REAL-TIME SEMANTIC ENRICHMENT OF VIDEO STREAMS IN THE AGE OF BIG DATA

M. Montagnuolo[1], P. Platter[2], A. Bosca[3], N. Bidotti[2], and A. Messina[1]

[1] RAI Radiotelevisione Italiana, Italy and [2] Agile Lab s.r.l., Italy
and [3] CELI - Language Technology, Italy

ABSTRACT

This paper describes AgileRAI, a framework for searching, organising and accessing multimedia data in a fast and semantic-driven way. AgileRAI supports real-time ingestion of video streams on which different machine learning techniques (such as global and local visual features extraction and matching) are applied in a parallel and scalable way. Extracted features are matched to a reference database of visual patterns (e.g. faces, logos, monuments) in order to produce a set of meta-tags describing the ingested contents. Furthermore, these tags are semantically enriched using open semantic data repositories. The system is designed with a scale-out pattern architecture based on Apache Spark, ensuring high-performance in Big Data management environments.

INTRODUCTION

In today's digital age, television content life cycle has a very long span: after being produced and broadcast, a copy of the content, possibly enriched with metadata, is stored in the archive to be reused when needed or to be published online. In order to maximise the reuse of those assets, the ability to efficiently search, organise and access content in a fast and semantic-driven way is an asset of fundamental importance for the broadcast and media industry. However, efficient access to large-scale video assets requires content to be conveniently annotated through an intellectually expensive and time-consuming process. Semantic Web technologies can provide solutions for enriching media content and improving search as they are designed to facilitate data integration and linking over a large number of data sources. Some broadcasters already started manually tagging contents, using DBpedia as a source of tag identifiers (1). However, given the ever-growing amount of content produced every day and available from the archive, in order to make the process scalable, a way to automatically assign tags is still needed.

We describe AgileRAI, a system for multimedia content analysis and linking that addresses these challenges by providing a scalable platform for innovative multimedia archiving and production services. This platform supports real-time ingestion of multiple video streams on which different machine learning techniques (such as global and local visual features extraction and matching) are applied in a parallel and scalable way. Extracted features are matched to a reference database of visual patterns (e.g. faces, logos, monuments) in order to produce, in real-time, a set of meta-tags describing the ingested contents. Furthermore, these tags are semantically enriched using open semantic

data repositories. The system is based on Apache Spark[1] (an open source, general-purpose data processing framework), ensuring high-performance, scalability and fault-tolerance in Big Data management environments. The remainder of the paper is organised as follows. As a first step we give an overview of related work and issues on multimedia visual search and enrichment. Then, we describe the AgileRAI architecture, including the design principles as well as the use-case implemented and tested at RAI labs. Finally, we conclude the paper summing up our work and findings.

RELATED WORK

For modern broadcast and media companies, the proper organisation and management of content, including archive, footage and production material, constitutes a strategic activity. Text-search systems based on manual annotation are widely used to access those contents. However, they have been proven to be less than effective when dealing with massive amount of data. In fact, manual content annotation is costly, time-consuming and error-prone, due to e.g. subjectivity of the annotations. New technologies are needed to increase the efficiency of documentation, access and (re-) use of multimedia archives.

Visual Analysis relies on the idea of indexing and matching image and video contents based on visual characteristics, in addition to manually generated metadata. Many methods have been developed to achieve this goal, such as those for key-point feature detectors and descriptors. For this purpose, the Scale-Invariant Feature Transform (SIFT) algorithm (2) is considered a pioneer work. In 2010, the Moving Picture Experts Group (MPEG) started a standardisation initiative called Compact Descriptors for Visual Search (CDVS)[2] that provides a robust and interoperable technology to create efficient visual search applications in image databases. The core building blocks of CDVS consist of global and local descriptor extractors and compressors based on selected SIFT features. Duan et al (3) provide an overview of the technical features of the related MPEG standard. Recently the interest is moving forward to the video domain with a new activity called Compact Descriptors for Video Analysis (CDVA). Intuitively, video analysis is a more challenging problem than still images due to temporal and spatial redundancy in video, which increases the amount of data that needs to be processed. One solution to handle this problem consists in subsampling a video stream into key-frames, thus translating the problem of video analysis into a (key-frame based) image analysis task. Because of that, video summarisation techniques are crucial as well as motion feature extraction and elaboration. Most recent approaches use deep learning to address video summarisation. As an example, Zhang et al (4) proposed the use of Long Short-Term Memory (LSTM) networks for automatic key-scene detection and key-frame extraction. However, such techniques require the optimisation of many parameters, which makes them difficult to adopt in practice, on real life data.

Metadata enrichment is the process aimed at enhancing, refining and enlarging information about multimedia assets. Most techniques for metadata enrichment rely on linked (open) data repositories. Linked Data refers to datasets using unambiguous

---

[1] See http://spark.apache.org (Last accessed: May 2nd, 2017).

[2] Now ISO/IEC 15938-14

identifiers that allow elements (or relations) in different datasets to be identified as referring to the same entity. Open Data refers to data that is available for anyone to use, reuse and redistribute without a corresponding licence acquisition cost. Data can be open but not linked, or linked but not open. Thus, we refer to the intersection between Open Data and Linked Data as Linked Open Data (LOD). Raimond et al (5) proposed an approach for automatically extracting topics from speech audio and identify them with DBpedia URIs using speech recognition, text processing and concept tagging techniques (applied to a large radio archive). Although extensively in use, there is not a general agreement on which LOD repository should be adopted in practice. A discussion on how to exploit those knowledge bases according to different user needs is provided in (6). One of the major drawbacks is that they are updated every several months or years. While this might be acceptable for some applications (e.g. encyclopedic archiving), it would definitely not be acceptable in specific contexts, such as news production, which need continuous, up-to-date information. To mitigate this issue, Viana and Pinto (7) proposed a solution based on 'gamification' mechanisms for collaboratively collecting timed metadata. The results presented in their work suggest that crowd-sourced annotation can correctly describe contents and be accurate in time. However, involving users to annotate large quantities of already existing media content, is surely not a trivial task.

AGILERAI SYSTEM ARCHITECTURE

The AgileRAI system architecture merges classical information analysis and retrieval components with the most advanced fast data architectures. The high-level architecture of the system may be considered as a cluster made of multiple back-end computation nodes and a single front-end node, as illustrated in Figure 1. The system is able to analyse different kinds of input streams (e.g. RTP flows, file-based storage systems, etc.) in order to recognise specific visual patterns like logos, paintings, buildings, monuments, etc. within the ingested streams. The architecture is designed to be parallel and scalable in order to ensure near real-time, frame-by-frame pattern detection and recognition in video data. Then, recognised patterns are enriched with semantic meta-information to ease the access and allow search and retrieval that involve high-level concepts. The following sub-sections describe how the architecture building blocks are built.

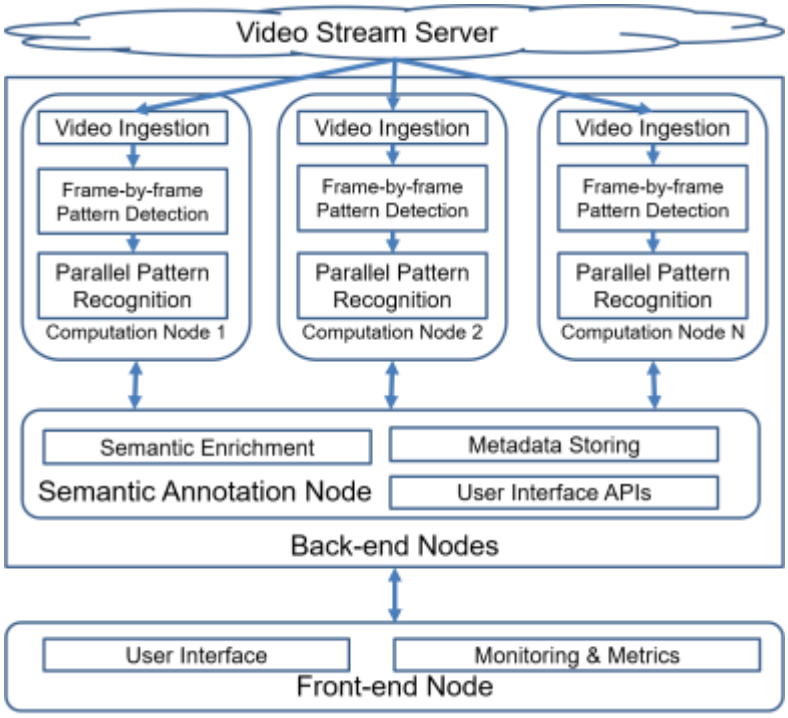

Figure 1 – AgileRAI high level architecture

Video Processing Pipeline

The video processing pipeline (i.e. back-end computation nodes in Figure 1) is based on the WASP (Wide Analytics Streaming Platform) architecture (8), an open source framework written in Scala. WASP leverages open source components, such as Akka[3] (an open source toolkit aimed at simplifying the development of concurrent applications in Java), Apache Kafka[4] (an open source platform for building real-time streaming data pipelines) and Apache Spark, to build complex, real-time big data applications. WASP allows developers to focus more on the business logic of the application, rather than on the technical, behind-the-scenes elements e.g. architecture backbone and components integration. For example, WASP is able to greatly simplify the implementation of exactly-once processing semantics, which means ensuring that each data point is processed exactly one time. This kind of guarantee is critical in many domains and very difficult to achieve.

The core abstractions of WASP are the Producers, which ingest data into the system, and the Pipegraphs, which model data flow and processing inside of the system; together they provide a flexible abstraction for complex stream processing. Producers are Akka-based and allow ingesting data from a variety of sources, with implementations available for the most common ones. Pipegraphs define pipelines as a series of processing units that read data from one or more endpoints, apply business logic to the data, and write to an output endpoint. The endpoints can be a number of storage systems from the Hadoop ecosystem, giving the users ample choice to use the right tool for each step; the processing itself can leverage Spark Streaming to scalably apply complex processing, or a lightweight streaming approach for simpler workloads. The various processing units inside a Pipegraph and between Pipegraphs can communicate through these endpoints to realise complex data flows. As well as Producers and Pipegrahs for streaming processing, WASP supports Batch jobs for periodically training machine learning models and other

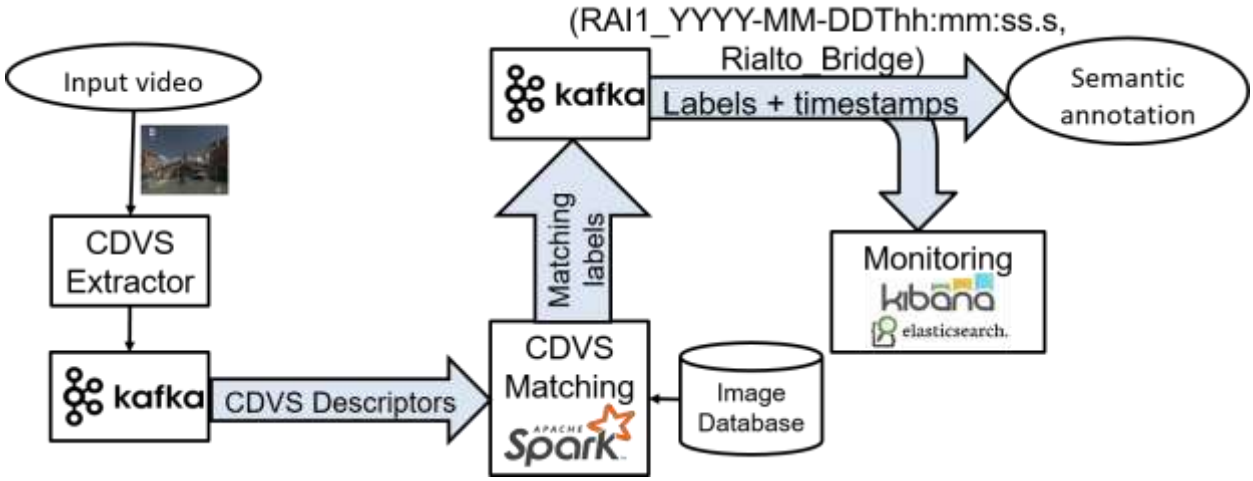

Figure 2 – AgileRAI video processing pipeline

[3] See http://akka.io (Last accessed: May 2nd, 2017).

[4] See https://kafka.apache.org (Last accessed: May 2nd, 2017).

such needs. The machine learning models built by batch jobs can be registered and versioned in a model database to be used in the Pipegraphs, enabling automatic model refresh. The entire system is orchestrated by an Akka-based actor system, enabling decoupling of the various functionalities and failure handling. Interested readers can refer to (8) for more information.

Figure 2 shows how the WASP framework has been used to implement the AgileRAI video processing pipeline. First, a Producer ingests the input RTP (or file-based) video and decodes it using the FFMPEG library in order to decode frames out of the incoming stream. Then, MPEG CDVS descriptors (3) are generated for each frame and pushed to a Kafka queue. It has to be noticed that while the current implementation integrates CDVS technology as a proof-of-concept, other machine learning tools and frameworks (e.g. face/logo detection and recognition) may be seamlessly plugged into the system thanks to the APIs provided by the WASP platform. Kafka acts as a kind of persistent, fault tolerant and distributed publish/subscribe layer, which allows decoupling video feature extraction from feature matching tasks. Then, in a Pipegraph, the descriptors queue is consumed by Spark-Streaming in order to match the incoming descriptors versus a reference database of images. These latter include visual objects of interest (e.g. paintings, buildings, monuments, etc.) under different scales, views, lighting conditions associated to unambiguous URIs of an existing Linked Data resource (e.g. `dbpedia:Rialto_Bridge`). The reference database may be updated, changed, replaced and/or modified at any time as needed. Being based on the Spark-Streaming module, the system allows to process incoming frames massively in near real-time. Furthermore, the decoupling of feature extraction from feature matching performed by Kafka, allows to stop the matching process (e.g. because the target image database is going to be updated) without the need of stopping the video ingestion and feature extraction processes. The output of Spark processing is a new Kafka queue made of tuples `(FrameID, Labels)`, where the `FrameID` is a unified identifier of the processed frame (i.e. input stream reference and timestamp data) and `Labels` is the list of labels of the matching visual patterns within the frame. Therefore, when such visual patterns are identified within the video stream, they are labelled with the same unique URI in the reference database, allowing for accessing the rich set of additional properties and relations present in the Linked Data source. The Kafka queue is finally sent to the semantic annotation node for enrichment, storing and publication, as well as delivered to an ElasticSearch cluster for monitoring purposes.

Semantic Enrichment Pipeline

The video semantic enrichment pipeline (i.e., the back-end semantic annotation nodes in Figure 1) is illustrated in Figure 3. The queue of `(FrameID, Labels)` tuples generated by the video processing pipeline is consumed by the enrichment pipeline. Since the incoming frames are labelled with the URIs of linked data resources, the system is able to access the rich set of properties and relations provided by the referenced web sources. As an example, if within the input video stream a monument is retrieved at a certain date and time with an unambiguous URI (e.g. `dbpedia:Rialto_Bridge`) the system may extract (and use as metadata for subsequent searches) all its properties like e.g. geographical location, year of construction, related external contents, etc.

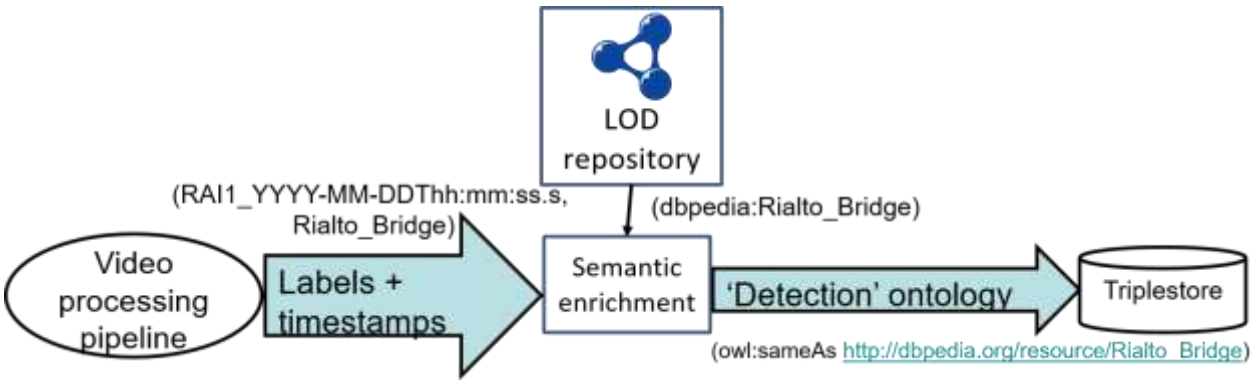

Figure 3 – Semantic enrichment pipeline

The expensive and time-consuming work of manual content annotation may be thus performed only on a (small) portion of data, e.g. when the target image dataset is created or updated, and not on the entire archived or broadcasted contents. Furthermore, the creation and annotation of the target image dataset may be automated itself, by e.g. using a Web image search engine to source visual training data for target queries (9). The collected information (i.e. source stream identifier, detection timestamps, linked data URIs) is stored as RDF triples in a triple store, in order to be semantically searchable and accessible by means of SPARQL queries. To this aim, a specific ontology was created. Figure 4 depicts an excerpt of the ontology showing some ontology classes (highlighted by circle notation) as well as some examples of their individuals (highlighted by diamond notation). The main class of the ontology is the `VisualObjectDetection` class, which is sub-class of the `Event` class defined by the W3C Time Ontology (10). Each time a visual pattern detection occurs a new individual `detection-X` of type `VisualObjectDetection` is created (e.g. `detection-0` in Figure 4). Metadata about date and time at which the detection occurred are stored by the individual `beginning-X` (e.g. `beginning-0` in Figure 4). Metadata about the type of detection (i.e. the video processing tool originating the detection, the asset in which the detection occurred, the type of object detected) are stored by the individual `action-X`. Continuing with the example in Figure 4, it may be seen that `detection-0` was generated by a visual analysis tool (in turn described by a specific `VisualAnalysis` class); detection

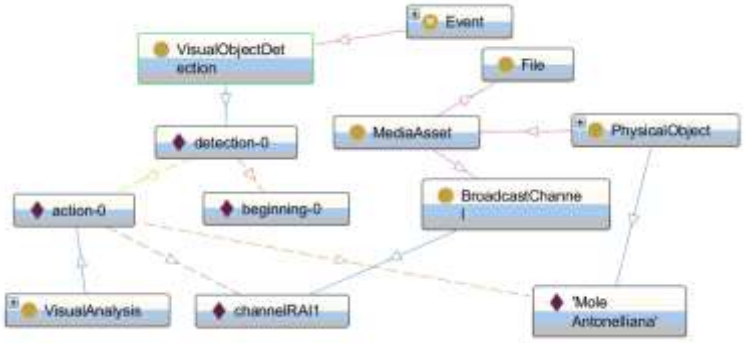

Figure 4 – Example of the AgileRAI ontology for metadata representation

occurred within the input media asset called `channelRAI1`; and the detected object was the `Mole_Antonelliana`, a building in Turin. Figure 4 shows also that the input media asset was acquired from a broadcast channel and stored to file. Figure 5 shows an excerpt of the OWL/XML representation of the described ontology individuals. Finally, the input stream is transcoded to HTML5 video formats. The transcoded video and the enriched metadata are sent to the GUI to be displayed to the user.

```
<!-- http://www.rai.it/CRIT/Detection#action-0 -->
<owl:NamedIndividual rdf:about="http://www.rai.it/CRIT/Detection#action-0">
  <rdf:type rdf:resource="http://www.rai.it/CRIT/Detection#VisualAnalysis"/>
  <Event:hasInput rdf:resource="http://www.rai.it/CRIT/Detection#channelRAI1"/>
  <Event:hasProduct rdf:resource="http://www.rai.it/CRIT/Detection#object-0"/>
</owl:NamedIndividual>

<!-- http://www.rai.it/CRIT/Detection#beginning-0 -->
<owl:NamedIndividual rdf:about="http://www.rai.it/CRIT/Detection#beginning-0">
  <timeline:atDateTime rdf:datatype="&xsd;dateTime">2017-03-19T01:01:00</timeline:atDateTime>
</owl:NamedIndividual>

<!-- http://www.rai.it/CRIT/Detection#channelRAI1 -->
<owl:NamedIndividual rdf:about="http://www.rai.it/CRIT/Detection#channelRAI1">
  <rdf:type rdf:resource="http://www.rai.it/CRIT/Detection#BroadcastChannel"/>
</owl:NamedIndividual>

<!-- http://www.rai.it/CRIT/Detection#detection-0 -->
<owl:NamedIndividual rdf:about="http://www.rai.it/CRIT/Detection#detection-0">
  <rdf:type rdf:resource="http://www.rai.it/CRIT/Detection#VisualObjectDetection"/>
  <Event:hasAction rdf:resource="http://www.rai.it/CRIT/Detection#action-0"/>
  <time-entry:begins rdf:resource="http://www.rai.it/CRIT/Detection#beginning-0"/>
</owl:NamedIndividual>

<!-- http://www.rai.it/CRIT/Detection#object-0 -->
<owl:NamedIndividual rdf:about="http://www.rai.it/CRIT/Detection#object-0">
  <rdf:type rdf:resource="&Event;PhysicalObject"/>
  <rdfs:label rdf:datatype="&xsd;string">Mole Antonelliana</rdfs:label>
  <owl:sameAs rdf:resource="http://yago-knowledge.org/resource/Mole_Antonelliana"/>
</owl:NamedIndividual>
```

Figure 5 – Example of the OWL/XML representation of metadata generated by the AgileRAI platform

Front-end User Interface and System Monitoring

A prototype GUI was developed to assist users in data visualisation and monitoring. The prototype consists in an HTML5 web page showing the live video of the input video stream being processed. A drop-down menu allows selecting between the different media assets processed by the system. Whenever a target visual pattern is detected in the stream, a set of contextual information about the visual pattern is presented on the screen. Figure 6 shows an example. The input video depicts some buildings and paintings of the cultural heritage of Italian cities. Some pictures of the same buildings and paintings were previously collected and tagged in the AgileRAI image reference dataset. Thus, the system is able to recognise their appearance within the input stream and collect further information listing e.g. their location, history, etc. Everything runs in real-time: from target detection to semantic enrichment and data visualisation. One of the most challenging aspects of the system implementation consists in ensuring a shared time-reference for all the components involved, in order to show the contextual information at the right time in the video from the perspective of the user. This means that the timestamps must be propagated all the way from the input video stream to the browser of the user and care must be taken that they are consistent across system boundaries and video transcoding; even environment details like clock skew between the machines can create problems. The backend services can be monitored and managed through the APIs, tools and interfaces provided by either the WASP platform or third parties (e.g. Spark monitoring interface, Kibana interface, etc.).

USE-CASE SYSTEM IMPLEMENTATION AND TEST

To test the system reliability and validity in real-life conditions we implemented the overall architecture on a virtual computing infrastructure. The host system is an enterprise server equipped with 80 logical cores on eight physical CPUs Intel(R) Xeon(R) @ 2.20GHz and

256 GB of RAM. The storage system is made of a fibre channel array providing about 800 GB storage space. We partitioned the host system in one virtual machine with four virtual CPUs and 4 GB of RAM for the semantic annotation and front-end nodes, and nine virtual machines each with eight virtual CPUs and 24 GB of RAM for the computation nodes. The available storage was equally partitioned among the created virtual machines. CentOS 7 64 bit was used as operating system for all the virtual machines.

The tests conducted were aimed at: (i) Demonstrating the feasibility/sustainability of a scalable system for visual analysis and retrieval based on the WASP technology; (ii) Showing the potential of the system when integrated with semantic databases.

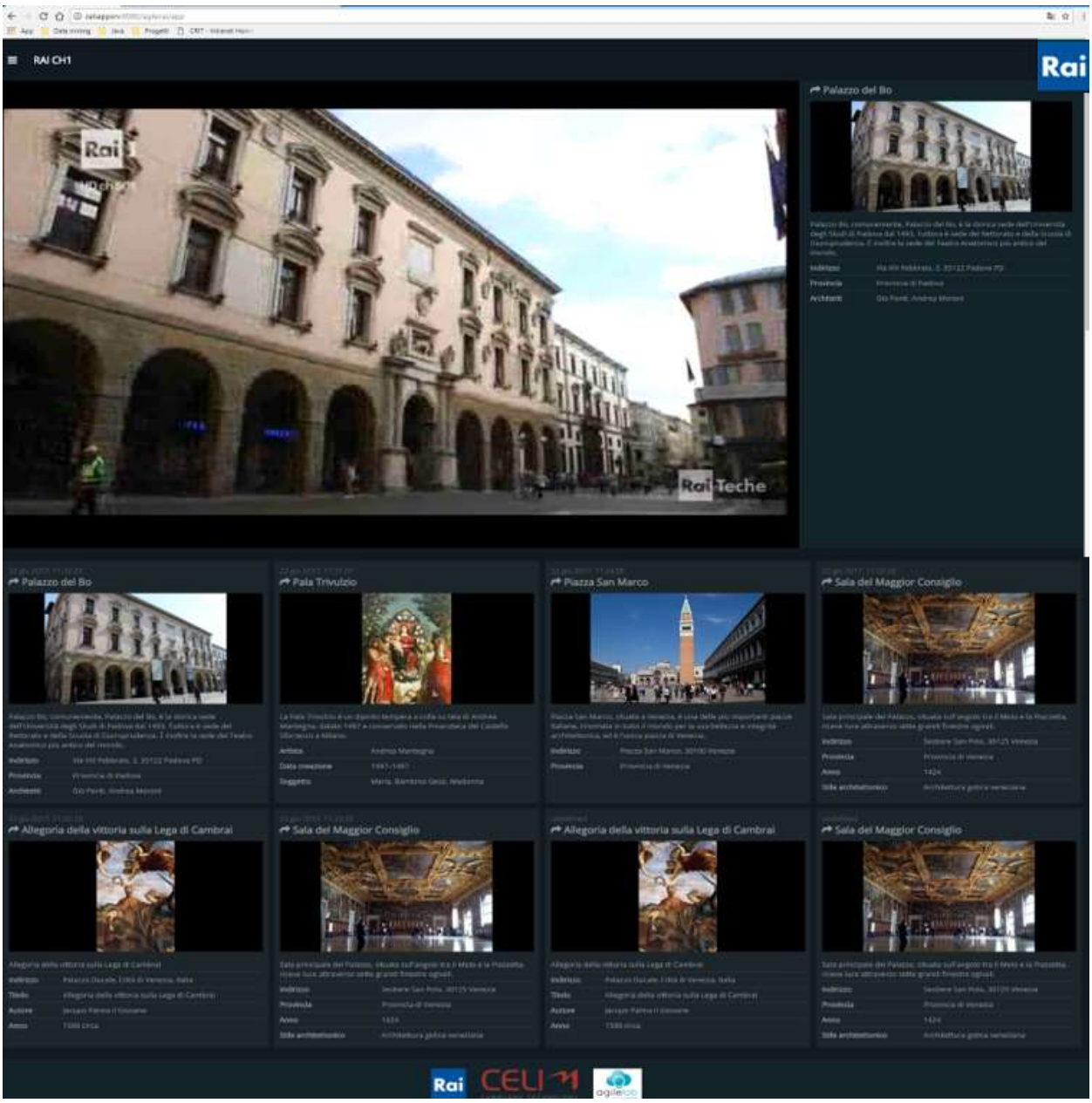

Figure 6 – AgileRAI prototype interface

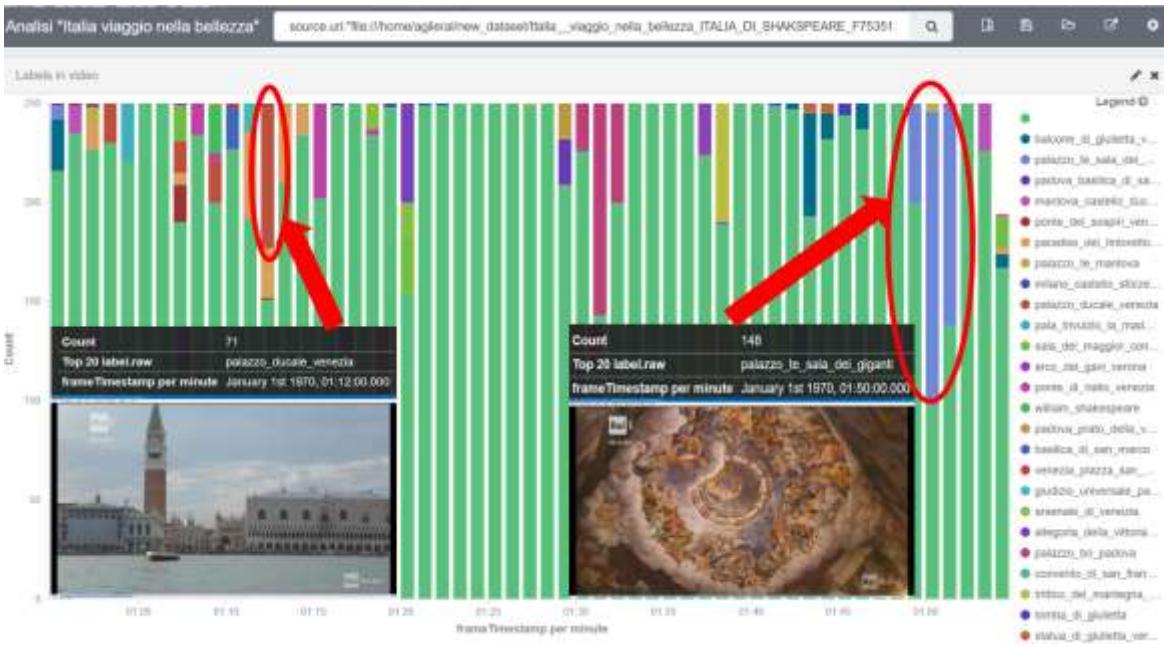

Figure 7 – Example of the interface for video browsing

Parallel computation is crucial to achieve scaling of capabilities. Thanks to the WASP framework, it is possible to manage several live stream videos in parallel and process them in a single cluster with a single application deployment. Furthermore, it is also possible to parallelise the computation down to frame level. Our tests demonstrated that, with the implemented virtual infrastructure, it was possible for each input stream to set a label detection rate up to four labels per second. This is sufficient to detect, identify and enrich relevant visual patterns appearing on video for more than 2-3 seconds. The target rate can be changed according to e.g. desired maximum latency or system resources consumed by each input stream. The use of Semantic Web technologies facilitates the process of data understanding, and therefore boosts content (re)-usability. This helps improve significantly the efficiency of content management and archiving processes, as well as increase productivity of content production and distribution workflows.

To show the potential of the system we developed two use-case scenarios that represent tangible business opportunities for modern media organisations. The first use-case (aka video browser) provides an interface to browse video content based on detected objects. The video timeline is represented as a histogram of occurrences of labels representing the detected visual patterns. This information may be used to quickly identify the main objects depicted in the video footage (e.g. buildings or paintings), thus speeding up the documentation process and reducing efforts of human annotators. Figure 7 shows an example where it is highlighted that the analysed video contains a view of the Doge's Palace in Venice between minute 12 and 13, and some fresco paintings located in Palazzo Te of Mantua from minute 50 to minute 53. The second use-case (aka dynamic semantic tagger) provides an interface (see Figure 6) that automates the process of enhancing broadcast programmes with rich metadata published on the Web through Linked Open Data datasets. This technology may allow broadcasters to add value to their archives at low cost through the re-use of existing and extracted metadata. Furthermore, it may

provide richer interactive television experience to the audience by e.g. delivery of enhanced, interactive contents through companion screens.

CONCLUSIONS

We presented AgileRAI, a platform that combines visual analysis techniques for the detection of visual patterns with Semantic Web technologies, for the identification of detected patters and their enrichment. The system uses the open source WASP framework to ensure scalability and tolerance in real-time, massive data management tasks. Implementation of two use-case scenarios demonstrated the feasibility and potential viability of the approach.

REFERENCES

1. Eggink J. and Raimond Y., 2013. Recent advances in affective and semantic media applications at the BBC., 14th Intl. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), pp. 1 to 4.
2. Lowe, D. G., 2004. Distinctive Image Features from Scale-Invariant Keypoints, Intl. Journal of Computer Vision, vol. 6. pp. 91 to 110.
3. Duan, L-Y., Chandrasekhar, V., Chen, J., Lin, J., Wang, Z., Huang, T., Girod, B. and Gao, W., 2016. Overview of the MPEG-CDVS Standard. IEEE Trans. Image Process. vol. 25. pp. 179 to 194.
4. Zhang, K., Chao, W.-L., Sha, F. and Grauman, K., 2016. Video summarization with long short-term memory. Proc. European Conf. on Computer Vision. pp. 766 to 782.
5. Raimond Y., Lowis C., Hodgson R., and Tweed J., 2012. Automated Semantic Tagging of Speech Audio. Proc. of the 21st Intl. Conf. on World Wide Web, pp. 405-408.
6. Färber, M., Bartscherer, F., Menne, C. and Rettinger, A. 2016. Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. Semantic Web Journal Special issue on Quality Management of Semantic Web Assets (Data, Services and Systems).
7. Viana, P. and Pinto, J.P., 2017. A collaborative approach for semantic time-based video annotation using gamification, Human-centric Computing and Information Sciences, 7:13. doi:10.1186/s13673-017-0094-5.
8. Agile Lab S.r.L., 2017. Wide Analytics Streaming Platform. GitHub Repository, https://github.com/agile-lab-dev/wasp (Last accessed: May 2nd, 2017).
9. Chatfield, K., Arandjelović, R., Parkhi, O. M. and Zisserman, A., 2015. On-the-fly Learning for Visual Search of Large-scale Image and Video Datasets, Intl. Journal of Multimedia Information Retrieval, vol. 4(2), pp 75 to 93.
10. Pan, F and Hobbs, J. R. 2004. Time in OWL-S. Proceedings of the AAAI Spring Symposium on Semantic Web Services, Stanford University, CA, pp. 29 to 36.