresponds to a shot detected by CERTH in the media. The body of the annotation is then an instance of a shot, and the target is the media fragment this shot is related to. Provenance information is also included in this class through the use of the properties `opmv:wasGeneratedAt` and `opmv:wasGeneratedBy`.

```
<http://data.linkedtv.eu/annotation/b3349cd5-f728-46fa-885b-93cdc01271da>
      a       oa:Annotation , prov:Entity ;
      oa:hasTarget <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t=22.04,32.4>
           ;
      oa:hasBody <http://data.linkedtv.eu/shot/d524373c-2158-43dc-be6d-518e72aacd3f> ;
      prov:wasAttributedTo <http://data.linkedtv.eu/organization/CERTH> ;
      prov:wasDerivedFrom <ftp://ftp.condat.de/Processing/SV/12_02_SV/FhG/EXMARaLDA/12_11_07> .
```

– The instance of the class `linkedtv:Shot` that is being referred in the previous annotation is explicitly typed.

```
<http://data.linkedtv.eu/shot/d524373c-2158-43dc-be6d-518e72aacd3f>
  a linkedtv:Shot ;
    rdfs:label "Sh6" .
```

– Instances of the class `oa:Annotation` can correspond to a LSCOM concept detected by CERTH in the media with a level of confidence represented by the `linkedtv:confidence` property.

```
<http://data.linkedtv.eu/annotation/30a8b4a1-3bc0-4930-bb2c-02edd26a725c>
      a       oa:Annotation , prov:Entity ;
      linkedtv:hasConfidence "0.419"^^xsd:float ;
      oa:hasTarget <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t
          =523.44,526.52> ;
      oa:hasBody lscom:Daytime_Outdoor ;
      prov:wasAttributedTo <http://data.linkedtv.eu/organization/CERTH> ;
      prov:wasDerivedFrom <ftp://ftp.condat.de/Processing/SV/12_02_SV/FhG/EXMARaLDA/12_11_07> .
```

– The instance `lscom:Daytime_Outdoor` that is being referred in the previous annotation is also an instance of the `linkedtv:Concept` class.

```
lscom:Daytime_Outdoor
      a       linkedtv:Concept ;
      rdfs:label "Daytime_Outdoor" ;
      owl:sameAs dbpedia-owl:Daytime_Outdoor .
```

– Instances of the class `oa:Annotation` can relate a particular media fragment with a name entity recognition result performed by EURECOM.

```
<http://data.linkedtv.eu/annotation/f09e1c35-57ec-4aa4-ac65-6eba2176f045>
      a       oa:Annotation , prov:Entity ;
      oa:hasTarget <http://data.linkedtv.eu/text/b70cc571-e986-45d5-96dc-2e13b2b5c897#
          offset_9413_9419_S-Bahn> ,
                  <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t
                      =962.759,965.36> ;
      oa:hasBody <http://data.linkedtv.eu/entity/e81bb960-9e4e-4f6e-b084-e4bfbfca27ce> ;
      prov:startedAtTime "2013-09-05T12:12:13.529Z"^^xsd:dateTime ;
      prov:wasAttributedTo <http://data.linkedtv.eu/organization/EURECOM> ;
      prov:wasDerivedFrom <http://data.linkedtv.eu/text/b70cc571-e986-45d5-96dc-2e13b2b5c897> .
```

– The instance of the class `linkedtv:Entity` that is being referred in the previous annotation has been typed as a `nerd:Location` and disambiguated with a DBpedia resource.

```
<http://data.linkedtv.eu/entity/e81bb960-9e4e-4f6e-b084-e4bfbfca27ce>
      a       nerd:Location , linkedtv:Entity ;
      rdfs:label "S-Bahn" ;
      linkedtv:hasConfidence "0.267278"^^xsd:float ;
      linkedtv:hasRelevance "0.16225"^^xsd:float ;
      dc:identifier "2205128" ;
      dc:source "textrazor" ;
      dc:type "DBpedia:Place,ArchitecturalStructure,Infrastructure,RouteOfTransportation,
          PublicTransitSystem;
      Freebase:/location/location,/metropolitan_transit/transit_system" ;
      owl:sameAs <http://de.dbpedia.org/resource/S-Bahn_Berlin> .
```

The data also contains other LinkedTV concepts such as `foaf:Person` detected by a Face Recognition algorithm provided by EURECOM, `linkedtv:Scene` detected by a scene detection algorithm provided by CERTH and `linked:ASR` detected by an ASR algorithm provided by Fraunhofer..

2.3 Annotating Sound and Vision Content

Sound and Vision has gained access to video of the Dutch TV program Tussen Kunst & Kitsch (Antiques Roadshow) which is a production of the public broadcaster AVRO[1]. To start with, the scenario has chosen a single episode of the show from 8 December 2010[2].

The Sound and Vision updated scenario is described in the Deliverables D6.2 [10]. The general aim of the scenarios is to describe how the information need of the Antiques Roadshow viewers can be satisfied from both their couch and on-the-go, supporting both passive and more active needs. Linking to external information and content, such as Europeana, museum collections but also auction information has been incorporated in these scenarios.

The legacy metadata for this program comes in the form of a spreadsheet. The automatic multimedia analysis results have been serialized in a Exmaralda file but also validated with ground truth results in another spreadsheet. In the following, we show how both type of metadata is converted in RDF using the LinkedTV core ontology.

2.3.1 Legacy Metadata

For this scenario, there is not much information offered by the providers, apart from the name of the television content and the channel that broadcasts it, so some extra data has been added manually to illustrate the example. These are the instances involved:

– One instance of the class `po:Episode`, that stores the title, the synopsis, the related subjects, and other basic attributes for the current material.

```
<http://data.linkedtv.eu/episode/TUSSEN_KUNST_AVR000080E2_115000_2850600>
 a po:Episode ;
 dc:title "Najaar" ;
 po:id "AVR000080E2_115000_2850600" ;
 po:microsites <http://cultuurgids.avro.nl/front/indextkk.html> ;
 po:short_synopsis "De nieuwe opnamedata en locaties van Tussen Kunst & Kitsch zijn weer bekend. Of
      je spulletjes nu waardevol zijn of niet, je mag drie voorwerpen meenemen naar de" ;
 po:subject "Tussen Kunst & Kitsch" , "Nelleke van der Krogt" , "Programma" ;
 po:version <http://data.linkedtv.eu/version/1_AVR000080E2_115000_2850600> .
```

[1] http://www.avro.nl
[2] http://cultuurgids.avro.nl/front/detailtkk.html?item=8237850

– One instance of the class `po:Brand`, that stores information about the brand this episode belongs to.

```
<http://data.linkedtv.eu/brand/AVRO>
  a po:Brand ;
  dc:title "Algemene Vereniging Radio Omroep" ;
  po:episode <http://data.linkedtv.eu/episode/TUSSEN_KUNST_AVR000080E2_115000_2850600> ;
  po:microsites <http://avro.nl/> .
```

– One instance of the class `po:Broadcast`, that establishes a relationship between a particular version of a program and the `po:Service` instance where this version is broadcasted on.

```
<http://data.linkedtv.eu/broadcast/1_7ffdb885-fcf4-44cd-80a7-7c137c8d457a>
  a po:Broadcast ;
  po:broadcast_of <http://data.linkedtv.eu/version/1_AVR000080E2_115000_2850600> ;
  po:broadcast_on <ftp://ftp.condat.de/NISV/> .
```

– One instance of the class `po:Service`, that represents the television channel where a particular program is broadcasted.

```
<ftp://ftp.condat.de/NISV/>
  a po:Service .
```

– One instance of the class `po:Version`, that represents the appearance of a program at a particular date and hour and in a particular format.

```
<http://data.linkedtv.eu/version/1_AVR000080E2_115000_2850600>
  a po:Version ;
  po:aspect_ratio "urn:ard:tva:metadata:cs:ARDFormatCS:2008:90.3" ;
  po:time [ a event:Interval ;
  event:end "2010-12-08T20:35:23"^^xsd:dateTime ;
  event:start "2010-12-08T21:20:48"^^xsd:dateTime ];
  linkedtv:hasMediaResource <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2> .
```

2.3.2 Multimedia Analysis Metadata

This video program has been completely processed by the WP1 multimedia analysis tool chain, yielding numerous metadata results serialized in the Exmaralda file. In the following, we show how each layer composing the Exmaralda file are converted in RDF using the LinkedTV ontology.

– First, we create an instance of the class `ma:MediaResource` that represents the particular media item and links it with its physical location in the LinkedTV platform.

```
<http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2>
  a ma:MediaResource ;
  ma:locator <ftp://ftp.condat.de/NISV/TUSSEN_KUNST_AVR000080E2_115000_2850600_MPEG_PAL_169_.mpg> .
```

– Instances of the class `ma:MediaFragment` represent the different spatio-temporal fragments that belong to a particular media resource. These media fragments could be related to other media fragments in a containment relationship (e.g. a scene contains shots). Keywords are also stored when the media fragment correspond to a shot.

```
<http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2034.12,2051.34>
  a ma:MediaFragment ;
  ma:hasKeyword
    [ a  linkedtv:keyword ;
      rdf:label "Expert"
    ]:
    [ a  linkedtv:keyword ;
      rdf:label "Object"
    ];
    ma:isFragmentOf <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2> ;
    ma:isFragmentOf <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=1195,2312>.
```

– The media fragments boundaries are themselves described explicitly using the Ninsuna ontology.

```
<http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2034,2051>
  a ma:MediaFragment, nsa:TemporalFragment ;
  nsa:temporalStart 2034^^xsd:int;
  nsa:temporalEnd 2051^^xsd:int.
```

– Instances of the class `oa:Annotation` describe the analysis result obtained from the different automatic processing tools. In this example, we can see an annotation that corresponds to a shot detected by CERTH in the media. The body of the annotation is an instance of a `linkedtv:Shot`, and the target is the media fragment this shot is related to. Provenance information is also included in this class through the use of the properties `opmv:wasGeneratedAt` and `opmv:wasGeneratedBy`.

```
<http://data.linkedtv.eu/annotation/Anno_199_CERTH_Shot>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-06-29T18:19:34.798Z"^^xsd:dateTime ;
  opmv:wasGeneratedBy
    [ a opmv:Process ;
    opmv:wasPerformedBy <http://data.linkedtv.eu/organization/CERTH>
    ];
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2034.12,2051.34> ;
  oa:hasBody <http://data.linkedtv.eu/shot/sht53> .
```

– The instance of the class `linkedtv:Shot` that is being referred in the previous annotation is explicitly typed.

```
<http://data.linkedtv.eu/shot/sht53>
  a linkedtv:Shot .
```

– Instances of the class `oa:Annotation` can correspond to a LSCOM concept detected by CERTH in the media, to which a level of confidence has been provided that will be represented using the `linkedtv:confidence` property.

```
<http://data.linkedtv.eu/annotation/Anno_199_CERTH_Concept>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-06-29T18:19:35.153Z"^^xsd:dateTime ;
  opmv:wasGeneratedBy
    [ a opmv:Process ;
      opmv:wasPerformedBy <http://data.linkedtv.eu/organization/CERTH>
    ];
  linkedtv:confidence "0.67"^^xsd:float ;
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2034.12,2051.34> ;
  oa:hasBody <lscom:Commentator_Studio_Expert> .
```

– The instance `lscom:Commentator_Studio_Expert` that is being referred in the previous annotation is also an instance of the `linkedtv:Concept` class.

```
<lscom:Commentator_Studio_Expert>
  a linkedtv:Concept .
```

– Instances of the class `oa:Annotation` can relate a particular media fragment with a face recognition result performed by EURECOM.

```
<http://data.linkedtv.eu/annotation/Anno_199_EURECOM_FaceRecognition>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-06-29T18:19:35.153Z"^^xsd:dateTime ; opmv:wasGeneratedBy
  [ a opmv:Process ;
    opmv:wasPerformedBy <http://data.linkedtv.eu/organization/EURECOM>
  ];
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2045&xywh=144,112,300,250>
      ;
  oa:hasBody <http://data.linkedtv.eu/person/person3735> .
```

– The instance of the class `foaf:Person` that is being referred in the previous annotation is also an instance of a `linkedtv:Person`.

```
<http://data.linkedtv.eu/person/person3735>
    a foaf:Person ;
  foaf:gender "m" ;
  foaf:nick "Jaap Polak" ;
  foaf:page <http://cultuurgids.avro.nl/front/detailtkk.html?item=8185498> .
<http://cultuurgids.avro.nl/front/detailtkk.html?item=8185498>
    a foaf:Document .
```

– Instances of the class `oa:Annotation` may relate a particular media fragment with a name entity recognition result performed by EURECOM.

```
<http://data.linkedtv.eu/annotation/Anno_199_EURECOM_NERD>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-03-29T18:21:36.163Z"^^xsd:dateTime ;
  opmv:wasGeneratedBy
    [ a opmv:Process ;
      opmv:wasPerformedBy <http://data.linkedtv.eu/organization/EURECOM>
    ];
  linkedtv:confidence "0.90"^^xsd:float ;
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2034,2051> ;
  oa:hasBody <http://data.linkedtv.eu/entities/YI89GFAZ> .
```

– The instance of the class `linkedtv:Entity` that is being referred in the previous annotation has been typed as a nerd:Place, and disambiguated with a DBpedia resource.

```
<http://data.linkedtv.eu/entitity/YI89GFAZ>
  a nerd:Place ;
  owl:sameAs <http://dbpedia.org/resource/India> .
```

– The same entity recognition service can also also produce annotations of persons:

```
<http://data.linkedtv.eu/annotation/Anno_200_EURECOM_NERD>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-03-29T18:21:36.163Z"^^xsd:dateTime ;
  opmv:wasGeneratedBy
    [ a opmv:Process ;
      opmv:wasPerformedBy <http://data.linkedtv.eu/organization/EURECOM>
    ];
  linkedtv:confidence "0.85"^^xsd:float ;
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=2034,2051> ;
  oa:hasBody <http://data.linkedtv.eu/entity/person3735>.
```

where `http://data.linkedtv.eu/entity/person3735` refers to Jaap Polak.

– Instances of the class `oa:Annotation` can relate a particular media fragment with a scene recognized by CERTH.

```
<http://data.linkedtv.eu/annotation/Anno_199_CERTH_Scene>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-01-22T18:21:40.153Z"^^xsd:dateTime ;
  opmv:wasGeneratedBy
    [ a opmv:Process ;
      opmv:wasPerformedBy <http://data.linkedtv.eu/organization/CERTH>
    ];
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=1194,2312> ;
  oa:hasBody <http://data.linkedtv.eu/scene/scn199> .
```

– The instance of the class `linkedtv:Scene` that is being referred in the previous annotation can be explicitly typed.

```
<http://data.linkedtv.eu/scene/scn199>
  a linkedtv:Scene .
```

– The instance of the class `ma:MediaFragment` indicates the spatio-temporal aspects of the scene detected but also that this scene is a sub-fragment of the complete media resource as expressed with the property `ma:isFragmentOf`:

```
<http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=1195,2312>
  a ma:MediaFragment;
  ma:isFragmentOf <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2> .
```

– Instances of the class `oa:Annotation` can relate a particular media fragment with the transcription generated by Fraunhofer.

```
<http://data.linkedtv.eu/annotation/Anno_199_FhG_ASR>
  a oa:Annotation , opmv:Artifact ;
  opmv:wasGeneratedAt "2012-06-29T18:19:35.153Z"^^xsd:dateTime ;
  opmv:wasGeneratedBy
    [ a opmv:Process ;
      opmv:wasPerformedBy <http://data.linkedtv.eu/organization/FhG>
    ];
  linkedtv:confidence "0.234"^^xsd:float ;
  oa:hasTarget <http://data.linkedtv.eu/media/TUSSEN_KUNST_AVR000080E2#t=1195,2312> ;
  oa:hasBody <http://data.linkedtv.eu/asr/TUSSEN_KUNST_AVR000080E2_asr_01> .
```

– The instance of the class `linkedtv:ASR` that is being referred in the previous annotation enables to store the string containing the transcription.

```
<http://data.linkedtv.eu/asr/TUSSEN_KUNST_AVR000080E2_asr_01> a linkedtv:ASR ;
  rdfs:label "Wij zijn zo gewend dat dingen gedrukt zijn Dat iedereen zegt: O, Indiase prenten! Maar
       het zijn geen prenten. Het zijn Indiase schilderingen. U bent in India geweest?Ja. Maar ik
      heb deze daar niet gekocht. Deze kocht ik op de veiling in Amsterdam. 15 jaar geleden. Nou, ik
       weet niet wat u betaald heeft, dat wil ik ook niet weten... Niet veel.Maar dat heeft u goed
      gedaan, denk ik. Want deze schildering... Een heel mooi vorstelijk portret. Het komt uit Noord
      -India. Hier heeft u een Indiaas miniatuur. Dat is ook uit het noorden. Ze zit daar prachtig
      op een mooie stoel. U ziet al dat het veel flamboyanter is dan de andere. Die doeken en dingen
       gaan al veel meer opzij. Dat vind je ook in het gebied van Jodhpur, Udaipur. Aan je goud zie
      je dat het een tamelijk late miniatuur is. Ze leggen er ook kleine pareltjes op. Die geven
      relief. En dat zit je al in de 19e eeuw. Als je goed kijkt heft het hier een kleine
      beschadiging. Miniaturen horen eigenlijk puntgaaf te zijn. Je moet denken aan 650 euro. Wel
      een hele mooie vondst. Welke periode is dit ongeveer? Ongeveer 1740. Hij is iets beschadigd.
      Dat heeft met de waarde te maken. Deze miniatuur: 1250 euro." .
```

3 Content annotation for content filtering

The seed video program content comes with metadata, some of it being automatically generated through several analysis services within the LinkedTV pipeline. Among this metadata, entities describing *what a media fragment is about* may be retrieved. The content analysis tools provide this description into a variety of LOD vocabularies and knowledge bases in order to disambiguate as much as possible the *things* talked about in the program with background knowledge.

This heterogeneity of annotations, however, triggers some challenges for applying an effective personalization layer. In order to cope with this problem, an homogeneous, user-centric reference knowledge base has been designed. The so-called LinkedTV User Model Ontology[3] has therefore been engineered within LinkedTV for the purpose of representing user-relevant information within the networked media domain. The next step is then to align the annotations produced by the automatic multimedia analysis processes and the LUMO core vocabulary. We describe in this section those mappings between the LUMO concepts and the entities used to semantically describe the seed media resources/fragments and their enrichment[4], which consist the recommendation candidates for the personalisation layer. Figure 3 illustrates the workflow that transforms and communicates content annotation to the personalised content filtering services.

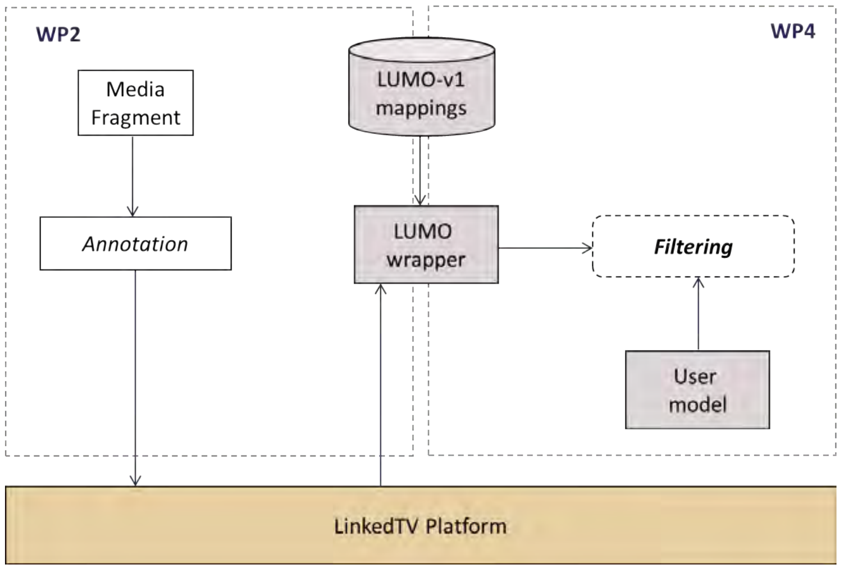


Figure 3: From content annotations to LUMO-based personalisation services

3.1 LUMO Mappings

The current version of LUMO (LUMO-v1) includes a mappings ontology that maps LUMO concepts to concepts of several existing open vocabularies. The LUMO mappings ontology is published at `http://data.linkedtv.eu/ontologies/lumo_mappings`.

The incentive of choosing the vocabularies that are included in these mappings were twofold:

– Create correspondences between LUMO and the vocabularies used by LinkedTV's analysis tools to semantically describe the content.

– Render LUMO inter-linkable with the most prominent vocabularies currently used by Semantic Web communities both in terms of semantic description of networked media content and of semantic user information/preference representation.

LUMO mappings thus serve a) for interpretation of content annotation to the LUMO vocabulary and b) as the means to facilitate re-use of LUMO by the Semantic Web.

[3]LUMO `http://data.linkedtv.eu/ontologies/lumo`. For more information about its engineering principles, see D4.4 [7].
[4]Entity extraction for enriching content is still on early stages of implementation within LinkedTV and it is foreseen that the enriched content will be itself annotated with entities in the next stages of development

LUMO mappings have been automatically generated via the LogMap tool [12] for scalable ontology matching and were further evaluated and revised manually. They are detached from LUMO in a separate ontology (module) in order to enable user modeling and inferencing (recommendation) to take place in the minimum representative vocabulary (i.e. only within the LUMO concept space), with regard to scalability and user privacy safeguarding issues (see D4.4 [7]).

Mappings are currently available to the main vocabularies that influenced the engineering of LUMO:

– DBpedia ontology: `http://dbpedia.org/ontology/`

– schema.org: `http://schema.org/docs/schemaorg.owl`

– NERD ontology: `http://nerd.eurecom.fr/ontology/`

– IPTC news codes (as owl-ified by the WebTLab[5]): `http://webtlab.it.uc3m.es/results/NEWS/subjectcodes.owl`

– GUMO ontology: `http://www.ubisworld.org/ubisworld/documents/gumo/2.0/gumo.owl`

Some of these ontologies provide mappings to each other (e.g. DBPedia and schema.org) and to other vocabularies (e.g. NERD). These interconnections are also incorporated in the LUMO mappings.

In effect, the existence of mappings between LUMO and two of the most prominent vocabularies currently used in the entity disambiguation and annotation of media content, namely NERD and the DBPedia ontology, already provides a firm basis upon which content annotation can be efficiently homogenized and received by the content recommendation services. Some graphical illustrations of mappings existent within LUMO Mappings can be seen in Figures 4 and 5.

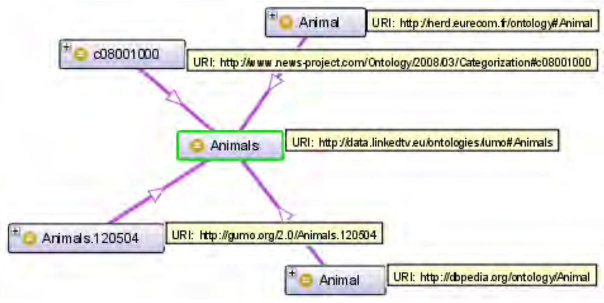

Animal
URI: http://nerd.eurecom.fr/ontology#Animal
c08001000
URI: http://www.news-project.com/Ontology/2008/03/Categorization#c08001000
Animals
URI: http://data.linkedtv.eu/ontologies/lumo#Animals
Animals.120504
URI: http://gumo.org/2.0/Animals.120504
Animal
URI: http://dbpedia.org/ontology/Animal

Figure 4: Mappings to lumo:Animals

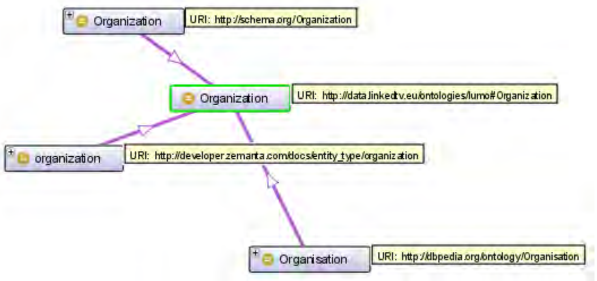

Organization
URI: http://schema.org/Organization
Organization
URI: http://data.linkedtv.eu/ontologies/lumo#Organization
organization
URI: http://developer.zemanta.com/docs/entity_type/organization
Organisation
URI: http://dbpedia.org/ontology/Organisation

Figure 5: Mappings to lumo:Organization

It should be noted that in Figure 4, *'c08001000'* is the serialization of the owl-ified IPTC news codes and corresponds to the IPTC subject code *'08001000'* with the human readable definition[6]:

```
Concept Id:  QCode = subj:08001000
URI = http://cv.iptc.org/newscodes/subjectcode/08001000
Name in en-GB is:  animal
Definition in en-GB is:  Animals of all types
```

Mappings to other open vocabularies are under consideration and engineering. In the following versions of the LUMO mappings, mappings to the following vocabularies and concepts will be considered:

– A minimal and finite set of DBPedia resources that correspond to non-named entities (therefore, in essence concepts); statistical measures will be employed to determine this finite set from the plurality of entities available as DBPedia resources.

– YAGO: (`http://www.mpi-inf.mpg.de/yago-naga/yago/`).

– Concepts identified by the visual analysis tools, a subset of TRECVID's[7] SIN task [16].

– LSCOM: `http://www.lscom.org/ontology/index.html`.

[5] `http://webtlab.it.uc3m.es/`
[6] Source: `http://cv.iptc.org/newscodes/subjectcode/`
[7] `http://trecvid.nist.gov/`

– An art/artifact related vocabulary such as AAT:
`http://www.getty.edu/research/tools/vocabularies/aat/`

Obviously, the inclusion of more vocabularies is opted as means of providing as complete coverage as possible and optimize retrieval of meaningful information about and from the content and will be influenced by the information offered from LinkedTV's content annotation tools. In addition, a future extension of LUMO mappings is planned where fuzzy mappings will be embedded either directly into LUMO, where each LUMO concept will be annotated with its respective mappings in all supported vocabularies, or a newer version and format of the separate mappings ontology will be engineered. The purpose is to take into account the semantic similarity of mapped concepts into the transformation of content annotation entities to LUMO concepts.

3.2 LUMO wrapper

For transforming content annotation into the LUMO concept space and communicating it to the filtering services of WP4, we have developed a LUMO wrapper. The LUMO wrapper has a dual role in the LinkedTV workflow: a) it is used internally in WP4 to convey implicit user information based on his/her transactions with the content to LUMO and b) to convey content annotation to LUMO for use by the content filtering services [7].

The conversion process involves reasoning with the *types* of the entities in the content annotation through the f-PocketKRHyper logical reasoner [8]. The entities form a "query" set which is used to search in the LUMO mappings knowledge base for the LUMO concepts that the query set is subsumed by. The output comprises of a set of instances of *LUMO concepts* that correspond to the entities in the content annotation. It has to be noted that if an entity's type is not found to have a correspondence to the LUMO concept space then this entity is discarded altogether in the final converted set passed on for filtering.

In the following subsections, we provide examples of entity conversion and final output to be received by the recommendation services.

3.2.1 Conversion examples

A close up to the DBPedia entities extracted from the subtitles of a *Tussen Kunst & Kitsch* show item reads:

```
"idEntity":791558, "label":"Geffen", "extractorType":"RecordLabel",
"nerdType":"http://nerd.eurecom.fr/ontology#Thing",
"uri":"http://dbpedia.org/resource/Geffen_Records",
"confidence":0.00216818, "relevance":0.5, "extractor":"dbspotlight",
"startChar":16424, "endChar":16430, "startNPT":1150.88, "endNPT":1153.48
```

This segment represents one of many entities recognized in this media resource. The system seeks for mappings between both the `extractorType` (which is usually the most granular type) and the `nerdType` of the entity to LUMO concepts within the LUMO mappings via the subsumption service of f-PocketKRHyper. If the mapping retrieved for `nerdType` subsumes the mapping retrieved for `extractorType` then only the most granular concept (i.e. only the mapping for `extractorType`) is maintained.

In any case, no retrieval task is initiated if the type at hand is null or the TOP level concept of NERD, i.e. `"extractorType": null` or `"nerdType":"http://nerd.eurecom.fr/ontology#Thing"`, which essentially denotes an entity for which it is unresolved whether it can instantiate anything within the LUMO concept space and therefore renders it non-usable for the purposes of personalisation within a designated concept space.

The instances of LUMO concepts identified are serialized in the form of "$\{instance : LUMOconcept = confidencedegree\}$". It has to be noted that the instance comprises merely of the label of the original entity, agnostic of its URI, since the filtering tool is in its term agnostic of any other vocabulary outside of its reasoning knowledge base, i.e. LUMO. In a similar mode, the $LUMOconcept$ type consists only of the concept's label and does not bare a URI since this is implied from the restriction to the LUMO concept space and thus to the default LUMO base URI. The rationale for omitting information such as full entity URIs lies in the need for minimizing the problem space and data models used in the inferencing (recommendation) layer, so as to render filtering as lightweight as possible and performable on the user client, again with regard to privacy safeguarding issues.

The transformation of the previous example would ultimately read:

```
"id":791558.0,"startChar":16424.0,"endChar":16430.0,"startNPT":1150.88,"endNPT":1153.48,
"concepts":"{Geffen:record_label=0.00216818}"
```

where `Geffen`is the original annotation entity's label and `record_label`is a concept in LUMO.

The input for the f-PocketKRHyper content filtering service would be the set of instances retrieved per media item, in a DL-based formalism, such that $\langle instance : LUMOconcept \rangle \geqslant confidencedegree$, or in the above example $\langle geffen : record\_label \rangle \geqslant 0.00216818$ [8]. In the particular notation employed by the personalization services that the filtering tool uses (i.e. KRSS[8]) the example above would read:

```
(INSTANCE geffen record_label >= 0.00216818)
```

Similarly, one of the entities recognized in a content item might map to more than one LUMO concepts according to the mappings produced via LogMap. When fuzzy mappings are implemented, it will become apparent that an entity may instantiate several concepts but with a different degree of similarity.

One such example from an RBB content item would read:

```
"idEntity":789878, "label":"Olympia", "extractorType":"Film",
"nerdType":"http://nerd.eurecom.fr/ontology#Product",
"uri":"http://dbpedia.org/resource/Olympia_1938_film",
"confidence":0.0206801, "relevance":0.5, "extractor":"dbspotlight",
"startChar":10794, "endChar":10801, "startNPT":898.04, "endNPT":901.2
```

Its interpretation to LUMO would read:

```
"id":789879.0, "startChar":10794.0, "endChar":10801.0, "startNPT":898.04, "endNPT":901.2,
"concepts":"Olympia:movie=0.0206801, Olympia:film=0.0206801, Olympia:product=0.0206801"
```

where `Olympia`is the original annotation entity's label and `movie, film, product`are concepts in LUMO.

So the list of entities passed onto the f-PocketKRHyper-based filterer as annotation for the specific media item would include all three concepts that "Olympia" instantiates:

```
(INSTANCE olympia movie >= 0.0206801)
(INSTANCE olympia film >= 0.0206801)
(INSTANCE olympia product >= 0.0206801)
```

Ultimately, each candidate content item's semantic profile would be passed onto the filterer. This profile would include the semantic description of the instance, a semantic axiom combining entities with basic provenance information (i.e. the content ID) and also automatically produced information that can help enrich the base annotation with meaningful LUMO-based non-taxonomical relations. The latter serves for automatically expanding the content filtering capabilities, like for example for the automatic detection of the topic/s that the content item at hand is about via the LUMO-based $hasTopic$ object property[9]. The semantic LUMO-based content annotation profile would therefore be of the form:

$\exists offers.(A \sqcap B \sqcap ...) \sqsubseteq objectID$
$\langle a : A \rangle \geqslant d_1$
$\langle b : B \rangle \geqslant d_2$
$\langle content, a : offers \rangle$
$\langle content, b : offers \rangle$
$\langle a, content : hasTopic \rangle$
$\langle b, content : hasTopic \rangle$

Where A and B are LUMO concepts, a and b are respective instances of these concepts and d_1, $d2$ are the degrees by which they respectively instantiate the concepts, $content$ is a generic instance, used to retrieve the logical relation between different instances of concepts and the content item as a whole via the default $offers$ object property, and $hasTopic$ is the LUMO object property that relates concepts to their topics, as modeled per the LUMO ontology.

[8]http://dl.kr.org/krss-spec.ps
[9]See D4.4, section 2.1.2 for an example on how topic detection can be achieved [7].

The KRSS example of the final semantic content profile of the RBB fragment/close-up presented before would read:

```
(IMPLIES (SOME offers (AND movie film product)) {objectID})
(INSTANCE olympia movie >= 0.0206801)
(INSTANCE olympia film >= 0.0206801)
(INSTANCE olympia product >= 0.0206801)
(RELATED content olympia offers)
(RELATED olympia content hastopic)
```

In future implementations, the normalization of confidence scores for entities stemming from different extractors which vary in range and computation of confidence values will be explored in order to achieve a comprehensive convergence of information.

3.2.2 RESTful Services

RESTful services are implemented in order to achieve communication between LUMO wrapper and the platform but also retrieval services along different parts of the workflow.

GET transformed annotation
Performs the transformation to LUMO for a given content item and receives the content item's LUMO-based annotation.
Description: GET /api/transform_annotation?cid={objectid}
HTTP method: GET
Content-Type: application/json
Response format:

```
{"[{"id":789879.0,"startChar":10794.0,"endChar":10801.0,"startNPT":898.04,"endNPT":901.2,
   "concepts":"{Olympia:movie=0.0206801, Olympia:film=0.0206801, Olympia:product=0.0206801}"}]",
"[{"id":789985.0,"startChar":16970.0,"endChar":16978.0,"startNPT":1557.48,"endNPT":1560.8,
   "concepts":"{Salzburg:settlement=0.0210687}"}]"
}
```

GET a KRSS content profile
Receives a given content profile from the server in the KRSS format. This is the input for f-pocketKRHyper's content matching service.
Description: GET /api/KRSS_content_annotation?cid={objectid}
HTTP method: GET
Content-Type: application/json
Response format:

```
{"KRSS_content_annotation":
"(IMPLIES (SOME offers (AND movie film product)) {objectID})
(INSTANCE olympia movie >= 0.0206801),
(INSTANCE olympia film >= 0.0206801),
(INSTANCE olympia product >= 0.0206801),
(INSTANCE salzburg settlement >= 0.0210687),
(RELATED content olympia offers),
(RELATED content salzburg offers),
(RELATED olympia content hastopic),
(RELATED salzburg content hastopic)"
}
```

PUT a KRSS annotation profile
Uploads a LUMO-based content annotation profile in the KRSS formalisation on the local server where the content filtering service resides. Currently supports uploading a file but will be converted into a URI-based call.
Description: PUT /api/upload_content_annotation
HTTP method: PUT
Content-Type: text/plain
Response format:

Requisite: a .txt file containing the LUMO-based annotation of the content item in KRSS
Result: Uploads/updates the selected profile on the server

4 LinkedTV Metadata Converter

The tool *TV2RDF*[10] is a REST API Web service that serializes the information available about a certain television content into RDF according to the LinkedTV core ontology. It has been developed by EURECOM and tested during the second year of the Project over a corpora of videos from the broadcasters S&V and RBB.

In a nutshell, this service takes as input the UUID of a `MediaResource` in the LinkedTV Platform and its corresponding metadata files, and produces a RDF representation of the whole information by using the concepts considered in the LinkedTV core ontology. The resulting serialization includes mainly (1) legacy information from the content providers, (2) subtitles and extracted name entities via the NERD framework [18, 19, 20, 21], and (3) data obtained after the execution of certain analysis techniques like shot segmentation, concept detection, or face recognition as serialized in an Exmaralda file in WP1.

The related content discovery and enrichment processes over the resulting annotation graph is out of the scope of this REST service. Those functionalities are provided in a separate service described in Section 5.

4.1 Implementation

This service has been developed while considering media resources as the main citizen in the workflow. Consequently, at the data storage level, we maintain the list of media resources that have been created inside a particular tv2rdf instance. For each resource, there are two types of items included: metadata files and serialization files. Both kind of documents are directly uploaded/accessed via the REST API layer.

The logic for converting metadata files into LinkedTV compliant RDF documents is encapsulated inside three different core components. First, all the results from the analysis algorithms generated by WP1 are processed inside the module *Exmaralda_Serialization*. At the moment the only format supported for this serialization phase is the EXMaRALDA format. This software component is in constant development mainly because new analysis algorithms are being implemented under the LinkedTV project, so the EXMaRALDA format and consequently the LinkedTV Ontology are evolving, sometimes significantly. Second, subtitles from which entities are extracted are converted into RDF within the component *Subtitle_Entity_Serialization*. The format accepted for this kind of metadata files is SRT[11]. This component is also in charge of invoking NERD for extracting the named entities over the transcripts. Finally, the legacy metadata information is processed in the module *Legacy_Serialization*. The information managed in this step always refer to the entire media resource, but not at the finer granularity of fragments.

Those three core components produce separated serialization files. However, the data populates a unique graph that is stored into the LinkedTV RDF triplestore[9]. The Figure 6 details this workflow.

The list of supported formats is planned to be extended in future versions of the TV2RDF service in order to broaden the scope of LinkedTV and to enable third parties to make their data available under the same ontology model.

Concerning the storage of the data, the system chosen is Mongodb[12], a cross-platform document oriented database solution classified as a “NoSQL” alternative. The reason behind this decision is that MongoDB performs really well with JSON-like files with dynamic schemas and makes the integration of data in certain types of applications (like the case of this Web Service) easier and faster. This software has been released under a combination of the GNU Affero General Public License and the Apache License so it is a free and open source software. Nowadays, MongoDB has been adopted as backend software by a number of major web sites and services, including Craigslist, eBay, Foursquare, SourceForge, and The New York Times, among others.

Regarding the implementation of the Web service, we have relied on two different initiatives that make easier to write scalable server applications. The first one is Grizzly[13], which supports Java New I/O API (NIO) and manages threads in order to allow a server to scale to thousands of users. The Grizzly NIO framework has been designed to help developers to build robust servers using NIO as well as offering extended framework components like Web Framework (HTTP/S), WebSocket, or Comet. Also if we want to have RESTful Web services that seamlessly support exposing data in a variety of

[10] `http://linkedtv.eurecom.fr/tv2rdf/api/`
[11] `http://en.wikipedia.org/wiki/SubRip`
[12] `http://www.mongodb.org/`
[13] `https://grizzly.java.net/`

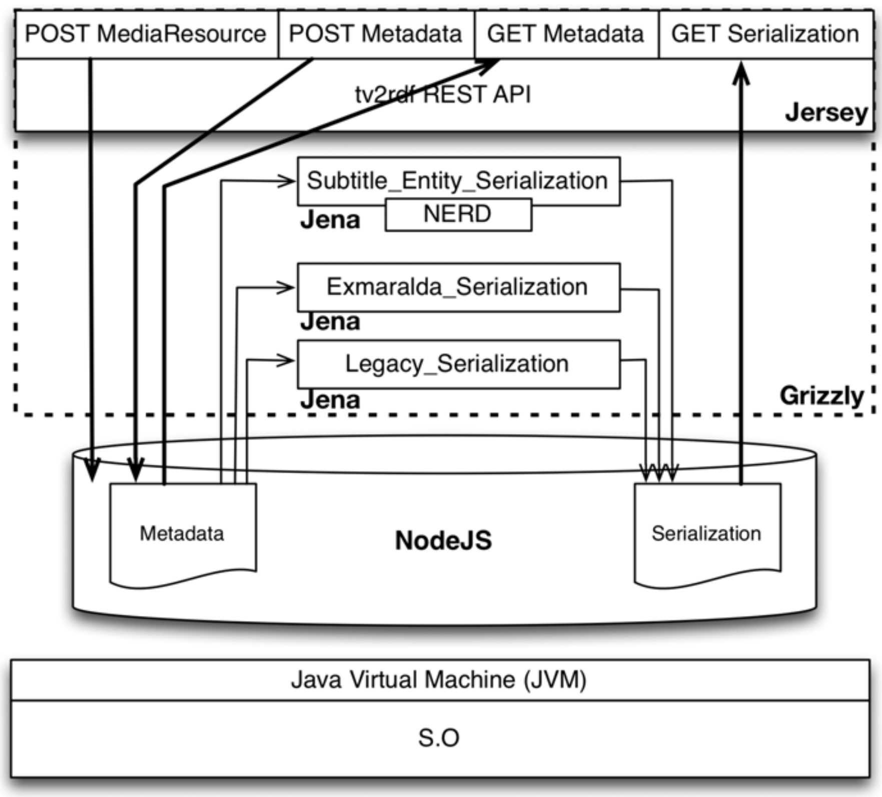


Figure 6: TV2RDF implementation details

representation media types and abstract the low-level details of the client-server communication, we need a toolkit like Jersey[14]. Jersey framework is an open source, production quality framework for developing REST services in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation. It exposes also numerous extensions so developers can effectively extend Jersey to best suit their needs.

Finally, there are two other libraries that are used in TV2RDF. The first one is nerd4java[15], a java library which provides a programmable interface to NERD for easily launch named entity extractions with different parameters. The second one is Apache Jena[16], a free and open source Java framework for building Semantic Web and Linked Data applications that makes easy to create and read Resource Description Framework (RDF) graphs, work with RDFS and OWL models to add extra semantics to RDF data, and serialise triples using well-known formats such as RDF/XML or Turtle. As a last remark, all those components have been integrated under the particularities of a Java environment and run over an instance of a Java Virtual Machine (JVM).

4.2 TV2RDF integration in the LinkedTV platform

The TV2RDF service has been integrated in the general LinkedTV workflow. In a nutshell, the REST API service interacts with three main actors taking part in the LinkedTV scenario (see Figure 7 for further details):

– LinkedTV Platform, for obtaining the video UUID, the locator of the media resource, and the namespace to be used for generating the instances URLs.

– Dataset containing metadata files from the providers. Those files include the subtitles and legacy metadata that need to be serialized into RDF.

– WP1 for obtaining the results from the analysis processes. The corresponding Exmaralda file generated by this word package has to be serialized as well so it is also used by TV2RDF when generating the RDF graph.

[14]https://jersey.java.net/
[15]https://github.com/giusepperizzo/nerd4java
[16]http://jena.apache.org/

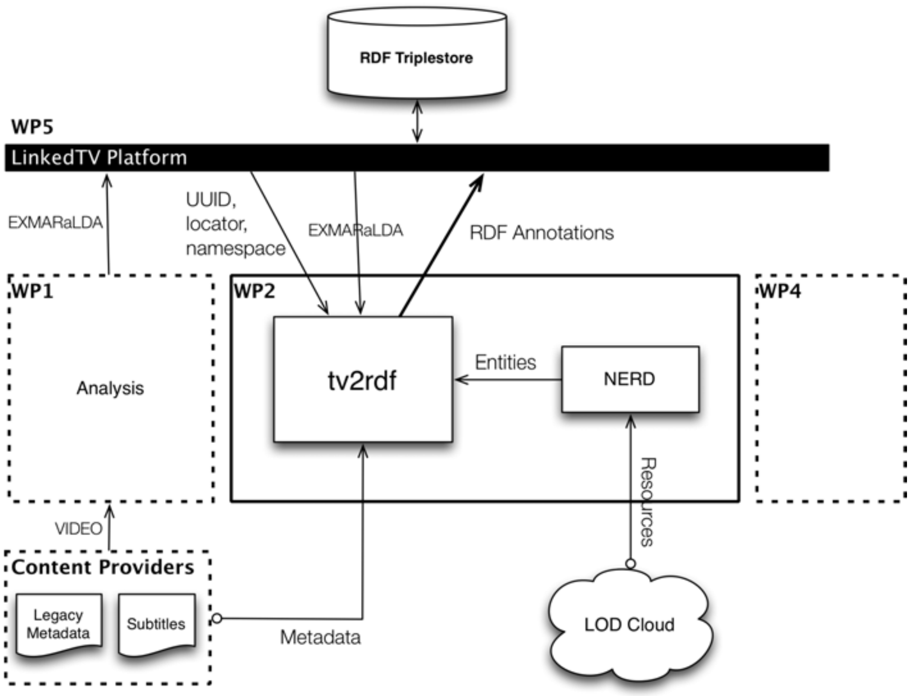


Figure 7: Integration of tv2rdf inside the LinkedTV workflow

Once all those LinkedTV components have provided TV2RDF with the necessary information, the internal serialization engine reads those particular formats, generates the corresponding media fragments, annotates them at different level of granularities, and links them with resources from the Web of Data cloud via NERD's entities. The final set of triples is serialized in the Turtle format and pushed to the LinkedTV platform.

After showing how the different components in LinkedTV request/get information to/from TV2RDF, we clarify the right timing in which those interactions have to be done in order to perform a valid RDF serialization of a `MediaResource`. As illustrated in Figure 8, everything starts with the ingestion of a television programme by the LinkedTV platform. When all the surrounding attributes (UUID, locator, namespace) have been generated and WP1 has finished its processing, the platform can send a POST request to TV2RDF in order to create the corresponding media resource. The next step consists of uploading the different metadata files associated to this particular television programme. On the TV2RDF side, this will automatically trigger the execution of the serialization processes that could imply a request to NERD in the case of processing subtitles. At the end of the execution, the results are ready to be retrieved by the LinkedTV platform, which can actually perform the three GET request to the REST service in order to download the corresponding Turtle files. In a last step, the Turtle files are loaded into the LinkedTV triplestore within the default graph `http://data.linkedtv.eu/graph/linkedtv`.

4.3 REST API calls supported in TV2RDF

4.3.1 Creating a Media Resource

This request creates a new media resource in TV2RDF to be serialized to RDF. It is necessary to provide the *UUID* (Universal Unique Identifier) of the media resource for uniquely identifying the television content to be processed. Optionally, it is also possible to specify the *locator* of the video (the logical address at which the resource can be accessed, a URL in Linked TV) and the *namespace* for generating the URI's of the instances generated during the serialization.

```
curl -X POST http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource --header "
    Content-Type:text/xml" -v
```

For specifying properties such as the locator and the namespace, one can use the query parameters as follows:

```
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource?locator=URL\
    _locator&namespace=http://data.linkedtv.eu/" --header "Content-Type:text/xml" -v
```

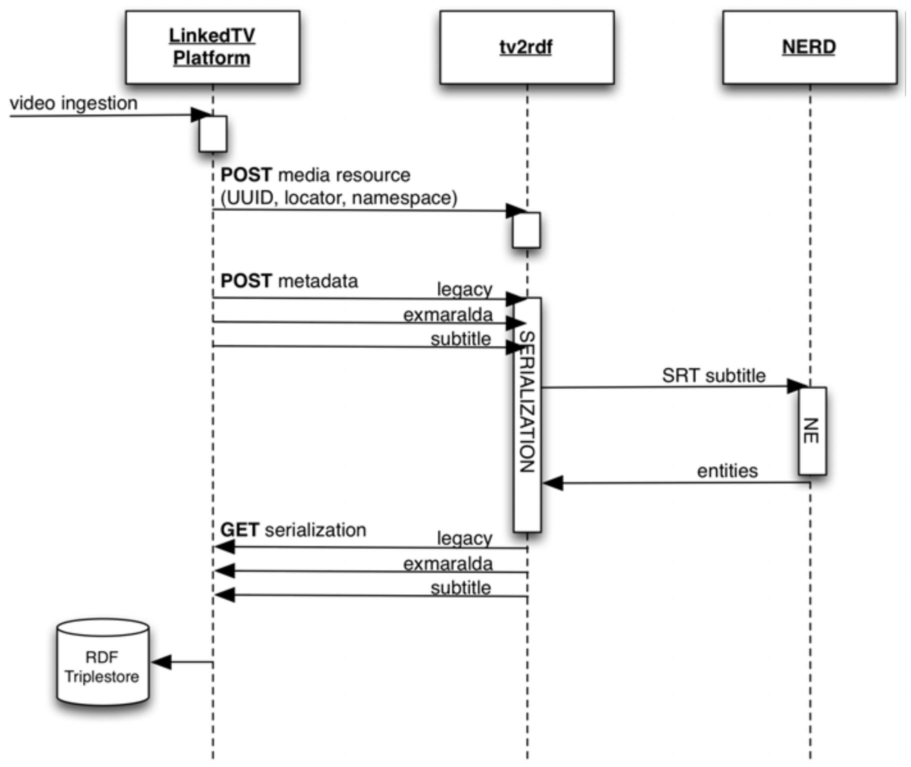


Figure 8: Sequence diagram of the serialization of a Media Resource by TV2RDF

4.3.2 Uploading Metadata Files for a Media Resource

This request allows to specify the metadata files describing a particular media resource: the legacy file, the subtitles file and the analysis results file. Immediately after the storage of the files in the server, the corresponding serialization process is automatically launched, so the RDF results will be available as soon as possible by performing one of the REST requests shown in Section 4.3.3. If these POST requests are executed multiple times, the files uploaded in the past are substituted by the ones specified in the current operation and the serialization processes are re-started again.

Legacy Metadata. This request upload the file that contains the information provided by the broadcasters for a particular media resource, and launch the process of converting it into RDF. Up to now, the only format supported by TV2RDF is TVAnytime[17].

```
curl -X POST --data-binary @LEGACY_file.tva  http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/
     UUID\_media\_resource/metadata?metadataType=legacy --header "Content-Type:text/xml" -v
```

Subtitles. This request upload the subtitle file for a particular Media Resource, and launch the process of extracting Named Entities from the time text and serialize them into RDF. Up to now, the format supported by TV2RDF is SRT[18].

```
curl -X POST --data-binary @SUBTITLES_file.srt  http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/
     UUID\_media\_resource/metadata?metadataType=subtitle --header "Content-Type:text/xml" -v
```

Analysis Results. This request upload the file with the results from the execution of various analysis techniques over a particular media resource. Up to now, the format supported by TV2RDF is Exmaralda[19].

```
curl -X POST --data-binary @EXMARALDA\_file.exb  http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource
     /UUID\_media\_resource/metadata?metadataType=exmaralda --header "Content-Type:text/xml"
```

[17]http://tech.ebu.ch/tvanytime
[18]http://en.wikipedia.org/wiki/SubRip
[19]http://www.exmaralda.org/

4.3.3 Getting Metadata files

It is possible to retrieve the original metadata files that were uploaded to TV2RDF for a certain media resource at any time, by performing a GET request instead of the corresponding POST calls. It is therefore possible to get, for instance, the version of metadata files used as input to launch the serialization.

Legacy Metadata. This request allows to download the legacy file corresponding to a particular media resource, if available.

```
curl -X GET  http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/metadata?
    metadataType=legacy --header "Content-Type:text/xml" -v
```

Subtitles. This request allows to download the subtitle file corresponding to a particular media resource, if available.

```
curl -X GET  http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/metadata?
    metadataType=subtitle --header "Content-Type:text/xml" -v
```

Analysis Results. This request allows to download the Exmaralda file corresponding to a particular media resource, if available.

```
curl -X GET  http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/metadata?
    metadataType=analysis --header "Content-Type:text/xml" -v
```

4.3.4 Getting Serialization results

This request allows to retrieve the Turtle files generated after the serialization of the different metadata files into RDF. Those results will be available after the corresponding core component has finished its processing, by just performing the corresponding GET request to the TV2RDF service. Hence, a client can repeatedly make those GET calls to the REST service until the resource is available (the 404 responses turn into a 200 OK and the file is downloaded from the server). It is also possible to see if a serialization result is available by using some of the REST calls explained in Section 4.3.5.

Legacy Serialization. This request allows to retrieve the serialization file that includes the legacy information from the providers. The main ontology behind this RDF excerpt is the BBC Programmes Ontology.

```
curl -X GET http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/serialization?
    metadataType=legacy --header "Content-Type:text/xml" -v
```

Subtitles and Entities Serialization. This request allows to retrieve the serialization file that includes the subtitles of the video and the named entities extracted using the NERD framework. The main ontologies behind are the NERD Ontology, The Open Annotation ontology, and the NIF ontology.

```
curl -X GET http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/serialization?
    metadataType=subtitle --header "Content-Type:text/xml" -v
```

Analysis Results Serialization. This request allows to retrieve the serialization file that includes the analysis results generated by WP1 over a particular video. The main ontologies behind are the W3C Ontology for Media Resources, The Open Annotation ontology, and the LSCOM ontology.

```
curl -X GET http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/serialization?
    metadataType=exmaralda --header "Content-Type:text/xml" -v
```

Complete Serialization. This request allows to retrieve the serialization file that includes the complete information available about a certain Media Resource.

```
curl -X GET http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/UUID\_media\_resource/serialization
    --header "Content-Type:text/xml" -v
```

4.3.5 Other Requests

Get MediaResource's description. With this request, it is possible to obtain a JSON serialization of the data available in the TV2RDF about a particular Media Resource: id, metadata files that have been uploaded, serialization files available, and base URL used for generating the different data instances of the graph.

```
curl -X GET http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/19a73f0a-d023-49f8-9203-cbd721053c55
      --header "Content-Type:text/xml" -v
```

Modify MediaResource's parameters. If some of the parameters (locator or namespace) need to be modified, one can used the same procedure as for creating a Media Resource. The parameters will be overwritten on the server side, and the serialization processes will be automatically re-launched (if the corresponding metadata files are available).

```
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/19a73f0a-d023-49f8-9203-
     cbd721053c55?locator=http://stream6.noterik.com/progressive/stream6/domain/linkedtv/user/rbb/
     video/59/rawvideo/2/raw.mp4&namespace=http://data.linkedtv.eu/" --header "Content-Type:text/xml"
      -v
```

Get a List of Media Resources. This operation returns the list of Media Resource's that have been created inside TV2RDF REST service.

```
curl -X GET http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/list --header "Content-Type:text/xml
     " -v
```

4.4 TV2RDF examples

In this section, we provide the REST API calls needed for serializing the 6 videos selected for demonstration during the second year review:
five videos from Sound and Vision (TUSSEN_KUNST_-AVR00006KUR_115000_3064640, TUSSEN_KUNST_-AVR00007O6H_115000_2848200, TUSSEN_KUNST_-AVR000080E2_115000_2850600, TUSSEN_KUNST_-AVR0000814C_115000_2808960, TUSSEN_KUNST_-AVR0000814D_115000_2814560) and one RBB video (rbbaktuell_20130619). Those programs have already been ingested in the LinkedTV platform via `http://api.linkedtv.eu/mediaresource/`.

The LinkedTV platform has already created an abstract UUID for each program which is synchronized with the physical locator of the corresponding video. We assume that WP1 has also finished to run different analysis algorithms, resulting in a set of Exmaralda files that will be used as input for TV2RDF. As depicted in the sequence diagram (Figure 8), the procedure can start by first creating the corresponding media resources into TV2RDF:

```
//MediaResource creation
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7
     dccd76f0001?locator=http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/avro/
     video/100/&namespace=http://data.linkedtv.eu" -v --header "Content-Type:text/xml"
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7
     dccd76f0002?locator=http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/avro/
     video/101/&namespace=http://data.linkedtv.eu" -v --header "Content-Type:text/xml"
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7
     dccd76f0003?locator=http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/avro/
     video/102/&namespace=http://data.linkedtv.eu" -v --header "Content-Type:text/xml"
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7
     dccd76f0004?locator=http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/avro/
     video/103/&namespace=http://data.linkedtv.eu" -v --header "Content-Type:text/xml"
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7
     dccd76f0005?locator=http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/avro/
     video/104/&namespace=http://data.linkedtv.eu" -v --header "Content-Type:text/xml"
curl -X POST "http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/b82fb032-d95e-11e2-951c-
     f8bdfd0abfbd?locator=http://stream17.noterik.com/progressive/stream17/domain/linkedtv/user/rbb/
     video/249/&namespace=http://data.linkedtv.eu" -v --header "Content-Type:text/xml"
```

The next step consist of uploading the corresponding metadata files, three per review video (subtitles, exmaralda files, and legacy files). Immediately after every file has been uploaded to TV2RDF, the serialization starts in the background making the results available through the REST API as soon as the corresponding software component has finished the processing.

```
//Posting Subtitles
curl -X POST --data-binary @TUSSEN_KUNST_-AVR00006KUR_115000_3064640.srt http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dccd76f0001/metadata?metadataType=subtitle -v
     --header "Content-Type:text/xml"
```

```
curl -X POST --data-binary @TUSSEN_KUNST_-AVR000070GH_115000_2848200.srt http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd099380002/metadata?metadataType=subtitle -v
     --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR000080E2_115000_2850600.srt http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd151100003/metadata?metadataType=subtitle -v
     --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR0000814C_115000_2808960.srt http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd247360004/metadata?metadataType=subtitle -v
     --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR0000814D_115000_2814560.srt http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd2d0650005/metadata?metadataType=subtitle -v
     --header "Content-Type:text/xml"
curl -X POST --data-binary @BERLIN-2013-06-19-22-00-37-06192145.srt http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/metadata?metadataType=subtitle -v
     --header "Content-Type:text/xml"
```

```
//Posting Exmaralda
curl -X POST --data-binary @TUSSEN_KUNST_-AVR00006KUR_115000_3064640.exb http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dccd76f0001/metadata?metadataType=exmaralda -v
      --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR000070GH_115000_2848200.exb http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd099380002/metadata?metadataType=exmaralda -v
      --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR000080E2_115000_2850600.exb http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd151100003/metadata?metadataType=exmaralda -v
      --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR0000814C_115000_2808960.exb http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd247360004/metadata?metadataType=exmaralda -v
      --header "Content-Type:text/xml"
curl -X POST --data-binary @TUSSEN_KUNST_-AVR0000814D_115000_2814560.exb http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd2d0650005/metadata?metadataType=exmaralda -v
      --header "Content-Type:text/xml"
curl -X POST --data-binary @rbbaktuell_20130619_sdg_m_16_9_512x288.exb http://linkedtv.eurecom.fr/
     tv2rdf/api/mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/metadata?metadataType=exmaralda -v
      --header "Content-Type:text/xml"
```

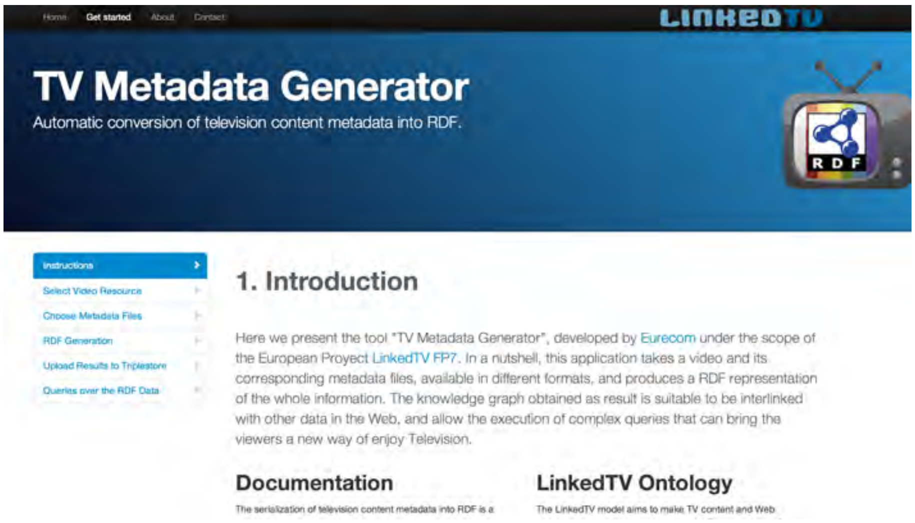
```
//Posting Legacy
curl -X POST --data-binary @rbbaktuell_20130619.html http://linkedtv.eurecom.fr/tv2rdf/api/
     mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/metadata?metadataType=legacy -v --header "
     Content-Type:text/xml"
```

Once, the metadata files have been uploaded and serialized, the corresponding Turtle files obtained as results can be accessed. In the following script, we show how to make the GET call and store the corresponding triples in a local file for further processing or re-loading into a RDF triplestore.

```
//getting subtitle
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dccd76f0001/
     serialization?metadataType=subtitle > TUSSEN_KUNST_-
     AVR00006KUR_115000_3064640_entities_subtitles.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd099380002/
     serialization?metadataType=subtitle > TUSSEN_KUNST_-
     AVR000070GH_115000_2848200_entities_subtitles.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd151100003/
     serialization?metadataType=subtitle > TUSSEN_KUNST_-
     AVR000080E2_115000_2850600_entities_subtitles.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd247360004/
     serialization?metadataType=subtitle > TUSSEN_KUNST_-
     AVR0000814C_115000_2808960_entities_subtitles.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd2d0650005/
     serialization?metadataType=subtitle > TUSSEN_KUNST_-
     AVR0000814D_115000_2814560_entities_subtitles.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/
     serialization?metadataType=subtitle > rbbaktuell_20130619_entities_subtitles.ttl;

//getting exmaralda
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dccd76f0001/
     serialization?metadataType=exmaralda > TUSSEN_KUNST_-AVR00006KUR_115000_3064640_exmaralda.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd099380002/
     serialization?metadataType=exmaralda > TUSSEN_KUNST_-AVR000070GH_115000_2848200_exmaralda.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd151100003/
     serialization?metadataType=exmaralda > TUSSEN_KUNST_-AVR000080E2_115000_2850600_exmaralda.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd247360004/
     serialization?metadataType=exmaralda > TUSSEN_KUNST_-AVR0000814C_115000_2808960_exmaralda.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd2d0650005/
     serialization?metadataType=exmaralda > TUSSEN_KUNST_-AVR0000814D_115000_2814560_exmaralda.ttl;
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/
     serialization?metadataType=exmaralda > rbbaktuell_20130619_exmaralda.ttl;

//getting legacy
curl http://linkedtv.eurecom.fr/tv2rdf/api/mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/
     serialization?metadataType=legacy > rbbaktuell_20130619_legacy.ttl;
```



TV Metadata Generator

Automatic conversion of television content metadata into RDF.

- Instructions
- Select Video Resource
- Choose Metadata Files
- RDF Generation
- Upload Results to Triplestore
- Queries over the RDF Data

1. Introduction

Here we present the tool "TV Metadata Generator", developed by Eurecom under the scope of the European Proyect LinkedTV FP7. In a nutshell, this application takes a video and its corresponding metadata files, available in different formats, and produces a RDF representation of the whole information. The knowledge graph obtained as result is suitable to be interlinked with other data in the Web, and allow the execution of complex queries that can bring the viewers a new way of enjoy Television.

Documentation

The serialization of television content metadata into RDF is a

LinkedTV Ontology

The LinkedTV model aims to make TV content and Web

Figure 9: Front-end user interface for serializing media resources in TV2RDF (under development)

4.5 Future Work

The RESTful TV2RDF application is currently under constant debugging and improvement cycles that aim to increase the reliability of the service and include new features that could be interesting in a multimedia and television scenario. We enumerate below some of the main features we are planning to implement during the upcoming months:

– Entities filtering. Once the subtitles are annotated using NERD, there are still many entities that are simply wrong or irrelevant for the actual context of the video being analyzed. Those entities should not be included in the resulting RDF graph, in order to gain in simplicity and consistency in the data.

– More formats supported. The television ecosystem is not limited to formats such as TVAnytime or SRT, but considers a bigger set of metadata schemas that currently play a role in the audiovisual market and should be supported in TV2RDF.

– User interface. TV2RDF is primarily a REST web service but we have also developed a more human-friendly interface (see Figure 9).

– LinkedTV core ontology evolution. The LinkedTV core ontology is evolving and it depends on a number of third party ontologies such as the Open Annotation ontology which itself is evolving. The TV2RDF service needs to be always compliant with those latest specifications.

5 LinkedTV Enrichment Service

In this section, we will present how television content ingested and serialized by the LinkedTV project is enriched with media resources extracted from external platforms, either coming from white-listed web sites or social networks. The software component responsible for this task is the so-called *TVEnricher*[20] web service, which is developed by EURECOM. This service is currently under an intensive process of development and testing based on some first evaluations and the feedback of other project partners. In a nutshell, TVEnricher detects suitable anchors within media resources for the enrichment based on volume and frequency of named entities detected in media fragments. The named entities become query terms that are used by the Media Collector[21] (see Section 4 of the Deliverable D2.5 [6]). The resulting enrichment results are then serialized in RDF using again the LinkedTV core ontology.

The technologies involved in the implementation of this web service are almost the same that were already mentioned for the TV2RDF service (see Section 4.1), except for the absence of a database system working at the server side. Indeed, no information is currently stored in TVEnricher since the methods for accessing the external platforms and serializing the results into RDF are executed on the fly every time an enrichment request is sent to the service. Response times are relatively high and vary from 30 to 180 seconds depending on the duration of the video and the number of relevant entities spotted in it. The bottleneck of the approach is clearly in the internal calls addressed to the Media Collector service, which takes some time while collecting items across the different considered sources. The complete workflow of TVEnricher can be summarized as follows:

– The LinkedTV Platform launches the enrichment process by providing a valid UUID to TVEnricher. A pre-requisite is that the annotation process of the corresponding media resource is completed, and the resulting RDF data has been already pushed in the triple-store.

– Given a media resource identifier, TVEnricher accesses the triple-store for retrieving all the entities identified by a linked data URI (e.g. the ones spotted by NERD) for this television program. From the set of entities retrieved from the graph, we rank and filter out a number of relevant entities. In the current version, the ranking operation is based on a naive approach consisting in giving more importance to the entities who appear the most frequently.

– For each entity, we proceed like follows:

  ◦ A search operation using the Media Collector is triggered by using as input the label of the entity. Up to now, we use a so-called *combined* strategy for retrieving as many media items as possible, including items from white-listed web sites and fresh media resources from social networks. Example: `http://linkedtv.eurecom.fr/api/mediacollector/search/combined/S-Bahn`

  ◦ We serialize the results into RDF (see Section 5.2):

    ∗ The media resources extracted are serialized according to the W3C Ontology for Media Resources.

    ∗ We align the results with a media fragment of a certain granularity (`linkedtv:Chapter`, `linkedtv:Scene`, `linkedtv:Shot`) inside the seed video. For making the alignment we consider the temporal proximity of the media fragment with the entity that triggered the enrichment. Consequently, TVEnricher needs to query the LinkedTV SPARQL endpoint again for getting these temporal boundaries. In the following, we use shots as anchors but we are aiming at using scenes or larger segments in future developments.

    ∗ The enrichment is serialized as an annotation between the seed video fragment (anchor) and the set of media resources retrieved.

  ◦ Finally, we integrate all the partial results into a single Turtle file that is pushed to the platform.

The Figure 10 illustrates this workflow while Figure 10 summarizes all the main services involved in the WP2 processing chain.

The sequence diagram in Figure 11 aims to clarify how the different components play their role over the time. The LinkedTV Platform initiates the conversation by providing the UUID of a media resource to be enriched. Immediately after, TVEnricher answers back for requesting an excerpt for the RDF

[20] `http://linkedtv.eurecom.fr/tvenricher/api/`
[21] `http://linkedtv.eurecom.fr/api/mediacollector/`

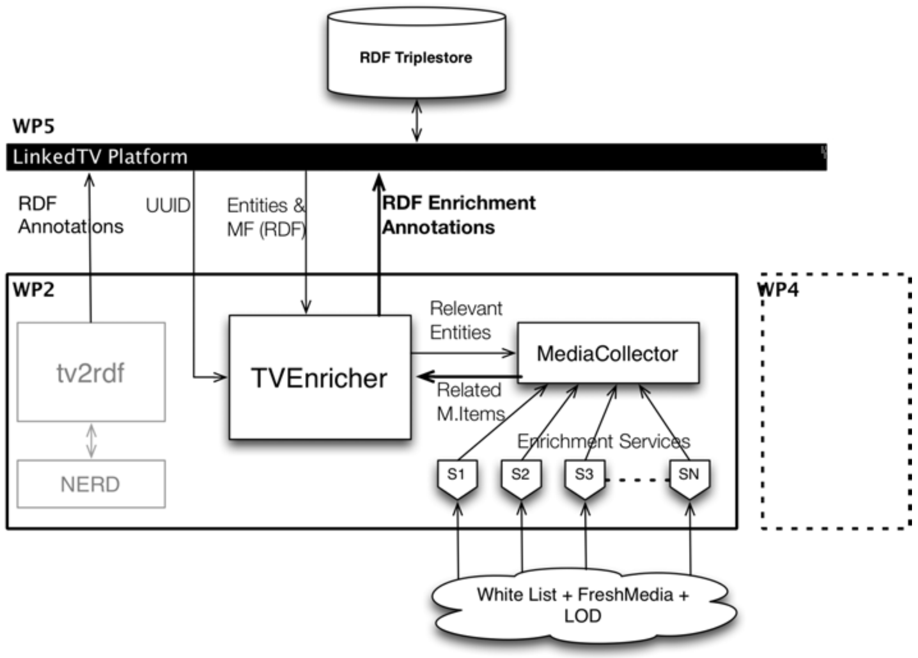


Figure 10: Diagram showing the role of the TVEnricher service within the LinkedTV Platform

data containing mainly entities to be used as anchors and media fragments to align items to. As it is illustrated in the figure, the main interaction occurs between the TVEnricher and the Media Collector[6] for collecting media items related to an entity. The latency can be more or less long depending on the duration of the seed video and the number of relevant entities detected.

5.1 REST API calls supported in TVEnricher

In the current version of TVEnricher, the REST API is kept as simple as possible: the only supported call is the one for triggering an enrichment over a media resource given its UUID inside the LinkedTV Platform. The call include two query parameters: (1) the granularity of the media fragments with which enrichments will be associated to, and (2) the namespace used in the instances of the dataset where the RDF representation of media resource to enrich is stored.

```
curl "http://linkedtv.eurecom.fr/tvenricher/api/mediaresource/UUID\_media\_resource/enrichment?
    granularity=Granularity&namespace=http://data.linkedtv.eu"  --header "Content-Type:text/xml" -v;
```

As we will further explain in Section 5.5, in the future, we plan to support additional parameters for specifying the nature of the sources where the related items will be extracted from: `fresh` sources (typically from social networks), `reliable` sources (typically from the Editor Tool [5] or white-listed web sites. The number of platforms supported in the Media Collector becoming larger and larger, and the time needed for the enrichment operation increasing, it could be necessary to switch to a two phases enrichment approach, consisting of posting the UUID of the media resource first, and adding requests for obtaining results later on.

5.2 Serialization and Metamodel.

This section describes the ontological approach for serializing media resource enrichments as it has been implemented in TVEnricher. Having in mind to keep the solution as simple as possible, those are the main serialization decisions that have been made:

– The list of media resources retrieved after each Media Collector search operation are modeled as instances of the *ma:MediaResource* class from the Ontology for Media Resources. As the ontology by itself does not cover entirely the set of attributes considered in the Media Collector JSON schema, we have used additional properties coming from the Dublin Core Metadata Element Set[22] and the LinkedTV Ontology. The Table 2 summarizes the mappings between both serialization formats:

[22] http://dublincore.org/documents/dces/

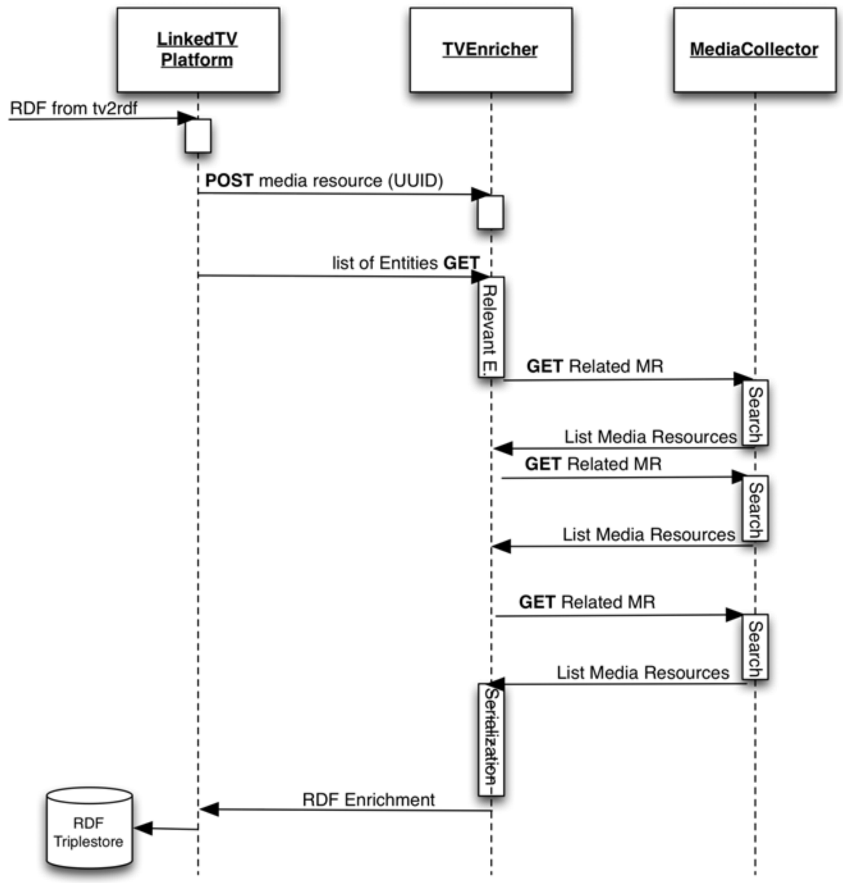


Figure 11: Sequence diagram of the enrichment serialization of a Media Resource by TVEnricher

– Those media resources considered as enrichment results are attached to the particular temporal fragment they seem to be more relevant to. In order to calculate this, we perform a SPARQL query for obtaining the instance of the *ma:MediaFragment* class (annotated with a desired level of granularity: *linkedtv:Chapter*, *linkedtv:Shot*, or *linkedtv:Scene*) for which the entity's temporal boundaries are closer to the interval defined by *nsa:temporalStart* and *nsa:temporalEnd*. The query looks like as follows:

```
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
    ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
SELECT ?mediaFragment
FROM <http://data.linkedtv.eu/graph/linkedtv>
  WHERE {
    ?granularity a linkedtv: "+ stringGranularityMF  +"  .
    ?annotationGranularity oa:hasBody ?granularity .
    ?annotationGranularity  oa:hasTarget ?mediaFragment.
    ?mediaFragment  a  ma:MediaFragment .
    ?mediaFragment  ma:isFragmentOf <+mediaFragmentURL+> .
    ?mediaFragment nsa:temporalStart  ?start.
    ?mediaFragment nsa:temporalEnd ?end .
    filter(+entityStart+ >= ?start) .
    filter("+entityStart+" < ?end) .
  }
```

Listing 1: SPARQL query

– Finally, an instance of the class *oa:Annotation* is generated for explicitly linking the collected media resources with the enriched media fragment. Apart from various provenance information according to the PROV[23] ontology, those annotations include two properties that worth to be mentioned: (1) *prov:wasDerivedFrom* that indicates the instance of the class *linkedtv:Entity*, which triggered the Media Collector search operation, and *oa:motivatedBy*, which specifies the reasons why the annotation was created, no matter the agents involved (in our case its value is set to `oa:linking` by default).

Table 2: Properties inside `ma:MediaResource` instances for representing Media Collector's JSON attributes

JSON attribute	Ontology property inside LinkedTV
mediaUrl	`ma:locator`
posterUrl	`linkedtv:hasPoster`
micropostUrl	`dc:isPartOf`
plainText	`dc:description`
userProfileUrl	`dc:creator`
type	`dc:type`
timestamp	`dc:date`
socialInteractions	`linkedtv:hasSocialInteraction`
likes	`linkedtv:likes`*
shares	`linkedtv:shares`*
comments	`linkedtv:comments`*
views	`linkedtv:views`*

The Figure 12 illustrates this process with an example of an enrichment serialization operation.

5.3 Enriching RBB Content

In this section we will show the RDF result of enriching the video *rbbaktuell_20130619* of the television show *Aktuell*, broadcasted by the German provider RBB. Specifically, this is the REST API needed to launch the process for this program:

```
curl "http://linkedtv.eurecom.fr/tvenricher/api/mediaresource/b82fb032-d95e-11e2-951c-f8bdfd0abfbd/
    enrichment?granularity=Shot&namespace=http://data.linkedtv.eu"  --header "Content-Type:text/xml"
     -v;
```

Having the UUID of the media resource (b82fb032-d95e-11e2-951c-f8bdfd0abfbd), the service retrieves the list of entities spotted by NERD for that particular video. TVEnricher will list those entities in order of relevance by considering to the number of times they have appeared over the entire show. In our example, those are the labels of the top 12 ranked entities are: Berlin, Insekten, Mücken, Barack Obama, Deutschland, Matthias Platzeck, S-Bahn, Maschine, Bornstedt, Freiheit, Krongut. Below we will show how RDF representation of the enrichment generated from one of those entities looks like.

We will take as example the RDF enrichment generated for the named entity **S-Bahn**. The first step to be done is to make a search query in Media Collector in order to start the collection of media resources in the different considered platforms: `http://linkedtv.eurecom.fr/api/mediacollector/search/RBB/S-Bahn`. Once the JSON response containing all the retrieved items is available, we start the RDF serialization of all the aspects concerning that set of media resources.

First, an instance of the class *oa:annotation* is generated for that particular entity. This annotation has for *oa:hasBody* the list of media resources found by the Media Collector and different provenance information such as the date of enrichment creation, the responsible tool (TVEnricher), etc.

```
<http://data.linkedtv.eu/annotation/2c60da24-d2fc-451a-888b-48f49bf0c75d>
      a        oa:Annotation , prov:Entity ;
      oa:hasBody <http://data.linkedtv.eu/media/656b2d1c-ded8-4e1f-a888-889beabeaff1> ,
                 <http://data.linkedtv.eu/media/808c5ec3-4fd6-428d-8dbf-9d7266184328> ,
                 <http://data.linkedtv.eu/media/1d115295-87a5-40b5-a512-62fbe1f60a98> ,
                 <http://data.linkedtv.eu/media/809a7b79-c02e-4db6-ae5f-6bbafee52a7c> ,
                 <http://data.linkedtv.eu/media/b4d7cadb-d33b-495f-9aae-4526cec3ba69> ,
                 <http://data.linkedtv.eu/media/38f4aefc-b176-4d64-900f-8ada4dc1a3c9> ,
                 <http://data.linkedtv.eu/media/e54ff382-6247-47ef-9446-ac222503dfc9> ,
                 <http://data.linkedtv.eu/media/023211d5-3275-44a9-bbd4-bd4b41e02a76> ,
                 <http://data.linkedtv.eu/media/3f072470-4981-4cbd-9339-595c501990ab> ,
                 <http://data.linkedtv.eu/media/5876ff1d-9ce0-40f1-98f6-05d7c2d9efb9> ,
```

[23] http://www.w3.org/TR/prov-o/

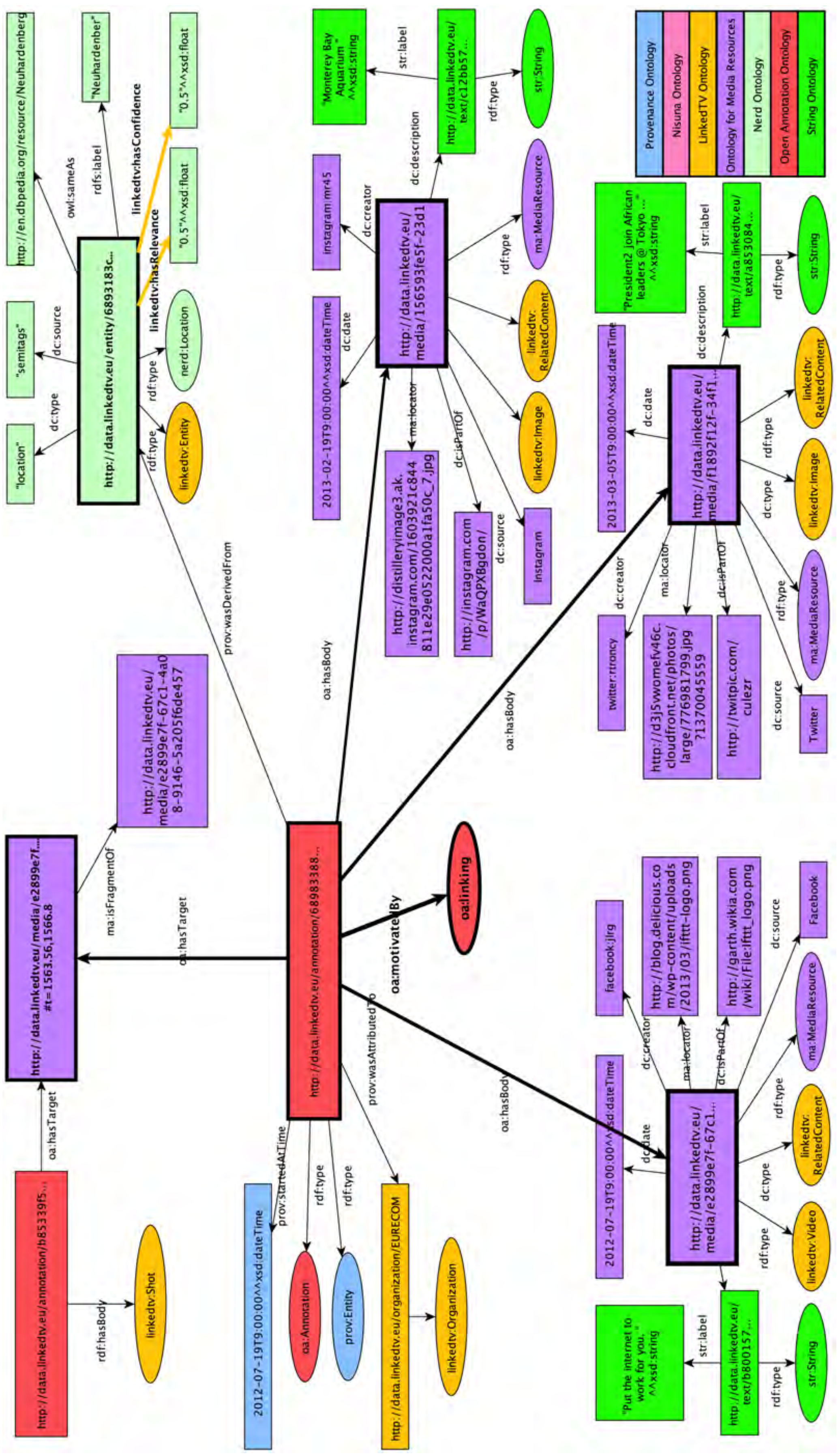

http://data.linkedtv.eu/annotation/68983388...
oa:motivatedBy
oa:linking
oa:hasTarget
http://data.linkedtv.eu/media/e2899e7f...#t=1563.56,1566.8
ma:isFragmentOf
http://data.linkedtv.eu/media/e2899e7f-67c1-4a08-9146-5a205f6de457
oa:hasTarget
http://data.linkedtv.eu/annotation/b85339f5...
rdf:hasBody
linkedtv:Shot
prov:startedAtTime
2012-07-19T9:00:00^^xsd:dateTime
rdf:type
oa:Annotation
prov:Entity
prov:wasAttributedTo
http://data.linkedtv.eu/organization/EURECOM
linkedtv:Organization
prov:wasDerivedFrom
http://data.linkedtv.eu/entity/6893183c...
dc:type
"location"
dc:source
"semitags"
owl:sameAs
http://en.dbpedia.org/resource/Neuhardenberg
rdfs:label
"Neuhardenber"
linkedtv:hasConfidence
"0.5"^^xsd:float
linkedtv:hasRelevance
"0.5"^^xsd:float
nerd:Location
linkedtv:Entity
oa:hasBody
http://data.linkedtv.eu/media/156593fe5f-23d1
dc:date
2013-02-19T9:00:00^^xsd:dateTime
dc:creator
instagram:mr45
dc:description
http://data.linkedtv.eu/text/c12bb57...
str:label
"Monterey Bay Aquarium" ^^xsd:string
str:String
ma:MediaResource
linkedtv:RelatedContent
linkedtv:Image
ma:locator
http://distilleryimage3.ak.instagram.com/1603921c844811e29e0522000a1fa50c_7.jpg
dc:isPartOf
http://instagram.com/p/WaQPXBgdon/
dc:source
Instagram
http://data.linkedtv.eu/media/f1892f12f-34f1...
2013-03-05T9:00:00^^xsd:dateTime
twitter:rtroncy
http://d3j5vwomefv46c.cloudfront.net/photos/large/776981799.jpg?1370045559
http://twitpic.com/culezr
Twitter
http://data.linkedtv.eu/text/a853084...
"President2 join African leaders @Tokyo ..." ^^xsd:string
http://data.linkedtv.eu/media/e2899e7f-67c1...
facebook:jlrg
http://blog.delicious.com/wp-content/uploads/2013/03/ifttt-logo.png
http://garth.wikia.com/wiki/File:Ifttt_logo.png
Facebook
linkedtv:Video
http://data.linkedtv.eu/text/b800157...
"Put the internet to work for you." ^^xsd:string
Provenance Ontology
Nisuna Ontology
LinkedTV Ontology
Ontology for Media Resources
Nerd Ontology
Open Annotation Ontology
String Ontology

Figure 12: Instances involved in the RDF serialization of a media resource enrichment

```
                  <http://data.linkedtv.eu/media/aa7bdf54-2853-410a-99a4-1a9805c447ef> ,
                  <http://data.linkedtv.eu/media/0e12a03b-5910-458f-b278-20dd62097fd3> ;
       oa:hasTarget <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t
           =962.36,964.84> ;
       oa:motivatedBy oa:linking ;
       prov:startedAtTime "2013-09-23T09:13:45.858Z"^^xsd:dateTime ;
       prov:wasAttributedTo <http://data.linkedtv.eu/organization/EURECOM/TVEnricher> ;
       prov:wasDerivedFrom <http://data.linkedtv.eu/entity/e81bb960-9e4e-4f6e-b084-e4bfbfca27ce> .
```

The property *prov:wasDerivedTo* points to the *linkedtv:Entity* instance which has triggered the enrichment process in order to have a direct access to other possible information such as the subtitle block where this entity has been spotted. The property *oa:motivatedBy* is set to *oa:linking* for specifying the nature of the relationship: an untyped link to a resource related to the target[24]. The target of the annotation (*hasTarget*) refers to the *ma:MediaFragment* instance to which those items should be related to. The corresponding Turtle code for that instance is:

```
 <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t=962.36,964.84>
       a       nsa:TemporalFragment , ma:MediaFragment ;
       nsa:temporalEnd "964.84"^^xsd:float ;
       nsa:temporalStart "962.36"^^xsd:float ;
       nsa:temporalUnit "npt" ;
       ma:duration "2.4800415"^^xsd:float ;
       ma:isFragmentOf <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd> .
```

All the media resources coming from the Media Collector are described in RDF as well. We use The Ontology for Media Resources for that purpose, as described in Section 5.2. An example for the entity “S-Bahn” is shown below:

```
 <http://data.linkedtv.eu/media/e54ff382-6247-47ef-9446-ac222503dfc9>
       a       ma:MediaResource , linkedtv:RelatedContent ;
       linkedtv:hasPoster <https://i1.ytimg.com/vi/29aSPW6xltM/default.jpg> ;
       linkedtv:hasSocialInteraction
               [ linkedtv:comments "22"^^xsd:int ;
                 linkedtv:likes "35"^^xsd:int ;
                 linkedtv:shares "0"^^xsd:int ;
                 linkedtv:views "41172"^^xsd:int
               ] ;
       dc:creator <https://www.youtube.com/channel/UC_3uchSRXVGSOPszleKUEBw> ;
       dc:date "2012-01-14T19:53:51Z"^^xsd:dateTime ;
       dc:description <http://data.linkedtv.eu/text/ea8283ec-5771-4149-9075-66b7db364ab3> ;
       dc:isPartOf <http://www.youtube.com/watch?v=29aSPW6xltM> ;
       dc:source <http://data.linkedtv.eu/organization/YouTube> ;
       dc:type linkedtv:Video ;
       ma:locator <https://www.youtube.com/embed/29aSPW6xltM> .
```

If we further analyze the different properties available for this media resource, we can easily see that it corresponds to a video, from the social platform Youtube, published on 2012-01-14. The video comes also within a web page accessible at the URL via the *dc:isPartOf* property, which at the same time contains a textual description serialized as a instance of a String.

```
 <http://data.linkedtv.eu/text/ea8283ec-5771-4149-9075-66b7db364ab3>
       a       prov:Entity , str:String ;
       str:label "Karlsplatz (Stachus) - Münchner S-Bahn. Dies ist ein Video über den S-Bahnhof
            Karlsplatz (Stachus). Der Karlsplatz (Stachus) ist ein Bahnhof der Stammstrecke, mitten in
             München. Zu sehen sind die S-Bahn Fahrzeuge der Linien 1, 2, 3, 4, 6, 7 und 8 und deren
            Ankünfte und Abfahrten. Da dieses Video recht lang ist hab ich hier ein paar gute Momente
            rausgesucht: -Laute Abfahrt der S4: 0:32 -Die S2 kommt aus dem Tunnel: 3:27 (
            Lichtspiegelung der Kamera bei 3:59) -Langsamer Start der S6/Lichtblitze zwischen
            Stromabnehmer und Oberleitung: 5:43 -Zwei S-Bahnen parallel gefilmt: 6:11 (S2; die 2. bei
            6:40 (S7)) -Abfahrt der S3 mit Ansage: 8:02 Der S-Bahnhof Karlsplatz (Stachus) darf nur
            mit einer gültigen Fahrkarte oder einer gültigen Bahnsteigkarte betreten werden. Für
            dieses Video wurde eine Bahnsteigkarte für 0,40 euro gelöst. Copyright:
            BergfelderVideos780 Weitere S-Bahnhöfe:"^^xsd:string .
```

All other media items from the Media Collector have been serialized in the same way. The Figure 13 depicts one media resource retrieved as an enrichment for the “S-Bahn” entity. This media resources shows one of the trains of this suburban metro-like railway system that serves city centre traffic as well as surroundings and nearby towns.

5.4 Enriching Sound and Vision Content

In this section we will show some RDF results of enriching the video *TUSSEN_KUNST_-AVR000080E2_115000_2850600* of the television show *Tussen Kunst en Kitsch* from the Dutch broadcaster Avro. Specifically, this is the REST API needed to launch the process over this program:

[24] `http://www.openannotation.org/spec/core/appendices.html#ExtendingMotivations` provides the list of possible values for this property

Figure 13: Screenshot of the media resource (video) at `https://www.youtube.com/embed/29aSPW6xltM`, relevant to the named entity “S-Bahn” - RBB scenario

```
curl "http://linkedtv.eurecom.fr/tvenricher/api/mediaresource/8a8187f2-3fc8-cb54-0140-7dd151100003/
     enrichment?granularity=Shot&namespace=http://data.linkedtv.eu"  --header "Content-Type:text/xml"
     -v;
```

Having the UUID of the media resource (8a8187f2-3fc8-cb54-0140-7dd151100003), the service retrieves the list of entities spotted by NERD for that particular video. TVEnricher will list those entities in order of relevance by considering the number of times they have appeared over the entire show. In our example, those are the labels of the top 8 ranked entities are: Amsterdam, Jan Toorop, Nederland, Parijs, Dat, Leuk and Tholen. Below we will show how RDF representation of the enrichment generated from one of those entities looks like.

We will take as example the RDF enrichment generated for the named entity **Jan Toorop**. The first step to be done is to make a search query in Media Collector in order to start the collection of media resources relevant for this search term: `http://linkedtv.eurecom.fr/api/mediacollector/search/SV/Toorop`. Once the JSON response containing all the retrieved items is available, we start the RDF serialization of all the aspects concerning that set of media resources.

First, an instance of the class *oa:annotation* is generated for that particular entity. This annotation has for *oa:hasBody* the list of media resources found by the Media Collector and different provenance information such as the date of enrichment creation, the responsible tool (TVEnricher), etc.

```
<http://data.linkedtv.eu/annotation/a2664015-3b62-47aa-b999-b3069a9054c0>
      a       oa:Annotation , prov:Entity ;
      oa:Motivation oa:linking ;
      oa:hasBody <http://data.linkedtv.eu/media/5ee11dc8-38cc-4034-807e-5eabdfb5bc84> , <http://data
          .linkedtv.eu/media/8b97a931-f7ea-4370-914a-73019eed62df> , <http://data.linkedtv.eu/media/
          f1cc03fe-be2e-41c3-bb9f-3bf9b77683c7> , <http://data.linkedtv.eu/media/263fddda-c7a7-47a0-
          bd6f-99be810874b1> , <http://data.linkedtv.eu/media/6415bd27-b025-4924-8e84-a256212bfc85>
          , <http://data.linkedtv.eu/media/4ea709f1-f11e-4af4-be85-7610b3b50633> , <http://data.
          linkedtv.eu/media/e3d59d8a-0a29-4fc5-80b0-34a31c4308af> , <http://data.linkedtv.eu/media/
          de89c3f1-e6f8-4d08-aabf-2a7a1a7f9e15> , <http://data.linkedtv.eu/media/b35fa154-e8b2-4b0b
          -8a21-aebc264072d6> , <http://data.linkedtv.eu/media/fb24909d-b765-4040-bacf-85b6345dc353>
           ;
      oa:hasTarget <http://data.linkedtv.eu/media/a8187f2-3fc8-cb54-0140-7dd151100003#t
          =785.56,790.56> ;
      prov:startedAtTime "2013-09-14T00:33:30.788Z"^^xsd:dateTime ;
      prov:wasAttributedTo
              <http://data.linkedtv.eu/organization/EURECOM> ;
      prov:wasDerivedFrom "http://data.linkedtv.eu/entity/226ba0ff-537a-485d-bbfa-f356eba105a9" .
```

As in the RBB use case, the property *prov:wasDerivedTo* points to the *linkedtv:Entity* instance which has triggered the enrichment process so we have a direct access to other possible information such as the text where that entity was retrieved from. The property *oa:motivatedBy* is set to *oa:linking* for specifying the nature of the relationship: again an untyped link to a resource related to the target.

```
<http://data.linkedtv.eu/media/a8187f2-3fc8-cb54-0140-7dd151100003#t=785.56,790.56>
      a       nsa:TemporalFragment , ma:MediaFragment ;
      nsa:temporalEnd "790.56"^^xsd:float ;
      nsa:temporalStart "785.56"^^xsd:float ;
      nsa:temporalUnit "npt" ;
      ma:duration "5.0"^^xsd:float ;
      ma:isFragmentOf <http://data.linkedtv.eu/media/8a8187f2-3fc8-cb54-0140-7dd151100003> .
```

An example of the serialization of a description of a video relevant to the entity “Jan Toorop” is shown below:

```
<http://data.linkedtv.eu/media/8b97a931-f7ea-4370-914a-73019eed62df>
```

Figure 14: Video showing multiple paintings by Jan Toorop (20 December 1858 3 March 1928), derived from the named entity “Toorop” - Sound and Vision scenario Source: `https://www.youtube.com/embed/QLLRMEF9Bt4`

```
        a       ma:MediaResource , linkedtv:RelatedContent ;
        linkedtv:hasPoster <https://i1.ytimg.com/vi/QLLRMEF9Bt4/default.jpg> ;
        linkedtv:hasSocialInteraction
                [ linkedtv:comments "5"^^xsd:int ;
                  linkedtv:likes "24"^^xsd:int ;
                  linkedtv:shares "0"^^xsd:int ;
                  linkedtv:views "1180"^^xsd:int
                ] ;
        dc:creator <https://www.youtube.com/channel/UCKmKCcwzzFTL8HgsOhPu5mg> ;
        dc:date "2012-03-13T17:10:28Z"^^xsd:dateTime ;
        dc:description <http://data.linkedtv.eu/text/e08ff4a0-4bcc-44b1-bba4-8bdd7cffdf3b> ;
        dc:isPartOf <http://www.youtube.com/watch?v=QLLRMEF9Bt4> ;
        dc:source <http://data.linkedtv.eu/socialplatform/YouTube> ;
        dc:type linkedtv:Video ;
        ma:locator <https://www.youtube.com/embed/QLLRMEF9Bt4> .
```

```
 <http://data.linkedtv.eu/text/e08ff4a0-4bcc-44b1-bba4-8bdd7cffdf3b>
        a       prov:Entity , str:String ;
        str:label "Jan Toorop. Jean Theodoor Toorop (20 December 1858 3 March 1928), better known as
            Jan Toorop, was a Javanese Dutch painter whose works straddle the space between the
            Symbolist painters and Art Nouveau. Jean Theodoor Toorop was born on 20 December 1858 in
            Purworejo, Java, Dutch East Indies. In 1872, he moved with his family to the Netherlands,
            where he studied in Delft and Amsterdam. In 1880 he became a student at the Rijksakademie
            in Amsterdam. From 1882 to 1886 he lived in Brussels, where he joined Les XX (Les Vingts),
             a group of artists centred around James Ensor. Toorop worked in various different styles
            during these years, such as Realism, Impressionism Neo-Impressionism and Post-
            Impressionism. After his marriage to an English woman, Annie Hall, in 1886, Toorop
            alternated his time between The Hague, England and Brussels, and after 1890 also the Dutch
             seaside town of Katwijk aan Zee. During this period he developed his own unique Symbolist
             style, with dynamic, unpredictable lines based on Javanese motifs, highly stylised
            willowy figures, and curvilinear designs. Thereafter he turned to Art Nouveau styles, in
            which a similar play of lines is used for decorative purposes, without any apparent
            symbolic meaning. In 1905 he converted to Catholicism and began producing religious works.
             He also created book illustrations, posters, and stained glass designs. Throughout his
            life Toorop also produced portraits, in sketch format and as paintings, which in style
            range from highly realistic to impressionistic. Toorop died on 3 March 1928 in The Hague,
            Netherlands. His daughter Charley Toorop (1891-1955) was also a painter. [from Wikipedia]
            Music"^^xsd:string .
```

The Figure 14 depicts one media resource retrieved as an enrichment for the “Toorop” entity. This media resource shows various paintings from this famous Dutch painter who worked on various different styles such as Realism, Impressionism Post-Impressionism, Symbolist or Art Nouveau.

5.5 Future Developments

TVEnricher is a recent WP2 tool with is still under a lot of developments. Due to the changing nature of the Web sources which feed the service with related content, it turns to be necessary to iteratively debugging and improving the current approach. Between the main challenges ahead, we would like to highlight the followings:

– Improve the Media Collector service, the main component behind the scene. This service is responsible for effectively retrieving related content from all the external platforms, a better indexing of those resources means an immediate increase in terms of quality for the corresponding enrichments.

– Code a more elaborated algorithm for finding relevant entities. The approach of considering an entity more relevant just because it appears more frequently in the corresponding transcript is too naive and sometimes even wrong. Implementing a more sophisticated strategies for ranking the extracted entities is a must for increasing the interlinking accuracy of TVEnricher.

– Use extra knowledge for launching the enrichment. Up to now, the only anchors that trigger the operation are the labels of the relevant entities. In the future, it could be interesting to use other available metadata such as LSCOM visual concepts, faces, or keywords, separately or in aggregated manner, for obtaining more precise results.

– Store the results of the calculated enrichments in the REST service. This requires to implement cache-based approaches on server side in order to save quota when querying particular social media platforms, to reduce the response time, and to offer the possibility of implementing a two phase strategy consisting in posting the UUID and later on retrieving the serialized results.

– Finally, all the considered platforms provide items from outside the LinkedTV ecosystem. But in some occasion it is interesting and makes a lot of sense to establish links to content (entire media resources, or more fine grained media fragments) existing in our own corpora.

6 Useful SPARQL Queries

We conclude this deliverable by showing how one can query both the annotations and the enrichments that have been serialized in RDF and stored in the LinkedTV platform. In the following, we provide 10 representative queries for accessing the data available inside the LinkedTV RDF graph. All the examples are applied over the media resource *http://data.linkedtv.eu/media/8a8187f2-3fc8-cb54-0140-7dd151100003* but are easy to generalize to any other media resource. For a better understanding of the different statements inside the SPARQL queries, please check Figure 2 and Figure 12 where those attributes and properties are graphically displayed.

6.1 Query 1: Get the list of `linkedtv:Shot` or `linkedtv:Chapter` for a media resource

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
PREFIX oa: <http://www.w3.org/ns/oa#>

SELECT ?mediaFragment
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
    ?mediaFragment ma:isFragmentOf <http://data.linkedtv.eu/media/8
       a8187f2-3fc8-cb54-0140-7dd151100003> .
  ?mediaFragment a ma:MediaFragment .
  ?annotation a oa:Annotation .
    ?annotation oa:hasBody ?shot.
    ?annotation oa:hasTarget ?mediaFragment .
    ?shot a linkedtv:Shot .
}
```

Listing 2: SPARQL query

6.2 Query 2: Get the list of keywords attached to a media fragment

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>

SELECT ?keywordtext
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
    <http://data.linkedtv.eu/media/b82fb032-d95e-11e2-951c-f8bdfd0abfbd
       #t=1463.28,1480.48>  ma:hasKeyword ?keyword .
  ?keyword a linkedtv:Keyword .
  ?keyword rdfs:label ?keywordtext
}
```

Listing 3: SPARQL query

6.3 Query 3: Get the list of LSCOM concepts attached to a media fragment

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontology/>
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
PREFIX oa: <http://www.w3.org/ns/oa#>
```

```
SELECT  ?keyURL
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
    ?annotation oa:hasTarget <http://data.linkedtv.eu/media/b82fb032-
       d95e-11e2-951c-f8bdfd0abfbd#t=1463.28,1480.48> .
    ?annotation oa:hasBody ?concept.
  ?concept a linkedtv:Concept .
  ?concept owl:sameAs ?keyURL .
}
```

Listing 4: SPARQL query

6.4 Query 4: Get the list of entities spotted inside the transcript of a media resource, showing for each entity their label and disambiguation URI

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
   ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX owl:     <http://www.w3.org/2002/07/owl#>

SELECT ?Entity, ?label, ?source, ?disURL
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
   ?MediaFragment ma:isFragmentOf <http://data.linkedtv.eu/media/
      b82fb032-d95e-11e2-951c-f8bdfd0abfbd> .
   ?MediaFragment a ma:MediaFragment .
   ?MediaFragment linkedtv:hasSubtitle ?subtitle .
   ?AnnotationEntity oa:hasTarget ?MediaFragment.
   ?AnnotationEntity oa:hasBody ?Entity.
   ?Entity a linkedtv:Entity .
   OPTIONAL { ?Entity rdfs:label ?label .  }
   OPTIONAL { ?Entity dc:source ?source . }
   ?Entity owl:sameAs ?disURL
}
```

Listing 5: SPARQL query

6.5 Query 5: Get the list of entities of type `nerd:Person` attached to a media resource

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
   ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX owl:     <http://www.w3.org/2002/07/owl#>
PREFIX nerd: <http://nerd.eurecom.fr/ontology#>

SELECT ?Entity, ?label, ?source, ?disURL
FROM <http://data.linkedtv.eu/graph/linkedtv>
```

```
WHERE {
   ?MediaFragment ma:isFragmentOf <http://data.linkedtv.eu/media/
      b82fb032-d95e-11e2-951c-f8bdfd0abfbd> .
   ?MediaFragment a ma:MediaFragment .
   ?MediaFragment linkedtv:hasSubtitle ?subtitle .
   ?AnnotationEntity oa:hasTarget ?MediaFragment.
   ?AnnotationEntity oa:hasBody ?Entity.
   ?Entity a linkedtv:Entity .
   OPTIONAL { ?Entity rdfs:label ?label .  }
   OPTIONAL { ?Entity dc:source ?source . }
   ?Entity owl:sameAs ?disURL .
   ?Entity  a nerd:Person.
}
```

Listing 6: SPARQL query

6.6 Query 6: Get the list of entities for a media resource which are temporally located inside a given period of time (a,b)

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
   ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX owl:     <http://www.w3.org/2002/07/owl#>

SELECT ?Entity, ?label, ?source, ?disURL
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
   ?MediaFragment ma:isFragmentOf <http://data.linkedtv.eu/media/
      b82fb032-d95e-11e2-951c-f8bdfd0abfbd> .
   ?MediaFragment a ma:MediaFragment .
   ?MediaFragment linkedtv:hasSubtitle ?subtitle .
   ?AnnotationEntity oa:hasTarget ?MediaFragment.
   ?AnnotationEntity oa:hasBody ?Entity.
   ?Entity a linkedtv:Entity .
   OPTIONAL { ?Entity rdfs:label ?label .  }
   OPTIONAL { ?Entity dc:source ?source . }
   ?Entity owl:sameAs ?disURL .
   ?MediaFragment nsa:temporalStart ?start ;
    nsa:temporalEnd ?end .
   filter(?start > 60 ) .
   filter(?end < 120 )
}
```

Listing 7: SPARQL query

6.7 Query 7: Get the list of the more frequent entities for a media resource

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
SELECT ?mr ?label ?url count(?url)
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
  ?mr a ma:MediaResource .
  ?mf ma:isFragmentOf ?mr .
  ?ann oa:hasTarget ?mf.
  ?ann oa:hasBody ?entity.
  ?entity a linkedtv:Entity .
  ?entity dc:source "textrazor" .
  OPTIONAL { ?entity rdfs:label ?label . }
  OPTIONAL { ?entity owl:sameAs ?url . }
}
GROUP BY ?mr ?label ?url
```

Listing 8: SPARQL query

6.8 Query 8: Get the list of enrichments triggered by a particular entity E given its label

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
   ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX owl:     <http://www.w3.org/2002/07/owl#>
PREFIX nerd: <http://nerd.eurecom.fr/ontology#>
PREFIX prov:    <http://www.w3.org/ns/prov#>

SELECT ?MediaResources ?URL
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
   ?MediaFragment ma:isFragmentOf <http://data.linkedtv.eu/media/
      b82fb032-d95e-11e2-951c-f8bdfd0abfbd>  .
   ?MediaFragment a ma:MediaFragment .
   ?MediaFragment linkedtv:hasSubtitle ?subtitle .
   ?AnnotationEntity oa:hasTarget ?MediaFragment .
   ?AnnotationEntity oa:hasBody ?Entity .
   ?Entity a linkedtv:Entity .
   ?Entity rdfs:label Berlin .
   ?AnnotationRelResource a oa:Annotation .
   ?AnnotationRelResource prov:wasDerivedFrom ?Entity .
   ?AnnotationRelResource oa:hasBody ?MediaResources .
   ?MediaResources ma:locator ?URL
}
```

Listing 9: SPARQL query

6.9 Query 9: Get the list of media fragments (initially, with any granularity) that have at least one enrichment result attached

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
   ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX owl:     <http://www.w3.org/2002/07/owl#>
PREFIX nerd: <http://nerd.eurecom.fr/ontology#>
PREFIX prov:    <http://www.w3.org/ns/prov#>

SELECT ?MediaFragment
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
   ?MediaFragment ma:isFragmentOf <http://data.linkedtv.eu/media/
      b82fb032-d95e-11e2-951c-f8bdfd0abfbd>  .
   ?MediaFragment a ma:MediaFragment .
   ?AnnotationRelResource a oa:Annotation .
   ?AnnotationRelResource oa:motivatedBy oa:linking .
   ?AnnotationRelResource oa:hasBody ?MediaResources .
   ?AnnotationRelResource oa:hasTarget ?MediaFragment .
   ?MediaResources a ma:MediaResource .
   ?MediaResources ma:locator ?URLs
}
```

Listing 10: SPARQL query

6.10 Query 10: Get the list of enrichment media resources interlinked to a certain media fragment

```
PREFIX linkedtv: <http://data.linkedtv.eu/ontologies/core#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX nsa: <http://multimedialab.elis.ugent.be/organon/ontologies/
   ninsuna#>
PREFIX ma: <http://www.w3.org/ns/ma-ont#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX owl:     <http://www.w3.org/2002/07/owl#>
PREFIX nerd: <http://nerd.eurecom.fr/ontology#>
PREFIX prov:    <http://www.w3.org/ns/prov#>

SELECT ?MediaResources ?URLs
FROM <http://data.linkedtv.eu/graph/linkedtv>
WHERE {
   ?AnnotationRelResource a oa:Annotation .
   ?AnnotationRelResource oa:motivatedBy oa:linking .
   ?AnnotationRelResource oa:hasBody ?MediaResources .
   ?AnnotationRelResource oa:hasTarget <http://data.linkedtv.eu/media/
      b82fb032-d95e-11e2-951c-f8bdfd0abfbd#t=80.12,128.96> .
   ?MediaResources a ma:MediaResource .
   ?MediaResources ma:locator ?URLs
}
```

Listing 11: SPARQL query

7 Conclusion and Future Work

In this deliverable, we described the stable version of the LinkedTV core ontology available at `http://data.linkedtv.eu/ontologies/core`. The main design decision has always been to design a lightweight model that would re-use as much as possible existing vocabularies and defining new classes and properties only when necessary. In particular, the Open Annotation is used for both annotating and enriching seed video content using the annotation motivation. Furthermore, we have completed a first mapping between this model and the LUMO ontology used as a central component for personalizing the LinkedTV experience.

We have developed the TV2RDF REST service (and a preliminary user interface) that converts in RDF (following the LinkedTV ontology) the following metadata elements: legacy metadata provided by the content provider (RBB, Sound & Vision), multimedia analysis results provided by WP1 and named entities extraction results provided by NERD.

We have finally developed the TVEnricher REST service that provides a set of media resources as suggestion of enrichments for media fragments created on a seed video program. Our approach is that named entities are the trigger for this enrichment. How multiple entities and visual cues can be aggregated for generating more relevant enrichments is part of our future work. We will largely explore this research direction during the third year of the project. The MediaEval Benchmarking Initiative in which we have participated (see Deliverable D2.5 [6]) provides a perfect setting for exploring and evaluating new ideas.

References

[1] LinkedTV consortium. Deliverable 2.2: Specification of lightweight metadata models for multimedia annotation, 2012.

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

[3] LinkedTV consortium. Deliverable 5.1: Linkedtv platform architecture, 2012.

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

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

[6] LinkedTV consortium. Deliverable 2.5: Specification of the linked media layer, 2013.

[7] LinkedTV consortium. Deliverable 4.4: User profile and contextual adaptation, 2013.

[8] LinkedTV consortium. Deliverable 4.5: Content and concept filter v2, 2013.

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

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

[11] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Transaction Interneternet Technology*, 2:115–150, May 2002.

[12] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *International Semantic Web Conference (ISWC'11)*, pages 273–288, 2011.

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.