D2.2.1/D3.2.1/D4.2.1/D5.2.1

First Specifications in Cross-Media Analysis, Metadata Publishing, Querying and Recommendations

Grant Agreement No:	610480
Project title:	Media in Context
Project acronym:	MICO
Document type:	D (deliverable)
Nature of document	R (report)
Dissemination level:	PU (public)
Document number:	610480/FHG, UMU, UP, SRFG, UOX/D2.2.1/D3.2.1/D4.2.1/D5.2.1/D/PU/a1
Responsible editor(s):	Patrick Aichroth, Johanna Björklund, Kai Schlegel, Thomas Kurz, Grant Miller
Reviewer(s):	S. Schaffert
Contributing participants:	SRFG, FHG, UP, UMU, ZA
Contributing workpackages:	WP2, WP3, WP4, WP5
Contractual date of delivery:	31 October 2014

First Specifications in Cross-Media Analysis, Metadata Publishing, Querying and Recommendations

Emanuel Berndl, Kai Schlegel, Christian Weigel, Patrick Aichroth, Johanna Björklund, Thomas Kurz, Rafa Haro, Marcel Sieland, Luca Cuccovillo

berndl@dimis.fim.uni-passau.de, schlegel@dimis.fim.uni-passau.de,
christian.weigel@idmt.fraunhofer.de, patrick.aichroth@idmt.fraunhofer.de, johanna@cs.umu.se,
schlegel@dimis.fmi.uni-passau.de, thomas.kurz@salzburgresearch.at, rharo@zaizi.com,
marcel.sieland@idmt.fraunhofer.de, luca.cuccovillo@idmt.fraunhofer.de

November 7, 2014

Abstract
This deliverable summarizes the first specifications in cross-media analysis, metadata publishing, querying and recommendations. It is a joint outcome of work packages WP2, WP3, WP4 and WP5, and describes the initial plans for planned and used MICO technologies as well as their combined operation.

Keyword List
Specification, Cross-Media Analysis, Metadata Publishing, Querying, Recommendations

Contents

List of Figures

List of Tables

1 Executive Summary

This document provides an initial specification draft for WP2-5 in MICO. At its core, it provides a standard ontology for publishing metadata based on several existing ontologies, adapted and extended to the MICO requirements, now available at `http://www.mico-project.eu/ns/platform/1.0/schema#` and complements by an annotation design based on the Open Annotation Data Model, including examples.

Moreover, a first set of prioritized textual, visual and audio extractors is presented, which aim at the first two selected showcases from InsideOut10 and Zooniverse. For each of them, design choices and evaluation criteria are outlined, which serve as a basis for future testing. The input and output is described and the resulting annotation design is derived from that, which provides the basis for the integration of extraction results. In addition, potential interactions with other extractors have been identified. Even within this initial set of extractors it has become obvious that there is significant potential for improving performance regarding quality and completeness through a combination of approaches.

Regarding querying, an extended version of the query language for the Semantic Web SPARQL is presented. Besides basic SPARQL functionality like full read-write support for structured data and novel path patterns, it supports fulltext search for several languages as well as spatio-temporal query functions. To improve the efficiency and to build a proper basis for further extensions, MICO is providing a SPARQL-to-SQL translation, which is integrated in the Open Source software Apache Marmotta.

Finally, several cross-media recommendations approaches are presented (content-based, collaborative filtering, ontology-based) and their technical specifications are introduced. The proposed recommendation models extend classical recommender system approaches with the aim of taking advantage of the semantic features extracted by the extractors and the multimedia extension of SPARQL. The benefits are a higher accuracy of the content suggestions and the possibility to group users with similar traits.

2 Introduction and Overall Considerations

This deliverable represents an initial specification draft from WP2 (cross-media extraction), WP3 (cross-media metadata publishing), WP4 (cross-media querying), and WP5 (cross-media recommendation).

In this chapter we briefly introduce the relevant terminology, general design options, and an explanation of the technology enablers (TE) selection for this initial draft. The main part of this document consists of the following parts, representing the output of the respective WP2-5:

- Task 2.2: Common Model for Cross-Media Extraction, which focuses on the description of a model for cross-media extraction. With respect to this deliverable, this activity provided the descriptions of a first set of extractors (uniquely identified as technology enablers, TE) in 5.
- Task 3.2: Ontology and Protocol for Cross-Media Publishing, which develops a standard ontology and protocol for publishing metadata about media objects and metadata from extractors. This activity provided the preliminaries and basics of the metadata model in 4, and the derived RDF annotation models for the TE in 5.
- Task 4.2: Cross-Media Query Language, which develops an extended version of SPARQL with capabilities for multimedia querying and querying results of extractors. This activity provided the language specification and evaluation model in 6.
- Task 5.2: Cross-Media Recommendation Models, which provides different approaches for calculating cross-media recommendations using the outcomes of WP2-WP4. This activity provided the descriptions of the overall recommendation approach and related TE) in 5.

2.1 Terminology

The document complies with the requirements analysis and system design methodology described in the requirement compendium document in month 6 of the project, reusing several of the entities described there:

- **Showcases (SC)**, which represent the main starting point for requirements analysis. SC are current or planned projects of Zooniverse (ZOO) and InsideOut10 (IO).
- **User Stories (US)**, which are derived from and/or referring to showcases. They serve as a starting point for requirements analysis, defining technology-free, high-level requirements in the form *As a ¡role¿, I want ¡goal/desire¿ (so that ¡benefit¿).*
- **Technology Enablers (TE)**, which represent the technical counterpart to user stories. They are used to express which technologies are required to support user stories and showcases. By linking TE to US, they make sure that what is "wanted" from a user perspective will be matched by enabling technologies, thereby creating the link that all further research and development work can refer to. They are especially for this document, because many relevant components and approaches, especially extractors (WP2) and recommendation approaches (WP5) are described and identified as TE in this document.

By using this approach, the deliverable links back all technical work (often represented on the level of TE) to the respective higher-level user stories (US) and showcases (SC), without the need to repeat the respective descriptions.

2.2 General Design Options and Best Practises

In this section we describe general design options an best practices that are shared on the whole MICO system.

2.2.1 Component Versioning

The MICO platform follows a component-based architecture paradigm (see D6.1.1 [SF14]). Independent analysis components (so called *extractors*) are orchestrated in a loosely coupled environment and generate an overall analysis results collaboratively. This form of architecture makes the workflow highly flexible and adaptable for various use cases but complex in other aspects, especially in component versioning on runtime.
Considering component functionality and interface definition we identified two different kinds of versioning that has to be managed:

(a) different versions of the same extractor in the one runtime, and

(b) same version of the same extractor on different runtimes.

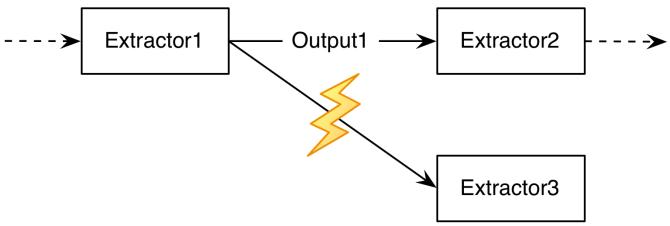
Regarding orchestration both can be handle in the same style. Following Semantic Versioning [PW13] the platform itself could decide which extractor version will be actually invoked (e.g. the newest version, the latest stable version, etc.). The version identification can be done e.g. by encoded this information in the extractor URI [Erl+12] at registration time.
For managing complex workflows or enabling things like backwards compatibility, the version information might be not sufficient. Richer descriptions can be provided e.g by using hypermedia APIs (e.g., Hydra [Lan14]) or ontological approaches [Mar+04].

It has to be mentioned that versioning aspects are not yet addressed in the current platform implementation but will be covered in future revisions.

2.2.2 Syntax vs. Semantics, Decoupling of Serialisation and Semantics

A common hidden problem and design choice that comes with architectures that are similar to the one that we use in MICO is the discrepancy between the semantic and the syntax of the output of the extractors. The problem is illustrated in figure 1.

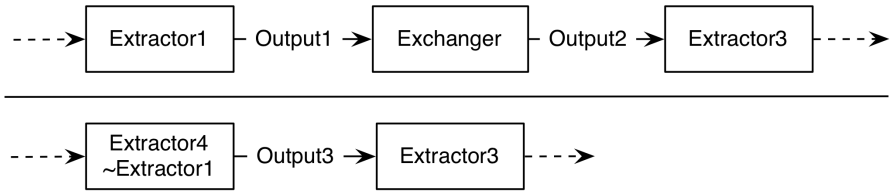
Figure 1 Syntax vs. Semantics Problem of Extractors and their output



Two extractors (`Extractor1` and `Extractor2`) already exist and work in a given workflow (irrespective if there are others in the workflow chain before or after the given ones). These two extractors

"communicate" over the output format `Output1` with its respective MIME type. Now, if there is a new extractor established (`Extractor3`) that can deal with the results of `Extractor1` semantically, it might be possible that the new extractor cannot deal with the `Output1` syntactically, because the extractor would have been needed to be implemented in the way of "understanding" `Output1` on the fly, which is not always the case or not always possible. There are basically two solutions for the problem, which are illustrated in figure 2.

Figure 2 Solutions for the Syntax vs. Semantics Problem



The upper solution introduces another extractor-like component (`Exchanger`), that will work in the way of a syntax exchanger. It transforms its input syntactically, while leaving the semantics of the result exactly as it is. After doing this, the newly introduced exporter can work with the converted output (`Output2`). Another possibility is the lower version in figure 2, which suggests to write an own extractor, which consecutively is very similar to `Extractor1`, that supports the intermediary result for the new workflow. Both solutions can be used for the problem, but in general they are very costly. To move around the problem every extractor should be implemented in a way to decouple its semantics as much as possible from the serialisation. By doing so, most extractors should be able to work with the supported output.

2.2.3 Feature Storage and Output Formats

The MICO architecture in its current design allows two possibilities for realising its feature storage. This deals with the way how the data that is processed and used inside the MICO platform will be stored. Data in this context means the multimedia files on the one hand, and the information, knowledge, or features that we deviate by processing the media files on the other hand. For the former, the process is quite easy, as we will just save the corresponding file as it is. The intermediary results as well as the end result of a extraction workflow is a little bit more difficult. These can be saved as either a binary file of the intermediary output or it can be transformed into RDF output. This consideration is quite important, as every result of an extraction step (even if not intended by its original processing workflow) can be a part of designing and combining different features in order to produce features of higher levels, and thusly take part in another workflow.

The problem on behalf of the RDF transformation lies in the loads of data that might be possible to be created as result of an extraction step. Transferring this into RDF triples might be a cumbersome task and additionally creates a ton of triples. The counterpart to saving RDF, storing the binary files, only requires very few triples in order to specify the location and type of the file. However, the downside is here that the binary file might be difficult to process for further extraction processes, which might lead to further implementation in following extractors.

Depending on the specific extractor or, to be more accurate, its output form, both possibilities might find use. The general way should be to be implementing both, as then every possible workflow can make use of the output form that suits them best.

Regarding the output formats, most of our extractors will have some kind of own proprietary format, but will make use of well known standards to store them. Two of the most commons ones are the Extensible Markup Language 1.0[Bra+08] and the JavaScript Object Notation Data Interchange Format JSON[Bra14]. Both are designed to be very easy to understand and also be machine readable. They are used as a syntax to store, exchange and publish data. While XML focuses on a tree-like tag structure, JSON is supposed to support an easier syntax by depending on a key value based form, which can also create a tree. Very simple examples can be seen in listings 1 (XML) and 2 (RDF), which show the same content formulated in RDF and in XML.

Listing 1: A simple XML example

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Listing 2: A simple RDF example

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Further common output formats will be added here incrementally.

2.2.4 Quality Measurement for Extraction Technologies

Proper quality measurement in the area of information extraction is an important instrument to guarantee a minimum level of result quality as well as to measure the impact of changes or improvements regarding algorithm and implementation. A common measure for extractors with a binary classification model is *Precision (1) and Recall (2)*, whereby t_p (true positives) describe correct results, f_p (false positives) describe results that are detected wrongly and f_n (false negatives) describe missing results. t_n (true negatives) describe results, which are correctly not identified, which is obviously hard to measure and therefor neither used for recall nor for precision. A third measurement, that combines precision and recall in the harmonic mean is F-measure also referred to as F1 score (3). Precision and recall are typically inversely related, which means that a improvement if precision influences the recall negative and vice versa. If a extraction technology produces fuzzy results (e.g. Sentiment Analysis 5.13) they are reduced to a binary problem by introducing a threshold.

(1) $$precision = \frac{t_p}{t_p + f_p}$$

(2) $$recall = \frac{t_p}{t_p + f_n}$$

(3) $$F = 2 * \frac{precision * recall}{precision + recall}$$

3 Prioritized Technology Enablers and Dependency Overview

Cross-media meta data publishing (WP3), querying (WP4) and recommendation (WP5) consist mostly of technology enablers (TE) which are cross-cutting, needed by almost all user stories (US) and showcases (SC). The situation is however very different for extraction (WP2), where a huge selection of potential TE and potential TE adaptations depends on the specific US / SC. Hence, an adequate TE selection / prioritization for the initial project development phase was made, which also applies to this deliverables.

The selection was based on two showcases, one from each application domain, represented by WP7 (Zooniverse) and WP8 (InsideOut10): **Snapshot Serengeti (SC-16)** from Zooniverse, which involves identifying various animals and their behaviour from camera trap images in the Serengeti National Park. Participating volunteers identify animals from a list of 48 species and give information of their numbers and activities. The goal is to help scientists better understand how the species interact with each other. And **News Video (SC-02)** from InsideOut10, which is focused on the production and consumption of videos created by large audiences as user-generated contents, both professional or not, and their publication on community portals or institutional news portal. The relevant technology enablers per showcase are identified in the following.

Snapshot Serengeti (SC-16) includes the need for the following prioritized TE:

- Object and Animal Detection (TE-202), to automatically detect images with no classifiable animals in it.
- Question Detection (TE-212), to detect questions directed to researchers and other forum members.
- Sentiment Analysis (TE-213), to distinguish between different types of discussions.
- Textual Feature Extraction (TE-215), to derive features from text fields to be used in cross-media classification.
- Chatroom cleaner (TE-216), to remove markup and standardize citation, punctuation, etc., as a preprocessing step for e.g. the phrase-structure parser.
- Phrase Structure Parser (TE-217), to assess how interesting / appealing / complex a picture is, to detect when a scientist should be prompted to look at a subject, based on annotations and information from comments, and to when Zoonibot should comment on a subject
- Interactive Wrapper Generator (TE-218), to identify volunteer types.
- Named Entity Recognizer (TE-220), to extract keywords related to e.g. species or activities, to know when Zoonibot should give an explanation, to assess how interesting / appealing / complex a picture is based on comments, and to detect when a scientist should be prompted to look at a subject.

News Video (SC-02), includes the need for the following prioritized TE:

- Low level Feature Extraction (TE-201), as a prerequisite for visual similarity.
- Face Detection and Recognition (TE-204), to detect faces in videos, in order to let them be identified by users, then to receive related information about the respective persons.

- A/V Error Detection and Quality Assessment (TE-205), to filter and rank videos based on A/V quality aspects.
- Temporal Video Segmentation (TE-206), for easier navigation / skipping between segments, annotating segments, and keyframe extraction
- Speech Music Discrimination (TE-207), as a preprocessing step for automatic speech recognition.
- Visual Similarity (TE-211), to detect perceptually similar video items and segments.
- Automatic Speech Recognition (TE-214), to extract text from speech and use this for search purposes.
- Audio Cutting Detection (TE-224), to detect audio cutting / manipulations and locations.
- Media Container Tag Extraction (TE-227), to extract technical metadata / tags from videos.

TE Dependency Overview

It is useful to see the interdependencies between technology enablers, as they are indicators for potential improvements via combining them, e.g. with respect to accuracy. The dotted arrows symbolise that a collaboration between the TEs has not been planned initially, but is possible. Continuous arrows indicate input from the source to the think, while double headed arrows indicate a close interaction between two technology enablers.

Figure 3 Technology enabler dependency overview

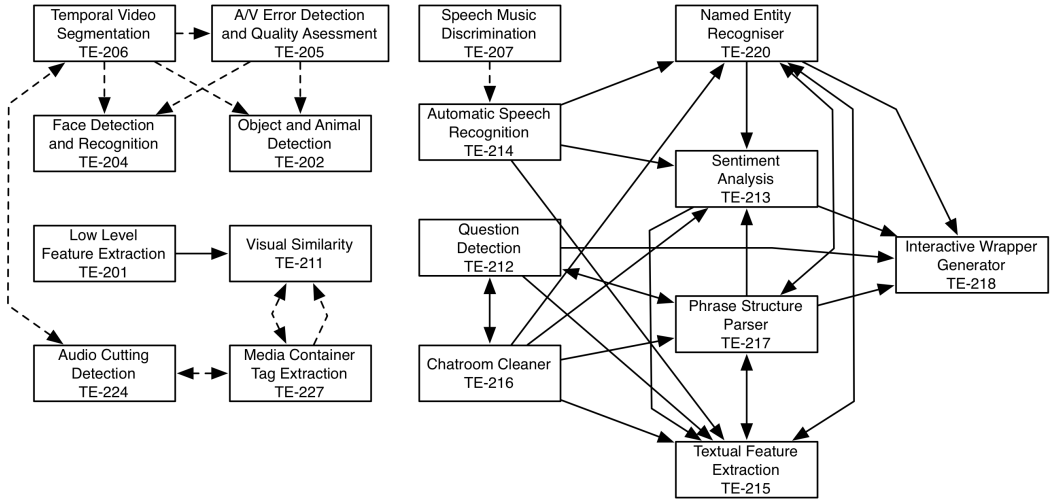


Figure 3 shows, that there are many possible interactions between the technology enablers. Many different workflow chains can be established with the goal of enhancing various evaluation criteria for the results of the extractors.

4 Preliminaries and Basics of the Metadata Model (WP3)

Every workflow and extraction process of the MICO platform will end in accumulations of information, which will for example consist of annotations, multimedia files, or their provenance information. All of this has to be stored and treated in a uniform way, which also allows the saved data and information to be accessible afterwards. For that reason, we developed a metadata model that will cover all of the aspects. This chapter introduces the model, starting with the ontologies of specifications that will be used (section 4.1), leading over to the basic structure for the modelling of the data structures used in the MICO platform (section 4.2). Small examples at the given characteristics will guide through the process and visualise the details for better understanding. Encountered problems and their solutions will also be depicted. Afterwards, section 4.3 will show an exemplary extraction process of a simple workflow, adding explanations for the underlying metadata modelling process next to its results.

4.1 Ontologies and Basic Structures

Our model will combine several existing ontologies. The base structure is posed by the Resource Description Framework RDF[MM04] (which we already covered in D3.1.1) and its schema RDFS[BG14]. RDF is a common standard for data interchange and merging. In terms of modelling annotations, the Open Annotation Data Model[SCS13] is used (we described this already in D3.1.1, further detail is given in section 4.2). As we are dealing with mainly multimedia items and files, an ontology for describing them is needed. Dublin Core DC[Boa12], especially with its subparts for **terms** and **types**, finds use there. For provenance modelling, we will make use of the PROV Ontology PROV-O[LSM13], which supports a wide range and variety of possibilities to illustrate provenance information. The FOAF Vocabulary Specification[BM14] is an ontology allowing to link people and model the relationships between them. MICO will make use of this ontology in order to depict and further enrich provenance information, especially for human agents that take part in MICO processes or workflows. For more concrete content descriptions, the Representing Content in RDF 1.0[KVA11] ontology is applied. WP5 and its recommendations introduce the ontology RDF Review Vocabulary[1], which is used to store for example feedback of users, ratings, comments, and votes. All the used ontologies with abbreviations (that will also be utilised in our examples and modelling) and their respective namespaces can be found in table 1.

Table 1: Used RDF Specifications, with corresponding namespace and abbreviation

Specification Name	Abbr.	Namespace
MICO Schema	mico:	http://www.mico-project.eu/ns/platform/1.0/schema#
MICO Example Instances	entity:	http://www.mico-project.eu/ns/platform/resource/
Open Annotation Data Model[2] [SCS13]	oa:	http://www.w3.org/ns/oa#
Representing Content in RDF[3] [KVA11]	cnt:	http://www.w3.org/2011/content#
Dublin Core Elements[4] [Boa12]	dc:	http://purl.org/dc/elements/1.1/

[1] http://vocab.org/review/terms.html
[2] http://www.openannotation.org/spec/core/
[3] http://www.w3.org/TR/Content-in-RDF10/
[4] http://dublincore.org/documents/dcmi-terms/

Dublin Core Terms	dcterms:	http://purl.org/dc/terms/
Dublin Core Types	dctypes:	http://purl.org/dc/dcmitype/
Friend-of-a-Friend Vocabulary[5] [BM14]	foaf:	http://xmlns.com/foaf/0.1/
Provenance Ontology[6] [LSM13]	prov:	http://www.w3.org/ns/prov#
Resource Description Framework[7] [MM04]	rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
RDF Schema[8] [BG14]	rdfs:	http://www.w3.org/2000/01/rdf-schema#
RDF Review Vocabulary[9]	ref:	http://purl.org/stuff/rev#

These will be combined with an own ontology defined for MICO. It will contain two abbreviations: `mico` will stand for the URL `http://www.mico-project.eu/ns/platform/1.0/schema#`, which defines classes, relationships, and properties for the MICO use case. `entity` for the URL `http://www.mico-project.eu/ns/platform/resource/` depicts the specific instances of the classes. The human and machine-readable specification of the schema is available at `http://www.mico-project.eu/ns/platform/1.0/schema`. The defined vocabulary items are the following:

- **mico:ContentItem** (Class)
  The class for a content item. This class must be associated with every content item.
- **mico:ContentPart** (Class)
  The class for content parts. This class must be associated with every content part.
- **mico:hasContentPart** (Relationship)
  The relationship between a content item and its (single or multiple) content parts. There MAY be 0 or more mico:hasContentPart relations per mico:ContentItem.
- **mico:hasContent** (Relationship)
  The relationship between a content part and its actual content. The content MUST be an annotation (type oa:Annotation). There MUST be exactly one mico:hasContent relation.
- **mico:requires** (Property)
  Property for an extractor registered at the MICO platform. Specifies which MIME type the input file of the given extractor must have. This will be changed to an enhanced approach in future version of the MICO platform.
- **mico:provides** (Property)
  Property for an extractor registered at the MICO platform. Counterpart to mico:requires, this specifies what MIME type the output of the given extractor has. This will also be changed in future versions of the MICO platform.

[5] http://xmlns.com/foaf/spec/
[6] http://www.w3.org/TR/prov-o/
[7] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
[8] http://www.w3.org/TR/rdf-schema/
[9] http://vocab.org/review/terms.html

- **mico:hasId** (Property)
  Property for an extractor registered at the MICO platform. Unique ID for a given extractor.
- **mico:hasQueueName** (Property)
  Property for an extractor registered at the MICO platform. Specifies the queue name that the given extractor listens to.
- **mico:hasVersion** (Property)
  Property for an extractor registered at the MICO platform. Specifies the current version as decimal value of the given extractor.
- **mico:hasLocation** (Property)
  Property that specifies the location of a given multimedia item by supporting its IRI.
- **mico:AnnotationBody** (Class)
  A class that is used as superclass for bodies of an oa:Annotation. Instances of this class are connected over the relationship oa:hasBody to the annotation.
- **mico:MultimediaBody** (Class)
  Subclass of mico:AnnotationBody used for multimedia uploads (e.g. images, audio or video).
- **mico:LowLevelFeatureBody** (Class)
  Subclass of mico:AnnotationBody used for a Low-Level Feature extractor result (see TE-201 in section 5.1).
- **mico:DetectionBody** (Class)
  Subclass of mico:AnnotationBody used for detection extractors (see TE-204 in section 5.2 and TE-202 in section 5.3).
- **mico:FaceDetectionBody** (Class)
  Subclass of mico:DetectionBody used for face detection extractors.
- **mico:EyeDetectionBody** (Class)
  Subclass of mico:DetectionBody used for eye detection extractors.
- **mico:NoseDetectionBody** (Class)
  Subclass of mico:DetectionBody used for nose detection extractors.
- **mico:MouthDetectionBody** (Class)
  Subclass of mico:DetectionBody used for mouth detection extractors.
- **mico:AnimalDetectionBody** (Class)
  Subclass of mico:DetectionBody used for animal detection extractors.
- **mico:RightEyeDetectionBody** (Class)
  Subclass of mico:DetectionBody used for right eye detection extractors.
- **mico:LeftEyeDetectionBody** (Class)
  Subclass of mico:DetectionBody used for left eye detection extractors.
- **mico:MouthCenterDetectionBody** (Class)
  Subclass of mico:DetectionBody used for mouth center detection extractors.

- **mico:FaceRecognitionBody** (Class)
  Subclass of mico:AnnotationBody used for face recognition extractors (see TE-204 in section 5.2).
- **mico:hasConfidence** (Property)
  Property that specifies the confidence value of an extraction result, supported as decimal.
- **mico:AVQBody** (Class)
  Subclass of mico:AnnotationBody used for A/V Error Detection and Quality Assessment extractors (see TE-205 in section 5.4). The purpose of the AVQ TE is to deliver technical measurements of the quality of an image or video with respect to different kinds of quality artifacts.
- **mico:hasRefinedType** (Property)
  Property that refines the type of the given extractor.
- **mico:TVSBody** (Class)
  Subclass of mico:AnnotationBody used for Temporal Video Segmentation extractors, which are able to find shot boundaries and key frames in shots for video sequences (see TE-206 in section 5.5).
- **mico:VSIBody** (Class)
  Subclass of mico:AnnotationBody used for Visual Similarity extractors. Visual Similarity may describe the same mood two videos represent, that the same kinds of objects or scene are shown or that two image or videos do have the same recording source but are modified by basic editing and encoding operations. (see TE-211 in section 5.6).
- **mico:SMDBody** (Class)
  Subclass of mico:AnnotationBody used for Speech Music Discrimination extractors. (see TE-207 in section 5.7).
- **mico:ACDBody** (Class)
  Subclass of mico:AnnotationBody used for Audio Cutting Detection extractors. (see TE-224 in section 5.8).
- **mico:MediaContainerTagBody** (Class)
  Subclass of mico:AnnotationBody used for Media Container Tag extractors. (see TE-227 in section 5.9).
- **mico:NERBody** (Class)
  Subclass of mico:AnnotationBody used for Named Entity Recogniser extractors. (see TE-220 in section 5.10). Named Entity Recognition (NER) classifies sequences of words in text into categories (so-called named entities).
- **mico:PSPBody** (Class)
  Subclass of mico:AnnotationBody used for Phrase Structure Parser extractors. (see TE-217 in section 5.11). A phrase structure parser provides the constituency relation of a natural language sentence.
- **mico:ASRBody** (Class)
  Subclass of mico:AnnotationBody used for Automatic Speech Recognition extractors (see TE-214 in section 5.12). Automatic speech recognition (ASR) consists in automatically transcribing what is being said in a video or audio stream.

- **mico:SentimentAnalysisBody** (Class)
  Subclass of mico:AnnotationBody used for Sentiment Analysis extractors (see TE-213 in section 5.13). Sentiment analysis consists in establishing the writer's judgement, affective state, or intended emotional communication.

- **mico:ChatroomCleanerBody** (Class)
  Subclass of mico:AnnotationBody used for Chatroom Cleaner extractors (see TE-216 in section 5.14). The chatroom cleaner takes the textual data from chatrooms and "cleans" it, making it more suitable for processing by text analysis tools, such as parsers, named entity recognisers, and sentiment analysers.

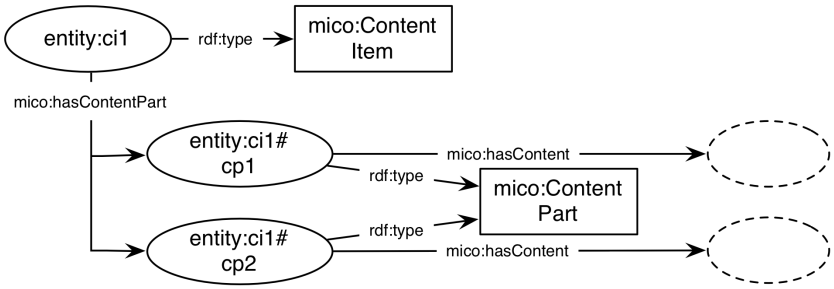
- **mico:IWGBody** (Class)
  Subclass of mico:AnnotationBody used for Interactive Wrapper Generator extractors (see TE-218 in section 5.15). Interactive Wrapper Generators find out how well the inherent structure in web pages and XML documents interact with the structure of extracted media metadata from the web page elements.

- **mico:TextualFeatureBody** (Class)
  Subclass of mico:AnnotationBody used for Textual Feature extractors (see TE-215 in section 5.16). Textual Feature extractors output a document-feature vector representing how many times a feature F was encountered in media asset M.

- **mico:QuestionDetectionBody** (Class)
  Subclass of mico:AnnotationBody used for Question Detection extractors (see TE-212 in section 5.17).

The whole metadata modelling in MICO deals with multimedia files that will be inserted into the MICO platform. There, these files will be saved and arranged in **Content Items**, which are divided in multiple **Content Parts** (see D6.1.1 and D6.2.1 for further detail). Initial files will create the first content part, while the workflow with its extraction results will create intermediate results. All of them will create their own distinctive annotation. Figure 4 shows a small example, how this is modelled.

Figure 4 The basic structure of content items and their respective content parts

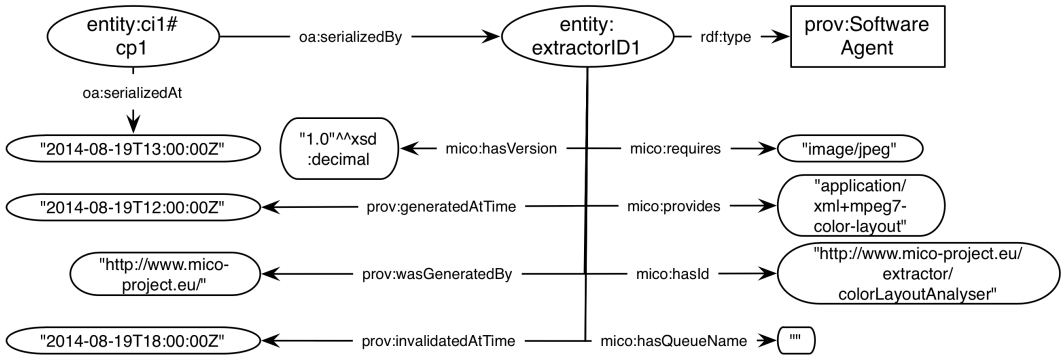


`entity:ci1` depicts the central content item, its type is an instance of the class `mico:ContentItem` respectively. Following the content item (indicated by the relationship `mico:hasContentPart` are two content parts, namely `entity:ci1#cp1` and `entity:ci1#cp2`, which also receive the corresponding type `mico:ContentPart`. Following examples where content items or content parts are present will

leave out the specific typing for reasons of clarity. The instances and their corresponding typing are self explanatory. Every content part will contain an annotation (only denoted by dotted lines here, concrete annotations will be covered later in this deliverable), which is illustrated by the relationship `mico:hasContent`.

Another part of the MICO workflow are the extractors and their results. These are used as an intermediary result or turned into a final annotation. The possible composition and interplay between the extractors is treated in D6.1.1 and D6.2.1, the technical description of the extractors can be seen in D2.1.1 and later in this document (see section 5). The information that will be created by linking one extractor to the MICO platform will pose the first part of provenance information. This will be crucial in order to have full provenance information about the specific workflow chain. Another provenance feature will be posed by the timestamp that is added for the given content part. Figure 5 shows a simple example of an extractor for a ColorLayout Feature[KY01].

Figure 5 A single extractor with its provenance information and a content part that is created by that extractor



The content part gets a timestamp when it has been received at the MICO platform, which is illustrated by the relationship `mico:serializedAt`. `oa:serializedBy` indicates, which agent has fulfilled the process of creating the annotation or doing the extraction. The agent can be a person creating an annotation, symbolised by the class `foaf:Person`, or an automated software procedure `prof:SoftwareAgent`. For a `prov:SoftwareAgent`, several properties will be stored:

- `mico:requires`: Defines the MIME type that the extractor is able to process. This is important because the MICO platform will only send jobs to this extractor with the given MIME type. The MIME type definition will be exchanged to an enhanced approach in later versions of the MICO platform.
- `mico:provides`: Counterpart to the `mico:requires`, this property supports the MIME type of the result that the extractor produces. This will also be enhanced in the same fashion as the requires property.
- `mico:hasId`: The unique ID for the extractor.
- `mico:hasQueueName`: The name of the queue that the extractor listens to. This impacts the RabbitMQ[10] implementation of the MICO platform (see D6.1.1 and D6.2.1).

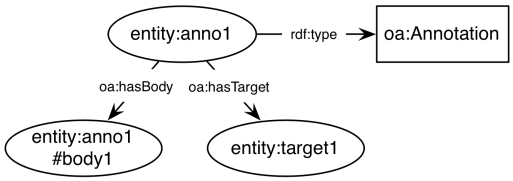
[10] http://www.rabbitmq.com/

- `mico:hasVersion`: Specifies the version of the extractor. Will be updated once a new version of the same extractor-type is registered at the MICO platform. We will leave out the specification as `xsd:decimal` in our further utilisations for the sake of clarity.
- `prov:generatedAtTime`: Timestamp when the extractor was successfully registered at a given MICO platform.
- `prov:wasGeneratedBy`: Indicates who registered the extractor at the given MICO platform.
- `prov:invalidatedAtTime`: Timestamp when the extractor is unregistered from the given MICO platform.

4.2 Annotation Structure based on the Open Annotation Model

The metadata that will be created and modelled throughout the whole project deals with content items, their distribution into smaller content parts as well as their raw file content. Every step is augmented with provenance information. One part which will be covered by RDF and metadata modelling are annotations, which will form the end of the extraction workflows in the MICO platform. The metadata model is based on the **Open Annotation Data Model** (OADM)[SCS13]. For that matter, this section will explain the OADM regarding its usage in the MICO environment. Starting at the basic shape of annotations, we will go over the sections and advantages of the OADM and explain them by showing simple and to-the-point examples. For every characteristic, only the key model feature will be visualised for better understanding. Details about other features that specific annotation could have will be left out. Every annotation starts with the base resource of the annotation itself, which is typed by the relationship `rdf:type` as being an `oa:Annotation`. Every annotation has got a body and a target, added by the relationships `oa:hasBody` and `oa:hasTarget` respectively. For the annotations there can be multiple bodies and targets. A general annotation has one body and one target, but there are special cases which depend on multiple targets. This will be covered later in this chapter. The body of an annotation depicts the content of the annotation itself. In the MICO environment these can be for example text annotations, area annotations, different proprietary standards and settings that will be converted into RDF. There will be examples of specific bodies later in this section and the concrete shapes of annotations will be explained in detail in chapter 5. The target of a body is the aim of the annotation, the OADM standard describes this by stating that the ”body is somehow about the target”. Often times in our use cases, the target of the annotations have already been addressed as content of a content part, so the connection from the annotation to its target is directed there (relationship `mico:hasTarget`). The base structure of an annotation is depicted in figure 6.

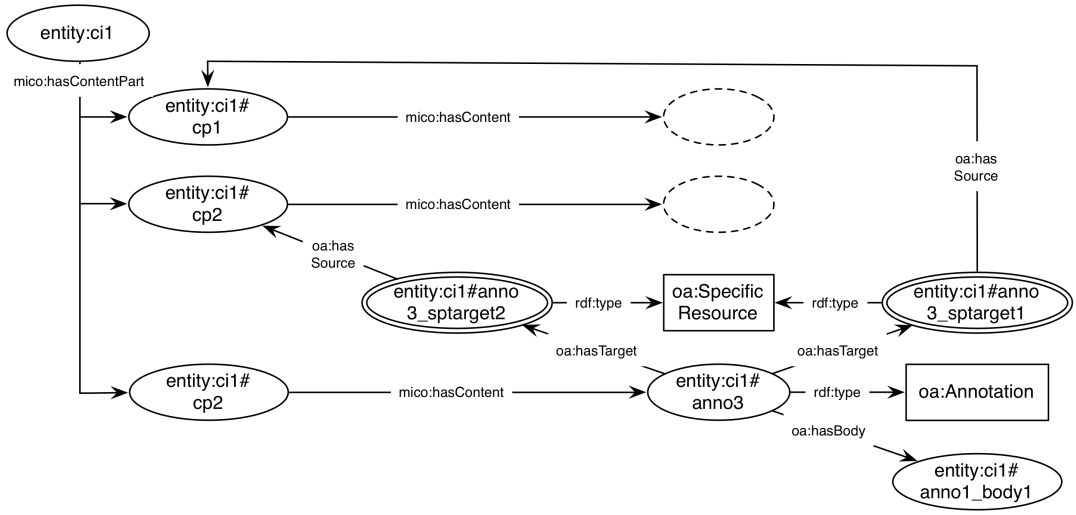
Figure 6 The core structure of every annotation



The targets of our extractor steps will not be referencing single entities or files. In general, the target of a result will be derived from another content part of the content item that is currently executed. In the

OADM as well as our structure, this will be established by introducing specific resources with selectors and specifiers. This allows to link a given annotation to the content part that supports the requested input. Figure 7 shows an example of a specific resource.

Figure 7 An example annotation using a specific resource as target



The annotation `mico:ci1#anno3` is extracted from the input contained at the first content part (`entity:ci1#cp1`) and second content part (`enitity:ci1#cp2`) of the content item `mico:ci1#cp1`. Therefore, the annotation has got two specific resources (type `oa:SpecificResource`), which will make use of the relationship `oa:hasSource` in order to link to the content parts that are used as input for the given annotation. Altogether, an annotation step can have no input (when it is the initial content part), a single input or multiple inputs.

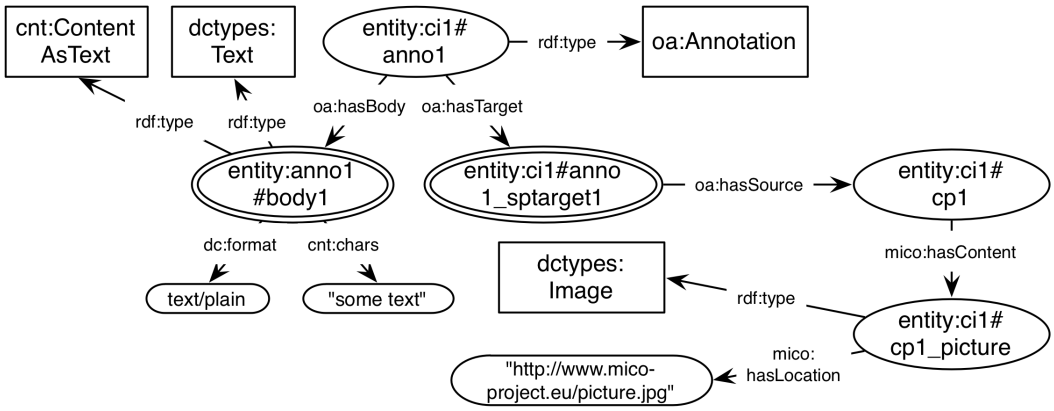
If necessary, these specific resources can then further be enhanced by specifying selectors for the supported target. A selector is used, when only subparts of the target are chosen to be the aim of the annotation. This can for example be an area of a picture or video (called fragments, which are conform to the W3C Media Fragments[11]), text parts, data positions or SVG selectors.

The next step of annotation modelling is posed by setting the type of the body and target. This allows the annotations to be more concrete and defined. Additionally, search capabilities are improved as the search can be directly aimed towards specific body or target types. For example, you can directly query the database for all given annotations that are attached to pictures. In MICO, examples of body types are defined by the extractors. These are for example face detection and recognition (see section 5.2) or Named Entity Recogniser (see section 5.10) extractors, and will be covered in detail in section 5. This list will be extended incrementally as we add more extractors to the architecture. Figure 8 shows an example containing a text-annotation that is about a given picture, `entity:ci1#cp1_picture`. Therefore, the body is typed as `dctypes:Text` **and** `cnt:ContentAsText`. This kind of modelling is deviated completely from the OADM and gives several advantages opposed to the possibility of writing the text of the annotation as content of the body node. In addition to the two defined types, the format is applied by the relationship `dc:format` and its content "`text/plain`". The actual text content of the

[11] http://www.w3.org/TR/media-frags/#standardisation-URI-fragments

annotation is supported as string literal and the corresponding relationship `cnt:chars`. The target, as it is a picture, is characterised with the type `dctypes:Image` which is enhanced by supporting the location of the picture through the relationship `mico:hasLocation`.

Figure 8 RDF example with typed body and target



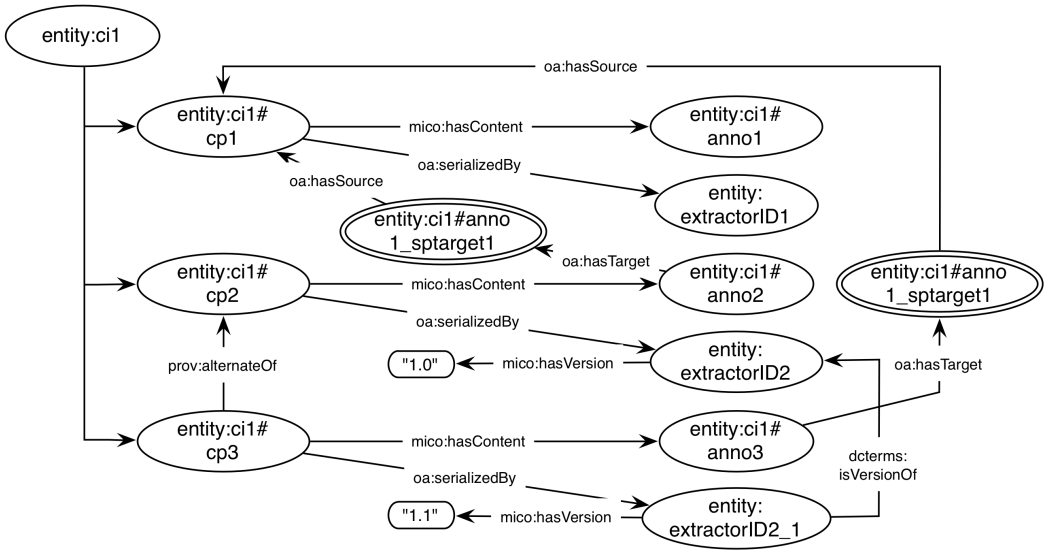
Another feature of the model is the versioning of annotations and extractors. This is needed when a registered extractor is changed, e.g. when the process of the extractor needs to be modified to be more efficient. When an extractor update happens, all results that were supported by the specific extractor need to be updated. Accordingly, all content items and their respective content part will be executed again in their corresponding workflow. The new content part will be linked to its old predecessor with the relationship `prov:alternateOf`. If the original content part, or more precisely its annotation, was the input for consecutive extraction calls, the new result will trigger those with the newly generated intermediary result. New content parts will be created respectively, but those will not have `prov:alternateOf` edges, as those are the result of a "normal" extraction workflow. Figure 9 shows an example of a versioned content part.

`entity:ci1#cp2` has its updated version in `entity:ci1#cp3` and they are therefore connected. As no further steps followed the one creating the second content part, no other content parts are created. Also, a new version for the extractor has been added, namely `entity:extractorID2_1`, which is an updated version for `entity:extractorID2`. The version number that is supported by the relationship `mico:hasVersion` is incremented in order to display this. In future implementations the reason of change will be modelled in more detail for an extractor.

4.3 Overall Annotation Example

In order to clarify the interaction and the composition of all the ontologies, the OADM and our extensions, this section will go over an example process of a workflow through the MICO platform. It will depict the metadata that is created, covering the annotations, the provenance information that develops from every step, as well as the entire RDF structure that is built up as a result of inserting one single content item into the platform. The RDF content is split up at representative resources for the sake of clarity. All these steps will be backed up by a sequence diagram, which shows the iterations taken in the platform in addition to their components. Figure 10 shows the picture that will be utilised to exercise

Figure 9 A content item with versioned content parts



the process. We will assume that a user has given an area-based annotation in the lower left corner, indicating that she or he thinks that the animal is a Dikdik[12].

The execution plan that is chosen for the example is quite simple and only contains two major steps (The extractors that are used here are only exemplary, but follow the general design of MICO extractors. Their invoked vocabulary will not be contained in the final metadata model):

- Extract an MPEG-7 ColorLayout feature[KY01] for the whole image. This step will invoke two separate extractor calls. The first extractor generates the ColorLayout feature as an XML file, which will be stored, too. The second instance will then take the XML as input and transform the contained information into RDF.
- Triplify the information of the given area-based text-annotation. This step does only require a single extractor iteration, as the annotation will be directly serialised as RDF content.

The corresponding workflow is depicted by the following pictures and described in their captions. The steps of creating the whole result are in the same order as the following figures.

Listing 3: Output of the ColorLayoutExtractor for figure 10

```
<VisualDescriptor xmlns="urn:mpeg:mpeg7:schema:2004"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ColorLayoutType">
  <YDCCoeff>38</YDCCoeff>
  <CbDCCoeff>22</CbDCCoeff>
  <CrDCCoeff>34</CrDCCoeff>
  <YACCoeff5>11 28 13 18 9</YACCoeff5>
  <CbACCoeff2>15 24</CbACCoeff2>
  <CrACCoeff2>18 9</CrACCoeff2>
</VisualDescriptor>
```

[12] http://www.awf.org/wildlife-conservation/dik-dik

Figure 10 The picture used for the overall example, showing two animals (Dikdiks) and a supported area annotation

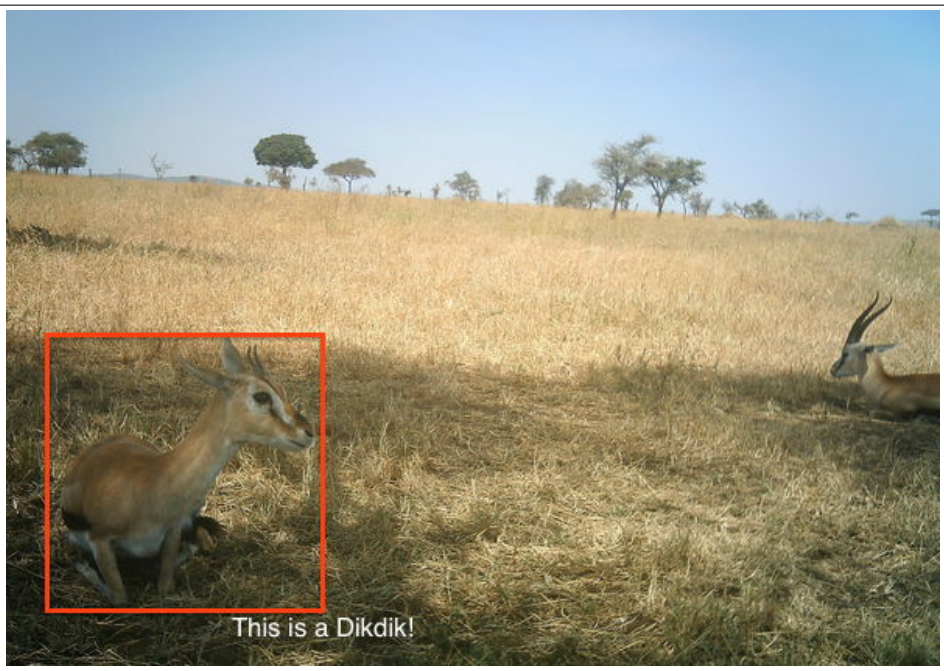


Figure 11 A sequence diagram illustrating all the corresponding components as well as their invocation is shown here. The components are (from left to right) the **Data Storage**, which is responsible for storing all types of "raw" data, followed by the **Metadata Storage** for all the RDF metadata content. The **Content Item API** depicts the base element of the architecture that is responsible for the core API as well as the triggering and handling of main processes. This component also communicates with both of the storages. In close collaboration is the **Message Manager**, which is responsible for maintaining the message queues, transportation, and protocols between the Content Item API and all the available extractors. In our case these are namely the **ColorLayoutExtractor**, **ColorLayoutTriplifier** and the **AreaAnnotationTriplifier**. For further illustration of the core components and their design, see D6.1.1 and D.6.2.1. The arrival of the content item triggers the whole chain, in our case, the annotated picture (which will be contained in a file structure, see D6.1.1 for further detail). This activity can be seen on the top process of the Content Item API in this figure. First steps will cover the storage of the raw image as well as its provenance entries. The invocation can be seen in the top left corner of the sequence diagram. The provenance information for the content parts is similar, so we will only cover that procedure once.

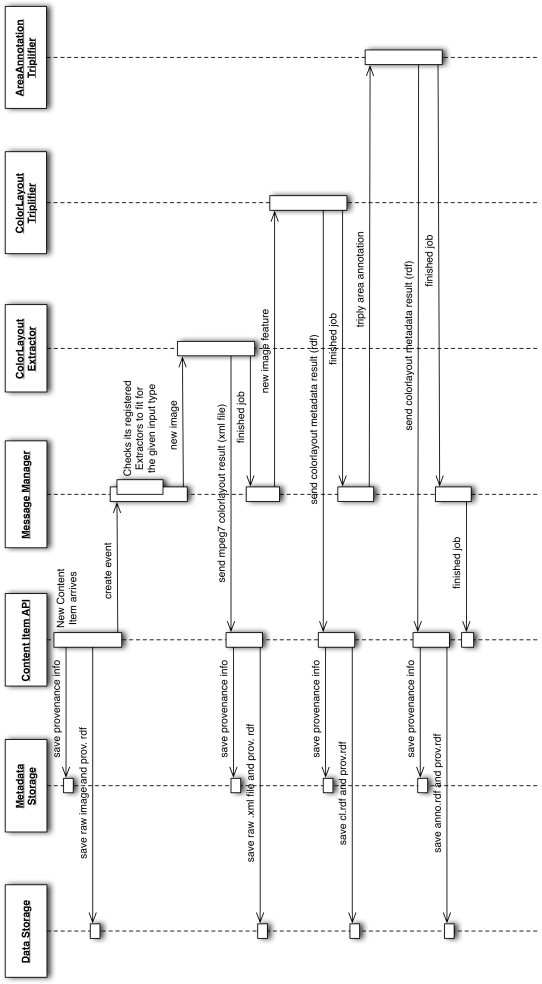


Figure 12 When a content part is received and created at the Content Item API, the RDF content of the item will be enhanced by adding the new content part and its own corresponding content with provenance information. The content of the content part is an annotation (`entity:ci1#anno1`), its body is of the type `mico:MultimediaBody`. This body type indicates, that the annotation contains an initial multimedia file. The annotation has no target. For the first content part, this can be seen on the top of this figure at the resource `entity:ci1#cp1`. The provenance for a content part consists of a timestamp indicating when the corresponding content part arrived at the Content Item API as well as the related extractor (or in the case of the initial item an indicator for that case). This is expressed by the two properties `oa:serialisedAt` and `oa:serialisedBy`. Next to the original content item and its parts, this figure also indicates many of the extractors that are in use in connection to the whole extraction process. Further details about the extractors will be added as soon as one extractor is registered at the Content Item API. We will also store provenance information for them. These are depicted in figure 13 for the exemplary process.

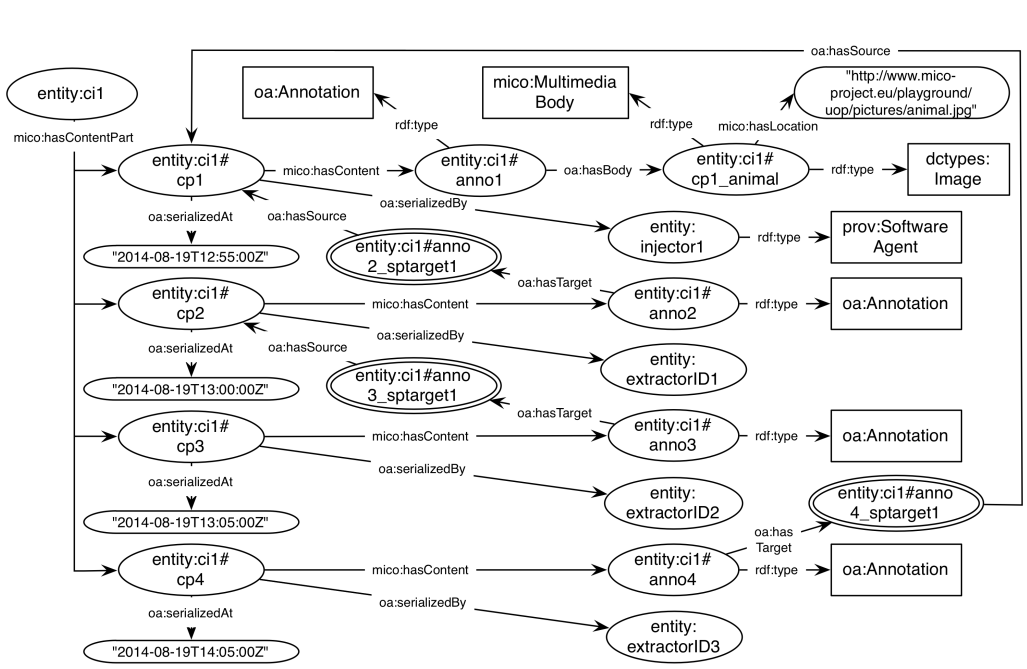


Figure 13 For every extractor, the required input format (`mico:requires`), the supported output format (`mico:provides`), its unique identifier (`mico:hasId`), a special queue name (`mico:hasQueueName`) and the point of time when the extractor was created/registered and unregistered (`prov:generatedAtTime` and `invalidatedAtTime`) as well as the organisation that hosts the specific extractor (`prov:wasGeneratedBy`) will be saved.

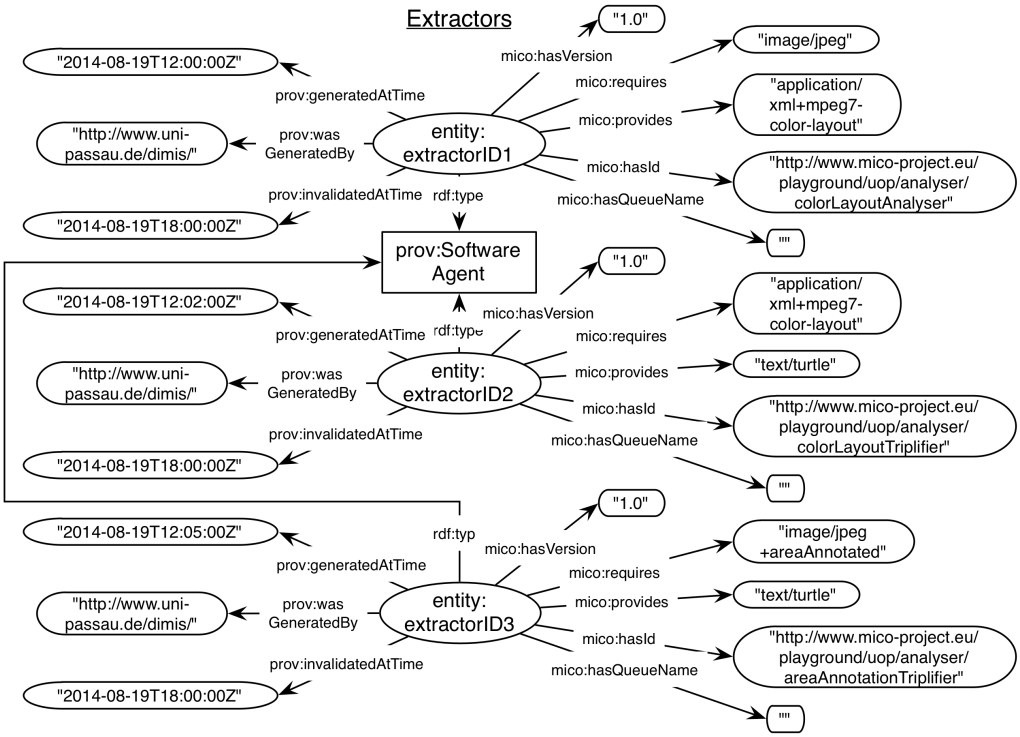


Figure 14 Following the initial tasks, the Content Item API sends a notification to the Message Manager, indicating that a new content item hast been initialised. In this process, the type of the content item is also transferred. With this type, the Message Manager knows what extractors of his can process and deal with the corresponding type. These extractors will then be triggered and notified where to find the associated input. In our example, the first extractor that will be triggered is the ColorLayoutExtractor. It needs an image file as input and will generate the ColorLayout feature as XML file output in combination with an annotation. That file will be saved together with its provenance information respectively. The direct XML output of the extractor can be seen in listing 3. The corresponding resource is `entity:ci1#anno2` in figure 12. The content of the annotation is depicted in this figure and its provenance information in figure 15.

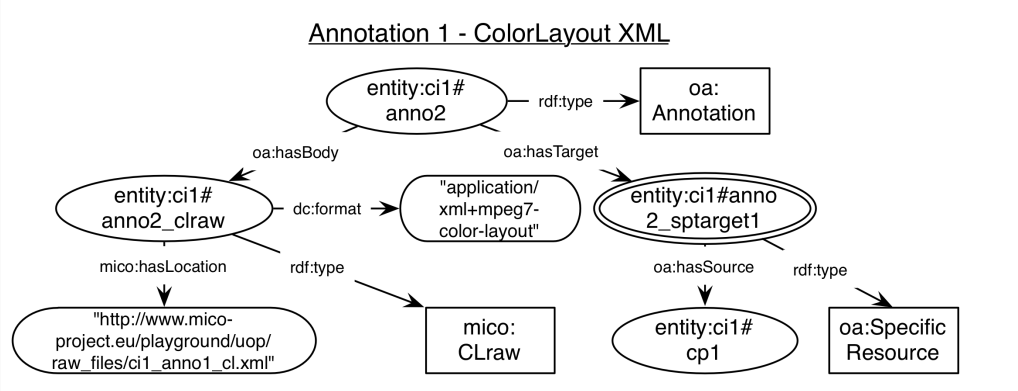


Figure 15 Provenance information for the annotation created by the ColorLayoutExtractor

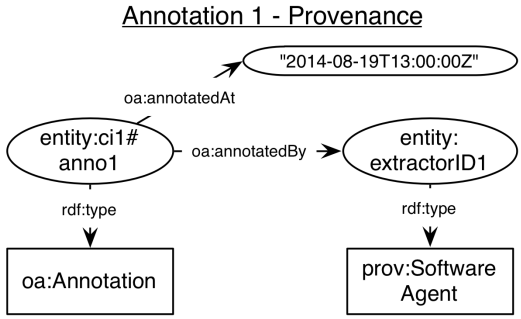


Figure 16 The intermediate result will be used by the next extractor, namely the ColorLayoutTriplifier and will create a ColorLayout annotation. The process of triggering and consecutively saving the raw file and its provenance information are in terms of shape equal to the preceding provenance steps. The result of the extraction is shown in this figure. Annotations will have extended provenance information, that is created by the extractor itself. For this annotation, this can be seen in figure 17. The provenance at the Content Item API tracks when the specific result has been received there, while the information at the annotation itself covers provenance facts of the extraction process. These two figures can be combined with figure 12 at the resource of the third annotation, `entity:ci1#anno3`.

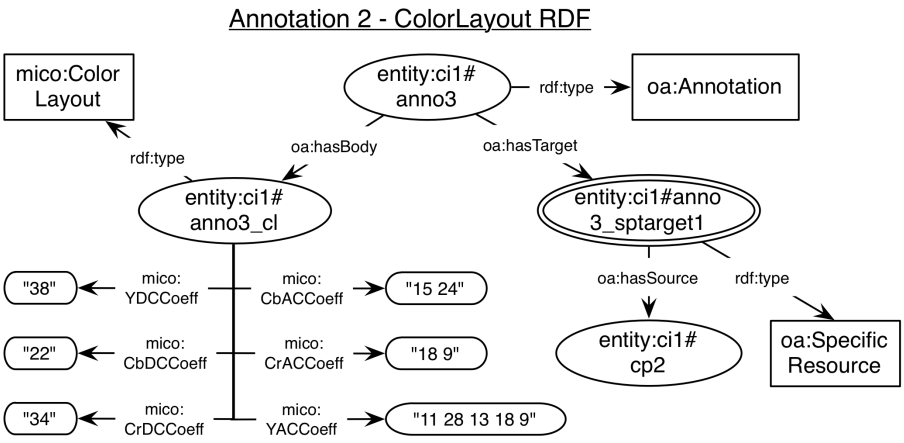


Figure 17 Provenance information of the result RDF of the ColorLayoutTriplifier

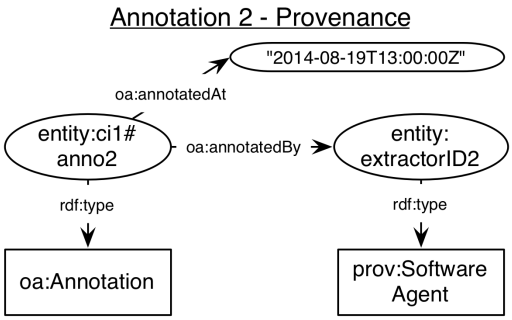


Figure 18 After the data corresponding to the first two steps has been stored, the ColorLayout feature step of the whole workflow has ended, and the area-based annotation is the next one. As this is already present in some kind of metadata form, it only has to be transferred into our metadata format and then be stored as well. The extractor connected to this undertaking is the AreaAnnotationTriplifier, its result will add the last part of the overall metadata graph in figure 12 at the resource `entity:ci1#cp4` and its result can be seen in this figure. The corresponding provenance information in figure 19.

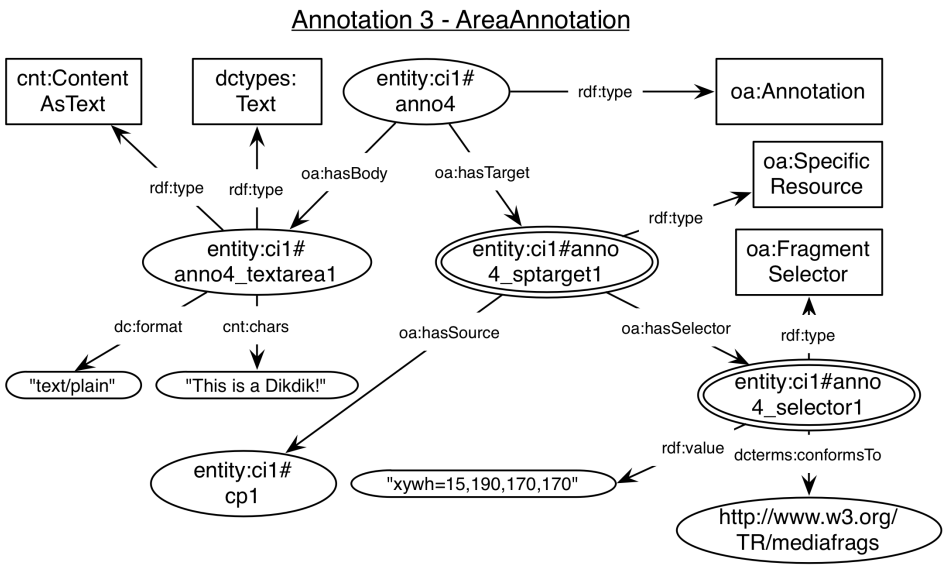
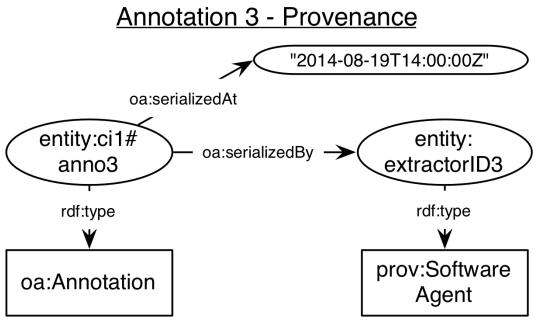


Figure 19 Provenance information of the result RDF of the AreaAnnotationTriplifier



5 Media Extractor Descriptions (WP2)

Overview: WP2 is providing the media extractors, which serve as a basis for respective work in WP3 (cross-media publishing), WP4 (cross-media querying) and WP5 (cross-media recommendation).

For this initial version of the deliverable, the focus is on the description of the first set of extractors (each of them represented by a technology enabler, or TE, as described in the requirements compendium), more specifically

- an overall description of the extractors, and possible related publications / implementations
- if applicable, interdependencies with other extractors
- relevant design choices, e.g. regarding the extraction process, implementation and feature storage
- evaluation criteria, which will be key for all further evaluation purposes, including subsequent measuring of improvements via combination of extractors
- input and output, which is then used to derive
- the respective annotation design (see WP3)
- a first assessment of the specific impact on querying (WP4)
- first ideas on the specific implications regarding recommendation (WP5)

It is clear that the provided information will be complemented over the course of the following projects, especially with respect to querying and recommendation implications: It is very likely that via further functional enrichment of the overall platform, and the addition of more content, many more possibilities will emerge.

5.1 Low level Feature Extraction - LFE (TE-201)

Introductory Description and Goal Features or descriptors are compact representations of audio or video signals. Being the first stage of an audio-visual analysis process they are both, an abstraction of the signal as well as a reduction of the signals dimensionality. They are then used for a variety of analysis tasks such as classification or matching. In MICO this technology enabler can be seen as more generic TE. The functionality towards a show case depends on the extractor chain to be used.

Relevant Publications and Implementations There are a number of possible implementations, both open source and proprietary. In the first MICO system version highly optimized extractors from Fraunhofer IDMT will be used since the output format combines well with the input to subsequent extractors. But there are also other implementations available such as OpenCV or OpenIMAJ. The features supported by the Fraunhofer extractor cover a wide range of descriptors both standard ones like the MPEG-7 descriptors (e.g. edge histogram, color layout etc.) as well as highly specialized SotA descriptors.

Dependencies and Interaction with other Technology Enablers As described above extracted auditive or visual low level features are used in many scenarios. In MICO we use them as input to the visual similarity TE, i.e. the video segment matcher 5.6 extractor. Other classification tasks are also possible although they do not have such a high priority as other TEs. Some TE combine the extraction of low level features and the task in which they are used into one extractor. The reason for this is, lies in both, software design and performance issues.

Design Choices The Fraunhofer feature extraction component uses the so called xpx system. The system works on a file basis and extracts a variety of features. The extraction process is highly configurable. The API is available in C++ and Java. If required by a show case due to time constraints it can be extended to a streaming data scenario. For the MICO system a wrapper for the C++ API will be provided.

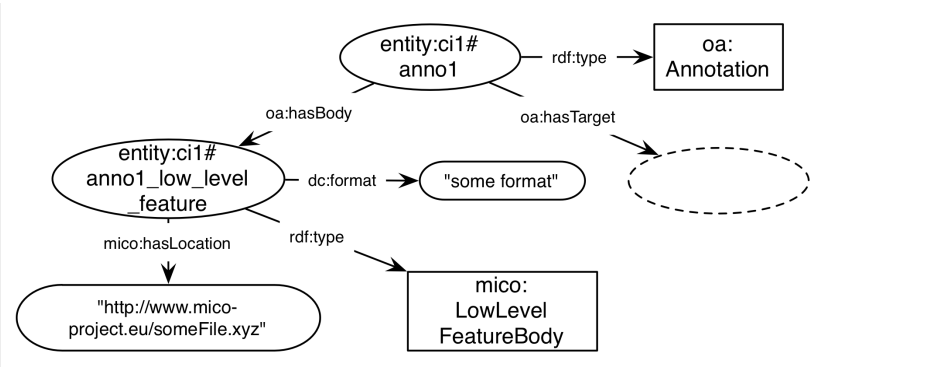
Quality of Service and Evaluation Criteria For pure low level feature extraction it is not possible to define criteria with respect to the overall extraction and query task. The results depend on the whole chain. In terms of non-functional requirements the ration between extraction time and media length is a comparative measure. Our aim is, that the extraction process should at least provide its service at a ration of 1:1 (i.e. real time) or better (e.g. 1:2).

Input and Output of the Relevant Components When using Fraunhofer xpx as basis for the extractor service, the input content part should be either an audio, image or video file. Since the extractor is generic, a meaningful configuration describing the extraction process must also be available. It is still under discussion how such extractor service specific configurations are handled within the Mico system. In this first version a pre-specified set of available extractor services will be used. The input for other extractors (e.g. OpenCV) will be restricted in terms of media type (e.g. only video or image).

The output of the low level feature extractor when using the Fraunhofer xpx component will a proprietary binary object containing some meta data about the descriptor and the feature coefficients.

Resulting Annotation Design For annotations with this kind of input, the information will only contain the binary file as it is, supported in an RDF fashion. As it is not yet known how to proceed with the results and consecutively the annotation, it is sufficient to safe the binary file. For binary files, the annotations will support a specific body type, its format, and the location of the file. An example can be seen in figure 20.

Figure 20 A basic generic annotation containing a binary file, exemplary for an extracted low level feature



The location of the file is supported by the relationship `mico:hasLocation`, the format by `dc:format` and finally the type of the specific body is modelled by `rdf:type`. In this case, the type of the annotation, and more specifically the body, is `mico:LowLevelFeatureBody`.

Impact on Multimedia Querying Within MICO scenarios were discussed and experimentally implemented that map the feature coefficients to the ontology model. While this might be useful for scientific querying and evaluation tasks (e.g. "return all descriptors where the first coefficient is below a specific threshold") in most other show cases this information is not needed. Thus in a usual scenario there will be no direct query at this semantic level. Thus meaningful queries cannot be done until the features are fed to subsequent extractor services.

Possibilities for Recommendation Just as in the querying case the usefulness of low level features coefficients for recommendation is very limited unless fed into further extractor services.

5.2 Face Detection and Recognition - FDR (TE-204)

Introductory Description and Goal The goal of this Technology Enabler is to build a system which reliably detects and identifies persons in visual footage (images and videos) based on their facial appearance. Face recognition systems are usually divided into three main parts: *Face and facial feature detection*, *alignment*, and *identification*. For face recognition in videos, however, once a face was automatically identified it has to be tracked through the video sequence. Thus, face recognition in video usually includes a fourth step, *face tracking*. This TE is relevant to US-12 (identify persons in videos) from the InsideOut show case.

Relevant Publications and Implementations For face and facial feature detection as well as multiple object tracking an external library called Sophisticated High-Speed Object Recognition Engine (SHORE^{TM}), developed by Fraunhofer Institute for Integrated Circuits (IIS), is utilized to localize faces in images and videos. SHORE^{TM}allows real-time robust detection and tracking of frontal faces. It utilizes a detection model comprising multiple consecutive classification stages with increasing complexity. Each stage comprises a feature extraction step and a look-up table based classifier built in an offline training procedure using Real-AdaBoost. Real-time capability is achieved by using simple and fast pixel-based features in the first stages and more sophisticated and therefore more complex descriptors in subsequent stages. A tracking-by-detection approach in combination with a Kalman-filter based algorithm is used within SHORE^{TM}to track detected faces through a video. For details the interested reader is referred to [KE06; EK11].

Face detection and tracking an affine transformation of the region of interest is performed based on the automatically detected eye and mouth coordinates. The face detection engine is based on Gabor feature extraction, feature space transformation using Locality Preserving Projections (LPP) [HN04] and a Sparse Representation Classification (SRC) [Wri+09] paradigm. For details the interested reader is referred to publications of Fraunhofer IDMT [Loo13; LE13].

Dependencies and Interaction with other Technology Enablers The FDR TE may benefit from TE that pre-process the image or video content in which the faces are about to be detected. TEs such as AVQ could help to increase detection quality.

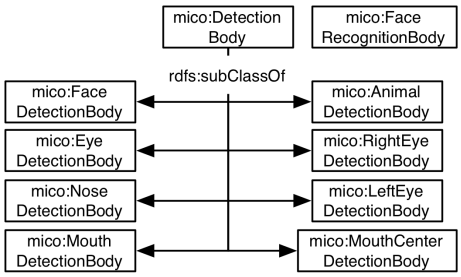
Design Choices Since face detection a a very narrow well researched task, detectors are usually self contained modules. In MICO an abstract C++ API for object detection and recognition will be used as interface while the underlying implementation can be different. Face recognition extractor results are provided by the same API. Internally the recognition uses the results of the detection as input for classification (i.e. labelling).

Quality of Service and Evaluation Criteria Evaluation of face detection is usually described in terms of precision, recall and the combining F-measure. Since result heavily depend on the application case and the content it is not easy to give a concrete figure as goal. With images a F-measure score of about 0.75 defines the state of the art algorithms. Since those algorithms always have parameters or giving a confidence value for the detection result another, more qualitative, way of evaluating a detector are so called ROC curves that describe the ratio between recall and precision under the change of the parameter.

Input and Output of the Relevant Components The input the FDR TE extractor is an image or video. The output is defined per frame. An extension modeling the whole video in one output content part (i.e. xml file) is considered in the future. The FDR extractor uses a simple annotation schema that describes an abstracts model for the purpose of object detection and recognition. The advantage of this is that the model can be used for any kind of objects that are detected in a arbitrary polygonal region of a 2D image, e.g. for animal detection 5.3. The hierarchy has been kept flat and the description of detected objects is separated from the labeling of those objects. Therefore the schema can be use for both, the result of detection task as well as the result of recognition (i.e. identification) tasks. The details of the model are described in appendix 9.1.

Resulting Annotation Design As the task of this extractor is twofold, detection as well as recognition, the aspect of annotation modelling will also be split into two annotation groups. Both will support fragment information, but the querying can be more efficient for the specific task. By supporting a class hierarchy of classes for the body and annotation, differences can be made in terms of what has been detected on the picture. This distinction is based on the possible outcomes of the extractors. The class hierarchy can be seen in figure 21.

Figure 21 The classes of bodies used for typing a detection or recognition based annotation

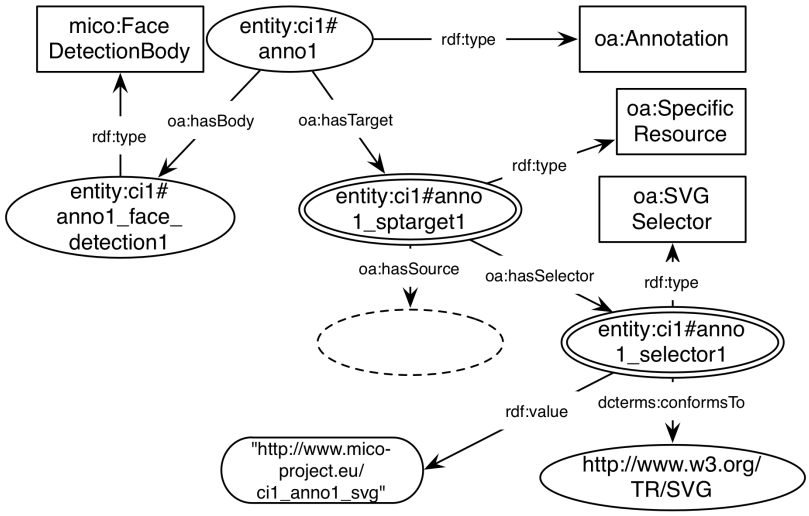


The type of the annotation depends on what has been detected by the specific extractor. For every `object` tag entry in the output, a detection annotation will be created. The positioning information will be saved as a fragment at the specific resource of the target. As this is supported as a polygon, we will make use of an SVG selector (the specification for SVG can be seen at [Dah+11]), which the OADM is conform to. The selector will be embedded directly via resolvable URI. There, the coordinates for the points of the polygon (or single point for right eye, left eye and mouth center detection) will be supported and can be used for further processes.

After the detection annotations have been created, if there are recognitions (which are supported by the label and mapping tags in the output), annotations will be created for those, too. The current

state only recognises faces, other object recognitions will be added incrementally. For a recognition annotation, the positioning is as important as for the detection, and it will be modelled in the same fashion as for detection via SVG selector. By doing this, the positioning information will be backed up by the name supported by the labels as well as a confidence value, which indicates how confident the extractor is about its result. Two annotations from the example of listing 15 modelling `OBJECTID1` in conjunction with its recognition of `LABELID1` are shown in figure 22, showing the detection, and figure 23 for the recognition.

Figure 22 Face detection annotation, depicting the first object of listing 15



The selector for the specific resource now is typed being an `oa:SVGSelector`, while the `dcterms:conformsTo` relationship specifies it to be conform to the SVG standard[Dah+11]. With the relationship `rdf:value` at the selector, the location of the SVG vector is stored. It can be resolved and then returns the following SVG conform vector. For figure 22 the vector would look like in listing 4.

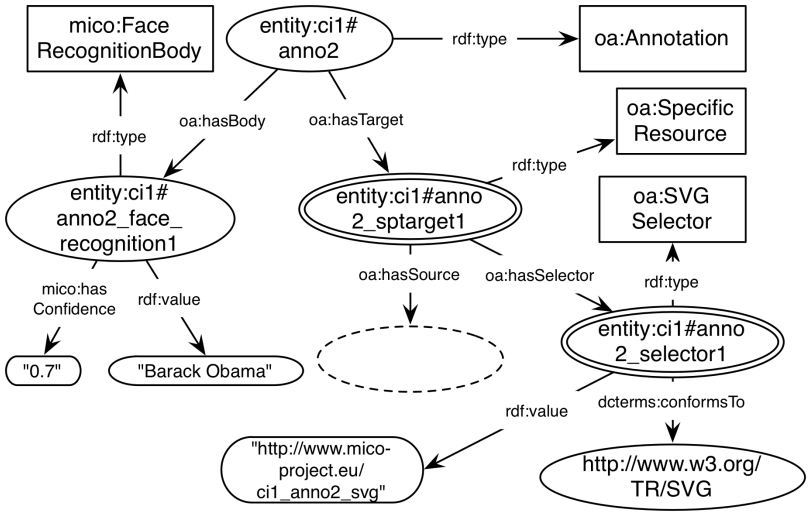
Listing 4: A simple SVG Vector for the Selector for figure 22

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="..." height="..."
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <!-- ... -->

  <polygon  points="0.2,0.3 0.2,0.5 0.4,0.5 0.4,0.3" />

  <!-- ... -->
</svg>
```

The polygon is described by a list of coordinates. If the face moves over the course of the video, the SVG vector will be enhanced by an animation that will implement the moving information.

Impact on Multimedia Querying Detection results can be used for query task which contain the constraint, that persons (i.e. faces) do (not) exist in images or videos. If recognition is used this can be

Figure 23 Face recognition annotation, depicting the label/mapping pair of listing 15



extended to incorporate specific person (e.g. actors) into a query. Combined with contextual information (e. g. named entity recognition) this can provide powerful queries.

Possibilities for Recommendation The existence of faces or specific persons in images or videos often signal a specific importance of that image or a part of the video and therefore could be candidates for recommendation. As for queries, combined with contextual information the recommendation might be better focused to a specific application scenario.

5.3 Object and Animal Detection - OAD (TE-202)

Introductory Description and Goal The objective of this Technology Enabler is to reliably detect objects - especially animals - in still images. The challenge here is that, in contrast to face detection, bodies of animals are highly deformable objects. Thus, the same object can take a variety of different appearances, depending on various extrinsic and intrinsic factors such as pose, viewing perspective, activity, and many more. An important requirement of the system is a low false negative rate in order to automatically sort out images which do not contain animals. The prioritized use cases to which this TE apply are US-31 (detection of no classifiable animals) and US-35 (animal number detection).

Relevant Publications and Implementations One of the most well known methods for generic object detection is the HOG-based detector [DT05]. Locally normalized Histogram of Oriented Gradients (HOG) descriptors provide excellent performance relative to other existing feature sets. First, a image pyramid is constructed where each stage represents the input image in different scales in order to detect objects in multiple resolutions. A sliding window approach with fixed detector size is then applied: In each position of the sliding window the HOG descriptor is first extracted and a linear SVM is utilized to classify if the object of interest is present in the particular region or not. Multiple detections are merged in a post-processing step using non-maximum suppression and a mean-shift algorithm based on Kernel

Density Estimation (KDE). Although this procedure is known to perform well for person detection, it was recently shown in [LS12] that this technique can be adopted in order to perform animal detection as well.

Dependencies and Interaction with other Technology Enablers The major task given by the Snapshot Serengeti show case is the detection of animals or, according to US-31 the detection of images without animals. For this purpose the extractor for animal detection does not need any other extractor. However, the results can be used in manifold ways. One idea is to present the empty images on purpose to keep users satisfied after complex annotation tasks. In this scenario, results from other extractor may (e.g. sentiment analysis) may be used get a better picture of the user in order to provide the correct recommendation.s

Design Choices Just as the Face Detection and Recognition TE (see 5.2), this TE uses the same generic API architecture and data model. It's implemented in C++ and a MICO API wrapper is provided. While the first implementations will focus on standard methods for people detection research efforts will be made towards deformable object or deep learning approaches.

Quality of Service and Evaluation Criteria Animal detection in images will be evaluated in terms of precision, recall and the combining F-measure. Since the detection of non-animals might be easier to accomplish than detecting specific kinds of animals we expect better results for the first. The algorithms will be tuned to have a higher recall rate at the cost of precision.

Input and Output of the Relevant Components The input to the TEs extractor when used for (non-)animal detection will be an image. It will use the detection parts same data model which has been used also by the FDR extractor (cmp. 5.2). The details of the model are described in appendix 9.1.

Resulting Annotation Design The annotation for animal- and object-detection is the same as for face detection (see section 5.2), only the specific class for the body is exchanged accordingly (see figure 21 for the classes).

Impact on Multimedia Querying There is little application to querying for the concrete task of non-animal detection.

Possibilities for Recommendation As described in the previous paragraph, the results of the non-animal detection will be used as recommendation basis for Snapshot Serengeti show case. If the detector will perform good enough for concrete animal detection, more detailed recommendations, e.g. according to users annotations preferences are imaginable.

5.4 A/V Error Detection and Quality Assessment - AVQ (TE-205)

Introductory Description and Goal The purpose of the AVQ TE is to deliver technical measurements of the quality of an image or video with respect to different kinds of quality artifacts. Those artifacts are caused by image and video coding steps (e.g. blocking, ringing) or failures during recording and post production (motion, blur, over-exposure, under-exposure, freeze frames, wrong field order). There are also audio quality measures (silence, clipping, etc.) although not as much as visual ones. The AVQ TE measures those and some more of these error without using a reference of the original video (if

existing at all). Within MICO the TE will be used to measure the quality of user generated content. The underlying user stories originates from the Inside Out show case US-21 (ranking and filtering based on quality). The TE is also be applicable to US-44 (removing artefacts such as bright stars and camera read errors) although currently no prioritized.

Relevant Publications and Implementations The Fraunhofer Broadcast Error Monitoring (BEM) framework is a collection of tools for measuring errors in audio, images and video. It will be used as central measurement tool for this TE.

Dependencies and Interaction with other Technology Enablers The AVQ TE can be generally used as pre-processing TE for successive TEs in order to increase the quality of visual (e.g. TE-204 FDR) and auditive (e.g. TE-207 SMD) extractors. For the IO-Showcase it provides direct ingest control for user generated content.

Design Choices The Fraunhofer BEM API is a ready to use C++ API that easily integrates with the MICO platform API. An exemplary wrapper has already been tested. As with an video and audio processing TE at this stage of the MICO project incoming streams are buffered and processed file based. Real stream processing can be integrated when required.

Quality of Service and Evaluation Criteria Since the perceived quality of a video is a subjective, application context dependent measure, only subjective evaluation could reliably asses this TE. Such experiments are quite elaborate and thus cannot be conducted within MICO. Therefore we are aiming to compare the results of the AVQ TE against established objective measurements (such as PSNR, SSIM). Although not perfectly aligned with human perception this will give at least some estimate of the TEs performance.

Input and Output of the Relevant Components The AVQ extractor accepts encoded video file containers as input content parts (such as avi, mp4) and outputs the annotation extracted from the video. The low level annotation model is rather simple and intended to be transferred to RDF triples by subsequent MICO extractors. The representation of the annotation is available either in json or xml format and specified as XML schema (see 9.2).

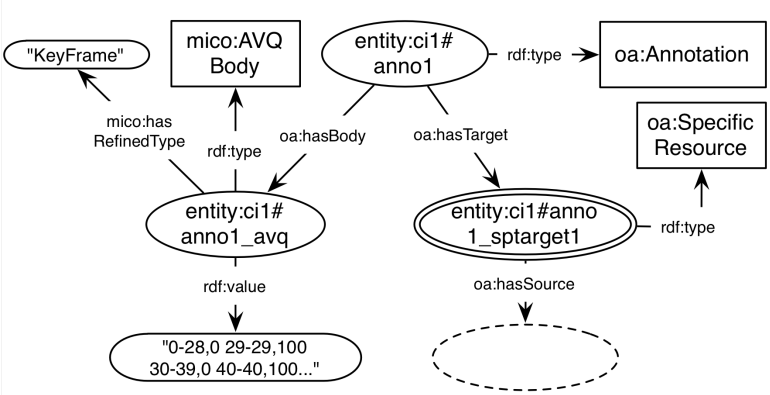
The measurement values are normed to ranges between 0 and 100 and given for every frame. While some results (blocking, blur) do have a continuous measurement space within this range some only return a true(100) or false(0) value (e.g. freeze, black bars). The ranges differ between error type and are *not comparable* to each other. If multiple successive frames contain the same value, they are combined to a frame range in order to reduce the amount of output data.

The interpretation of the measurement values is closely related to the application scenario. Special profiles are to be developed in order to represent a quality measurement value that corresponds to the expected human perception. One research task within MICO will be to correlate the weighted measurement values to objective (reference based) quality measures and the subjectively perceived quality within the particular show case (cmp. also previous paragraph).

Resulting Annotation Design For extractors that process videos and deliver ranges of frames or shots as result, the annotation is straightforward. The body will be typed according to the given extractor type (in this case `mico:AVQBody`) and it will support the extracted values via the `rdf:value` relationship directly at the body as well as the name of the module. Its content will be a whitespace separated list,

modelling every frame or frame range with its corresponding value (shape for one entry: `starting frame-ending frame,value`). The name of the specific analysis will be supported by the relationship `mico:hasName`. From the example given in listing 17, the annotation (covering only the frames from line 6 to 9) would look as illustrated in figure 24.

Figure 24 Annotation for AVQ result of example shown in listing 17



Impact on Multimedia Querying A/V quality can be used in queries when used in terms of simplified model such as a traffic light (green: best quality, yellow: acceptable, red: no go). In order to achieve such level of abstraction profiles for the application case of user generated content (see previous paragraphs) are needed which are under develop in MICO. Queries like 'Show me all video above acceptable quality' are then possible.

Possibilities for Recommendation The same thoughts as for querying can be applied to a recommendation paradigm. E.g. according to a specific user profile video with acceptable quality can be recommended. While this alone may not deliver the expected outcome, the combination with other measure of quality or interestingness (e.g. the quality of the commit message, comments or discussions about a video).

5.5 Temporal Video Segmentation - TVS (TE-206)

Introductory Description and Goal The Temporal Video Segmentation technology enabler is a C++ extractor able to find shot boundaries and key frames in shots for video sequences. US-07 t(thumbnails), US-19 (see, validate and edit segmentation of videos).

Relevant Publications and Implementations Within the first version of the MICO system a proprietary version provided by Fraunhofer IDMT will be used. For purposes of evaluation a free version is available.

Dependencies and Interaction with other Technology Enablers The TVS-extractor will be used for sequencing of video into their respective shots. These shots can be used directly by the higher

level application or processed by subsequent TEs / Extractors such as A/V Error Detection and Quality Assessment 5.4. The key frame extraction can be used directly for visualization purposes or by TEs such as the face detection 5.2. Moreover, the segmentation can be checked against audio cutting locations 5.8, thereby improving detection / localization accuracy.

Design Choices The TVS library brought into the project as background knowledge of Fraunhofer IDMT has been kept in its native C++ form. It's designed to analyze the base band of a video (i.e. raw pixel data). Internally it uses an adaptive majority voting approach based on different visual low level features extracted from the video.

A wrapper in order to include it into the MICO system prototype has been developed. In this early version the wrapper persists the whole incoming content part (i.e. the coded video stream) temporarily in order decode it. This makes development and deployment of an early running prototype possible. Future versions will use stream based decoding in order to parallelize the decoding and extraction work.

For decoding purposes the wrapper employs the open source ffmpeg library which is able to decode almost any video container and codec format.

Quality of Service and Evaluation Criteria As for all methods of detection, the shot detection can be evaluated in terms of precision, recall and F-measure compared to a comprehensive set of annotated ground truth. While hard cuts are easier to detect and to annotate, smooth transitions are still subject to research. The key frame extraction is a very subjective evaluation task, since the meaning of a key frame is 'a representative frame from that shot.' which is, obviously, hard to define by objective means.

Input and Output of the Relevant Components The TVS extractor accepts encoded video file containers as input content parts (such as avi, mp4) and outputs the annotation extracted from the video. In later version new content parts such as the key frame as image are also imaginable. The low level annotation model is rather simple and intended to be transferred to RDF triples by subsequent MICO extractors. The representation of the annotation uses the same model as AVQ (5.4)and is specified in appendix 9.2.

Resulting Annotation Design The annotations for the temporal video segmentation has the same shape as for AVQ (see section 5.4), only its body will be typed as `mico:TVS` accordingly.

Impact on Multimedia Querying Shot boundaries and key frames are extracted information about the video structure. Key frames encode importance on a semantically low level. In a query task shot information can be used to make search index entities smaller (e. g. searching only in shots instead of whole video files). Both, shot information and key frames may help to visualize query results in a better way. Anyhow the main purpose of this technology enabler is to server as pre-processor for other TEs.

Possibilities for Recommendation If shot detection is used as a measurement for the kind of editing, this information can be used for recommendation purposes.

5.6 Visual Similarity - VSI (TE-211)

Introductory Description and Goal The term "visual similarity" can be interpreted manifold for videos. It may describe the same mood two videos do represent, that the same kinds of objects or scene is are shown or that two image or videos do have the same recording source but are modified by basic

editing and encoding operations. This TE addresses exactly the latter case. It detect parts of a set of videos that do share content that originated from the same source but has been modified in some way. A good examples are exclusive news contents from some agency that are edited and used by a number of broadcasters. The goal of this TE is to find such re-used video content and provide the temporal relationships between the content parts. The TE is related to US-09 (finding perceptually similar items) and US-20 (annotate new video segments and extract topics).

Relevant Publications and Implementations For the purpose of sequence matching in video usually global spatio-temporal descriptors are used. An comprehensive overview of state of the are technologies has been given in deliverable D2.1.1. The video segment matcher (vsm) API of Fraunhofer IDMT implements matching algorithms and will be used for this TE. A modification toward more robust descriptors and matcher might be developed during MICO depending on the content used in particular show cases.

Dependencies and Interaction with other Technology Enablers In order to achieve competitive processing times, compact low level features as provided by TE-201 are required as input, and the frame rate of media container tag extraction (TE-227) can be used to match the frame rates between compared items here. Finally, for some A/V material, the matching video segments identified by this component can confirm cutting locations in the audio stream of the content (TE-224), thereby improving detection / localization accuracy, and vice versa.

Design Choices In order to find perceptually identical segments in videos, the extraction and matching processes have been separated into two extractors. Since extraction can be an elaborate task, its run only once and only the very basic version of an descriptor is extracted an persisted. Adaptation (e.g. quantization) of the descriptor are done during insertion into the in memory data base. The matching process is tuned for speed. If scalability will become an issue for MICO, the in-memory (one machine) solution might be extended to a share memory solution which could also benefit from massive parallelization.

Quality of Service and Evaluation Criteria Evaluation of the matching performance will be specified in terms of Precision, Recall and F-measure compared against a ground truth data set. Two way of identifying a match are possible: comparing start and end position with a given tolerance (usually the extraction step width) or measuring the overlap of matching segments to the ground truth. Since annotation of ground truth data for multiple (i.e. hundreds) of videos is almost impossible, within MICO, Fraunhofer developed scripts that synthetically create video that share segments. from a set of original videos.

Input and Output of the Relevant Components The underlying vsm implementation used in the first version of this TE is able to read the binary output of TE-201 (when the extractor implemented with xpx is used). Currently the matcher uses a in memory data base for ultra-fast access and matching operations. Each descriptor within this data base is referenced by an unique id (TUID) which is also stored within the binary low level feature. If the extractor is called in insert mode the low level feature is stored in memory (input binary low level feature for a specific video). If the extractor is called in query mode (input the query TUID and the TUIDs to match against) the matching is performed and a new content part with result of the operation as json (see snippets vsm_match_results.json according to

schema vsm_api_result_json_schema.json). The result contains the TUIDS of the descriptors representing the video content parts and the time stamps of the matched sequences. The time stamps are given in time base units w.r.t. 1 second.

Resulting Annotation Design The triplification for named visual similarity analysis will support an huge amount RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:VSI` and the corresponding format, supported by the property `dc:format`.

Impact on Multimedia Querying The relationships between videos sharing content from the same source can be used to extend the search space of an text based query by delivering video that are "visually" connected to the queried video. A content base search is also possible but it must be kept in mind that matching videos against large data bases requires a lot of computational power in order to achieve reasonable answering times.

Possibilities for Recommendation It could be possible to use this TE to detect e.g. news videos that share the same source material, reporting about related issues. Hence, it could be used to recommend material to related topics.

5.7 Speech Music Discrimination - SMD (TE-207)

Introductory Description and Goal The Speech Music Discrimination component is able to discriminate between audio segments that contain speech, music, speech and music, or silence. In MICO, this can be used to improve automatic speech recognition.

Relevant Publications and Implementations The component uses a huge set of features which is processed by Inertia Ratio Maximization using Feature Space Projection (IRMFSP) to select best features of this set. A Gaussian mixture model (GMM) is used to classify the segments, see e.g. [BHS10].

- Mathieu Barthet, Steven Hargreaves, and Mark B. Sandler. "Speech/Music Discrimination in Audio Podcast Using Structural Segmentation and Timbre Recognition." In: *CMMR*. Ed. by Sølvi Ystad et al. Vol. 6684. Lecture Notes in Computer Science. Springer, 2010, pp. 138–162. ISBN: 978-3-642-23125-4. URL: `http://dblp.uni-trier.de/db/conf/cmmr/cmmr2010.html#BarthetHS10` [BHS10]

Dependencies and Interaction with other Technology Enablers The output of this extractor can be used to improve the performance of the Automatic Speech Recognition (ASR) TE (5.12), enabling to operate only on segments that contain speech.

Design Choices The component represents existing background know-how of Fraunhofer IDMT, implemented in C++.

Quality of Service and Evaluation Criteria To evaluate the approach, a 5-fold cross-validation can be used. When this detector is configured to differentiate between 4 classes (speech, music, silence, speech+music), expected accuracy should be above 70%.

Input and Output of the Relevant Components This component receives an audio file or a video file with audio track as input, providing an xml file as output which lists the content segments with speech, music, speech+music and silence. The representation of the annotation uses the same model as AVQ (5.4) and is specified in the appendix 9.2.

Resulting Annotation Design See above, the annotations for speech music discrimination are the same as for AVQ (see section 5.4).

Impact on Multimedia Querying This TE could be used e.g. to limit the search space on content fragments that contain only speech, or only music, and then to apply specific queries related to the content types.

Possibilities for Recommendation At the moment, no relevant recommendation cases are foreseen.

5.8 Audio Cutting Detection (TE-224)

Introductory Description and Goal Audio recordings, and especially speech content, can convey important information about real-world events, and become increasingly available and relevant, thanks to the ubiquity of audio recording devices[13] and content distribution technologies / platforms. However, user-generated content can also easily be edited intentionally or unintentionally, thereby distorting its semantic meaning, which can create major problems, e.g. for journalistic uses of material.

The main goal of audio cutting / tampering detection approaches is therefore to determine whether a recording has been edited / tampered with or not, and to locate respective cuts, using different approaches:

1. The *Inverse decoder* detects changes in the underlying lossy codec[14], and/or in the framing grid applied by the lossy codec itself.
2. *Stable tone analysis* detects the presence of jumps and/or knees in the frequency trajectory of the target stable tone, or local non-linear behavior of the phase trajectory of the target stable tone.
3. *Microphone discrimination* detects a change in the audio recording device involved between two adjacent preselected intervals.

Relevant Publications and Implementations Fraunhofer IDMT can provide respective implementations to the aforementioned approaches. The most relevant publications related to that are:

- Daniel Gärtner et al. “Efficient Cross-Codec Framing Grid Analysis for Audio Tampering Detection.” In: *Audio Engineering Society Convention 136*. Apr. 2014 [Gä+14]
- Luca Cuccovillo et al. “Blind Microphone Analysis and Stable Tone Phase Analysis for Audio Tampering Detection.” In: *Audio Engineering Society Convention 135*. Oct. 2013 [Cuc+13b]
- L. Cuccovillo et al. “Audio tampering detection via microphone classification.” In: *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*. Sept. 2013, pp. 177–182 [Cuc+13a]

[13] including smartphones
[14] Currently MP3, AAC, MP3PRO and HE-AAC are supported.

- Sebastian Mann et al. “Combining ENF Phase Discontinuity Checking and Temporal Pattern Matching for Audio Tampering Detection.” In: *2. Workshop Audiosignal- und Sprachverarbeitung*. 2013 [Man+13]

Dependencies and Interaction with other Technology Enablers For some cases, the information about detected codec and coding bitrate traces from the inverse decoder can be used to cross-validate the information via media container tag extraction (TE-227), and vice versa. Moreover, for some A/V material, the audio cutting locations provided can be checked against segments identified by visual similarity / matching (TE-211) or temporal video segmentation (TE-206) in the video stream, thereby improving localization accuracy, and vice versa.

Design Choices The three components were implemented in C++. In order to ensure modularity, each component is independent from the other ones, and can be executed independently. The three outputs can be merged in a second step.

Quality of Service and Evaluation Criteria The evaluation of the components can be performed by means of standard statistic functions, such as precision and recall.

The *Inverse decoder* ACD (ID-ACD) and the *Stable tone analysis* ACD (ST-ACD) share a common formulation for True Positive (TP), False Positive (FP) and False Negative (FN) detection. For both components, a detected discontinuity *belongs* to a cutting point if it lies not farther than τ msec apart from the reference location of the latter; as a consequence, as stated in [Gä+14]:

1. A discontinuity is correct (TP) if it *belongs* to a cutting point present in the ground truth.

2. A discontinuity is not correct (FP) if it *does not belong* to a cutting point present in the ground truth.

3. A cutting point is not retrieved (FN) if no *belonging* discontinuity is detected.

More detailed information can be found in [Gä+14] for ID-ACD, and in [Cuc+13b; Man+13] for ST-ACD.

The *Microphone discrimination* ACD (MD-ADC) requires a different formulation, due to the fact that the MD-ACD is not directly detecting cutting points, but confirming or denying that adjacent intervals were recorded by using the same device:

1. A cut is correctly recognized (TP) if the neighboring intervals were recorded by different devices, and the MD-ADC detects a change of the microphones involved.

2. A cut is not correctly recognized (FP) if the neighboring intervals were recorded by the same device, and the MD-ADC detects a change of the microphones involved.

3. A cut is correctly rejected (TN) if the neighboring intervals were recorded by the same device, and the MD-ADC does not detect a change of the microphones involved.

4. A cut is not correctly rejected (FN) if the neighboring intervals were recorded by different devices, and the MD-ADC does not detect a change of the microphones involved.

MD-ADC as a stand-alone component can be evaluated by means of precision, recall, and accuracy. In addition to this, a Receiver Operating Characteristic (ROC) curve can be built, by varying a user-defined tolerance threshold: This threshold, defines the maximum dissimilarity necessary to classify two adjacent intervals as been recorded by different devices.

So far MD-ADC has been evaluated in the literature only jointly with ST-ACD in [Cuc+13b]. MD-ADC, however, is derived by an audio tampering detection method involving the same technique, but based on microphone classification[15]. A complete evaluation of the previous detector is reported in [Cuc+13a].

Input and Output of the Relevant Components The three independent components involved in Audio Cutting Detection (ACD), addressing respectively Inverse decoding, Stable tone analysis and Microphone discrimination, provide as output one XML file each. The three XML files were generated by a single XSD, and can thus be combined in a single XML.

The only input required by the *Inverse decoder* ACD is a WAV audio file. The current XML output of the extractor can be seen in listing 5:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AudioTamperingDetectionAnnotation xsi:noNamespaceSchemaLocation="ATDAnnotation.xsd"
                                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <GeneralProperties>
    <Duration>00:00:50.214</Duration>
    <Channels>2</Channels>
    <SamplingRate>44100</SamplingRate>
  </GeneralProperties>
  <InverseDecoding>
    <Segment>
      <Id>0</Id>
      <Start>00:00:00.000</Start>
      <Duration>00:00:18.800</Duration>
      <CodingTraces>
        <Codec>MP3</Codec>
        <FrameOffset>48</FrameOffset>
        <Bitrate>64</Bitrate>
      </CodingTraces>
    </Segment>
    <Segment>
      <Id>1</Id>
      <Start>00:00:18.800</Start>
      <Duration>00:00:09.200</Duration>
      <CodingTraces>
        <Codec>AAC</Codec>
        <FrameOffset>255</FrameOffset>
        <Bitrate>128</Bitrate>
      </CodingTraces>
    </Segment>
    <Segment>
      <Id>2</Id>
      <Start>00:00:28.000</Start>
      <Duration>00:00:22.214</Duration>
      <CodingTraces>
        <Codec>MP3</Codec>
        <FrameOffset>325</FrameOffset>
        <Bitrate>48</Bitrate>
      </CodingTraces>
    </Segment>
  </InverseDecoding>
</AudioTamperingDetectionAnnotation>
```

Listing 5: Example output of the Inverse Decoder ACD component

[15] Which has a lower applicability, since relies on a closed-set assumption with respect to the devices involved

The *Stable tone analysis* ACD requires a WAV input audio file, and the frequency in Hz of the target stable tone. The current XML output of the extractor can be seen in listing 6:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AudioTamperingDetectionAnnotation xsi:noNamespaceSchemaLocation="ATDAnnotation.xsd"
                                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <GeneralProperties>
    <Duration>00:00:50.214</Duration>
    <Channels>2</Channels>
    <SamplingRate>44100</SamplingRate>
  </GeneralProperties>
  <StableToneAnalysis>
    <Segment>
      <Id>0</Id>
      <Start>00:00:00.00</Start>
      <End>00:00:18.40</End>
      <Duration>00:00:18.40</Duration>
    </Segment>
    <Segment>
      <Id>1</Id>
      <Start>00:00:18.40</Start>
      <End>00:00:50.21</End>
      <Duration>00:00:31.82</Duration>
    </Segment>
    <Discontinuity>
      <Location>00:00:18.40</Location>
      <SourceFeature>Frequency</SourceFeature>
      <Threshold>1.0277</Threshold>
      <Value>2.04919</Value>
    </Discontinuity>
  </StableToneAnalysis>
</AudioTamperingDetectionAnnotation>
```

Listing 6: Example output of the Stable Tone Analysis ACD component

The *Microphone discrimination* ACD requires a WAV input audio file, an input segmentation for the test file, and a threshold value. A correlation value between each adjacent segment determined by the input segmentation[16] will be computed, and a discontinuity detected if the output correlation value is lower than the threshold. The current XML output of the extractor can be seen in listing 7:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AudioTamperingDetectionAnnotation xsi:noNamespaceSchemaLocation="ATDAnnotation.xsd"
                                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <GeneralProperties>
        <Duration>00:00:50.214</Duration>
        <Channels>2</Channels>
        <SamplingRate>44100</SamplingRate>
    </GeneralProperties>
    <MicrophoneDiscrimination>
        <Segment>
            <Id>0</Id>
            <Start>00:00:00.00</Start>
            <End>00:00:18.43</End>
            <Duration>00:00:18.43</Duration>
        </Segment>
        <Segment>
            <Id>1</Id>
            <Start>00:00:18.43</Start>
            <End>00:00:27.55</End>
            <Duration>00:00:09.12</Duration>
        </Segment>
```

[16]The input segmentation can be automatic, e.g., the one provided by the Inverse decoder or by the Stable tone analysis, or can be set manually by the user.

```
        <Segment>
            <Id>2</Id>
            <Start>00:00:27.55</Start>
            <End>00:00:50.21</End>
            <Duration>00:00:22.66</Duration>
        </Segment>
        <Discontinuity>
            <SegmentId1>0</SegmentId1>
            <SegmentId2>1</SegmentId2>
            <SourceFeature>CorrelationCoefficient</SourceFeature>
            <Threshold>0.99</Threshold>
            <Value>0.890839</Value>
        </Discontinuity>
    </MicrophoneDiscrimination>
</AudioTamperingDetectionAnnotation>
```

Listing 7: Example output of the Microphone Discrimination ACD component

The XML schema specifying the current output XML annotation is reported in the appendix, in Section 9.3.

Resulting Annotation Design The triplification for named audio cutting detection analysis will support an huge amount RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:ACD` and the corresponding format, supported by the property `dc:format`.

Impact on Multimedia Querying The segments that are detected by this TE can be used to limit queries to the identified audio segments, which can e.g. be used for cross-validation with temporal video segmentation (see dependencies), and to link segments to their different original sources.

Possibilities for Recommendation At the moment, no relevant recommendation cases are foreseen.

5.9 Media Container Tag Extraction (TE-227)

Introductory Description and Goal Many multimedia containers for A/V content include tags and technical information, which can be retrieved using media container tag extraction. The respective metadata can be used for user search / filtering, or to support subsequent analysis steps (e.g. via determining the frame rate).

Relevant Publications and Implementations The current implementation is based on the mediaInfo library [SAR02].

Dependencies and Interaction with other Technology Enablers The output can be used by other A/V extractors like segment matching (TE-211) to align frame rate / determination of the appearance time of a frame in a video sing the frame rate value extracted by this component. The information about codec and coding bitrate can be used to cross-validate the results of the inverse decoder approach for audio cutting detection (TE-224) for improved performance, and vice versa.

Design Choices The extractor service will be implemented in C++, as the underlying library mediaInfo is also developed in C++ and available on many platforms.

Quality of Service and Evaluation Criteria The detector should be able to extract almost all ID3 tags and the most relevant information about the applied audio and video codecs.

Input and Output of the Relevant Components This component can process several audio and video file formats and extract general data like the number and the length of audio or video tracks of an content part, as well as codec-specific information like bitrate and encoder settings. An example of the current XML output of the extractor can be seen in listing 8:

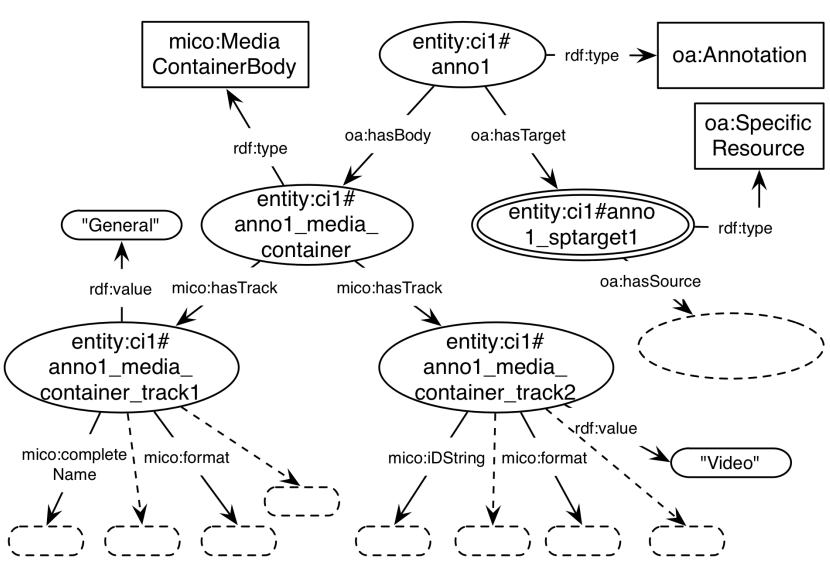
```
<?xml version="1.0" encoding="UTF-8"?>
<Mediainfo version="0.7.68">
  <File>
    <track type="General">
      <CompleteName>C:\tmp\nch\youtube_crawls\test.mp4</CompleteName>
      <Format>MPEG-4</Format>
      <Format_Profile>Base Media</Format_Profile>
      <CodecID>isom</CodecID>
      <FileSize_String>5.02 MiB</FileSize_String>
      <Duration_String>1mn 36s</Duration_String>
      <OverallBitRate_String>437 Kbps</OverallBitRate_String>
      <Encoded_Application>Lavf55.21.100</Encoded_Application>
    </track>

    <track type="Video">
      <ID_String>1</ID_String>
      <Format>AVC</Format>
      <Format_Info>Advanced Video Codec</Format_Info>
      <Format_Profile>High@L1.2</Format_Profile>
      <Format_Settings_CABAC_String>Yes</Format_Settings_CABAC_String>
      <Format_Settings_RefFrames_String>4 frame2</Format_Settings_RefFrames_String>
      <CodecID>avc1</CodecID>
      <CodecID_Info>Advanced Video Coding</CodecID_Info>
      <Duration_String>1mn 36s</Duration_String>
      <BitRate_String>429 Kbps</BitRate_String>
      <Width_String>320 pixel3</Width_String>
      <Height_String>180 pixel3</Height_String>
      <DisplayAspectRatio_String>16:9</DisplayAspectRatio_String>
      <FrameRate_Mode_String>CFR</FrameRate_Mode_String>
      <FrameRate_String>25.000 fps3</FrameRate_String>
      <ColorSpace>YUV</ColorSpace>
      <ChromaSubsampling>4:2:0</ChromaSubsampling>
      <BitDepth_String>8 bit3</BitDepth_String>
      <ScanType_String>Progressive</ScanType_String>
      <Bits_Pixel_Frame_>0.298</Bits_Pixel_Frame_>
      <StreamSize_String>4.92 MiB (98%)</StreamSize_String>
      <Encoded_Library_String>x264 core 140 r2377 1ca7bb9</Encoded_Library_String>
      <Encoded_Library_Settings>cabac=1 / ref=3 / deblock=1:0:0 / analyse=0x3:0x113 / me=hex / 8x8dct=1 /
        subme=7 / psy=1 / psy_rd=1.00:0.00 / mixed_ref=1 / me_range=16 / chroma_me=1 / trellis=1 /
        deadzone=21,11 / fast_pskip=1 / chroma_qp_offset=-2 / threads=12 / lookahead_threads=1 / nr=0 /
        decimate=1 / interlaced=0 / bluray_compat=0 / constrained_intra=0 / bframes=3 / b_pyramid=2 /
        b_adapt=1 / b_bias=0 / direct=1 / weightb=1 / open_gop=0 / weightp=2 / keyint=25 / keyint_min=2 /
        intra_refresh=0 / rc_lookahead=25 / rc=crf / mbtree=1 / crf=20.0 / qcomp=0.60 / qpmin=0 / qpmax=69 /
        qpstep=4 / ip_ratio=1.40 / aq=1:1.00 / scenecut=40 / cqm=0 / sliced_threads=0
      </Encoded_Library_Settings>
    </track>

    <track type="Audio">
      <ID_String>2</ID_String>
      <Format>AAC</Format>
      <Format_Info>Advanced Audio Codec</Format_Info>
      <Format_Profile>LC</Format_Profile>
      <CodecID>40</CodecID>
      <Duration_String>1mn 36s</Duration_String>
      <Duration_LastFrame_String>23ms</Duration_LastFrame_String>
      <BitRate_Mode_String>CBR</BitRate_Mode_String>
      <BitRate_String>1401 bps</BitRate_String>
      <Channel_s__String>2 channel2</Channel_s__String>
      <Channel_s__Original_String>1 channel1</Channel_s__Original_String>
```

```
      <ChannelPositions>Front: C</ChannelPositions>
      <SamplingRate_String>44.1 KHz</SamplingRate_String>
      <Compression_Mode_String>Lossy</Compression_Mode_String>
      <StreamSize_String>16.5 KiB (0%)</StreamSize_String>
    </track>

  </File>
</Mediainfo>
```

Listing 8: Example output of the Media Container Tag Extraction component

Resulting Annotation Design The annotations for this technology enabler include all tags supported in the XML output as its own RDF MICO vocabulary items, which will be a property with the respective name. Every track will represent an entity which supports its type in addition to its given content. An outline for the annotation conform to the example of listing 8 is shown in figure 25.

Figure 25 Exemplary annotation for the media container output of listing 8



Impact on Multimedia Querying The extracted tags can be used for queries related to technical parameters of material.

Possibilities for Recommendation The extracted tags can be used for suggesting to the users similar media content to those that he has already enjoyed. Under the assumption that users tend to like similar items to the ones liked in the past, the tags extracted can be used for feeding a similarity function for media content and suggest to the users the most similar content available in the platform.

5.10 Named Entity Recognizer (TE-220)

Introductory Description and Goal Named Entity Recognition (NER) classifies sequences of words in text into categories (so-called named entities), such as person, company names, organizations, locations, quantities, etc. The input to such a NER systems is unannotated text such as "I run some similar cameras in Arizona, and have had a well-endowed young man "flash" at the camera, but haven't seen any such exhibitionism on the Serengeti cams!" [17] and the output highlights pre-defined named entities, i.e. "I run some similar cameras in $[Arizona]_{location}$, and have had a well-endowed young man "flash" at the camera, but haven't seen any such exhibitionism on the $[Serengeti]_{location}$ cams!" A NER system is often a necessary first step for natural language processing tasks like document categorization and sentiment analysis.

Relevant Publications and Implementations Named entity recognition is integrated in Stanbol and also the Stanford CoreNLP tool[18] includes a statistical named entity recognition tool. The Stanford NER is a Java-based CRF (Conditional Random Field) classifier which allows training one's owns models.

The (Stanford NER) relevant publications are

- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling." In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 363–370 [FGM05]

Dependencies and Interaction with other Technology Enablers NER is technically and contentwise linked to sentiment analysis (TE-213), textual feature extraction (TE-215) and phrase structure parser (TE-217).

Design Choices The Stanford CoreNLP tools, which NER is a part of, generates for each input one XML file which is one of the projects preferred formats (see Section 2.2.3). There will be a release of a version of the Stanford CoreNLP package that outputs JSON.

Quality of Service and Evaluation Criteria The Stanford NER tool is open source and includes good named entity recognizer for English, in particular for person, organization and location entities. According to (Finkel et al. 2005) the statistical Stanford NER results in an error reduction of up to 9% over state-of-the-art systems on the CoNLL 2003 English named entity recognition dataset and the CMU Seminar Announcements information extraction dataset. An evaluation between Stanbol NER and Stanford NER might be of interest.

Input and Output of the Relevant Components The Stanford CoreNLP that also includes NER, creates for each input text one XML file. Figure 26 shows the XML output for the text "Young buffalo?" encoding phrase structure (see Section 5.11 for more detailed information on phrase structure), named entities and part of speech (POS). Part of speech is the linguistic category of words, such as verb, noun, preposition, etc. For example, in Figure 26 the POS for the noun "buffalo" is encoded in line 22 as `<POS>NN</POS>`, whereas no named entity is identified as encoded in line 23 with `<NER>O</NER>`.

[17] The text is a comment taken from Snapshot Serengeti discussion board

[18] The Stanford CoreNLP package integrates part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, the sentiment analysis, and the bootstrapped pattern learning tools.

Resulting Annotation Design The triplification for named entity recognition analysis will support an huge amount RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:NERBody` and the corresponding format, supported by the property `dc:format`.

Impact on Multimedia Querying Named Entity Recognition makes it possible to identify persons, company names, organizations, locations, quantities, etc. in text.

5.11 Phrase Structure Parser (TE-217)

Introductory Description and Goal A phrase structure parser provides the constituency relation of a natural language sentence. The input to such a parser is a natural language sentence such as "Are these young zebra on the ground?" and the output of such a parser is the syntactic structure of the input sentence. The syntactic structure of a sentence is often depicted as a tree illustrating the constituency relation, that is, the grouping of words into so-called phrases. The words "these", "young", "zebra", for instance, have the syntactic categories, *determiner*, *adjective*, *noun*, respectively and are grouped into a so-called *nominal phrase* (abbreviated as NP). Identifying the syntactic structure of a natural language sentence is often a first and crucial step for natural language processing tasks, such as automatic summarization, machine translation, document categorization, sentiment and discourse analysis, to name a few. Probabilistic parsers produce the most likely syntactic analysis for a given input sentence. These statistical parsers allow automatic and high-accurarcy parser output for natural language input.

Relevant Publications and Implementations We use the Stanford natural language parser as phrase structure parser. The Stanford parser is an open source parser and the Stanford parser package is a Java implementation of high-accuracy probabilistic context-free grammar (PCFG) parser, lexicalized dependency parser and a lexicalized PCFG parser. The phrase structure tree output of the parser can be displayed. The (Stanford parser) relevant publications are

- Dan Klein and Christopher D. Manning. "Accurate Unlexicalized Parsing." In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. ACL '03. Sapporo, Japan: Association for Computational Linguistics, 2003, pp. 423–430 [KM03]
- Richard Socher et al. "Parsing With Compositional Vector Grammars." In: *ACL*. 2013 [Soc+13]
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. "Generating typed dependency parses from phrase structure parses." In: *IN PROC. INT'L CONF. ON LANGUAGE RESOURCES AND EVALUATION (LREC)*. 2006, pp. 449–454 [MMM06]

Dependencies and Interaction with other Technology Enablers Phrase structure parser is technically and contentwise linked to question detection (TE-212), sentiment analysis (TE-213), textual feature extraction (TE-215), interactive wrapper generator (TE-218), named-entity recognizer (TE-220), and summarization (TE-221).

Design Choices The Stanford CoreNLP tools, which the parser is a part of, generates for each input one XML file which is one of the projects preferred formats (see Section 2.2.3). There will be a release of a version of the Stanford CoreNLP package that outputs JSON.

Quality of Service and Evaluation Criteria The Stanford parser is an open source, high-accuracy parser that is not errorless, but works in general well. The performance of the unlexicalized PCFG parser is 86.36% (labeled recall (LP), labeled precision (LR), F1 (harmonic mean of LR and LP) (see [Klein and Manning, 2003]). According to [Socher et al. 2013], the Stanford parser (starting at version 3.2) obtains an F1 score of 90.4%.

Input and Output of the Relevant Components The Stanford CoreNLP[19] that also includes the phrase structure parser, creates for each input text one XML file. Figure 26 shows the XML output for the text "Young buffalo?" encoding phrase structure, named entities (for more detailed NER information see Section 5.10) and part of speech (POS)[20].

For example, in Figure 26 the phrase structure of the sentence "Young buffalo?" is encoded in line 36 as `<parse>(ROOT(FRAG(NP(NNP Young))(NP(NN buffalo)).(.?)))</parse>`.

Resulting Annotation Design The triplification for phrase structure parser analysis will support an huge amount of RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:PSPBody` and the corresponding format, supported by the property `dc:format`.

5.12 Automatic Speech Recognition (TE-214)

Introductory Description and Goal Automatic speech recognition (ASR) consists in automatically transcribing what is being said in a video or audio stream. In MICO, ASR will be used in the 'News' showcase to make content searchable, to create links to related content, and to help generate recommendations. In particular, user story US-18 of showcase SC-02 reads "As a user, I'd like to search content items using automatic speech recognition".

Based on discussions with showcase owners InsideOut10, we can expect a very large dictionary of words, several simulatanelously active speakers, noisy material and sometimes also background sounds such as music, traffic, etc. With these assumptions in mind, we conducted a preliminary study to selet a suitable open-source library, and decided on Kaldi.

Kaldi is a C++ library that was originally designed for speech researchers but it is now starting to be used in transcription applications. The ambition for Kaldi is to be open-ended enough that different algorithms can be supported; a recent addition to kaldi is a neural-net library which is believed to be the state of the art algorithm at the present. In contrast to pocketsphinx, the model files for Kaldi are large (just under 1 GB), and its installation requires the numerical library ALPHA. Kaldi used to be supported for Windows, but is currently only guaranteed to build on Linux.

For the present, we ignore the problem of segmentation (that is, finding the most appropriate way of dividing the audio stream into smaller chunks to feed the ASR engine). We believe that additionnal effort on the segmentation side could improve word rate accuracy (WACC) by 2-3 percent. Due to time restraint, we also omit corrective feedback, which could improve the WACC by another 5 percent.paragraph

[19] The Stanford CoreNLP package integrates part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, the sentiment analysis, and the bootstrapped pattern learning tools.

[20] Part of speech is the linguistic category of words, i.e. verb, noun, adjective, preposition etc.

Figure 26 The XML output for the text “Young buffalo” created by Stanford CoreNLP encoding POS, parse structure and NER.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="CoreNLP-to-HTML.xsl" type="text/xsl"?>
<root>
 <document>
  <sentences>
   <sentence id="1">
    <tokens>
     <token id="1">
      <word>Young</word>
      <lemma>Young</lemma>
      <CharacterOffsetBegin>0</CharacterOffsetBegin>
      <CharacterOffsetEnd>5</CharacterOffsetEnd>
      <POS>NNP</POS>
      <NER>O</NER>
      <Speaker>PER0</Speaker>
     </token>
     <token id="2">
      <word>buffalo</word>
      <lemma>buffalo</lemma>
      <CharacterOffsetBegin>6</CharacterOffsetBegin>
      <CharacterOffsetEnd>13</CharacterOffsetEnd>
      <POS>NN</POS>
      <NER>O</NER>
      <Speaker>PER0</Speaker>
     </token>
     <token id="3">
      <word>?</word>
      <lemma>?</lemma>
      <CharacterOffsetBegin>13</CharacterOffsetBegin>
      <CharacterOffsetEnd>14</CharacterOffsetEnd>
      <POS>.</POS>
      <NER>O</NER>
      <Speaker>PER0</Speaker>
     </token>
    </tokens>
    <parse>(ROOT (FRAG (NP (NNP Young)) (NP (NN buffalo)) (. ?))) </parse>
   </sentence>
  </sentences>
 </document>
</root>
```

Relevant Publications and Implementations The ASR system Kaldi is described in:

- Daniel Povey et al. "The Kaldi Speech Recognition Toolkit." In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. IEEE Signal Processing Society, Dec. 2011 [Pov+11]

Within MICO, the following articles have been published on ASR:

- H. Björklund and E. Lundh. "Experiences from Development with open-source speech-recognition libraries." In: *MICO project homepage*. 2014. URL: `www.mico-project.eu` [BL14b]

- H. Björklund and E. Lundh. "Automatic speech recognition in media asset management." In: *Swedish Language Technology Conference*. 2014 [BL14a]

Dependencies and Interaction with other Technology Enablers Automatic speech recognition can be followed by e.g. Sentiment analysis (TE-213), and NER (TE-220), and textual feature extraction (TE-215).

Design Choices To adhere to the general design principles outlined in Section 2.2, the output of the ASR analyzer is stored in JSON-LD format. This keeps syntax and semantics separate (as suggested in Sec. 2.2.2), and is the simpler of the two serialization formats suggested in Sec. 2.2.3.

Quality of Service and Evaluation Criteria Word accuracy rate (WACC) needs to be upward of 60% to be useful for indexing purposes, to exceed 80% to be helpful for transcriptions.

Input and Output of the Relevant Components The input to ASR analyser is audio in WAW 16kb format. The output is a time stamped transcription in the srt format commonly used for subtitles (Figure 28), which within MICO can be published in JSON (Figure 29).

Resulting Annotation Design The triplification for speech to text analysis will support an large amount of RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:ASRBody` and the corresponding format, supported by the property `dc:format`.

Impact on Multimedia Querying By enriching video content with transcriptions, it is possible to search the content based on what is being said, and to navigate to the time points in the video where the utterances are made. The transcription needs not be completely accurate to be of use in this context.

Possibilities for Recommendation By processing the transcription further through e.g. named entity recognition or summarisation, it is possible to find key aspects of the video that can be used as a basis for generating recommendations.

Figure 27 A key frame from a video submitted for ASR. The subject matter is a presidential debate between Mick Romney and Obama, a type of material on which ASR usually performs well. In practise, only the audio track is passed to the ASR component.

5.13 Sentiment Analysis (TE-213)

Introductory Description and Goal In the Zooniverse Snapshot Serengeti and Galaxy Zoo showcases, one of the goals is to decide when to call the attention of the scientists to a post or series of posts. For this purpose, it can be useful to decide if there is a strong polarity to the discussion, e.g. a heated debate, in other words, to do a sentiment analysis on the text.

Sentiment analysis consists in establishing the writer's judgement, affective state, or intended emotional communication. The simplest form is perhaps polarity-detection, in which we are satisfied to separate between positive and negative statements.

The prototype implementation uses existing open source software and performs polarity-detection. Later versions will employ new techniques, involving phrase structure analysis, in order to improve accuracy. There will also be specialized versions, that target other polarities, such as scientific vs. non-scientific content and controversy vs. agreement.

Relevant Publications and Implementations The prototype implementation is done in Java and adapts software from the Stanford CoreNLP software library (`http://nlp.stanford.edu/software/corenlp.shtml`). This library is described in the following article:

- Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit." In:

Figure 28 The output of ASR analysis using Kaldi.

```
1
00:00:00,000 --> 00:00:05,000
THE SHOW ON THE ECONOMY THIS IS THEORETICALLY NOW IN SECOND SEGMENT
STILL ON THE ECONOMY

2
00:00:05,000 --> 00:00:14,000
AND THAT'S SPECIFICALLY ON WHAT TO DO ABOUT THE FEDERAL DEFICIT IN
THE FEDERAL DEBT AND THE QUESTION [NEEDS AND DEMANDED SUNDAY,BEING]

3
00:00:14,000 --> 00:00:23,000
[U,A]S AND YOU GOVERNOR ROMNEY AND YOU YOU GO FIRST BECAUSE FOR THE
ONE FOR US ON THAT MEN WANT TO RUN AND THE QUESTION IS THAT ONE OF
THE DIFFERENCES BETWEEN THE TWO OF YOU

4
00:00:23,000 --> 00:00:25,000
YOU AS TO HOW [HE WOULD ,YOU ]GO ABOUT
```

Figure 29 A snippet of an ASR file in JSON.

```
{"captions":[
{"duration":3000,"content":"This is really a two-hour presentation I give
to high school students,","startOfParagraph":true, "startTime":0},
{"duration":1000,"content":"cut down to three minutes.","startOfParagraph":
false,"startTime":3000},{"duration":3000,"content":"And it all started one
day on a plane, on my way to TED,","startOfParagraph":false,"startTime":4000},
{"duration":1000,"content":"seven years ago.","startOfParagraph":false,
"startTime":7000},
{"duration":1000,"content":"And in the seat next to me","startOfParagraph":
false,"startTime":8000},
{"duration":4000,"content":"was a high school student, a teenager,",
"startOfParagraph":false,"startTime":9000}, ... ]}
```

Figure 30 A snippet of an output file in JSON.

```
{"sentiments":[
{"content":"I am glad saw the movie.","polarity":0.7},
{"content":"I had waited for a long time to go.","polarity":-0.1},
{"content":"On friday I finally had the chance.","polarity":0.1},
{"content":"Unlike the last Marvel movie, this one lived up to my
expectations", "polarity":0.4},
 ... ]}
```

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014, pp. 55–60 [Man+14]

Dependencies and Interaction with other Technology Enablers The sentiment analyzer takes data that has been cleaned by a chatroom cleaner (TE-216, Section 5.14). Later versions, that employ syntactic analysis, will need to call a parser (TE-217).

Design Choices Java technology was selected because the Stanford CoreNLP library is defined in Java and Java is directly supported by the MICO framework. The component can be set to produce output either as text or in JSON-LD format. The latter is one of the projects prefered formats; see Section 2.2.3.

Quality of Service and Evaluation Criteria There is no absolute true answer when performing sentiment analysis. Often, two human beings can differ in their opinion about whether a certain statement should be interpreted as positive or negative. This may, e.g., be because they disagree as to whether the statement is to be read literally or ironically.

For most statements, it is relatively clear to humans whether they are positive, negative, or neutral. In order to test the quality of a sentiment analysis tool, we can therefore use a corpus of statements that have been analyzed and annotated by a human being and consider the software to be correct whenever it agrees with the human annotator.

Given such a corpus, the quality of service is measured using the F1 score, like described in 2.2.4. Our goal is to have a score of at least 80%.

Input and Output of the Relevant Components The component takes a table with texts cleaned by a chatroom cleaner (TE-216, Section 5.14) and outputs a table where each text has been assigned a numerical value, indicating the sentiment (see Figure 30). Due to large output sizes, the files will be stored in binary format.

Resulting Annotation Design As stated in the previous section, the output files of the sentiment analysis are very large, so the annotation will stick to modelling the storage of the corresponding binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:SentimentAnalysisBody` and the corresponding format, supported by the property `dc:format`.

Impact on Multimedia Querying Annotating chatroom data with the results of sentiment analysis makes it possible to search for discussions that are of particular interest, e.g., where opinions differ.

Figure 31 A snippet of an output file in JSON.

```
{"sentences":[
{"raw":"I am glad saw the movie ;-)","cleaned":"I am glad saw the movie."},
{"raw":"I had waited for a long time to go.","cleaned":"I had waited
for a long time to go."},
 ... ]}
```

Combined with interactive wrapper generation (TE-218, Section 5.15), such searches can become highly specialized.

5.14 Chatroom cleaner (TE-216)

Introductory Description and Goal The chatroom cleaner takes the textual data from Zooniverse chatrooms and "cleans" it, making it more suitable for processing by text analysis tools, such as parsers, named entity recognizers, and sentiment alalyzers. It can be set to simply remove smileys and standardize the text (for, e.g., parsing) or to extract stop words (for sentiment analysis).

Relevant Publications and Implementations The prototype implementation is done in Java and adapts software from the Stanford CoreNLP software library (`http://nlp.stanford.edu/software/corenlp.shtml`). This library is described in the following article:

- Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2014, pp. 55–60 [Man+14]

Dependencies and Interaction with other Technology Enablers The chatroom cleaner works as a preprocessor for TEs that take want raw natural language text as inputs, e.g., parser (TE-217), named entity recognizer (TE-220), and sentiment analyzer (TE-213, Section 5.13).

Design Choices Java technology was selected because the Stanford CoreNLP library is defined in Java and Java is directly supported by the MICO framework. Since the component works only with text, the input and output is pure text. This can be wrapped in XML or JSON, to bring it into one of the preferred project formats (see Sec. 2.2.3).

Quality of Service and Evaluation Criteria There are no easily quantifiable criteria for the quality of the chatroom cleaner. It should be easy to use and provide output that can be directly used by other TEs. In the first round of development, the quality of the cleaner will be assessed manually, by inspection of example input and output. In later stages, the usefulness of the output to other TEs will be evaluates in order to determine if improvements in quality are necessary.

Input and Output of the Relevant Components Both the input and the output to the component is pure text. In later versions, the output will be in JSON format and include sentence-by-sentence translations from raw text to cleaned text; see Figure 31.

5.15 Interactive Wrapper Generator (TE-218)

Introductory Description and Goal When annotated data is sparse, interactive learning can be a viable alternative. The main idea is to use the knowledge of a domain expert, e.g., the system administrator for a discussion forum, as efficiently as possible, since human time is normally a very expensive resource. The system expert annotates some data, but not necessarily very much. The annotated data is used to learn a query. The expert then views the results of running the query on a test data set and gives additional input to the learning system. This process is iterated until the expert is satisfied with the query result.

The main objective of this TE is to find out how well the inherent structure in web pages and XML documents interact with the structure of extracted media metadata from the web page elements. Can tree-shaped meta-data be usefully integrated into the overall tree structure to allow interactive learning of cross-media extractors to be efficient?

Relevant Publications and Implementations This TE is not yet developed, but a prototype, dealing only with pure text, is under development. It is described in the following publication:

- Henrik Björklund. "Interactive Learning of Syntax-Based Natural Language Queries." In: *Swedish Language Technology Conference*. 2014 [Bjö14]

Dependencies and Interaction with other Technology Enablers The interactive wrapper generator relies heavily on a natural language parser (TE-217). It can also make use of the output of other extraction tools such as named entity recognizers (TE-220), sentiment analyzers (TE-213, Section 5.13), and question detectors (TE-212).

Design Choices In accordance with the ambition of having a component-based system (see D6.1.1 [SF14]) and keeping syntax and semantics separated (see Section 2.2.2), the wrapper generator will act as a component, taking input and producing output. During the interaction with the user, the functionality of the component, i.e., the relationship between input and output, will change, but the formats it works with will stay the same.

The component will take JSON-LD, one of the projects prefered formats (see Section 2.2.3) as input and also produce JSON-LD as output. The exact format of the output will vary according to the choices of the user. Since the outputs will be large, they will be stored in a binary format.

Quality of Service and Evaluation Criteria As the name suggests, the interactive wrapper generator is supposed to be used interactively by a user. The goal is to automate the intention of the user, i.e., to construct a query in the form of an automaton that extracts the information the user is interested in. For this reason, the only reasonable measure of quality is user satisfaction.

In the first round of development, the intended users are system administrators at Zooniverse, and they will be the judges of whether the component has sufficient quality to be useful.

Impact on Multimedia Querying A high-quality interactive wrapper generator can enable users to make very complex and specific multimedia queries without the need for detailed technical knowledge of querying techniques.

5.16 Textual Feature Extraction (TE-215)

Introductory Description and Goal MICO aims towards cross-media analysis, in which many different aspects of a media asset are combined in classification and for annotation. To feed a high-level classifier, that uses e.g. visual and textual features, we suggest the use of an abstract form of tech enabler called High-Level Feature Extraction (HLFE). In essence, this is a family of custom built extractors. Each take as input an annoted media asset M and a set of features F, and outputs a document-feature vector representing how many times each feature in F was encountered in M.

Relevant Publications and Implementations Support vector machines (SVM) seem particularly useful for combining features from different domains. SVMs were first described in:

- Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks." In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125 [CV95]

Dependencies and Interaction with other Technology Enablers Question detection is depends on the chatroom cleaner (TE-216), and can export features from e.g. Question detection (TE-212), Sentiment analysis (TE-213), and NER (TE-220).

Design Choices In accordance with the guidelines of Section 2.2, the output of Textual Feature Extractors will be stored as JSON-LD. This separate syntax from semantics (see Sec. 2.2.2), and is on of the preferred project formats (see Sec. 2.2.3).

Quality of Service and Evaluation Criteria The HLFEs are simple extractors that basically filter out interesting aspects of previous analysis steps to make the result digestible for a classifier. As such, both accuracy and recall should be 100 percent, and the running time should be lower than bascially every other extractor.

Input and Output of the Relevant Components The input to a HLFE is an annotated asset, the output is a feature vector for that asset.

Resulting Annotation Design The triplification for named textual feature extraction will support a big amount RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:TextualFeatureBody` and the corresponding format, supported by the property `dc:format`.

Impact on Multimedia Querying By combining different aspect in analysis, queries can be made with respect to abstract properites such as "An esthetically pleasing picuture of a lion", rather than posing separate queries about what the image shows, and what the users in the forum thinks about the images.

Possibilities for Recommendation The same properities that make HLFEs useful for querying also make them good for recommendations: The allows to several different aspects of a media asset to be summarized with one label.

Figure 32 A snippet of an ASR file in JSON.

```
{"sentences":[
{"content":"I am glad saw the movie.","question":0.0},
{"content":"Please suggest what I should see next.","question":1.0},
{"content":"Did you for instance enjoy 'Gone Girl'?","question":1.0},
{"content":"But who knows if we like the same things?", "question":0.1}, ... ]}
```

5.17 Question Detection (TE-212)

Introductory Description and Goal The purpose of the TE Question Detection is to find questions in text. This can be useful for calling the administrator's attention to a forum dicussion or, together with further analysis, for classifying users by the kind of questions that they pose.

Relevant Publications and Implementations Sentence detection is integrated in Stanbol and the Open NLP library. Once a text has been segmented into sentences, we can easily find a large portion of the questions by simply looking for sentences that ends with a question mark.

Some relevant publications are

- Kai Wang and Tat-Seng Chua. "Exploiting Salient Patterns for Question Detection and Question Retrieval in Community-based Question Answering." In: *Proceedings of the 23rd International Conference on Computational Linguistics*. COLING '10. Beijing, China: Association for Computational Linguistics, 2010, pp. 1155–1163 [WC10]

- Helen Kwong and Neil Yorke-Smith. "Detection of Imperative and Declarative Question–answer Pairs in Email Conversations." In: *AI Commun.* 25.4 (Oct. 2012), pp. 271–283. ISSN: 0921-7126 [KYS12]

Dependencies and Interaction with other Technology Enablers Question detection is technically and contentwise linked to the chatroom cleaner (TE-216) and textual feature extraction (TE-215).

Design Choices In accordance with the guidelines of Section 2.2, the output of the Question Detection TE will be stored as JSON-LD. This separate syntax from semantics (see Sec. 2.2.2), and is on of the preferred project formats (see Sec. 2.2.3).

Quality of Service and Evaluation Criteria Quality of service is measured using the regular F1 score, that is: $2 \cdot precision \cdot recall/(precision + recall)$. Seeing that only looking at the presence of question marks gives us an F1 score of approx. 0.87, this can be seen as a treshold for reasonable quality.

Input and Output of the Relevant Components The input is a text file, the output is given as JSON-LD, e.g. in the form suggested in Figure 32. The format is simply a sequence of the sentences in the input file, as detected by the Apache Stanbol segmentation component, and for each sentence a value indication how strong a question it is (some questions may for instance be purely rhetorical).

Resulting Annotation Design The triplification for question detection will support a big amount RDF triples, therefore in the first version we will store only the binary file. This shape of annotation is similar to the annotations for low level feature extraction (see section 5.1), only its body will be exchanged with `mico:QuestionDetectionBody` and the corresponding format supported by the property `dc:format`.

Impact on Multimedia Querying The Question Detection tool is of course an appropriate analyser when the aim is to dectect questions in a forum debate.

Possibilities for Recommendation It may be interesting to see if other, related, questions have been answered elsewhere, and if so recommend the answer to the user.

6 SPARQL-MM Query Model (WP4)

In the last chapter we described the representation model for annotated multimedia content in detail. We use a semistructured format (namely RDF) which gives us the possibility to easily extend and adopt the model to current and upcoming use cases. With SPARQL we have a standardized query language for this kind of data structure, which is very powerful regarding expressivity but lacks specific features that are necessary for a proper multimedia (meta-) data retrieval, as we described in MICO Deliverable 4.1.1 (State of art in Cross-media querying). In this section we introduce SPARQL-MM as an extension for the de-facto standard query language in the Semantic Web world (SPARQL), which aims to bridge this gap by introducing special features like fulltext-search and spatio-temporal filter and aggregation functions.
The work in workpackage four has been aligned to the Technology Enablers we defined in MICO Deliverables 7.1.1 and 8.1.1 (Use Case Requirements). In the current state we fullfill:

TE-403: Support storage and retrieval for structured data We achieved this aim by introducing SAPRQL as query language for the MICO platform. SPARQL allows to store and retrieve data, which follows the data structure described in section 4.

TE-406: SPARQL 1.1 Support We achieved this aim by supporting the full SPARQL 1.1. specification in the MICO system, which includes full read-write support, novel path patterns, etc. To get a better efficiency we developed a SPARQL-to-SQL translation and implemented it in Apache Marmotta[21] and described in the MICO Deliverable 6.2.1 (System Architecture).

TE-402: Support fulltext search on structured datasets We achieved this by adding fulltext facilities to Apache Marmotta (the storage system of the MICO platform).

TE-401: Support query by spatio-temporal relationship We achieved this aim by extending SPARQL to spatio-temporal properties, relations and aggregations, which was the main effort in the first year of the project.

TE-405: Provide a SPARQL query interface We achieved this goal by developing, integrating and adapting *Squebi* [Kur14][22] as SPARQL query editor.

In MICO Deliverables 3.1.1 and 4.1.1 (State of art) we gave an overview on RDF, SPARQL 1.1, SPARQL extensions, and Multimedia Query Languages. We take this as prerequisite for this section, which mainly focuses on SPARQL extensions for spatio-temporal queries. The section includes:

- the description of the fulltext search support in Apache Marmotta,
- the introduction of basic object classes for spatio-temporal properties, relations and aggregations,
- the definition of spatio-temporal SPARQL functions for properties, relations and aggregations, and
- a wrap up of the implementation details of the first implementation iteration including an example use case.

The sample data and queries are related to MICO use cases, which gives the reader a proper understanding of how to use SPARQL-MM in real world scenarios.

[21] http://marmotta.apache.org/
[22] https://github.com/tkurz/squebi

Table 2 Namespace prefixes used in section 6

mo	http://linkedmultimedia.org/sparql-mm/ns/1.0.0/ontology#
mf	http://linkedmultimedia.org/sparql-mm/ns/1.0.0/function#
mm	http://marmotta.apache.org/vocabulary/sparql-functions#
ma	http://www.w3.org/ns/ma-ont#
dct	http://purl.org/dc/terms/
skos	http://www.w3.org/2004/02/skos/core#
xsd	http://www.w3.org/2001/XMLSchema#
animal	http://example.org/animal/

Through the section we use XML namespace prefixes defined in Table 2. In all RDF and SPARQL examples we take this prefixes as given, so we do not add additional prefix definitions. All RDF examples in this section are using the Terse RDF Triple Language (turtle) [Turtle] for serialization. The notation of SPARQL functions is aligned to the W3C specification document [HS13]. All functions that we introduce has an identifier that is composed as follows: 'F' {number_of_TE}-{type_of_function}-{alphabetic_serial} (e.g. *F402-SA-a*). The function types are defined in Table 3. A detailed description of all functions in human and machine readable format (following the sparql-service-description extension for describing SPARQL extensions and function libraries[23]) can be found on the source repository of our reference implementation[24]. Each spatio-temporal function is identified by a unique URI but all together share the same base URI `<http://linkedmultimedia.org/sparql-mm/1.0.0/function#>`.

[23] http://www.ldodds.com/schemas/sparql-extension-description/
[24] http://github.com/tkurz/sparql-mm/

6.1 Fulltext query and search in Apache Marmotta

Full-text search works over the literal values of nodes and differs from normal literal queries or regexp filters in that it applies language-specific lingustic processing (e.g. stemming and stop-word elimination). The Marmotta Full-text search currently offers two SPARQL functions that can be used in the FILTER part of a query and return boolean values (found or not found).

Function F402a: fulltext-search

```
xsd:boolean mm:fulltext-search (
   simple literal text
   ,simple literal query
   [,simple literal language-range]
)
```

This functions searches `text` for the words occurring in `query`, optionally applying the language-specific processing for the given language; query is a simple text literal (list of words) without any boolean connectors; words are AND connected, i.e. all words have to be found in the text for a successful match.

Function F402b: fulltext-query

```
xsd:boolean mm:fulltext-query (
   simple literal text
   ,simple literal query
   [,simple literal language-range]
)
```

This function searches `text` using the boolean query string passed in `query`, optionally applying language-specific processing for the given language; query is a boolean query string following the syntax used by PostgreSQL Note that full-text search is only available when using backend databases that support this functionality (currently only PostgreSQL and MySQL). Only PostgreSQL has real support for language specific processing. Also note that performance heavily depends on the availability of an appropriate full-text index in the database. The SPARQL module will automatically create full-text indexes for the languages configured in the Configuration used for creating the triple store.

With the following example we try to find all images that shows a lion by searching for the word *lion* occurring in the `ma:title` field of an image resource using the literal language of `ma:title`:

```
SELECT ?image ?title WHERE {
  ?image a ma:ImageResource .
  ?image ma:title ?title .
  FILTER( mm:fulltext-search(str(?title), "lion", lang(?title)) )
}
```

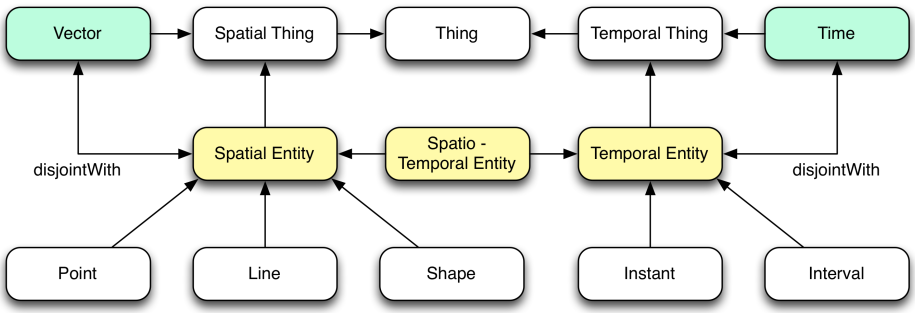
Listing 9: A fulltext query with Apache Marmotta

6.2 Class and Property Model for the spatio-temporal extension

SPARQL-MM defines some core classes that are necessary to describe spatio-temporal properties as well as relational and aggregational functions. There are some vocabularies which could be used here, complex ones like MPEG7 [MKP02] or very simple ones like Ninsuna [VD+14]. With our ontology we tried to get the tradeoff between expressivity and complexity. We are going to provide mappings for such existing vocabularies to our basic classes and give hints, how they can be extended to fulfill further requirements in the next iteration of SPARQL-MM. We just present some basic classes and properties in text that are necessary to describe the spatio-temporal functions we want to introduce. The ontology is currently in progress and might be adapted to upcoming issues. Figure 33 shows the main class model, a formal version can be found online under `http://linkedmultimedia.org/sparql-mm/1.0.0/ontology`.

We introduce two classes *mmo:SpatialEntity* and *mmo:TemporalEntity* (yellow) to describe spa-

Figure 33 SPARQL-MM basic classes and relations



tial as well as temporal instances and two classes (green) to describe the actual values of the instances, whereby *mmo:Vector* is a superclass of all classes of multidimensional vectors and *mmo:Time* is a superclass of possible time representations (e.g. Normal Play Time NPT [SRL98]). As you can see, they are disjoint with each other.

Spatial Entity

The class *mmo:SpatialEntity* is defined by the following:

```
mmo:SpatialEntity rdf:type owl:Class ;
  rdfs:label "Spatial Entity" ;
  rdfs:comment "A superclass of any spatial entities like
          point, line, polygone, curcle, etc." ;
  rdfs:subClassOf mmo:SpatialThing ;
  owl:disjointWith mmo:Time .
```

Listing 10: class Spatial Entity

Temporal Entity

The class *mmo:TemporalEntity* is defined by the following:

```
mmo:TemporalEntity rdf:type owl:Class ;
  rdfs:label "Temporal Entity" ;
  rdfs:comment "A superclass of any the temporal entity like instant, interval, etc." ;
  rdfs:subClassOf mmo:TemporalThing ;
  owl:disjointWith mmo:Time ;
  owl:disjointUnionOf (
    mmo:Instant
    mmo:Interval
  ) .
```

Listing 11: class Temporal Entity

Vector

The class *mmo:Vector* is defined by the following:

```
:Vector rdf:type owl:Class ;
  rdfs:label "Vector" ;
  rdfs:comment "A superclass for vectors." ;
        rdfs:subClassOf :SpatialThing ;
        owl:disjointWith mmo:SpatialEntity .
```

Listing 12: class Vector

Time

The class *mmo:Time* is defined by the following:

```
:Time rdf:type owl:Class ;
  rdfs:label "Time" ;
  rdfs:comment "A superclass for any kind of time specification." ;
  rdfs:subClassOf :TemporalThing ;
        owl:disjointWith mmo:TemporalEntity .
```

Listing 13: class Time

The ontology describes several subclasses and properties for non abstract entities. Figure 34 e.g. shows a class *Circle* with properties *hasXY* (describing the center point) and *hasRadius* (describing the radius). It should be mentioned that we abstract from real units (e.g. percentage, pixel, etc), which makes the model more flexible for function definition. A second example in Figure 35 shows a basic class *Interval* which has two defined relations to class *Time* (start and endpoint).

6.3 Extension Functions

In the following we consider 2 kind of functions, spatial and temporal, that are both again separated in 3 different types, namely relations, aggregations and features. A definition of these 6 types is given in

Figure 34 Sample object Circle

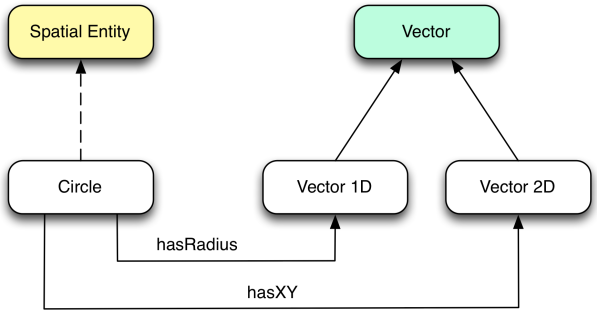


Figure 35 Sample object Interval

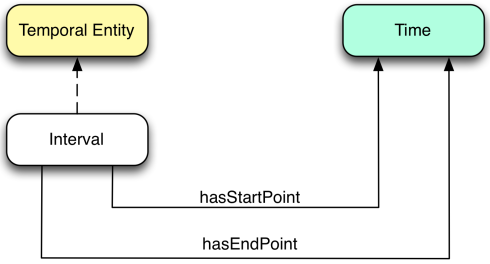


Table 3 SPARQL-MM Function Types

	Spatial
Relation	Type: *SR* how 2 spatial objects relate to each other (e.g. A right beside B)
Aggregation	Type: *SA* how 2 or more spatial objects can be aggregated with each other (e.g. intersection of A and B)
Feature	Type: *SF* features of spatial objects (e.g. area of A)

	Temporal
Relation	Type: *TR* how 2 temporal objects relate to each other (e.g. A before B)
Aggregation	Type: *TA* how 2 or more temporal objects can be aggregated with each other (e.g. intermediate of A and B)
Feature	Type: *TF* features of temporal objects (e.g. duration of A)

Table 3. It has to be mentioned that there might be overlaps between one or more types (e.g. spatio-temporal overlap), which we are going to discuss later. To provide the required functionality, we need to determine a model for spatial and temporal relations.

6.3.1 Types and models for spatial relations

Relations between spatial objects can be separated in three main classes, which are a) topological relations (how two spatial objects relates to each other, e.g. contains), b) directional relations (how a spatial object *a* relates to a spatial object *b* e.g. rightBeside), and distance relations (the attributes of the relation itself e.g. nearby). In the following we describe models for topological and directional relations and specify SPARQL-MM functions based on these models. Currently we do not consider distance relations, because in comparison to the topological and directional relations they are fuzzy and therefore does not seamlessly integrate into SAPRQL (unless e.g. extending SPARQL to fuzzy logic).

Topological Relations

A standard model to describe relations between spatial objects in a 2 dimensional geometric model is the Dimensionally Extended nine-Intersection Model (DE-9im) [CFO09]. The model is based on a 3x3

Figure 36 Clementini-Matrix

$$DE9IM(a,b) = \begin{bmatrix} dim(I_a \cap I_b) & dim(I_a \cap B_b) & dim(I_a \cap E_b) \\ dim(B_a \cap I_b) & dim(B_a \cap B_b) & dim(B_a \cap E_b) \\ dim(E_a \cap I_b) & dim(E_a \cap B_b) & dim(E_a \cap E_b) \end{bmatrix} \tag{4}$$

intersection matrix (Clementini-Matrix, Figure 36), which allows to specify the spatial relation of of two geometric objects according to interior (I), boundary (B) and exterior (E). The result of `dim(x)` is the maximum value of all matching intersection pattern, whereby `-1` is the value of $\emptyset$, `0` the dimension of point intersection, `1` the line intersection and `2` the dimension of area intersection. Additionally `*` represents a wildcard and indicates that the actual value does not have any influence on the current problem.

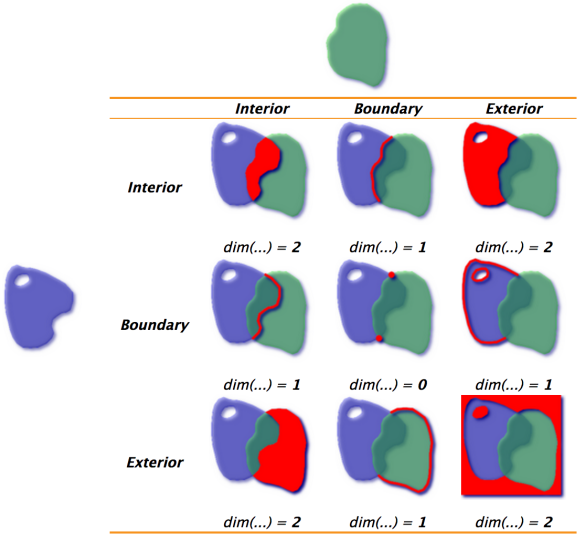
Figure 37[25] shows the matrix and the dimension results for two example shapes. To get a compact string representation it is common to concatenate the pattern values from left-to-top and from top-to-bottom. Hence, the resulting pattern string for the example is `212101212`. As the raw model is complex to use, a set of spatial predicates has been defined, which describes well known object relations. Each predicate is mapped to one or more matrix forms / pattern strings, so a clear semantic is bound to former fuzzy natural language terms. Therefore, the spatial predicates build a proper grounding for SPARQL-MM spatial relations. It is obvious that the pattern strongly depend on the dimension of the involved geometric objects. To reduce the amount patterns the resulting dimension is reduced to values $\{$`T,F,*`$\}$, if it does not change the underlying semantics.

For example, the *contains* predicate is described by the pattern `T*****FF*`, which means that:

(a) $dim[I(a) \cap I(b)]$ is true (a and b has an interior in common)

[25] `http://postgis.org/documentation/manual-svn/using_postgis_dbmanagement.html`

Figure 37 Example for DE9im



(b) $dim[E(a) \cap I(b)]$ is false (the exterior of a and the interior of b has nothing in common)

(c) $dim[E(a) \cap B(b)]$ is false (the boundary of b and the exterior of a has nothing in common)

(d) all the other fields of the matrix does not matter

In the following we list the spatial predicates with their patterns:

equals [T*F**FFF*]

disjoint [FF*FF****]

touches [FT*******], [F**T*****], [F***T****]

contains [T*****FF*]

covers [[T*****FF*], [*T****FF*], [***T**FF*], [****T*FF*]

intersects [T********], [T*******], [***T*****], [****T****]; logical inversion of disjoint

within [T*F**F***]; within(a,b) = contains(b,a)

coveredBy [T*F**F***], [*TF**F***], [**FT*F***], [**F*TF***]; coveredBy(a,b) = covers(b,a)

crosses [T*T******] for dim(a)<dim(b); [T*****T**] for dim(a)>dim(b); [0********] for dim(any)

overlaps [T*T***T**] for dim=0 or dim=2; [1*T***T**]for dim=1.

Now we are able to describe a topological function set for SPARQL-MM:

Function F402-SR-a: spatialEquals

```
xsd:boolean mf:spatialEquals (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-b: disjoint

```
xsd:boolean mf:disjoint (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-c: touches

```
xsd:boolean mf:touches (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-d: spatialContains

```
xsd:boolean mf:spatialContains (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-e: covers

```
xsd:boolean mf:covers (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-f: intersects

```
xsd:boolean mf:intersects (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-g: within

```
xsd:boolean mf:within (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-h: coveredBy

```
xsd:boolean mf:coveredBy (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-i: crosses

```
xsd:boolean mf:crosses (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-j: spatialOverlaps

```
xsd:boolean mf:spatialOverlaps (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Some of the methods are overloaded, which means that they are used for temporal as well as spatial relations. Therefore we added a prefix *spatial* to some of them.
In addition to the functions we defined a set of topological aggregation functions. As we currently only consider rectangle shapes, we took a reasonable subset, which has to be extended in the further work. This functions are:

Function F402-SA-a: boundingBox

```
lmo:SpatialEntity mf:boundingBox (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SA-b: intersection

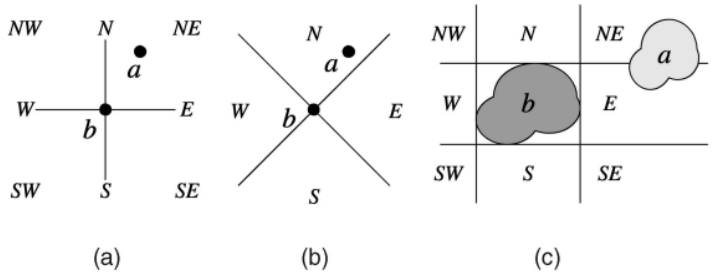
```
lmo:SpatialEntity mf:intersection (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

It has to be mentioned that the implementation of the two aggregation functions does not have to be delimited to two parameters but as both functions are assoziative, commutative and distributive they can be nested.

Directional Relations

Like for topological relations we have to find predicates for directional relations to specify proper functions for SPARQL-MM. Directional relations describe, how a primary object a is placed relative to a reference object b based on a coordinate system (for example, object a is south of object b). There are

Figure 38 Models of directional relations



several models, which describe directional relations in different spaces. Figure 38 (taken from [Ski+07]) shows 3 of the most common ones. 38a (projection-based model) and 38b (cone-based model) are defining relations between punctual objects but can be easily extended to spatial object by approximating an extended representative point (e.g. the centroid). Both models partition the space around the reference object *b* into a number of mutually exclusive areas. Other models like 38c (Projection-based Directional Relation model, PDR) extend the definition to spatial objects which provides more preciseness and expressivity but increases the number of relations that can be expressed (for PDR there are 511 possible relations), which disqualifies it as basis for directional predicates. We decided to took the projection-based model because a) it is easy to understand for users, b) it allows us to specify intuitive predicate names and c) it can be calculated very efficiently (e.g. by indexing the centroid for any spatial object). To make it even more intuitive, we refrain from using words from the geographical domain (*Modelname*) and replaced it with daily used words (name of *Function*) as follows:

Abbr.	Modelname	Function
W	West	$leftBeside(a,b) = a.x < b.x$
E	East	$rightBeside(a,b) = a.x > b.x$
N	North	$above(a,b) = a.y > b.y$
S	South	$below(a,b) = a.y < b.y$
NW	Northwest	$leftAbove(a,b) = leftBeside(a,b) \wedge above(a,b)$
NE	Northeast	$rightAbove(a,b) = rightBeside(a,b) \wedge above(a,b)$
SW	Southwest	$leftBelow(a,b) = leftBeside(a,b) \wedge below(a,b)$
SE	Southeast	$rightBelow(a,b) = rightBeside(a,b) \wedge below(a,b)$

Now we are able to describe a directional function set for SPARQL-MM:

Function F402-SR-k: leftBeside

```
xsd:boolean mf:leftBeside (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-l: rightBeside

```
xsd:boolean mf:rightBeside (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-m: above

```
xsd:boolean mf:above (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-n: below

```
xsd:boolean mf:below (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-o: leftAbove

```
xsd:boolean mf:leftAbove (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-p: rightAbove

```
xsd:boolean mf:rightAbove (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-q: leftBelow

```
xsd:boolean mf:leftBelow (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

Function F402-SR-r: rightBelow

```
xsd:boolean mf:rightBelow (
   lmo:SpatialEntity a
   ,lmo:SpatialEntity b
)
```

6.3.2 A model for temporal relations

Figure 39 Allen's 13 basic temporal relations

precedes	meets	overlaps	finished by	contains	starts	equals	started by	during	finishes	overlapped by	met by	preceded by
												
p	m	o	F	D	s	e	S	d	f	O	M	P

The standard model for temporal relation was introduced by Allen's interval algebra for temporal reasoning [All83]. The algebra defines thirteen basic relations between two time intervals, whereby a time point can be interpreted as a interval with duration 0. Figure 39[26] illustrates these relations. It has to be mentioned that 6 pairs of relations are converse, which is represented through upper-lower case letters (e.g. *precedes* p is converse to *preceded by* P). With this model we are able to describe a topological function set for SPARQL-MM:

Function F402-TR-a: precedes

```
xsd:boolean mf:precedes (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-b: meets

```
xsd:boolean mf:meets (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-c: overlaps

```
xsd:boolean mf:overlaps (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-d: finishedBy

```
xsd:boolean mf:finishedBy (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

[26] http://www.ics.uci.edu/~alspaugh/cls/shr/allen.html

Function F402-TR-e: contains

```
xsd:boolean mf:contains (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-f: starts

```
xsd:boolean mf:starts (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-g: equals

```
xsd:boolean mf:equals (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-h: startedBy

```
xsd:boolean mf:startedBy (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-i: during

```
xsd:boolean mf:during (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-j: finishes

```
xsd:boolean mf:finishes (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-k: overlapedBy

```
xsd:boolean mf:overlapedBy (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-l: metBy

```
xsd:boolean mf:metBy (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TR-m: precededBy

```
xsd:boolean mf:precededBy (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Like for topological relations we defined a reasonable set of topological aggregation functions, which are:

Function F402-TA-a: boundingBox

```
lmo:TemporalEntity mf:boundingBox (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TA-b: intersection

```
lmo:TemporalEntity mf:intersection (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

Function F402-TA-c: intermediate

```
lmo:TemporalEntity mf:intermediate (
   lmo:TemporalEntity a
   ,lmo:TemporalEntity b
)
```

It has to be mentioned that F402-TA-a and F402-TA-b can be extended to more parameters in the implementation, whereas F402-TA-c is restricted to two parameters.

6.3.3 Temporal and Spatial properties

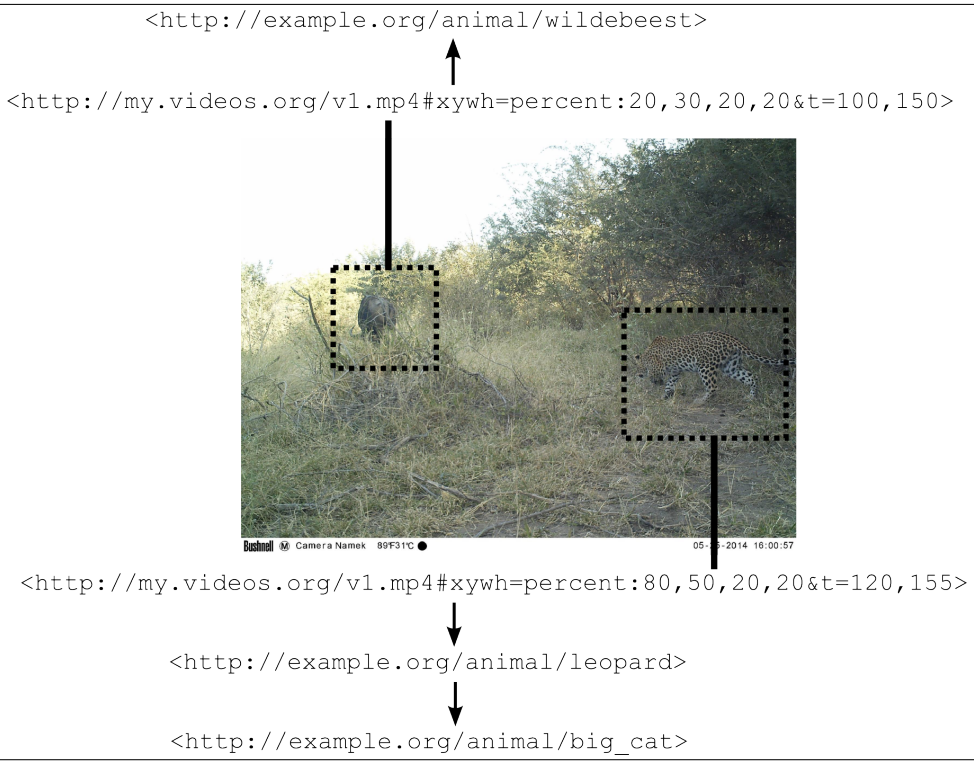
During the first year we focused on relational and aggregational functions. It is obvious that it is useful and necessary to support temporal and spatial feature functions (*SF* and *TF*) like $area(shape)$ or $duration(interval)$. This will be considered in the next iteration of SPARQL-MM.

6.4 Implementation Details and Example

Currently the implementation [Kur+14] supports Media Fragment URIs [Tro+12] for specifying Spatial and Temporal objects. This W3C specification provides a media-format independent, standard means of addressing media fragments on the Web using Uniform Resource Identifiers. It supports particular name-value pairs, e.g. `('t='start','end)` for temporal and `('xywh=',x','y','width','height)` for regional fragments. We currently support all functions that are mentioned above but continuously extend the function set (e.g. by adding spatio-temporal feature functions). The first iteration uses the OpenRDF Sesame[27] API and its extension interfaces, which makes it backend agnostic but requires expensive and inefficient in-memory calculations. We plan to improve this by a backend specific implementation for Marmotta Triplestore[28] in combination with SQL/MM [Sto03].

Figure 40 outlines an example of an annotated video showing a hunting scene in the Serengeti.

Figure 40 A sample for an annotated video



We use Media Fragment URIs to link annotations to specific spatio-temporal parts of the video. To keep it simple we do not use the expressive but more complex annotation model described in section 4. Of course SPARQL-MM is capable to query this structure, too. In our example a *wildebeest* appears from second `100` to `150` on the left side, while a *leopard* is marked from second `120` to `155` on the right side. If a user wants to retrieve the (spatio-temporal) snippet, that covers a specific hunting scene, she may issue

[27] `http://www.openrdf.org/`
[28] `http://marmotta.apache.org/kiwi/triplestore.html`

a query like: ”Give me the spatio-temporal snippet that shows a *big cat* right beside a *wildebeest*”. Using SPARQL-MM functions we can now formulate this need as a SPARQL query like in Listing 14. We use *mm:overlaps* to get fragments that appear in the identical temporal sequence. *mm:rightBeside* handles the spatial relation and *mm:boundingBox* merges every two fragments that match the filters. Figure 41 shows the resulting spatial box, the temporal interval starts on second `120` and ends on second `150`.

```
SELECT (mf:boundingBox(?l1,?l2) AS ?hunting_scene) WHERE {
    ?f1 a ma:MediaFragment; ma:locator ?l1; dct:subject ?a1 .
    ?a1 skos:broader animal:big_cat .
    ?f2 a ma:MediaFragment; ma:locator ?l2 .
    ?f2 dct:subject animal:wildebeest .
    FILTER mf:rightBeside(?l1,?l2)
    FILTER mf:overlaps(?l1,?l2)
}
```

Listing 14: A spatio-temporal query with SPARQL-MM

Figure 41 A resulting spatio-temoral fragment

<http://my.videos.org/v1.mp4#xywh=percent:20,30,80,45&t=120,150>

7 Cross-Media Recommendation Models (WP5)

7.1 Introduction

The main goal of WP5 is to investigate how the outcomes of WP2-WP4 (cross-media extraction, cross-media metadata publishing and cross-media querying) can be used for overcoming the state-of-the-art Content Recommendation approaches. The materialization of this goal should be the development of a Recommender Framework in MICO that will be based on recommendations models that have been proposed for covering all the recommendations related uses cases requirements.

Primarily, the framework will track the activity of the users in terms of their interactions with the content managed by MICO for suggesting unknown content in the platform that could be interesting for the user. The interactions data details vary depending on the context or showcase. For Zooniverse, the interaction data will consist on the users actions over the images or subjects: classification, comments, etc. For Shoof, interaction data can include contents like/dislike, ratings, contents inspection in a certain context, etc. The framework will analyze this data following the proposed models for providing different services like personalized content suggestions and content/users clustering.

As we will see in the following sections, certainly, WP2-WP4 outcomes can enhance the previous models mainly by providing more contextual metadata to the items and by increasing the expressiveness of the content querying and management. In the following three sections, the Recommendation Models that will be used in MICO are described. For each one, how to take advantage of the rest of work packages outcomes is explained. Also the uses cases that fit with the proposed model are identified. Finally, the last section describe all the Technology Enablers necessary for developing a Cross-Media Recommender framework in MICO that implements the proposed recommendations models. For each one, its role within the recommender is explained and implementation options are proposed.

7.2 Content-Based Recommendations

Content-based recommender systems learn to recommend items that are similar to those that the user liked in the past or those similar to their profiles. The similarity of items is calculated based on the features associated with the compared items. Each item is usually represented as a weighted vector of features, in which each component measures the ”importance” or ”informativeness” of the corresponding feature in the item content description. Global similarity functions can then, somehow, use this vectorial representation for calculating the similarity between items using, for example, the cosine of the composed vectors. Also, instead of using a global similarity scoring, a common practice is to let the user select over a set of features for retrieving similar items using only those. For instance, in order to suggest movies to an user, the system can ask the user to lookup for movies with the same genre, actors, director, etc.

Regardless the similarity approach, Content-Based Recommenders usually suffer of serious limitations precisely because of the lack of items' metadata quality and quantity, specially if the system is dealing with media based content. Manual annotation of media content is a tedious task and it becomes prohibitive when the number of items starts to growth in a significant way. This issue can be tackled in MICO by relying on WP2's extractors. WP2 will provide a set of media content analyzers that automatically can extract semantic and contextual metadata for better categorizing the content. This metadata will be used by WP5's Recommendation Algorithms for delivering meaningful suggestions.

The uses cases that are consistent with this model are US-27, US-30, US-37, US-43 and US-58 for Zooniverse showcase and US-02, US-04, US-05, US-08, US-16 for Shoof showcase.

7.3 Collaborative-Filtering Recommendations

Collaborative-Filtering approaches tend to cluster together users with similar tastes or preferences, under the assumption that users with common traits (in their demographic data, behaviour, taste, opinions, etc.) may enjoy similar items. Suggestions are based on similar users preferences. Once an user has been grouped within a certain number of similar users, recommendations consist on items that these users have consensually highly rated. In the literature, different approaches for clustering the users have been proposed. One of the main features to take into account for grouping the users are their profiles. For instance, users with a similar occupation, geographical location and so on, are supposed to enjoy similar things. Beyond users profiles, a strong indicator of users preferences are their opinions or stances on the items. Because of this, in Collaborative-Filtering approaches, user preferences are usually harvested in the form of items ratings. These ratings can be explicitly provided by the users (5-starts models, Like/Unlike, Numeric Values, etc.) or implicitly inferred by analyzing the user behavior. The ratings submitted by a user are used as an approximate representation of his tastes, interests and needs. These ratings are matched against ratings submitted by all other users, obtaining the user's set of "nearest neighbours" who finally will be used to base the recommendations.

Collaborative-Filtering algorithms can benefits from MICO technology in many ways. First of all, by better contextualizing the content, users can better decide on their preferences while they are browsing the systems., identifying what the content is about by the visualization of contextual metadata extracted from the content. In this way, users can early discriminate relevant content allowing more accurate ratings score. In the other hand, WP2's analyzers can be used for improving the rating schema. For instance, at Snapshot Serengueti, one of the Zoouniverse projects, users can not only interact with the subjects (classify animals images), but also comment and discuss about the images. Those comments can be analyzed in different ways for providing different feedback information. It is possible to perform Sentiment Analysis on the comments to check their polarity. If the comments are positive, we can certainly consider to increase the user rating for that image and the opposite if the comment is negative. Comments can be analyzed for categorizing the user type: experts users tend to use more technical vocabulary and/or provide more evidences about the images. To identify the user level of expertise can help the clustering users for grouping together more similar users and then deliver recommendations between them.

The uses cases that are consistent with this model are US-29, US-49, US-51, US-55, US-59 for Zooniverse showcase and US-03, US-10 and US-11 for Shoof showcase.

7.4 Ontology-Based Hybrid Approach

The use of domain ontologies in Recommender Systems has been also widely study in this research field ([CBC08]). The main advantages of using a domain ontology are two: the capability of inferring knowledge from the relations defined in the ontologies and the possibility of representing both users' preferences and items using the same concept space model defined by the ontologies. This model is an improvement of the previous ones by using a domain ontology as resource.

An ontology can be modeled as a semantic network of interrelated concepts and the user profiles can initially be described as weighted lists measuring the users' interests for those concepts. Merging all the information in the same knowledge graph, the links between users and concepts can be exploited for extracting relations among users according to common interests. The ontology-based representation is richer and less ambiguous than a content or item-based model and provides further formal, computer-processable meaning on the concepts. Furthermore, ontology standards support inference mechanisms that can be used to enhance recommendations, so that, for instance, a user interested in lizards images,

a subclass of reptiles, is also likely to enjoy other reptiles images like snakes.

Domain ontologies can be used also to improve users' preferences and items representation model. As items are represented as vectors of features in typical Content-Based approaches, this model can be extended by representing the items as weighted vectors of concepts from the ontology. Users' preferences can be also represented in the same way, since when the user rate an item is showing interest in all the concepts that represent that item. Furthermore, preferences' vectors can be extended by spreading the interest in a concept with related concepts within the domain ontology. This technique is known as *Constrained Spreading Activation* ([CK87]). Figure 42 illustrates how it works. The expansion is self-controlled by applying a decay factor to the intensity of preference each time a relation is traversed. Therefore, the user can have weighted preferences also for initially unknown concepts. The expansion can be also controlled by traversing only through certain properties between the concepts, in order to avoid noisy representations.

Figure 42 Constrained Spreading Activation with a decay factor of 0,26

7.5 Technology Enablers

7.5.1 TE-501. User Activity and Context Monitor

Introductory Description and Goal All Recommendation models are based on user activity in the past. Therefore, the first thing any recommender system needs is a tool for tracking, harvesting and storing users' behavior and users-items interactions. The source, amount and structure of the information that has to be collected is totally dependent of the use case. For example, certain web applications

collect the data while the user is browsing and store it in a persistence system for later feeding the recommendation system. It is also quite common to rely on log files parsing for gathering the data. In the other hand, not all the systems can provide exactly the same type of information. Users' profiles data can, of course, change from one application to another. For enabling collaborative filtering approaches, items' ratings data has to be provided, but sometimes it is either not possible/convenient to force the users to give explicit ratings or there is not straightforward way to inference them.

Since it is clear that data preparation is a custom issue of each final system, the goal of this technology enabler is to provide services and resources for representing and storing items and users' preferences data in an homogeneous and standard way within the MICO platform. Item's metadata structuring and storage is already covered by WP3, therefore, users' preferences data structuring and storage should be consistent with WP3 guidelines in order to enable to process them in the same way for recommendations. In summary, this technology enabler will ease showcases' systems to feed the MICO platform with users' preferences data.

Dependencies and Interaction with other Technology Enablers As the monitoring service in the platform will be storing the users' preferences as item's and user's metadata using RDF, it will need to use the outcomes of TE-403 (Support storage and retrieval for structured data), where an API for **Metadata Storage** is expected to be provided.

Similarly, as the data is stored in a triplestore, TE-406 (Marmotta SPARQL 1.1 support) outcomes can be used for querying it and provide SPARQL based recommendations.

Design Choices The main outcome will be a **Activity Monitoring API** implemented as REST services integrated in the MICO platform. The API will be very simple and will allow integrators to send users' preferences data in a concrete fixed format. The data structure used for representing users' preferences must fit with the proposed recommendation models for WP5 but also must be consistent with the representations foundations in MICO. For those reasons, a reasonable approach is to use an ontology for representing users preferences. Between several options, the best vocabulary we have identified for our purposes is the **Review ontology** ([Rev]), already described in section 4 (**Preliminaries and Basics for the Metadata Model**) and summarized here in the following table:

Term	URI	Description
Comment	http://purl.org/stuff/rev#Comment	A comment on a review.
Feedback	http://purl.org/stuff/rev#Feedback	Feedback on the review.
Review	http://purl.org/stuff/rev#Review	A review of a content.
commenter	http://purl.org/stuff/rev#commenter	The commenter on the review.
hasReview	http://purl.org/stuff/rev#hasReview	Associates a work with a a review.
hasComment	http://purl.org/stuff/rev#hasComment	Used to associate a review with a comment on the review.
hasFeedback	http://purl.org/stuff/rev#hasFeedback	Associates a review with a feedback on the review.
maxRating	http://purl.org/stuff/rev#maxRating	A numeric value.
minRating	http://purl.org/stuff/rev#minRating	A numeric value.
positiveVotes	http://purl.org/stuff/rev#positiveVotes	Number of positive usefulness votes (integer).
rating	http://purl.org/stuff/rev#rating	A numeric value.
reviewer	http://purl.org/stuff/rev#reviewer	The person that has written the review.
text	http://purl.org/stuff/rev#text	The text of the review.
title	http://purl.org/stuff/rev#title	The title of the review.

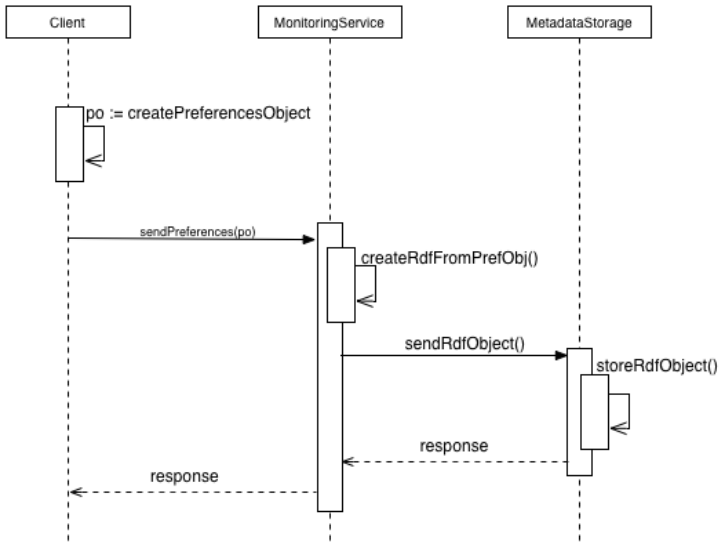
totalVotes	http://purl.org/stuff/rev#totalVotes	Number of usefulness votes (integer).
type	http://purl.org/stuff/rev#type	The type of media of a work under review.

The API will allow final systems to send data structured according to this ontology. In this context, a possible workflow for feeding the platform with users' preferences data could be the following:

- **Data Preparation**: clients have to collect somehow interactions between users and items. The interaction data must be wrapped into an object respecting the Review Ontology model. Typical information to be collected are: reviewed item (*ItemID*), reviewer (*UserID*), review comments, review rating, etc. This object doesn't need to be formatted necessarily using RDF. Others formats like JSON or XML can be used for easing the representation in the client side and the API can support them.
- **Data Import**: isolated Review's objects or a set of them can be sent to the MICO platform using the **Activity Monitoring API**.
- **Data storage**: finally, at the platform side, the data is transformed (if needed) to RDF and stored using the **Metadata Storage** API. It is important to note that the reviewed items and the reviewer users are supposed to be recorded in the platform previously, therefore this storage stage implies a linkage of analyzed items in the platform with users reviews.

This workflow is specified as a sequence diagram in the Figure 43.

Figure 43 Sequence Diagram for Content Review Metadata Storage



Quality of Service and Evaluation Criteria Depending on the use case, the **Monitoring API** could expect to be operating in a data intensive or streaming environment. Therefore, the main non-functional requirements and quality aspects should be related with the data management:

- **Scalability**: implemented services for collecting the preferences data should offer a good performance independently of the amount of data received. For instance, at Shoof's showcase, the amount of data can growth exponentially with the number of users registered in the system. All the interactions between the users and the content would be sent through the Monitoring API and all this information must be persisted at the MICO storage system, which the main component is a triplestore. Triplestores are not well suited for storing and managing real-time or streaming data. Therefore, a heuristic for efficiently flushing the data must be applied.

- **Freshness**: newly submitted preferences data should be taking into account by recommendations algorithms as soon as possible. This requirement is quite coupled with the previous because actually it implies to make the data available in the platform as soon as possible.

- **Consistency**: the API will be exactly the same for any system using it, independently of the kind of data this system manages. All the data will be mapped to the same ontology and will be treated in the same way by the recommendation algorithms.

7.5.2 TE-502. User Similarity Calculator

Introductory Description and Goal To cluster similar users is a critical step in any Collaborative Filtering approach, where item's suggestions are based precisely on the preferences of the most similar users to a given one. Most of the identified uses cases related to recommendations in MICO request this feature. For grouping together similar users, a variety of Clustering algorithms have been proposed in the context of Recommender systems ([AT05]). As in any other clustering algorithm, one of the main aspects here is the distance or similarity function. Generally, a distance function within a clustering algorithm determines how close are two items in a certain space. The algorithms then, tend to group together those items that are more similar to each other than to those in others groups.

When it comes to recommendations, a distance function defined over users always try to measure or predict how similar are their tastes. Three kind of features are normally computed by user's distance functions:

- **User Ratings (User Behavior)**: there is a set of algorithms in the literature that process users-items interactions to calculate the similarity between users under the assumption that users who tend to like/dislike items in the same way are more similar between them.

- **User Profiles**: users in the same range of age, with common hobbies or interests or living in the same area tend to have the same tastes.

- **Contextual Information**: like location, time, interactions with other users, etc.

User Similarity functions are partially use case dependent in MICO. For computing user ratings information, state-of-the-art approaches can be used, but this information has to be combined with custom use cases' data. For instance, at Snapshot Serengueti, subject's comments analysis have to be taken into account for grouping together experts users. The goal then is to provide a Framework for easily integrate and combine this custom data into a baseline User Similarity approach based on classical user's preferences computation.

Relevant Publications and Implementations Various approaches have been used to compute the similarity between users in collaborative recommender systems. In most of these approaches, the similarity between two users is based on their ratings of items that both users have rated. The two most popular approaches are *correlation* and *vector-based*. In the correlation-based approach, the similarity between items is given by their *correlation* which measures the linear relationship between objects (how they tend to appear together). While there are several correlation coefficients that may be applied, the *Pearson correlation* is the most commonly used. In the vector-based approach ([BKR08]), the two users are treated as two vectors in m-dimensional space, where m is equals to the number of items that both users have rated. Then, the similarity between two vectors can be measured by computing the cosine of the angle between them or by calculating the Euclidean distance between the vectors.

Several other measures have been proposed to compute similarities between users or items. One of them is the *Mean Squared Difference* ([SM95]), which evaluates the similarity between two users u and v as the inverse of the average squared difference between the ratings given by u and v on the same items.

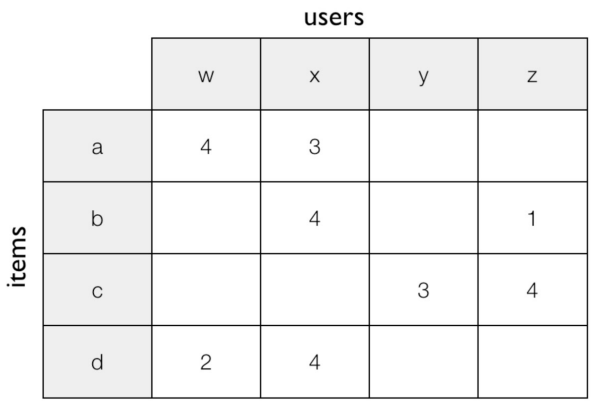
Dependencies and Interaction with other Technology Enablers TE-501 (User-Activity and Context-Monitor) is the only dependency, which means that it is only necessary to have the data in place. Low-level access to the triplestore will be also necessary in order to take all the preferences data for feeding the clustering algorithms.

Design Choices *Correlation* or *vector-based* implementations are nowadays provided by many open-source Recommender systems like Apache Mahout ([Mah]), Prediction.io ([Pre]) or MyMediaLite ([Med]). All these implementations use Ratings data for computing the similarity between users, using boolean or numeric values as rates. Also, most of these frameworks allow to define custom or complementary domain-dependent similarity functions. Independently of the selected framework for the final implementation, it is important to expose an API for easing the inclusion of custom data for calculating the similarity between users. Two possible cases are expected:

- **Client Custom Rating Models**: state-of-the-art distance functions for recommender systems work well with classical Ratings models where the interactions between users and items are weighted by a rating value explicitly provided by the user or implicitly calculated based on user behavior. Most of the MICO uses cases are not consistent with explicit ratings. One possibility is to force integrators to always define their Rating model and only provide final rating values. In this way, it is not necessary to extend the recommender framework which eventually could be a complex task. In that way, the data collected through TE-501 will modeled by a matrix similar to the one shown by Figure 44 and classical correlation or vector-based functions will be used.
- **Custom Distance Functions**: on the other hand, for others use cases, it doesn't even make sense to use distance functions based on ratings data and user similarity must be defined in terms of users' profiles, skills or behavior. For instance, as explained above, for grouping together experts users at Snapshot Serengueti, it is necessary to analyze things like the used vocabulary or the classification skills.

Quality of Service and Evaluation Criteria In general, the evaluation of Recommender systems is a subjective task and completely dependent of the application domain. Regarding user similarity functions, the evaluation criteria is even more subjective because it is difficult to establish an empirical

Figure 44 Users-Items Ratings Collaborative Filtering Matrix



measurement of how much an user is similar to another user, being then complicated even for humans to build a gold standard for evaluation.

7.5.3 TE-503. Item Similarity Calculator

Introductory Description and Goal For both Content-Based and Item-based Collaborative Filtering approaches, an item similarity function is mandatory. Under the assumption that users tend to like items that are similar to others items that the user have liked in the past, the item similarity or distance functions in Content-Based recommender systems are used to directly deliver new items suggestions to users. The similarity of items is calculated based on the features associated with the compared items. In the other hand, Item-based Collaborative Filtering approaches, predict the rating of an user for an item based on the ratings of the user for similar items. In such approaches, two items are similar if several users of the system have rated these items in a similar fashion.

In MICO, items consist mainly of media content. When the information has no structure (e.g. text) some kind of pre-processing step is needed to extract structured relevant information. That is supposed to be the responsibility of WP2's extractors: to analyze the content for extracting relevant items' features. These features are actually descriptors of the content that provide meaningful details indicating what the content is about. Item Similarity functions in the context of MICO recommenders must explode this information for providing more accuracy similarity scores for the items.

Relevant Publications and Implementations Most content-based recommender systems use relatively simple retrieval models, such as keyword matching or the *Vector Space Model* (VSM) with basic *TF-IDF* weighting in the case of text or weighted features vectors for other media content. VSM is a spatial representation of content. In that model, each item is represented by a vector in a n-dimensional space, where each dimension corresponds to a term/feature from the overall vocabulary/set of features of a given item collection. Within this model, many similarity measures have been derived to describe the proximity of two vectors; among those measures, cosine similarity is the most widely used.

In the other hand, as in User-based Collaborative Filtering, for Item-based approaches, it is also possible to use *vector-based* or *correlation* distance functions.

Dependencies and Interaction with other Technology Enablers Item similarity functions accuracy depends mainly on the amount and quality of the items' extracted features. In this sense, there could be a dependency with those extractors from WP2 that can be applied to each of the items. In the other hand, the recommender could be querying the MICO backend for retrieving item's metadata. Therefore, SPARQL related technology enablers like TE-406 makes a dependency.

Design Choices As stated in section 4 (**Preliminaries and Basics for the Metadata Model**), all the extracted features from the content analyzed in MICO will be stored as triples following the specified aforementioned **Metadata Model**. These features will be part of the items' vectors that will feed the similarity functions. There are some issues to be addressed in order to develop efficient similarity functions:

- **Feature Selection**: the set of features must be consistent along all the items for the same use case. Noisy features should be discarded and the rest of them should be weighted for all the items. Therefore, an administrator user should be able to customize the recommender by properly selecting the set of features that the system has to use.
- **Similarity Matrix Calculation**: at "recommendation time", the recommender should have access to precomputed similarity values between all the items. Because real-time calculation is an unrealistic strategy because of performance reasons, nightly calculations of the similarity matrix is a common choice for both, content-based and collaborative filtering approaches.
- **Content-Based and Item-Based CF combination**: Item-Based CF distance functions can be enhanced with content-based similarity functions. Different combination strategies can be found in the literature ([BH04], [MMN02]).

Quality of Service and Evaluation Criteria For evaluating Item similarity functions it is possible to apply methods typically related with Information Retrieval like *Precision/Recall* and *Mean Average Precision*. For performing this kind of evaluation, every single item can be used as queries. For each query, a ranking of most similarity items can be obtained, providing a list of items sorted by their similarity scores. The relevance of the results should be judged by an expert in the domain where the recommender system is being applied.

7.5.4 TE-504. Cross-Modal Content Recommender

Introductory Description and Goal The users and items **Similarity Functions** along with the preferences data collected through the **Monitoring API** will be finally used for enabling the required recommendations algorithms for fulfilling all the WP5 uses cases. The Cross-Modal Content Recommender will consist on a **Recommender Framework** implementing the described Recommendations Models for MICO. This framework will be integrating the described similarity functions within the proper recommendations algorithms depending on the implemented model.

The Recommender Framework will be used through an API for recommending contextual media content. This API will contain a set of services covering the following functionality:

- Retrieve related/similar content to a given one.
- Suggest personalized content to an user based on their profiles and their activity in the systems.
- Get a list of most similar users to a given one.

- Inspect users' preferences

This functionality is actually intentionally coupled with the different proposed recommendations models. To retrieve related or similar content is the materialization of a content-based approach that uses the defined item similarity functions for finding the similar and unknown content for an specific user. For those uses cases where the content analysis is not enough and user behavior has to be taken into account, an implementation of the collaborative-filtering model can be used. For example, at US-59 (As a Zooniverse admin, I'd like to be able to recommend different projects to volunteers based on their previous experiences) Zooniverse's users will be grouped together based on their interactions with the subjects (if they tend to classify the same images, if they have discussed the same images, if they have the same expertise level, etc.). Once the users have been clustered, it is important to provide a service for inspecting the clusters. In this way, a Zooniverse application can access to the similar users to a given one and recommend him those projects where the most similar users have been working actively in the past.

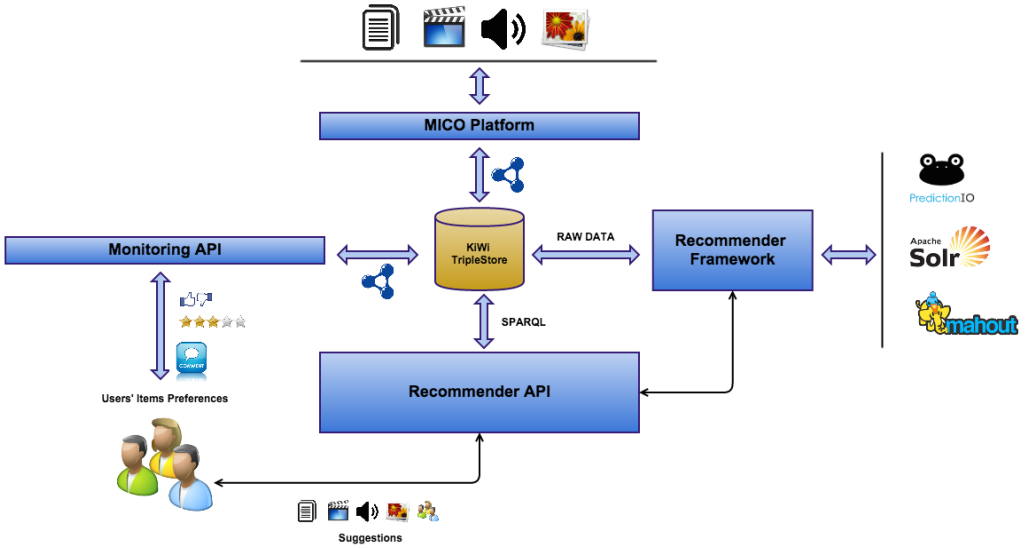
Relevant Publications and Implementations The literature is rich on approaches for both content-based and collaborative filtering models. Regarding content-based models, the trends have changed from directly delivering similar content from a similarity matrix to use Machine Learning techniques for learning User Profiles ([BP96]). The problem of learning user profiles can be cast as a binary text categorization task: each document/item has to be classified as interesting or not with respect to the user preferences. The classifier is trained using the items that the user has liked in the past as positive examples and the ones that the user has not liked as negative examples. As training features, the extracted item's features are used. Typical Machine Learning algorithms like Naïve Bayes can later be applied.

In the other hand, collaborative filtering methods can be grouped in the two general classes of neighborhood and model-based methods. In neighborhood-based collaborative filtering the user-item ratings stored in the system are directly used to predict ratings for new items. As explained in the previous technology enablers, this can be done in two ways known as user-based and item-based recommendations. In contrast to neighborhood-based systems, model-based approaches use the ratings to learn a predictive model. The general idea is to model the user-item interactions with factors representing latent characteristics of the users and items in the system, like the preference class of users and the category class of items. This model is then trained using the available data, and later used to predict ratings of users for new items.

Dependencies and Interaction with other Technology Enablers The **Recommender Framework** depends, in one way or another, on all WP5 technology enablers. First of all, it will be using the data collected through the **Monitoring API**. User Similarity functions will be used for supporting Collaborative Filtering models. Item Similarity functions will be used for supporting both, Collaborative Filtering and Content-Based approaches.

Design Choices To design and implement a Recommender Framework in the context of MICO involves to address a set of technical issues mainly related to how to manipulate the data and make it works within any available recommendation framework in the market. Figure 45 shows an architecture for the Recommender Framework in the context of the MICO architecture. First, it is necessary to put the data (items + preferences/reviews metadata) available for the final recommendation system. Initially, all the data will be stored in a triplestore provided by Apache Marmotta. Depending on the framework used for the implementation, different data importers would need to be implemented for taking the data

Figure 45 MICO Recommender Framework Architecture



out of the triplestore and place it in the proper place. Because SPARQL could be quite inefficient for crawling all the data, a native access to the Marmotta database would have to be used. Then, this data needs to be converted to the formats supported by the destination. Any Collaborative-Filtering approach would have to use the data for training the recommender from time to time. Therefore, the data is processed in a batch way, not in real time. The most typical strategy in this case is to perform the training in the nights and provide to the Recommender Framework a model using the data collection snapshot at the previous day.

For the MICO Recommender itself, depending on the model, different frameworks or tools can be used for supporting the implementation. For Content-Based models, because of its native ability to manipulate vectors, the most straightforward implementation could be based on *Apache Solr* ([Sol]). Among other facilities, Apache Solr provides a feature called *MoreLikeThis* that suggests the user similar content to the one the user has selected. Usually applied to text by using TF-IDF, *MoreLikeThis* is easily customizable for other kind of content. It is only necessary to store the extracted content's features as document's fields in Solr and define a similarity function or let the vector-based model in Solr to do the work. In terms of data management, to use Solr implies to permanently index and/or store the item's metadata in any of the file systems supported by Solr.

As exposed in the architecture, simple SPARQL queries can be also used for looking up for similar content. Similarity functions can be directly implemented as custom SPARQL functions or SPARQL-MM can be directly used for searching content with the same or approximate multimedia features. This option would suppose the least development effort because the main part of the work is already done by the platform by storing in RDF all the items metadata.

Finally, it is possible to opt for a Machine Learning based recommender by learning user profiles using the user preferred items. Any Machine Learning framework like Weka, Apache Mahout, scikit-learn, etc can be used. All these systems normally also needs to (temporally) store all the training data for building the models.

Regarding the Collaborative-Filtering model, there are a couple of good candidates. Apache Mahout ([Mah]) is an open source project which goal is to build scalable machine learning libraries. It provides a variety of Classification, Clustering and Recommendations algorithms ready to work on top of a MapReduce environment like Apache Hadoop or Apache Spark. Regarding recommendations, it basically provides an API for Collaborative-Filtering including typical similarity functions like user and item based correlation. The API can be extended for using custom preferences data, similarity functions and clustering algorithms. Depending on the volume of data, Mahout will manage the data in memory or in HFDS.

PredictionIO is another good option. PredictionIO ([Pre]) is an open source machine learning server for software developers to create predictive features, such as personalization, recommendation and content discovery. It supports both Collaborative-Filtering and Machine Learning content-based approaches. PredictionIO's software stack is more complicated, involving the deployment of others systems like Apache Spark, ElasticSearch and HBase. Its mean advantage is its ease of use and extend.

Quality of Service and Evaluation Criteria For evaluating Recommender Systems, often it is easiest to perform offline experiments using existing data sets and a protocol that models user behavior to estimate recommender performance measures such as prediction accuracy. A more expensive option is a user study, where a small set of users is asked to perform a set of tasks using the system, typically answering questions afterwards about their experience. It is also possible to run large scale experiments on a deployed system. Such experiments evaluate the performance of the recommenders on real users who are oblivious to the conducted experiment.

The quality of service and evaluation criteria completely depends on the use case. As different applications have different needs, the designer of the system must decide on the important properties to measure for the concrete application at hand. However, it is possible to define some global evaluation properties valid for any recommendation approach. *Prediction accuracy* is by far the most discussed property in the recommendation system literature. A basic assumption in a recommender system is that a system that provides more accurate predictions will be preferred by the users. Typically, three prediction accuracy measures are used: measuring the accuracy of ratings predictions, measuring the accuracy of usage predictions and measuring the accuracy of rankings of items.

Another property widely used for evaluating a recommender system is the *coverage*. As the prediction accuracy of a recommendation system, especially in collaborative filtering systems, in many cases grows with the amount of data, some algorithms may provide recommendations with high quality, but only for a small portion of the items where they have huge amounts of data. The term coverage can refer to **Item Space Coverage**, i.e. the proportion of items that the recommendation system can recommend, or can refer to **User Space Coverage** which is the proportion of users or user interactions for which the system can recommend items.

Besides the above properties, there is a set of properties typically identified as important for getting user feedback:

- **Confidence**: system's trust in its recommendations or predictions (prediction score, normally directly extracted from a classifier).
- **Trust**: user's trust in the system recommendations. It includes any feature that can help the user check the recommendations are reasonable.
- **Novelty**: Novel recommendations are recommendations for items that the user did not know about.

- **Serendipity**: measure of how surprising the recommendations are for the user (amount of relevant information that is new to the user in a recommendation).
- **Utility**: domain-dependent measure for estimating how useful for the final system are being the recommendations. For example, for e-commerce websites, the utility can be measured by the proportion of revenue coming directly from suggested items.

8 Conclusion and Further Planning

The deliverable provides an initial specification draft for WP2, WP3, WP4 and WP5 by describing:

- a first set of media extractors including interdependencies (indicating potential combinations / improvements), design choices, evaluation criteria, input and output, annotation design
- a standard ontology for publishing metadata based on several existing ontologies, then adapted and extended for MICO use cases, which is available at `http://www.mico-project.eu/ns/platform/1.0/schema#`
- an annotation design based on the Open Annotation Data Model, including examples
- an extended version of SPARQL with capabilities for multimedia (metadata) querying
- different approaches for calculating cross-media recommendations, and related components.

The developed specs are an important basis for the research, development and evaluation activities in the next project phase.

During the next project phase, the list of media extractors will be further extended and complemented, and the metadata publishing protocol, extended SPARQL and recommendation approaches are likely to be further enhanced, all of them taking into account the evaluation results in WP2-5. The respective update to this will be provided in month 24.

References

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." In: *IEEE Trans. on Knowl. and Data Eng.* 17.6 (June 2005), pp. 734–749. ISSN: 1041-4347. DOI: 10.1109/TKDE.2005.99. URL: http://dx.doi.org/10.1109/TKDE.2005.99.

[All83] J.F. Allen. "Maintaining Knowledge About Temporal Intervals." In: *Communications of the ACM* 26.11 (1983), pp. 832–843.

[BG14] Dan Brickley and R.V. Guha. *RDFS RDF Schema 1.1*. W3C Recommendation. http://www.w3.org/TR/rdf-schema/. W3C, Feb. 2014.

[BH04] Justin Basilico and Thomas Hofmann. "Unifying Collaborative and Content-based Filtering." In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: ACM, 2004, pp. 9–. ISBN: 1-58113-838-5. DOI: 10.1145/1015330.1015394. URL: http://doi.acm.org/10.1145/1015330.1015394.

[BHS10] Mathieu Barthet, Steven Hargreaves, and Mark B. Sandler. "Speech/Music Discrimination in Audio Podcast Using Structural Segmentation and Timbre Recognition." In: *CMMR*. Ed. by Sølvi Ystad et al. Vol. 6684. Lecture Notes in Computer Science. Springer, 2010, pp. 138–162. ISBN: 978-3-642-23125-4. URL: http://dblp.uni-trier.de/db/conf/cmmr/cmmr2010.html#BarthetHS10.

[BKR08] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. "Mediation of User Models for Enhanced Personalization in Recommender Systems." In: *User Modeling and User-Adapted Interaction* 18.3 (Aug. 2008), pp. 245–286. ISSN: 0924-1868. DOI: 10.1007/s11257-007-9042-9. URL: http://dx.doi.org/10.1007/s11257-007-9042-9.

[BL14a] H. Björklund and E. Lundh. "Automatic speech recognition in media asset management." In: *Swedish Language Technology Conference*. 2014.

[BL14b] H. Björklund and E. Lundh. "Experiences from Development with open-source speech-recognition libraries." In: *MICO project homepage*. 2014. URL: www.mico-project.eu.

[BM14] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification*. Tech. rep. http://xmlns.com/foaf/spec/. xmlns, Jan. 2014.

[BP96] Daniel Billsus and Michael Pazzani. "Learning Probabilistic User Models." In: *In Proceedings of the Workshop on Machine Learning for User Models, Sixth International Conference on User Modeling, Chia*. Springer, 1996.

[Bjö14] Henrik Björklund. "Interactive Learning of Syntax-Based Natural Language Queries." In: *Swedish Language Technology Conference*. 2014.

[Boa12] Dublin Core Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. http://dublincore.org/documents/dcmi-terms/. DMCI, June 2012.

[Bra+08] Tim Bray et al. *XML Extensible Markup Language*. W3C Community Draft. http://www.w3.org/TR/2008/REC-xml-20081126/. W3C, Nov. 2008.

[Bra14] T. Bray. *JSON The Javascript Object Notation Data Interchange Format*. RFC 7159. RFC Editor, 2014, pp. 1–15. URL: http://tools.ietf.org/html/rfc7159.

[CBC08] Iván Cantador, Alejandro Bellogín, and Pablo Castells. "A Multilayer Ontology-based Hybrid Recommendation Model." In: *AI Commun.* 21.2-3 (Apr. 2008), pp. 203–210. ISSN: 0921-7126. URL: http://dl.acm.org/citation.cfm?id=1460172.1460184.

[CFO09] Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. “A Small Set of Formal Topological Relationships Suitable for End-User Interaction.” In: *SSD*. Ed. by David J. Abel and Beng Chin Ooi. Vol. 692. Lecture Notes in Computer Science. Springer, Oct. 5, 2009, pp. 277–295. ISBN: 3-540-56869-7. URL: http://dblp.uni-trier.de/db/conf/ssd/ssd93.html#ClementiniFO93.

[CK87] Paul R. Cohen and Rick Kjeldsen. “Information retrieval by constrained spreading activation in semantic networks.” In: *Information Processing & Management* 23.4 (1987). Special Issue: Artificial Intelligence and Information Retrieval, pp. 255 –268. ISSN: 0306-4573. DOI: http://dx.doi.org/10.1016/0306-4573(87)90017-3. URL: http://www.sciencedirect.com/science/article/pii/0306457387900173.

[CV95] Corinna Cortes and Vladimir Vapnik. “Support-Vector Networks.” In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125.

[Cuc+13a] L. Cuccovillo et al. “Audio tampering detection via microphone classification.” In: *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*. Sept. 2013, pp. 177–182.

[Cuc+13b] Luca Cuccovillo et al. “Blind Microphone Analysis and Stable Tone Phase Analysis for Audio Tampering Detection.” In: *Audio Engineering Society Convention 135*. Oct. 2013.

[DT05] Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection.” In: *International Conference on Computer Vision & Pattern Recognition*. Ed. by Cordelia Schmid, Stefano Soatto, and Carlo Tomasi. Vol. 2. June 2005, pp. 886–893. URL: http://lear.inrialpes.fr/pubs/2005/DT05.

[Dah+11] Erik Dahlström et al. *SVG Scalable Vector Graphics 1.1*. W3C Community Draft. http://www.w3.org/TR/SVG/Overview.html. W3C, Aug. 2011.

[EK11] A. Ernst and C. Küblbeck. “Fast Face Detection and Species Cassification of African Great Apes.” In: *International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. Klagenfurt, Austria: IEEE, Aug. 2011, pp. 279–284. ISBN: 9781457708459. DOI: 10.1109/AVSS.2011.6027337. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6027337http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6027337.

[Erl+12] Thomas Erl et al. *SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST*. Prentice Hall Press, 2012.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. “Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling.” In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL ’05. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 363–370.

[Gä+14] Daniel Gärtner et al. “Efficient Cross-Codec Framing Grid Analysis for Audio Tampering Detection.” In: *Audio Engineering Society Convention 136*. Apr. 2014.

[HN04] Xiaofei He and Partha Niyogi. “Locality Preserving Projections.” In: *Advances in Neural Information Processing Systems 16*. Ed. by S. Thrun, L.K. Saul, and B. Schölkopf. MIT Press, 2004, pp. 153–160. URL: http://papers.nips.cc/paper/2359-locality-preserving-projections.pdf.

[HS13] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. 2013. URL: http://www.w3.org/TR/sparql11-query/.

[KE06] Christian Küblbeck and Andreas Ernst. “Face detection and tracking in video sequences using the modifiedcensus transformation.” In: *Image and Vision Computing* 24.6 (2006). Face Processing in Video Sequences, pp. 564 –572. ISSN: 0262-8856. DOI: http://dx.doi.org/10.1016/j.imavis.2005.08.005. URL: http://www.sciencedirect.com/science/article/pii/S0262885605001605.

[KM03] Dan Klein and Christopher D. Manning. “Accurate Unlexicalized Parsing.” In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. ACL ’03. Sapporo, Japan: Association for Computational Linguistics, 2003, pp. 423–430.

[KVA11] Johannes Koch, Carlos A Velasco, and Philip Ackermann. *CNT Representing Content int RDF 1.0*. W3C Recommendation. http://www.w3.org/TR/Content-in-RDF10/. W3c, May 2011.

[KY01] Eiji Kasutani and Akio Yamada. “The MPEG-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval.” In: *Image Processing, 2001. Proceedings. 2001 International Conference on*. Vol. 1. IEEE. 2001, pp. 674–677.

[KYS12] Helen Kwong and Neil Yorke-Smith. “Detection of Imperative and Declarative Question–answer Pairs in Email Conversations.” In: *AI Commun.* 25.4 (Oct. 2012), pp. 271–283. ISSN: 0921-7126.

[Kur+14] Thomas Kurz et al. “SPARQL-MM-Extending SPARQL to Media Fragments.” In: *The Semantic Web: ESWC 2014 Satellite Events* (2014), pp. 236–240.

[Kur14] Thomas Kurz. *Squebi*. ISWC2014 Developers Workshop, co-located with the 13th International Semantic Web Conference, Trentino, Italy. 2014.

[LE13] Alexander Loos and Andreas Ernst. “An Automated Chimpanzee Identification System Using Face Detection and Recognition.” In: *EURASIP Journal on Image and Video Processing: Special Issue on Animal Behaviour Understanding in Image Sequences* 2013.49 (2013). ISSN: 1687-5281. DOI: 10.1186/1687-5281-2013-39.

[LS12] Andreas Laschka and Friedericke Schneemann. “Detektion von Objekten am Beispiel von Tierkörpern.” PhD thesis. Technical University of Ilmenau, Fraunhofer IDMT, 2012.

[LSM13] Timothy Lebo, Satya Sahoo, and Deborah McGuinness. *PROV-O The PROV Ontology*. W3C Recommendation. http://www.w3.org/TR/prov-o/. W3C, Apr. 2013.

[Lan14] Markus Lanthaler. *Hydra Core Vocabulary*. Community Group Unofficial Draft. W3C, 2014. URL: http://www.hydra-cg.com/spec/latest/core/.

[Loo13] Alexander Loos. “Chimpanzee Identification Using Global and Local Features.” In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, Canada, 2013.

[MKP02] J.M. Martinez, R. Koenen, and F. Pereira. “MPEG-7: The Generic Multimedia Content Description Standard, part 1.” In: *IEEE Multimedia* 9 (2002). ISSN: 1070-986X. DOI: 10.1109/93.998074.

[MM04] Frank Manola and Eric Miller. *RDF RDF Primer*. W3C Community Draft. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. W3C, Feb. 2004.

[MMM06] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. “Generating typed dependency parses from phrase structure parses.” In: *IN PROC. INT’L CONF. ON LANGUAGE RESOURCES AND EVALUATION (LREC)*. 2006, pp. 449–454.

[MMN02] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. “Content-boosted Collaborative Filtering for Improved Recommendations.” In: *Eighteenth National Conference on Artificial Intelligence*. Edmonton, Alberta, Canada: American Association for Artificial Intelligence, 2002, pp. 187–192. ISBN: 0-262-51129-0. URL: http://dl.acm.org/citation.cfm?id=777092.777124.

[Mah] *Apache Mahout*. https://mahout.apache.org/. Accessed: 2014-11-02.

[Man+13] Sebastian Mann et al. “Combining ENF Phase Discontinuity Checking and Temporal Pattern Matching for Audio Tampering Detection.” In: *2. Workshop Audiosignal- und Sprachverarbeitung*. 2013.

[Man+14] Christopher D. Manning et al. “The Stanford CoreNLP Natural Language Processing Toolkit.” In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2014, pp. 55–60.

[Mar+04] David Martin et al. “OWL-S: Semantic markup for web services.” In: *W3C member submission* 22 (2004), pp. 2007–04.

[Med] *MyMediaLite Recommender System Library*. http://www.mymedialite.net/. Accessed: 2014-11-02.

[PW13] Tom Preston-Werner. *Semantic Versioning 2.0.0*. http://semver.org/. 2013. URL: http://semver.org/.

[Pov+11] Daniel Povey et al. “The Kaldi Speech Recognition Toolkit.” In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. IEEE Signal Processing Society, Dec. 2011.

[Pre] *PredictionIO, Build Smarter Software with Machine Learning*. http://prediction.io/. Accessed: 2014-11-02.

[Rev] *RDF Review Vocabulary*. http://vocab.org/review/terms.html. Accessed: 2014-11-02.

[SAR02] MediaArea.net SARL. *MediaInfo*. Oct. 2002-2014. URL: http://mediaarea.net/en/MediaInfo (visited on 10/30/2014).

[SCS13] Robert Sanderson, Paolo Ciccarese, and Herbert Van de Sompel. *OADM Open Annotation Data Model*. W3C Community Draft. http://www.openannotation.org/spec/core/. W3C, Feb. 2013.

[SF14] Sebastian Schaffert and Sergio Fernández. *D6.1.1 System Architecture and Development Guidelines*. Deliverable. MICO, 2014. URL: http://www.mico-project.eu/wp-content/uploads/2014/06/Del-6.1.1-MICO-Architecture.pdf.

[SM95] Upendra Shardanand and Pattie Maes. “Social Information Filtering: Algorithms for Automating &Ldquo;Word of Mouth&Rdquo;” in: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223931. URL: http://dx.doi.org/10.1145/223904.223931.

[SRL98] Henning Schulzrinne, A. Rao, and R. Lanphier. *RFC2326 - Real Time Streaming Protocol (RTSP)*. 1998. URL: http://www.ietf.org/rfc/rfc2326.txt.

[Ski+07] Spiros Skiadopoulos et al. “A family of directional relation models for extended objects.” In: *Knowledge and Data Engineering, IEEE Transactions on* 19.8 (2007), pp. 1116–1130.

[Soc+13] Richard Socher et al. “Parsing With Compositional Vector Grammars.” In: *ACL*. 2013.

[Sol] *Apache Solr*. http://lucene.apache.org/solr/. Accessed: 2014-11-02.

[Sto03] Knut Stolze. “SQL/MM Spatial - The Standard to Manage Spatial Data in a Relational Database System.” In: *BTW*. 2003, pp. 247–264. URL: http://dblp.uni-trier.de/db/conf/btw/btw2003.html\#Stolze03.

[Tro+12] Raphaël Troncy et al. *Media Fragments URI 1.0 (basic)*. W3C Recommendation. http://www.w3.org/TR/2012/REC-media-frags-20120925/. W3C, Sept. 2012.

[VD+14] Davy Van Deursen et al. “Experiencing standardized media fragment annotations within HTML5.” In: *Multimedia Tools and Applications* 70.2 (2014), pp. 827–846.

[WC10] Kai Wang and Tat-Seng Chua. “Exploiting Salient Patterns for Question Detection and Question Retrieval in Community-based Question Answering.” In: *Proceedings of the 23rd International Conference on Computational Linguistics*. COLING ’10. Beijing, China: Association for Computational Linguistics, 2010, pp. 1155–1163.

[Wri+09] John Wright et al. “Robust Face Recognition via Sparse Representation.” In: *IEEE Transactions on Parallel and Pattern Analysis and Machine Intelligence* 31.2 (2009), pp. 210–27. ISSN: 0162-8828.

9 Appendencies

9.1 Result data model for TE-202(OAD), TE-204 (FDR)

Figures 46 and 47 show the low level annotation model which is created as new content parts by the extractors implementing TE-202(Object and Animal Detection), TE-204 (Face Detection and Recognition).

Figure 46 The general schema for object detection and recognition annotations.

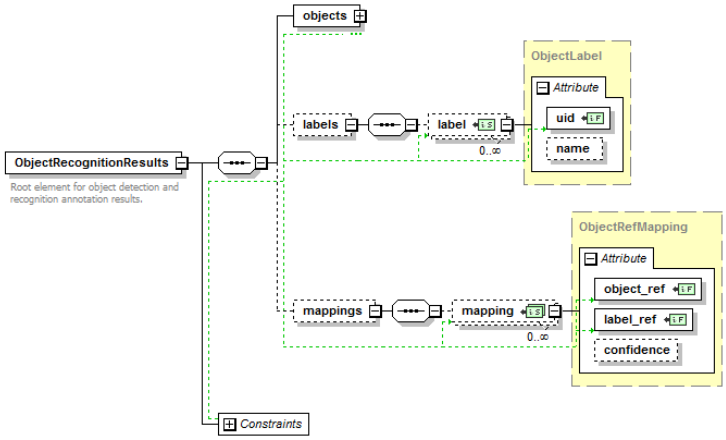
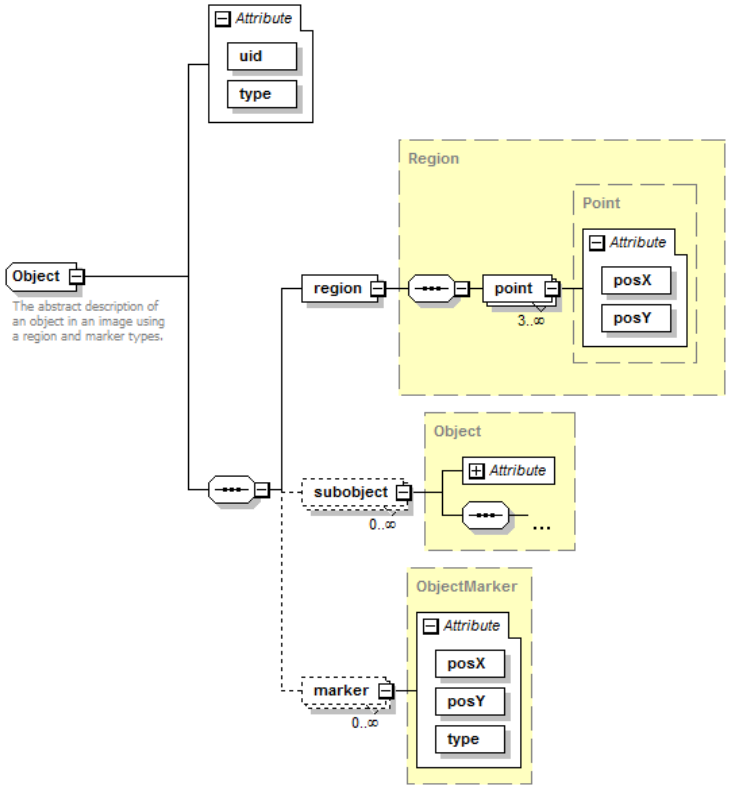


Figure 47 The detailed schema of the object type.



Listing 15 shows an exemplary xml annotation. The example shows a possible outcome for a face recognition extraction for one image. The corresponding schema is shown in listing 16.

```
<?xml version="1.0" encoding="UTF-8"?>
<ObjectRecognitionResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ObjectRecognitionResults.xsd">
  <objects>
    <object uid="OBJECTID1" type="FACE">
      <region>
        <point posX="0.2" posY="0.3"/>
        <point posX="0.2" posY="0.5"/>
        <point posX="0.4" posY="0.5"/>
        <point posX="0.4" posY="0.3"/>
      </region>
      <subobject uid="OBJECTID13" type="EYE">
        <region>
          <point posX="0.25" posY="0.31"></point>
          <point posX="0.25" posY="0.32"></point>
          <point posX="0.26" posY="0.32"></point>
          <point posX="0.26" posY="0.31"></point>
        </region>
      </subobject>
      <marker posX="0.255" posY="0.355" type="LEFT_EYE"/>
    </object>
    <object uid="OBJECTID2" type="FACE">
      <region>
        <point posX="0.7" posY="0.25"/>
        <point posX="0.7" posY="0.4"/>
        <point posX="0.9" posY="0.4"/>
        <point posX="0.9" posY="0.25"/>
      </region>
    </object>
  </objects>
  <labels>
    <label uid="LABELID1" name="Barack Obama"/>
    <label uid="LABELID2" name="Angela Merkel"/>
    <label uid="LABELID3" name="Martin Luther King"/>
  </labels>
  <mappings>
    <mapping object_ref="OBJECTID1" label_ref="LABELID1" confidence="0.7"/>
    <mapping object_ref="OBJECTID1" label_ref="LABELID3" confidence="0.1"/>
    <mapping object_ref="OBJECTID2" label_ref="LABELID2" confidence="0.8"/>
  </mappings>
</ObjectRecognitionResults>
```

Listing 15: Exemplary face recognition result

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning" elementFormDefault="qualified"
attributeFormDefault="unqualified" vc:minVersion="1.1">
  <!-- Root element -->
  <xs:element name="ObjectRecognitionResults">
    <xs:annotation>
      <xs:documentation>Root element for object detection and
      recognition annotation results.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="objects">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="object" type="Object" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="labels" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="label" type="ObjectLabel" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="mappings" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="mapping" type="ObjectRefMapping" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:key name="ObjectKey">
      <xs:selector xpath="objects/object"/>
      <xs:field xpath="@uid"/>
    </xs:key>
    <xs:key name="LabelKey">
      <xs:selector xpath="labels/label"/>
      <xs:field xpath="@uid"/>
    </xs:key>
    <xs:keyref name="ObjectKeyRef" refer="ObjectKey">
      <xs:selector xpath="mappings/mapping"/>
      <xs:field xpath="@object_ref"/>
    </xs:keyref>
    <xs:keyref name="LabelKeyRef" refer="LabelKey">
      <xs:selector xpath="mappings/mapping"/>
      <xs:field xpath="@label_ref"/>
    </xs:keyref>
  </xs:element>
  <!-- Complex type definitions -->
  <xs:complexType name="Object">
    <xs:annotation>
      <xs:documentation>The abstract description of an object in
      an image using a region and marker types.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="region" type="Region" minOccurs="1" maxOccurs="1"/>
      <xs:element name="subobject" type="Object" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="marker" type="ObjectMarker" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="uid" type="xs:ID" use="required"/>
    <xs:attribute name="type" type="ObjectType" use="required"/>
  </xs:complexType>
  <xs:complexType name="Point">
    <xs:annotation>
      <xs:documentation>Describes a discrete position in an image</xs:documentation>
```

```
    </xs:annotation>
    <xs:attribute name="posX" type="NormalizedPositionValue" use="required"/>
    <xs:attribute name="posY" type="NormalizedPositionValue" use="required"/>
  </xs:complexType>
  <xs:complexType name="Region">
    <xs:annotation>
      <xs:documentation>An area of an image defined by multiple points</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="point" type="Point" minOccurs="3" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ObjectMarker">
    <xs:annotation>
      <xs:documentation>Describes a discrete position that is part of an object</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="Point">
        <xs:attribute name="type" type="ObjectMarkerType" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ObjectLabel">
    <xs:annotation>
      <xs:documentation>A structure describing the identity (the label)
      of a detected object.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="uid" type="xs:ID" use="required"/>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="ObjectRefMapping">
    <xs:attribute name="object_ref" type="xs:IDREF" use="required"/>
    <xs:attribute name="label_ref" type="xs:IDREF" use="required"/>
    <xs:attribute name="confidence" type="ConfidenceValue" use="optional"/>
  </xs:complexType>
  <!--Simple type definitions -->
  <xs:simpleType name="ConfidenceValue">
    <xs:annotation>
      <xs:documentation>A type for confidence values.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:float">
      <xs:minInclusive value="0.0"/>
      <xs:maxInclusive value="1.0"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NormalizedPositionValue">
    <xs:annotation>
      <xs:documentation>A type for values of point specification
      in a normalized range from 0..1.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:double">
      <xs:minInclusive value="0.0"/>
      <xs:maxInclusive value="1.0"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ObjectType">
    <xs:annotation>
      <xs:documentation>Pre define types of objects. Those are subject to
      change and depend on what the extractor is able to detect.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="FACE"/>
      <xs:enumeration value="EYE"/>
      <xs:enumeration value="NOSE"/>
      <xs:enumeration value="MOUTH"/>
      <xs:enumeration value="ANIMAL"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ObjectMarkerType">
    <xs:annotation>
```

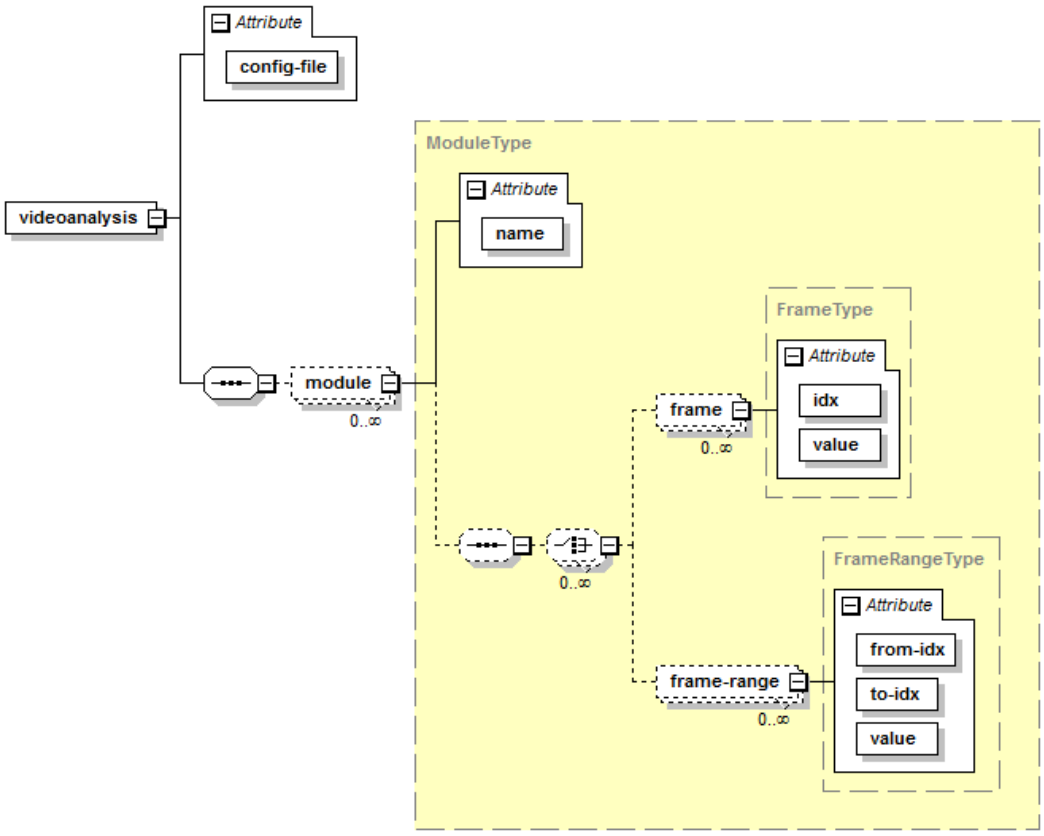
```
      <xs:documentation>Pre defined types of object markers.  Those are subject to
      change and depend on what the extractor is able to detect.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="LEFT_EYE"/>
      <xs:enumeration value="RIGHT_EYE"/>
      <xs:enumeration value="MOUTH_CENTER"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Listing 16: Schema for object detection and recognition annotation

9.2 Result data model for TE-205(AVQ), TE-206 (TVS), TE-207(SMD)

Figure 48 shows the low level annotation model which is created as new content parts by the extractors implementing TE-205(A/V Error Detection and Quality Assessment), TE-206 (Temporal Video Segmentation) and TE-207(Speech-Music Discrimination). Listing 17 shows an exemplary xml annotation.

Figure 48 The extractor annotation schema for TE-205(AVQ), TE-206 (TVS), TE-207(SMD)



The corresponding schemas are shown in listing 18 and 19.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<videoanalysis xmlns:temporalvisegm="http://www.idmt.fraunhofer.de/temporalvisegm"
config-file="result_tvs_neu.conf.xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
si:noNamespaceSchemaLocation="videoanalysisQuality.xsd">
  <module name="KeyFrame">
    <frame-range from-idx="0" to-idx="28" value="0"/>
    <frame idx="29" value="100"/>
    <frame-range from-idx="30" to-idx="39" value="0"/>
    <frame idx="40" value="100"/>
    ...
    <frame idx="629" value="100"/>
    <frame-range from-idx="630" to-idx="690" value="0"/>
  </module>
  <module name="Shot">
    <frame-range from-idx="0" to-idx="45" value="0"/>
    <frame idx="46" value="100"/>
    <frame-range from-idx="47" to-idx="169" value="0"/>
    <frame idx="170" value="100"/>
    ...
    <frame-range from-idx="506" to-idx="575" value="0"/>
    <frame idx="576" value="100"/>
    <frame-range from-idx="577" to-idx="690" value="0"/>
  </module>
</videoanalysis>
```

Listing 17: Shortened result example for TVS

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:namespace="http://www.idmt.fraunhofer.de/namespace">
  <xs:include schemaLocation="videoanalysisTypes.xsd"/>
  <!-- root -->
  <xs:element name="videoanalysis">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="module" type="ModuleType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="config-file" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <!-- types -->
  <xs:complexType name="TimebaseType">
    <xs:attribute name="num" type="xs:integer" use="required"/>
    <xs:attribute name="denum" type="xs:integer" use="required"/>
  </xs:complexType>
  <xs:complexType name="ModuleType">
    <xs:sequence minOccurs="0" maxOccurs="1">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="frame" type="FrameType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="frame-range" type="FrameRangeType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="name" type="ModuleNames" use="required"/>
  </xs:complexType>
  <xs:complexType name="FrameType">
    <xs:attribute name="idx" use="required">
      <xs:simpleType>
        <xs:restriction base="IndexType"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="value" type="FloatValueType" use="required"/>
  </xs:complexType>
  <xs:complexType name="FrameRangeType">
    <xs:attribute name="from-idx" use="required">
      <xs:simpleType>
        <xs:restriction base="IndexType"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="to-idx" use="required">
      <xs:simpleType>
        <xs:restriction base="IndexType"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="value" type="FloatValueType" use="required"/>
  </xs:complexType>
</xs:schema>
```

Listing 18: Main schema definition

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:namespace="http://www.idmt.fraunhofer.de/namespace">
  <!-- types for further explanations of elements/attributes from
     Time-, Quality-, and Config-Scheme are defined here -->
  <!-- types -->
  <xs:simpleType name="IndexType">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="FloatValueType">
    <xs:restriction base="xs:float">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ModuleNames">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Blur"/>
      <xs:enumeration value="Blocking"/>
      <xs:enumeration value="Ringing"/>
      <xs:enumeration value="Edge"/>
      <xs:enumeration value="Interlace"/>
      <xs:enumeration value="Freeze"/>
      <xs:enumeration value="OverExposure"/>
      <xs:enumeration value="UnderExposure"/>
      <xs:enumeration value="BlackFrame"/>
      <xs:enumeration value="Noise"/>
      <xs:enumeration value="FieldOrder"/>
      <xs:enumeration value="HBlackbars"/>
      <xs:enumeration value="VBlackbars"/>
      <xs:enumeration value="KeyFrame"/>
      <xs:enumeration value="SceneBorder"/>
      <xs:enumeration value="Shot"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ParameterTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="BlockingResolution"/>
      <xs:enumeration value="BlockingLevel"/>
      <xs:enumeration value="BlurResolution"/>
      <xs:enumeration value="BlurLevel"/>
      <xs:enumeration value="RingingLevel"/>
      <xs:enumeration value="InterlacePrefilterThreshold"/>
      <xs:enumeration value="InterlaceThreshold"/>
      <xs:enumeration value="FreezeMinDuration"/>
      <xs:enumeration value="FreezeThreshold"/>
      <xs:enumeration value="keyframeMaximumDistance"/>
      <xs:enumeration value="keyframeMinimumDistance"/>
      <xs:enumeration value="keyframeMinimumDifference"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Listing 19: Types schema definition

9.3 Result data model for TE-224 (ACD)

As reported in Section 5.8, the three independent components realizing TE-224 (Audio Cutting Detection) provide an output compliant to the same XML schema.

The schema is is shown in listing 20:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="AudioTamperingDetectionAnnotation">
    <xs:annotation>
      <xs:documentation>XSD for annotating the ATD (audio tampering detection) outcome</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="GeneralProperties">
          <xs:annotation>
            <xs:documentation>General information regarding the audio item</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="URI" type="xs:anyURI" minOccurs="0">
                <xs:annotation>
                  <xs:documentation>URI of the audio item</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="Duration">
                <xs:annotation>
                  <xs:documentation>Item duration, given as time in hh:mm:ss.ms</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string"/>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="Channels" type="xs:int">
                <xs:annotation>
                  <xs:documentation>Number of audio channels used within the item</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="SamplingRate" type="xs:int">
                <xs:annotation>
                  <xs:documentation>Sampling rate used within the item</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="InverseDecoding" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Segment" type="Segment" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>Segment within content item</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="StableToneAnalysis" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Segment" type="Segment" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>Segment within content item</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="Discontinuity" maxOccurs="unbounded">
```

```
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Location" type="xs:string">
                    <xs:annotation>
                      <xs:documentation>Discontinuity Location, given as time in hh:mm:ss.ms</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="SourceFeature">
                    <xs:annotation>
                      <xs:documentation>Frequency or Phase</xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Frequency"/>
                        <xs:enumeration value="Phase"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="Threshold" type="xs:float">
                    <xs:annotation>
                      <xs:documentation>Threshold value not respected that led to the detection</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="Value" type="xs:float">
                    <xs:annotation>
                      <xs:documentation>Numeric value violating the threshold</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="MicrophoneDiscrimination" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Segment" type="Segment" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>Segment within content item</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="Discontinuity" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="SegmentId1" type="xs:int">
                    <xs:annotation>
                      <xs:documentation>Id of the first segment involved</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="SegmentId2" type="xs:int">
                    <xs:annotation>
                      <xs:documentation>Id of the second segment involved</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="SourceFeature" type="xs:string">
                    <xs:annotation>
                      <xs:documentation>currently only CoCo</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="Threshold" type="xs:float">
                    <xs:annotation>
                      <xs:documentation>Threshold value not respected that led to the detection</xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="Value" type="xs:float">
                    <xs:annotation>
                      <xs:documentation>Numeric value violating the threshold</xs:documentation>
                    </xs:annotation>
                  </xs:element>
```

```
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="Segment">
  <xs:sequence>
    <xs:element name="URI" type="xs:anyURI" minOccurs="0">
      <xs:annotation>
        <xs:documentation>URI of the segment within the audio item</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Id" type="xs:int">
      <xs:annotation>
        <xs:documentation>integer id of the segment, unique within each detection output</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Start">
      <xs:annotation>
        <xs:documentation>Segment start, given as time in hh:mm:ss.ms</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string"/>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="End" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Segment end, given as time in hh:mm:ss.ms</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string"/>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="Duration">
      <xs:annotation>
        <xs:documentation>Segment duration, given as time in hh:mm:ss.ms</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string"/>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="CodingTraces" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Information about codec traces in the segment</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Codec">
            <xs:annotation>
              <xs:documentation>Codec used for encoding, currently 5 types:
                    MP3, MP3PRO, AAC, HE-AAC, unknown  </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="AAC"/>
                <xs:enumeration value="HE-AAC"/>
                <xs:enumeration value="MP3"/>
                <xs:enumeration value="MP3PRO"/>
                <xs:enumeration value="unknown"/>
              </xs:restriction>
```

```
              </xs:simpleType>
            </xs:element>
            <xs:element name="FrameOffset" type="xs:int" minOccurs="0">
              <xs:annotation>
                <xs:documentation>Frame offset created during encoding;
                      always present if a codec has been detected</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="Bitrate" type="xs:int" minOccurs="0">
              <xs:annotation>
                <xs:documentation>Bitrate used for encoder;
                      always present if a codec has been detected</xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Listing 20: XML Schema Definition for the output of ACD components