EUMSSI: Multilayered analysis of multimedia content using UIMA, MongoDB and Solr

Jens Grivolla and **Maite Melero** and **Toni Badia** [1]

Abstract. We present a scalable platform that allows for distributed processing of large quantities of multimedia content. The EUMSSI platform provides support for both synchronous and asynchronous analysis processes and thus allows for *on-demand* services as well as long running batch processes. Analysis services for speech, video and text are integrated in the platform, as well as transversal services that combine and enrich the existing outputs from various modalities. It builds on established open source projects such as UIMA, MongoDB and Solr and the project outcomes are published under permissive open source licenses.

1 Introduction

For reasoning with and about the multimedia data, the EUMSSI platform needs to recognize entities, such as actors, places, topics, dates and genres. A core idea is that metadata resulting from analyzing one media helps reinforce the aggregation of information from other media. For example, an important issue in speech recognition is the transcription of previously unknown (out-of-vocabulary) words. This is particularly important when dealing with current news content, where person and organization names, and other named entities that may not appear in older training corpora, are among the most critical parts of the transcription. Existing text, tags and other metadata, as well as information automatically extracted from these sources, are used to improve and adapt the language models. Further, OCR on video data, speech analysis and speaker recognition mutually reinforce one another.

The combined and integrated results of the audio, video and text analysis significantly enhance the existing metadata, which can be used for search, visualization and exploration. In addition, the extracted entities and other annotations are exploited for identifying specific video fragments in which a particular person speaks, a new topic begins, or an entity is mentioned. Figure 1 illustrates some of the different layers of analysis that may exist for a video content item.

The EUMSSI system currently includes a wide variety of analysis components (many of which leverage and improve upon existing open source systems), such as automatic speech transcription (ASR), person identification (combining voice and face recognition, OCR on subtitles for naming, and Named Entity Recognition and Linking), and many natural language processing approaches, applied to speech transcripts as well as original written content or social media, e.g. NER (Stanford NLP), Entity Linking (DBpedia Spotlight), keyphrase extraction (KEA), quote extraction, topic segmentation, sentiment analysis, etc.

[1] Universitat Pompeu Fabra, Spain, email: <firstname>.<lastname>@upf.edu

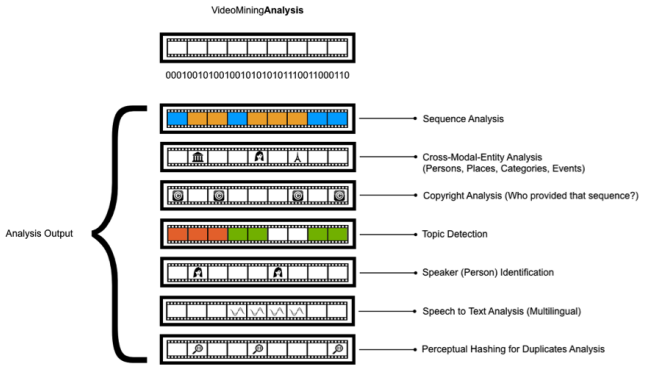

Figure 1. Video Mining Analysis

2 Architecture overview

The EUMSSI platform has been developed using UIMA, MongoDB, Solr and other Open Source technologies to manage complex workflows involving online (on-demand) and offline (batch) processing, with mutual dependencies between the different modalities. The three main challenges of the core platform are:

- Enabling the integration and combination of different annotation layers using UIMA and its CAS format
- Managing the processing workflow using MongoDB and UIMA
- Providing efficient and scalable access to the analyzed content for applications and demonstrators using Solr

The EUMSSI architecture was designed with a few core principles and requirements in mind:

- Simplicity: The platform should not be overly complex, in order to make it maintainable as well as to rapidly have a working system that all involved parties can build on
- Robustness: Failures, even hardware failures, should not have disastrous consequences
- Portability: It should be possible to easily migrate the platform to a different system
- Flexibility: It must be possible to quickly extend the platform, in particular by adding new analysis processes or content sources
- Scalability: The platform must be able to support large-scale content processing, as well as efficiently provide results to end users

As a result, the EUMSSI platform relies on open source technologies with a proven track record of reliability and scalability as its foundation.

The EUMSSI platform functions as a set of loosely coupled components that only interact through a common data back-end (MongoDB) that ensures that the system state is persisted and can be robustly recovered after failures of individual components or even the whole platform (including hardware failures).

All components run independently and can be seen as basically "stateless" in that they maintain only the information necessary for immediate execution. As such it is possible to restart individual components without affecting the overall system, making it relatively easy to ensure the overall reliability of the platform.

All new content coming into the system is first normalized to a common metadata schema (based on schema.org) and stored in a MongoDB database to make it available for further processing. Analysis results, as well as the original metadata, are stored in UIMA's CAS format[2] to allow integration of different aligned layers of analysis as well as in a simplified format that is then indexed with Solr. The applications use the Solr indexes for efficient and scalable access to the analyzed content, as well as statistical metrics over the whole document collection or specific subsets that can be used for exploration and visualization.

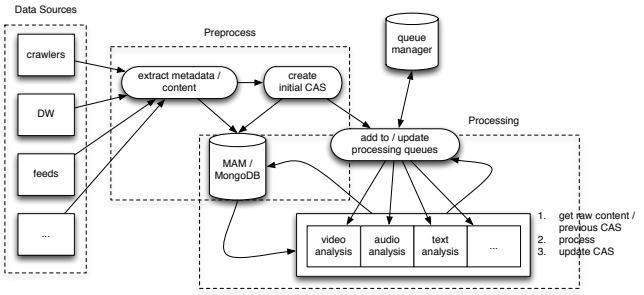

Figure 2. Architecture design

The process flow, pictured in Figure 2, can be summarized as follows:

1. new data arrives (or gets imported)
2. preprocessing stage
   (a) make content available through unique internal identifier
   (b) create initial CAS with aligned metadata / text content and content URI
   (c) mark initial processing queue states
3. processing / content analysis
   (a) distributed analysis systems query queue when they have processing capacity
   (b) retrieve CAS with existing data (or get relevant metadata from wrapper API)
   (c) retrieve raw content based on content URI
   (d) process
   (e) update CAS (possibly through wrapper API)
   (f) create simplified output for indexing
   (g) update queues
      i. mark item as processed by the given queue
      ii. mark availability of data to be used by other analysis processes
4. updating the Solr indexes whenever updated information is available for a content items

[2] Unstructured Information Management Architecture: http://uima.apache.org/

Note that this architecture design mainly depicts the data analysis part of the EUMSSI system. The applications for end users are built upon the Solr indexes that are automatically synchronized with the analysis results.

Crawlers, preprocessors and API layer are maintained as part of the core EUMSSI platform. The **MongoDB database** is installed separately and managed from within the platform components (with little or no specific configuration and setup), and the same goes for some external dependencies such as having a Tomcat server on which to run the API layer.

Analysis components for video and audio are fully external and independent and communicate with the platform through the API layer. Text analysis and cross-modality components are implemented as UIMA components and run as pipelines integrated into the platform using custom input (CollectionReader) and output (CASConsumer) modules that read an existing CAS representation of the document from the MongoDB back-end, and write back a modified CAS with added annotations (and possibly layers/views) as well as extracted or "flattened" metadata that can be used by other components (e.g. a list of all detected entities in the document).

Crawlers make external data sources available to the platform. Some crawler components are run only once to import existing datasets, whereas others feed continuously into the platform. **Preprocessing** takes original metadata from the different sources and transforms it into a unified representation with a common metadata vocabulary.

The **EUMSSI API** abstracts away from the underlying storage (MongoDB and CAS data representation) to facilitate access for external components such as video and audio processing. It acts as a light-weight layer that translates between the internal data structure and REST-like operations tailored to the needs of the components.

Indexing takes care of making the metadata (from the original source as well as automatically extracted) available to demonstrators and applications by mirroring the data on a Solr server that is accessible to those applications. It is performed using mongo-connector[3], leveraging built-in replication features of MongoDB for low-latency real time indexing of new (even partial) content, as well as content updates.

Components that are part of the core platform can be found on GitHub and are organized into directories corresponding to the type of component. More detailed information about those components may be found in their respective README.md files.

2.1 Design decisions and related content analysis platforms

Apart from integrating a wide variety of analysis components working on text, audio, video, social media, etc., at different levels of semantic abstraction, a key aspect of EUMSSI is the integration and combination of those different information layers. This is the main motivation for using UIMA as the main underlying framework, as described in section 3. This also has the advantage of providing a platform for building processing pipelines that has low overhead when running on a single machine (all information is passed in-memory), while still enabling distributed and scaled-out processing when necessary.

On the other hand, it quickly became apparent that not all kinds of analysis are a good fit for such a workflow, leading to the hybrid approach described in section 4. Having the workflow control in the

[3] https://github.com/mongodb-labs/mongo-connector

same database as the data itself eliminates some of the potential failures of more complex queue management systems by ensuring consistency between the stored data and its analysis status. It also means that efforts in guaranteeing availability and performance can focus on optimizing and allocating resources for the MongoDB database (for which best practices are well established).

While there are commercial content management systems on the market, some of which allow for the integration of some automatic content analysis, none of them have the flexibility of the EUMSSI platform, and in particular none are aimed at facilitating cross-modality integration.

Some recent research projects approach similar goals. MultiSensor[4] combines analysis services through distributed RESTful services based on NIF as an interchange format, incurring higher communication overheads in exchange for greater independence of services (compared to the UIMA-based parts of EUMSSI). LinkedTV[5] has a similar approach to EUMSSI (also using MongoDB and Solr), integrating the outputs of different analysis processes in a common MPEG-7 representation in the *consolidation* step, however (it appears) with far less mutual integration of outputs from different modalities. MediaMixer[6] focuses on indexing Media Fragments[7] with metadata to improve retrieval in media production, and BRIDGET[8] provides means to link (bridge) from broadcast content to related items, partly based on automatic video analysis.

3 Aligned data representation

Much of the reasoning and cross-modal integration depends on an aligned view of the different annotation layers, e.g., in order to connect person names detected from OCR with corresponding speakers from the speaker recognition component, or faces detected by the face recognition.

The Apache UIMA[9] CAS (common analysis structure) representation is a good fit for the needs of the EUMSSI project as it has a number of interesting characteristics:

- Annotations are stored "stand-off", meaning that the original content is not modified in any way by adding annotations. Rather, the annotations are entirely separate and reference the original content by offsets
- Annotations can be defined freely by defining a "type system" that specifies the types of annotations (such as *Person, Keyword, Face,* etc.) and the corresponding attributes (e.g. *dbpediaUrl, canonicalRepresentation, ...*)
- Source content can be included in the CAS (particularly for text content) or referenced as external content via URIs (e.g. for multimedia content)
- While each CAS represents one "document" or "content item", it can have several *Views* that represent different aspects of that item, e.g. the video layer, audio layer, metadata layer, transcribed text layer, etc., with separate source content (SofA or "subject of annotation") and separate sets of annotations
- CASes can be passed efficiently in-memory between UIMA analysis engines
- CASes can be serialized in a standardised OASIS format[10] for storage and interchange

Annotations based directly on multimedia content (video and audio) naturally refer to that content via timestamps, whereas text analysis modules normally work with character offsets relative to the text content. It is therefore fundamental that any textual views created from multimedia content (e.g. via ASR or OCR) refer back to the timestamps in the original content. This is done by creating annotations, e.g. tokens or segments, that include the original timestamps as attributes in addition to the character offsets.

As an example, we may have a CAS with an audio view which contains the results of automatic speech recognition (ASR), providing the transcription as a series of tokens/words with a timestamp for each word as an additional feature.

In this way it is possible to apply standard text analysis modules (that rely on character offsets) on the textual representation, while maintaining the possibility to later map the resulting annotations back onto the temporal scale.

So called *SofA-aware* UIMA components are able to work on multiple views, whereas "normal" analysis engines only see one specific view that is presented to them. This means that e.g. standard text analysis engines don't need to be aware that they are being applied to an ASR view or an OCR view; they just see a regular text document. SofA-aware components, however, can explicitly work on annotations from different views and can therefore be used to integrate and combine the information coming from different sources or layers, and create new, integrated views with the output from that integration and reasoning process.

4 Synchronous and asynchronous workflow management

In EUMSSI we decided to use a dual approach to workflow management, allowing for synchronous (and even on-demand) analysis pipelines as well as the execution of large batch jobs which need to be run asynchronously, possibly scheduled according to the availability of computational resources.

We opted for UIMA as the basis for synchronous workflows, as well as the data representation used for integrating different analysis layers. On the other hand, a web-based API allows other analysis processes, such as audio and video analysis, to retrieve content and upload results independently, giving them complete freedom to schedule their work according to their specific needs.

4.1 Analysis pipelines using UIMA

UIMA provides a platform for the execution of analysis components (*Analysis Engines* or *AEs*), as well as for managing the flow between those components. CPE or uimaFIT[11] [2] can be used to design and execute pipelines made up of a sequence of AEs (and potentially some more complex flows), and UIMA-AS[12] (*Asynchronous Scaleout*) permits the distribution of the process among various machines or even a cluster (with the help of UIMA DUCC[13]).

Within the EUMSSI project we have developed and integrated a number of UIMA analysis components, mostly dealing with text analysis and semantic enrichment. Whenever possible, components

[4] http://multisensorproject.eu/
[5] http://linkedtv.eu
[6] http://mediamixer.eu
[7] https://www.w3.org/2008/WebVideo/Fragments/
[8] http://ict-bridget.eu
[9] http://uima.apache.org/
[10] http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html
[11] https://uima.apache.org/uimafit.html
[12] http://uima.apache.org/doc-uimaas-what.html
[13] http://uima.apache.org/doc-uimaducc-whatitam.html

from the UIMA-based DKPro project [1] were used, especially for the core analysis components (tokenization, part-of-speech, parsing, etc.). In addition to a large number of ready-to-use components, DKPro Core provides a unified type system to ensure interoperability between components from different sources. Other components developed or integrated in EUMSSI were made compatible with this type system.

4.2 Managing content analysis with MongoDB

There are some components of the EUMSSI platform, however, that do not integrate easily in this fashion. This is the case of computationally expensive processes that are optimized for batch execution. A UIMA AE needs to expose a *process()* method that operates on a single CAS (= document), and is therefore not compatible with batch processing. This is particularly true for processes that need to be run on a cluster, with significant startup overhead, such as many video and audio analysis tasks.

It is therefore necessary to have an alternative flow mechanism for offline or batch processes, which needs to integrate with the processing performed within the UIMA environment.

The main architectural and integration issues revolve around the data flow, rather than the computation. In fact, the computationally complex and expensive aspects are specific to the individual analysis components, and should not have an important impact on the design of the overall platform.

As such, the design of the flow management is presented in terms of transformations between data states, rather than from the procedural point of view. The resulting system should only rely on the robustness of those data states to ensure the reliability and robustness of the overall system, protecting against potential problems from server failures or other causes. At any point, the system should be able to resume its function purely from the state of the persisted data.

To ensure reliability and performance of the data persistence, we use the well-established and widely used database system MongoDB, which provides great flexibility as well as proven scalability and robustness.

Figure 3 shows the general flow of the EUMSSI system, focusing on the data states needed for the system to function.

In order to avoid synchronization issues, the state of the data processing is stored together with the data within each content item, and the list of pending tasks can be extracted at any point through simple database queries. We therefore only depend on the MongoDB database (which can be replicated across several machines or even a large cluster for performance and reliability) to fully establish the processing state of all items. For example, the queues for analysis processes can be constructed directly from the "processing.queues.*queue_name*" field of an item by selecting (for a given queue) all items that have not yet been processed by that queue and that fulfill all prerequisites (dependencies).

The analysis results are stored in CAS format (optionally with compression). In order to avoid potential conflicts or race conditions between components (most analysis processes run independently of one another), the different layers are stored in separate database fields as independent CASes. Components that work across layers then merge the separate CASes into a single one (as separate Views) in order to combine the information. The "meta.extracted" section of a document is used to store the simplified analysis results that are automatically synchronized with the Solr index, and can also be used as inputs to other annotators (such as detected Named Entities as input to speech recognition), to avoid the overhead of extracting that information from the CAS on demand.

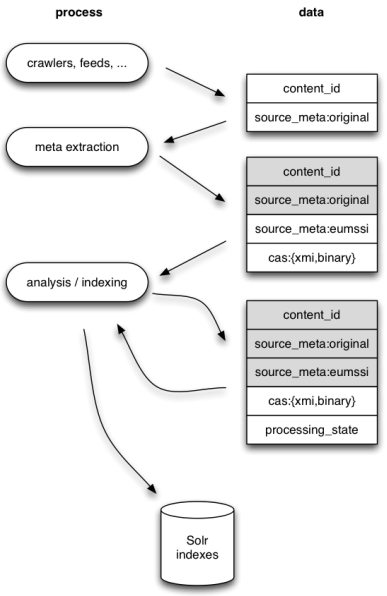

Figure 3. data flow and transformations

In its simplest form, the processes responsible for the data transitions are fully independent and poll the database periodically to retrieve pending work. Those processes can then be implemented in any language that can communicate comfortably with MongoDB.

4.3 Multimodal multilayer data integration and enrichment

The integration of data from different analysis layers is usually done by loading the CAS representations generated by different prior processes and merging them as individual Views in a single CAS. Layers that work on different representations, e.g. speaker recognition, audio transcript and OCR, are aligned by using timestamps associated with the segments or tokens. As a result, new integrated views can be created, combining the different information layers. Metadata is also enriched by adding information to existing annotations or creating new ones, e.g. with information obtained from SPARQL DBpedia lookups.

4.4 Indexing for scalable data-driven applications

The final applications do not use the information stored in MongoDB directly, but rather access Solr indexes created from that information to respond specifically to the types of queries needed by the applications. Those indexes are updated whenever new analysis results are available for a given item, through the use of *mongo-connector* which keeps the indexes always up-to-date with the content of the "meta.source" and "meta.extracted" sections.

5 Standards and interoperability

EUMSSI uses established protocols and uses freely available and widely used open source software as its underpinnings, in addition to publishing in-project developments under permissive open source licenses through popular platforms such as GitHub.

The API for external analysis components is REST-like and uses JSON for communication, whereas the end applications access the

data through Solr's REST-like API (which supports various result formats). Metadata is represented using a vocabulary built upon *schema.org* and the internal representations in UIMA use the DKPro type system as a core. Entity linking is performed against the DBpedia, thus yielding Linked Open Data URIs for entities and allowing for the use of SPARQL and RDF to access additional information.

There is now a starting initiative to establish a "standard" type system for UIMA, with initial conversations pointing towards building upon the DKPro type system for this purpose. Various institutions have expressed their interest in endorsing such a type system, leading to a major step forward in improving interoperability between UIMA components from different sources.

6 Demonstrators and applications

Using the data stored in the system, and made available through Solr indexes, as well as on-demand text analysis services, a variety of applications can be built that provide access to the content and information. Two very different applications were built within the EUMSSI project, serving as technology demonstrators as well as having real-world use.

The **storytelling tool** provides a web interface that allows a journalist to work on an article in a rich editor, when writing an news article or preparing a report. The system then analyses the text the journalist is writing in order to provide relevant background information. In particular, the journalist can directly access the Wikipedia pages of entities that appear in the text, or find related content in the archives (including content from outside and social media sources). A variety of graphical widgets then allow to explore the content collection, finding relevant video snippets, quotes, or presenting relevant entities and the relations between them.

The tool is built on web technologies (HTML, Javascript, ...) and in particular AJAX-Solr[14] which manages the communication with the Solr backend that provides the data. Most of the functionality in the widgets is based on using Solr queries (automatically generated or manually specified) to select a relevant set of items (articles, videos, etc.) from the index.

The **second-screen application**, on the other hand, is aimed at viewers of TV or streaming content, at home who would like to easily access background information, or have their viewing augmented with entertainment (or edutainment) activities such as automatically generated quizzes relating to the currently viewed content. The EUMSSI second screen application is implemented as a server-side application that connects a 'first screen' client, which shows the video in an HTML5 player, with one or more 'second screen' clients (which may be on the same machine, or on a separate laptop, tablet or smartphone). All possible information and questions that can be shown to the user, are stored in JSON format in a WebVTT file (WebVTT is a W3C standard for displaying timed text, such as subtitles, in connection with the HTML5 video[15]).

All second screen clients that are logged in using the same identifier as the video show the same content at the same time. The video client sends the relevant content to the server at the moment specified in the VTT file. The server-side agent forwards the content to the second screen clients.

[14] `https://github.com/evolvingweb/ajax-solr`
[15] `https://w3c.github.io/webvtt/`

7 Conclusions

In the EUMSSI project we have developed a platform capable of handling large amounts of multimedia, with support for online and offline processing as well as the alignment and combination of different information layers. The system includes many interactions between different modalities, such as doing text analysis on speech recognition output, or adding Named Entities from surrounding text to the vocabulary known to the ASR system, among others.

The platform has proven capable of handling millions of content items on modest hardware, and is designed to allow for easily adding capacity through horizontal scaling.

The source code of the platform is publicly available at `https://github.com/EUMSSI/`. Additional documentation can be found in the corresponding wiki at `https://github.com/EUMSSI/EUMSSI-platform/wiki`.

ACKNOWLEDGEMENTS

The work presented in this article is being carried out within the FP7-ICT-2013-10 STREP project EUMSSI under grant agreement n° 611057, receiving funding from the European Union's Seventh Framework Programme managed by the REA-Research Executive Agency http://ec.europa.eu/research/rea.

REFERENCES

[1] Richard Eckart de Castilho and Iryna Gurevych, 'A broad-coverage collection of portable nlp components for building shareable analysis pipelines', in *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pp. 1–11, Dublin, Ireland, (August 2014). Association for Computational Linguistics and Dublin City University.

[2] Philip V. Ogren and Steven J. Bethard, 'Building test suites for UIMA components', *SETQA-NLP '09 Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, 1–4, (June 2009).