Enhancing and Re-Purposing TV Content for Trans-Vector Engagement

Deliverable 1.2 (M20)

Data Ingestion, Analysis and Annotation

Version 2.0

This document was produced in the context of the ReTV project supported by the European Commission under the H2020-ICT-2016-2017 Information & Communication Technologies Call Grant Agreement No 780656

DOCUMENT INFORMATION

Delivery Type	Report
Deliverable Number	1.2
Deliverable Title	Data Ingestion, Analysis and Annotation
Due Date	M20
Submission Date	August 31, 2019
Work Package	WP1
Partners	CERTH, MODUL Technology, webLyzard
Author(s)	Konstantinos Apostolidis, Nikolaos Gkalelis, Evlampios Apostolidis, Alexandros Pournaras, Vasileios Mezaris (CERTH), Lyndon Nixon, Adrian M.P. Brașoveanu (MODUL Technology), Katinka Boehm (webLyzard)
Reviewer(s)	Basil Philipp (Genistat)
Keywords	Data Ingestion, TV Program Annotation, TV Program Analysis, Social Media Retrieval, Web Retrieval, Video Analysis, Concept Detection, Brand Detection
Dissemination Level	PU
Project Coordinator	MODUL Technology GmbH Am Kahlenberg 1, 1190 Vienna, Austria
Contact Details	Coordinator: Dr Lyndon Nixon (nixon@modultech.eu) R&D Manager: Prof Dr Arno Scharl (scharl@weblyzard.com) Innovation Manager: Bea Knecht (bea@zattoo.com)

Revisions

Version	Date	Author	Changes
0.1	30/6/19	V. Mezaris, K. Apostolidis	Created template and ToC
0.2	18/7/19	L. Nixon	Initial input (Section 2)
0.3	26/7/19	L. Nixon	Completed inputs (Sections 2 and 3)
0.4	27/7/19	K. Apostolidis, E. Apostolidis	Added parts for Sections 4.1, 5 and 6
0.5	27/7/19	A. Pournaras	Added parts for Section 6
0.6	29/7/19	A. Brasoveanu	Started NER/NEL text for Section 3
0.7	30/7/19	N. Gkalelis	Input for Section 4.2
0.8	31/7/19	K. Apostolidis	Revision of Sections 4 and 5
0.85	1/8/19	K. Boehm	Checked and completed SKB descriptions
0.95	14/8/19	L. Nixon	Revised sections and checked all MODUL input
1.0	19/8/19	A. Brasoveanu	Finished NER/NEL text with evaluations
1.5	22/8/19	B. Philipp	QA review
2.0	27/8/19	K. Apostolidis, L. Nixon	Post-QA updates

Statement of Originality

Contents

EXECUTIVE SUMMARY

This deliverable is an update of D1.1. It covers the topics of content collection across vectors, annotation and knowledge graph alignment, concept-based video abstractions, and brand detection. Specifically, on content collection across vectors it reports on the current status of the collection process including quality management. Concerning annotation and knowledge graph alignment, it updates on the implementation of an accurate NLP & NEL pipeline for annotating the collected data with respect to keywords and Named Entities and aligning annotaions to our Semantic Knowledge Base (SKB) as well as external knowledge sources. With respect to concept-based video abstractions, it presents a fast learning-based method for video fragmentation to shots and scenes, a new deep learning architecture for improved concept detection that is suitable for learning from large-scale annotated datasets such as YouTube8M, and an extended set of concept pools that are supported by the ReTV Video Analysis service. Concerning brand detection, this deliverable discusses the newly-adopted method for brand (logo) detection in ReTV, the extended pools of brands that were specified, and a method that was developed for advertisement detection. Finally, this document also presents in detail the ReTV WP1 Video Analysis REST service, which implements the methods for concept-based video abstraction and brand detection in a complete, integration-ready software component.

ABBREVIATIONS LIST

Abbreviation	Description
API	Application Programming Interface: a set of functions and procedures that allow the creation of applications which access the features or data of an application or other service.
DCNN	Deep Convolutional Neural Network: a type of artificial neural network.
EPG	Electronic Program Guides: menu-based systems that provide users of television with continuously updated menus displaying broadcast programming or scheduling information for current and upcoming programming.
HTTP POST/GET	Types of method in the Hypertext Transfer Protocol (HTTP). The HTTP POST method is used to send data to a server to create/update a resource. The HTTP GET method is used to request data from a specified resource.
IPTV	Internet Protocol Television: is the delivery of television content over Internet Protocol (IP) networks.
JSON	JavaScript Object Notation: a data-interchange format.
LSTM	Long Short Term Memory networks: a type of recurrent neural network.
MTL	Multi-task learning: a field of machine learning in which multiple learning tasks are solved at the same time, exploiting commonalities and differences across tasks.
NEL	Named Entity Linking
NER	Named Entity Recognition
NLP	Natural Language Processing: subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human (natural) languages.
OTT	Over The Top: content providers that distribute streaming media as a standalone product directly to viewers over the Internet, bypassing telecommunications that traditionally act as a distributor of such content.
RDF	Resource Description Framework: a method for conceptual description or modeling of information that is implemented in web resources.
REST	Representational State Transfer: an architectural style that defines a set of constraints to be used for creating web services.
RNN	Recurrent Neural Network: a type of an artificial neural network.
SKB	Semantic Knowledge Base: a RDF-based triple store for a knowledge representation of keywords and entities during in annotation of documents
TVoD	Transactional Video on Demand: a distribution method by which customers pay for each individual piece of video on demand content.
URL	Uniform Resource Locator: a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

1. Introduction

This deliverable reports on the work done from month M11 to M20. In this reporting period all tasks, (i.e., T1.1: Content Collection Across Vectors, T1.2: Concept-Based Video Abstractions, T1.3: Brand Detection in Video, and T1.4: Annotation and Knowledge Graph Alignment) were active.

Firstly, the content collection was started by MODUL regarding an initial set of TV broadcasters and TV series, using the annotation model described in D1.1 of ReTV [60]. Specifically, we both crawl the Web for news and TV related content as well as collect social media content via TV related channels and a list of terms for public posts by users (Section 2). All collected data is annotated according to keywords and Named Entities, supported by our own Knowledge Graph (Section 3). We report on the current status of the data collection and annotation pipeline including quality management for online data sources and an evaluation of our Named Entity annotation accuracy. In the direction of video fragmentation and annotation, CERTH developed new technologies for learning-based video fragmentation to shots and scenes, and a new deep learning architecture for improved concept detection that is suitable for learning from large-scale annotated datasets such as YouTube8M (Section 4). Alongside this, the set of concept pools that are supported for concept detection was expanded. State-of-the-art object detection frameworks were tested for brand recognition. We report (i) the selection of the framework that best suits the project's requirements, (ii) the extended pools of brands whose detection is supported, and (iii) a method that was developed for advertisement detection (Section 5). The ReTV WP1 Video Analysis service, which implements the methods for concept-based video abstraction and brand detection in a complete software component accessible via REST calls, is presented in Section 6, where the service's APIs are documented and usage instructions and examples are provided. The deliverable concludes (Section 7) with a brief summary and outlook.

2. Content Collection Across Vectors

2.1. Status

The ReTV data collection is a vital contribution to subsequent data analyses, predictive analytics and content recommendation work. We build up a significant body of TV-related content in the Metadata Repository of partner webLyzard through the set-up, configuration and running of various 'data mirrors' (components for the query and retrieval of data from different vectors) - each collecting at regular intervals online documents matching their search configuration on the pre-defined Websites or platforms. The collected documents can be searched and browsed in the TVP Visual Dashboard. In this chapter, we report on the status of the data collection by source and language.

Since Web and social media data retrieval requires a clear scope (no-one can or wants to download the entire Web or an entire social media platform for data analysis) we identified the TV broadcasters and TV series of interest for an initial ReTV data collection implementation. We started with a "seed" list of 25 TV series with unambiguous titles and some expected online discussion; these are reproduced below:

Program title	*Broad-caster*	*Genre (if known)*	*Country*	*Language*	*Current broadcast schedule*
Heimatjournal	RBB	Travel	DE	DE	Sa 1900
Gute Zeiten Schlechte Zeiten	RTL	Soap Opera	DE	DE	Mo-Fr 1940
Germany's Next Topmodel	Prosieben	Reality	DE	DE	Thu 2015
Bauer ledig sucht	3+	Reality	CH	DE	Do 2015
C'est ma question!	RTS Un	Quiz	CH	FR	Mo-Fr 1820
Berlin direkt	ZDF	Politics	DE	DE	So 1910
Auslandsjournal	ZDF	Politics	DE	DE	Wed 2215
Abendschau	RBB	News	DE	DE	Daily 1930
Rundschau	SRF1	News	CH	DE	Wed 2005
Schweiz Aktuell	SRF1	News	CH	DE	Mo-Fr 1900
Le 19h30	RTS Un	News	CH	FR	Mo-Fr 1930
PULS4News	PULS4	News	AT	DE	Daily 1845 & 2000
zibb	RBB	Magazine	DE	DE	Mo-Fr 1830
Café PULS	PULS4	Magazine	AT	DE	Mo-Fr 0530-0900
rbbPraxis	RBB	Health	DE	DE	Wed 2015
Unkraut	BR	Environ-ment	DE	DE	Mo 1900 (fortnightly)
Game of Thrones	RTL2	Drama	DE	DE	Currently not being broadcast
DOK	SRF1	Documen-tary	CH	DE	various
Täter Opfer Polizei	RBB	Crime	DE	DE	Wed 2100, So 1900
SUPER.MARKT	RBB	Consumer	DE	DE	Mo 2015
Young Sheldon	Prosieben	Comedy	DE	DE	Mo 2045
taff	Prosieben	Boulevard	DE	DE	Mo-Fr 1700
Brandenburg aktuell	RBB	News	DE	DE	Daily 1930
Big Bang Theory	Prosieben	Comedy	DE	DE	Mo 2015 (new episodes)
Sandmännchen	RBB	Childrens	DE	DE	Daily 1755

While these already covered 10 broadcasters in the DACH region (Germany, Austria, Switzerland) we also expanded our broadcaster lists to all primary channels in those 3 countries for consideration. We collected Websites and social media channels for the broadcasters and TV series we had listed, as well as drew up term lists for unambiguous references to them in public social media content.

This resulted in a list of 105 Radio/TV websites, however we found only 42 contained exclusively Radio/TV programming related content. TV series had always their own sub-sections within the TV channel Website. For the remaining 63 Websites - dubbed 'hybrid' - we found that almost all had separate sections for their programming and for public interest news, and some had additional sections which needed blacklisting from a Web crawl (e.g. schedules which are available anyway via our EPG feed). So we manually examined each hybrid Web site to determine matching URL patterns to crawl for news, TV/Radio content and to blacklist (not include in the ingested Web documents). Our news Web document collection, already running since previous projects, used 213 Websites from the DACH region (German language), is now extended by the news sections of the 63 hybrid Websites; there are also 191 Websites in our English language collection. Our new TV/Radio Web document collection is crawling 19 Austrian, 45 German & 25 Swiss (German) websites after testing the crawler for extracting successfully the main text of pages.

In social media, we have focused on the TV series and the channels they are broadcast upon, both to ensure an effective focus in the data on the "seed" content and to avoid any issues with API limits: 39 Facebook channels, 31 Twitter channels and 29 YouTube channels were identified and added to the respective data mirrors. We also use a list of terms in social media to query for any public content mentioning a seed TV program or broadcaster. We have 27 broadcaster terms and 36 TV program terms for both English and German language content.

The scale of data collection can be seen via the TVP Visual Dashboard. As reported in D1.1, this also includes the EPG data from ZATTOO. We present here the number of documents collected from 17 September 2018 (launch of ReTV data collection) up to 10 July 2019 (296 days), both as a total (rounded to the nearest hundred) and a daily average (rounded to the nearest integer).

	EN total	EN avg	DE total	DE avg	FR total	FR avg
News (Web)	3.4 mil	11 486	4.3 mil	14 527		
TV/Radio (Web)			84 600	286		
Twitter	11.4 mil	38 514	361 100	1 220	391 500	1 323
Facebook	900	3	70 700	239	40	0
YouTube	7 300	25	7 300	25	80	0
EPG			118 900	402		

Since both content partners RBB (Germany) and GENISTAT / ZATTOO (Switzerland) prioritize German, our data collection is also largely focused on documents in this language. All TV/Radio websites being crawled currently are in the German language. Twitter is a clear exception, as queries for global TV series such as Game of Thrones or Big Bang Theory return large numbers of English language tweets. With regard to Facebook, we collect only the posts on the listed TV related Pages which are for a German speaking audience - regardless there are occasional posts in English too; a few posts are classified as French due to the use of largely French terms (e.g. "Tour de France Video!"). Our YouTube data mirror is configured to collect equally videos from EN and DE as a response to how the YouTube API limit works: in the initial configuration, as API responses do not clearly differentiate content by language and therefore the language detection is handled on our side, we typically collected majority English language content up to the API limit at the expense of having other language content in our repository. Therefore we implemented a 'split-evenly' mechanism to ensure that there was a fairer split in the video collection by language. The EPG data, as can be seen from the table, provides us with 400 documents on average per day, where each document is one EPG entry, with channel, time, program title and description.

2.2. Updates from Previous Report

All of the data collection described above in the Status section was either started or planned by the publication of D1.1 (October 2018). Since then, besides regular quality checks (see next sub-section), we also installed a Redash [1] instance to visualise the data collection by each mirror. Fig. 1 shows the data collection volume by mirror for 2019 (January-July), where a set of peaks can be clearly seen - these are English language tweets (tv_twitter_en) and match the weekly broadcast times of the last series of Game of Thrones, showing the massive online interest in this TV program.

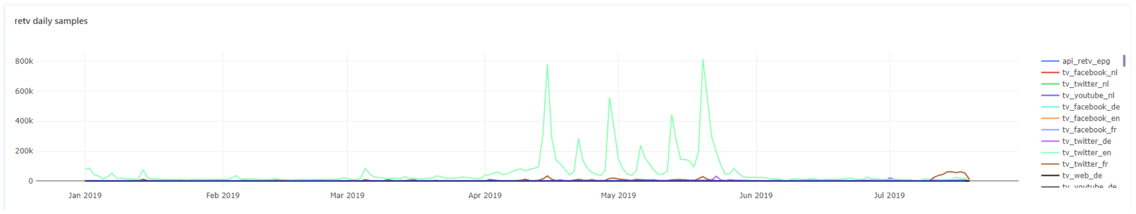


Figure 1: Visualisation of data collection by mirror in our Redash instance

Recently, according to a request from the use case partner NISV, we also added Dutch language sources. This also required updating our data ingestion pipeline (NLP/NER) to support the Dutch language so that keyword detection could be applied to the new documents. Based on a list of 13 public Dutch TV broadcasters [2] and a shortlist of 9 Dutch TV programs, we added 46 News and 21 Radio/TV Web sources to our Web crawl (12 sites are overlapping as hybrid sites); in social media we added 24 Facebook Pages, 27 Twitter accounts and 20 YouTube channels; we monitor 44 terms on public social media. Collection started on 7 July 2019 and we are observing at present the following totals and daily averages in document collection (data up to 23 July 2019 - 16 days, rounded to the nearest hundred):

	NL total	NL avg
News (Web)	27 900	1 744
TV/Radio (Web)	5 500	344
Twitter	12 400	775
Facebook	900	56
YouTube	150	9

[1] An open source tool to query and visualize data `https://redash.io`
[2] `https://en.wikipedia.org/wiki/Dutch_public_broadcasting_system#List_of_broadcasters`

2.3. Evaluation of Data Quality

After the initial launch of each data mirror, we monitored at regular intervals the collected content for quality and relevance. This led to occasional changes to the configuration of a data mirror, such as removing or changing a term, adding or removing channels or correcting the Web crawler set-up to extract the right text from a page on a certain Web site. This subsection outlines our data quality checks.

The social media mirrors (Facebook, Twitter, YouTube) generally work as expected. The predefined channels can be manually checked prior to addition to a mirror, e.g. the last few posted content items. Frequency of posting is not an issue but long dormant channels are removed as the queries generally cost something towards the API limit even if no content is returned. High frequency channels could be an issue as we need to respect API limits but generally our content collection, due to limiting to shortlists of broadcasters and TV programs, works within the standard limits. What is important is that the usual postings on the given channels contain a minimum amount of text (for the analysis; a channel that e.g. just posts links or video embeds without text would not be valuable for our collection) and that the usual topic is relevant for the ReTV analytics (either as news content or a radio/TV programming related content). Regarding the term lists, the volume of data that may be collected is more variable with peaks when certain terms (e.g. a TV program) experience a spike in online interest (e.g. the final episodes of both Game of Thrones and Big Bang Theory generated such spikes). For Twitter, we use the real time Streaming API to catch tweets that contain our terms and it is quite generous in terms of API limits. YouTube has become an issue at times, with the API limit being reached, thus our mirror caps the number of responses per query so that a single term does not dominate the collected video links (as would, e.g. Game of Thrones). We have sampled social media content at different times to check the relevancy of returned content with respect to our term lists. Already in the formulation of those lists, we take ambiguity into consideration by searching manually for each planned term on Twitter and YouTube, and checking the relevance of the top 20 documents to the intended use of the term. For example, RBB may be for us the German broadcaster Rundfunk Berlin Brandenburg but finds other uses in social media, such as being an often cited EP by globally popular K-Pop band Red Velvet (as an acronym of 'Really Bad Boy'). The various possible references to RBB particularly reduce our relevance in YouTube searches, and we do not use RBB alone as a term, preferring unambiguous n-grams such as 'rbb abendschau' or synonymous usage like 'blossnichtlangweilen' on Twitter (RBB's official hashtag). As such, we have not had to make any subsequent corrections to our term lists to date.

The Web mirror needs a configuration per Website and can be less predictable in advance, since there is a difference between a human browsing of a broadcaster's site and a Web crawl by a robot tasked to only extract the actionable content for analysis. Following the launch of the Web content collection, we checked the first 1000 documents in each mirror (TV/Radio sites in DE, AT and CH respectively). The following outlines the different cases that can be encountered:

- No pages from a Website – the crawler needs rules for where to start the crawl and how to follow links (target pages have to match a given URL pattern). If no pages are being crawled, this configuration is checked and corrected.
- Irrelevant pages from a Website – some pages appearing in the sample are not directly related to TV/radio programming (a common example is a contact form). As above, we seek to correct the Website configuration, usually by adding such pages to a blacklist.

- Pages from the Website belong in a different mirror – we separated crawls for news content from Radio/TV content so that also the data can be analysed separately. Some sites were found to be incorrectly categorized, especially some TV/Radio sites also carried news content but had not been categorized initially as hybrid, with their news content added to the news Website mirror.
- No relevant content from a Website – some sites proved to be returning content that was not relevant for an analysis, e.g. we found several radio station Websites which only have static pages describing their scheduled shows in very general terms (such as "The best of pop and rock from 7 to 9pm"), whereas we seek regularly updated descriptions of the content of the shows. Such Websites were removed from the mirror.
- Relevant content not extracted correctly – for the textual analytics, for each Webpage we store in the metadata repository the title (generally the Webpage title) and a description (the relevant text extracted from the page). Issues can arise where the title or descriptive text is not correctly extracted due to some non-standard approach in the Webpage structure. For example, we found that the page titles extracted from `https://tv.orf.at` were initially the program start times, as these were given as a title for the inline article text, rather than the HTML title, where ORF was using meta tags with Facebook's Open Graph 'title' property. Some sites returned for their descriptions text such as 'this site needs JavaScript' because they were making this browser JavaScript check when the crawler was accessing them and not getting the expected response (that confirmed JavaScript support), so the crawler was redirected to such pages instead of the article it was intended to crawl.

When we encounter such cases, we register them as issues in an internal tracking software (Gitlab) until they are resolved.

2.4. Outlook

Our data collection pipeline is active in 4 languages, across 3 social media platforms and crawling several hundred Websites as well as daily EPG data since September 2018. ReTV clients now have access to over 7 million news documents and over 12 million TV/radio specific documents that can be browsed and analysed in the TVP Visual Dashboard. The data collection is regularly quality checked and drives our data annotation and analytics, contributing to the prediction and recommendation services.

3. Annotation and Knowledge Graph Alignment

3.1. Status

The ReTV annotation model and the associated Knowledge Graph for alignment of entities has been introduced in deliverable D1.1 [60]. Binary videos, as released to ReTV for the video summarization and publication work, are analyzed and annotated as outlined in the following chapter. Here, we focus on the annotation of the TV related content collected by our data mirrors as described in the previous chapter: Web, social media and EPG data. Our data ingestion pipeline consists of two annotation services for all of the collected documents:

- the NLP (Natural Language Processing) pipeline extracts a short list of keywords from the text of the document;
- the NER/NEL (Named Entity Recognition and Linking) pipeline aligns textual object mentions with identifiable Named Entities such as Persons, Organisations and Locations.

Every document passing the pipeline is annotated with keywords (n-grams) provided enough meaningful text is present; while no minimum is specified, a text of tweet-sized length will probably produce 3-6 keywords. Web documents tend to be of longer textual content and could return a large number of keyword candidates, therefore keyword extraction caps the number of keywords at 10. All candidates are ranked by statistical significance with respect to a reference corpus for the domain, with the 10 most significant keywords used in the annotation.

Whether a document is annotated with entities will depend on the content, as entities are a much smaller vocabulary space than keywords - while a keyword can be any linguistic formulation, an entity (or more specifically, Named Entity (NE)) is specifically a real world thing that has an officially recognized and widely understood name that can be used to identify it. A formal definition may be taken from the 'rigid designator' in the philosophy of language (term from Saul Kripke), which determine that a Named Entity is only any referent which designates the same thing in all possible worlds. So in saying that 'Donald Trump is the president of the USA', we would say that the referent *Donald Trump* is a NE since it refers to the same thing in all possible worlds but 'president' is not, as the thing it refers to can differ according to the different worlds, i.e. at different time periods or in different locations. Without pursuing any further the philosophy of language, we can say that Named Entities for computer-based recognition tasks tend to be taken from the classes of Persons, Organizations and Locations. Since the names (or labels) given to NEs in natural language may be ambiguous (e.g. *Michael Jackson* as a person name might refer to the music star or 44 other well-known persons according to the Wikipedia disambiguation page), computer-based NER aims to allocate different identifiers to distinct NEs, and the NER community has moved towards globally shared identifiers using the web infrastructure, since a web URL should resolve to the same resource for any user from any location (and, arguably, at any time if URL persistence is maintained). Wikipedia became a popular choice since it would give every resource it described a separate article at an unique URL: *Michael Jackson*, the singer, is found at `https://en.wikipedia.org/wiki/Michael_Jackson` whereas *Michael Jackson*, the former head of the British Army, is found at `https://en.wikipedia.org/wiki/Mike_Jackson_(British_Army_officer)`. Thus the task became not just to correctly identify the reference to an entity in the text (*Michael Jackson* refers to a Person entity), but also link it via identifier to the correct NE from a shared vocabulary such as Wikipedia. In recent years, there has been an emergence of online, public Knowledge Graphs (KGs) where concepts are represented as graph nodes with unique URIs as identifiers that can be resolved by human or machine agents to acquire additional metadata about them. Two of the best known and most generic in terms of conceptual coverage are DBPedia `https://dbpedia.org` - a KG representation of Wikipedia - and WikiData `https://wikidata.org` - a crowdsourced approach to structured metadata about concepts.

Recognyze [95] is a graph-based NER/NEL engine that supports multiple Knowledge Bases (e.g., Wikidata, DBpedia, Wikipedia), languages or annotation styles (e.g., NER, NEL, take longest available string, abbreviations, etc). Since each entity type can have different type of name variants (e.g., abbreviations, aliases, multilingualism, hypocorism or partial matches), the support for multiple annotation styles became the defining feature of the latest version. Recognyze is now the core of a larger ecosystem dedicated to almost all the Knowledge Extraction tasks related to entities:

- The *NER* pipelines are dedicated to the extraction of NEs and their types. In general such pipelines are external to Recognyze and we provide support for Spacy [3] or Stanford's Core NLP [4] for various use cases, if needed.
- The *NEL* pipeline represent the core of Recognyze and is dedicated to linking candidate entity mentions to their respective entry from a KB. Some of the implemented disambiguation algorithms include: Louvain clustering, HITS and name analyzers [95]. Currently it supports multiple KBs and languages.
- *Data enrichment* - An engine called Jairo is built on top of Recognyze in order to provide additional details about the identified entities.
- *Benchmarking* - Since assessing the performance of a NEL system is a complex issue due to a large number of errors in KBs or gold standards [13] a set of components for running visual evaluations was designed jointly with HTW Chur during Recognyze implementation. The available package called Orbis [67] is available as an open-source tool.
- *Error analysis* - In order to improve Recognyze, the errors detected during evaluations are collected and analyzed.
- *Slot Filling* (SF) is defined as a sequence learning problem in which you are required to fill all the available information from a particular entity from a given text (e.g., for a movie you might be required to also specify its director, screenwriter or stars, if these snipets of information are available). Dedicated SF components can be easily built on top of Recognyze.

The generated annotations - associated keywords and found entities - are stored with the rest of the document metadata in the metadata repository (an Elastic Search index maintained by webLyzard) according to the annotation model presented in D1.1 for RelatedContent. Keywords are associated to the entire document by the wl:key property. Entities are referenced as part of the textual fragment where the entity was found (associated to the document by the wl:sentence property).

Entity references use DBPedia as a vocabulary space - this is a Knowledge Graph representation of the information stored in Wikipedia, where each article (HTML webpage) is converted into

[3] `https://github.com/explosion/spaCy`

[4] `https://github.com/stanfordnlp/CoreNLP`

a resource (graph node). These resources have associated metadata which is also extracted from the articles, particularly from the structured information known as the infoboxes. This metadata can be used in the data analysis (e.g. selecting documents with a certain entity type or entities with certain characteristics) so we replicate the resources from DBPedia which are used in ReTV annotations in a local Knowledge Graph which we have named the Semantic Knowledge Base (SKB). This is more efficient than relying on querying the public DBPedia endpoints. The SKB uses the RDF data model, stores the data in a persistent Triple DataBase (TDB) from Apache Jena and runs on an instance of Apache Fuseki to provide an HTTP interface to the data (read/write) [5]. A synchronisation service has been implemented between the metadata repository and the SKB so that when a new entity is annotated to a document in the repository, that entities DBPedia resource is replicated to the SKB. Apart from the entities, the SKB is also used to store keywords as resources in the knowledge graph. Keywords are represented in a lexical model where multiple forms of a keyword in natural language can be grouped into a single lexeme [6]. This allows us to capture alternative and grammatical forms of the keyword (e.g. singular and plural, British and American English spelling or verbs in various tenses) as well as cross-lingual translations, supporting both keyword disambiguation (aligning different written forms to a single keyword) and multi-lingual document search by keyword.

The original lexical data was obtained from OmegaWiki, an open source, free to use, multilingual dictionary, which provides word senses and the base word forms for our supported core languages. This data was enhanced with additional word forms through NLP processes, such as lemmatization and stemming, and the results were cross-checked with our document repositories. Additional information, such as POS tags, spelling differences and common misspellings were attached retrospectively with the aid of external resources, such as Wikipedia.

3.2. Updates from Previous Report

3.2.1. NLP

Our NLP pipeline has been extended with support for the Dutch language. We added support for Dutch stopwords [7] and tested the keyword extraction. Our Dutch partner NISV provided additional feedback since they are Dutch native speakers, confirming the correct functioning of the NLP pipeline.

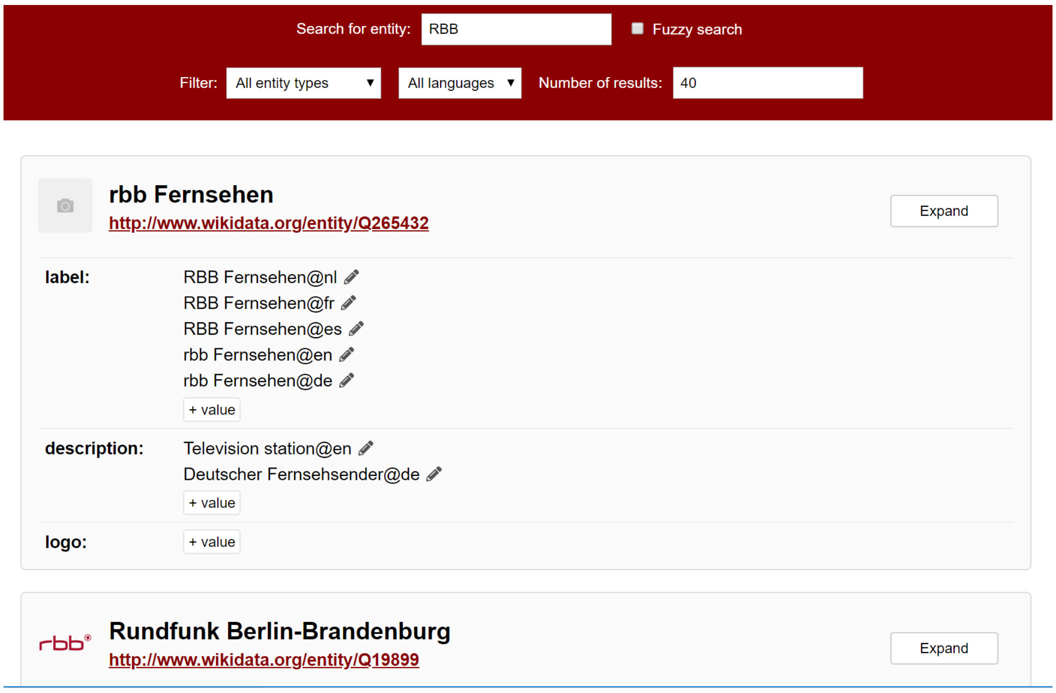
3.2.2. SKB

We have developed a Web interface for searching and browsing resources in the SKB, known to us as the SKBBrowser. It allows us to simply search within the SKB for entities by substring matching on property values ('search by entity name'), by the entity's unique identifier ('search by URI') or specifically across entities of type Event ('search for events') which allows additionally a date range restriction. The screenshot shows a search for 'RBB' across entities, and note how we have both a WikiData entity for RBB the TV channel and a separate entity for RBB the German public broadcaster (Organisation) (Fig. 2).

[5] `https://jena.apache.org/documentation/fuseki2/`

[6] The set-up was reported in the InVID project, Deliverable 2.3 'Social media filtering and extraction, preprocessing and annotation, final version', Section 2.2 'Keywords (Semantic Knowledge Base)'

[7] `https://github.com/stopwords-iso/stopwords-nl/blob/master/stopwords-nl.txt`


Search for entity: RBB
Fuzzy search
Filter: All entity types | All languages | Number of results: 40

rbb Fernsehen
http://www.wikidata.org/entity/Q265432

label:
RBB Fernsehen@nl
RBB Fernsehen@fr
RBB Fernsehen@es
rbb Fernsehen@en
rbb Fernsehen@de
+ value

description:
Television station@en
Deutscher Fernsehsender@de
+ value

logo:
+ value

Rundfunk Berlin-Brandenburg
http://www.wikidata.org/entity/Q19899

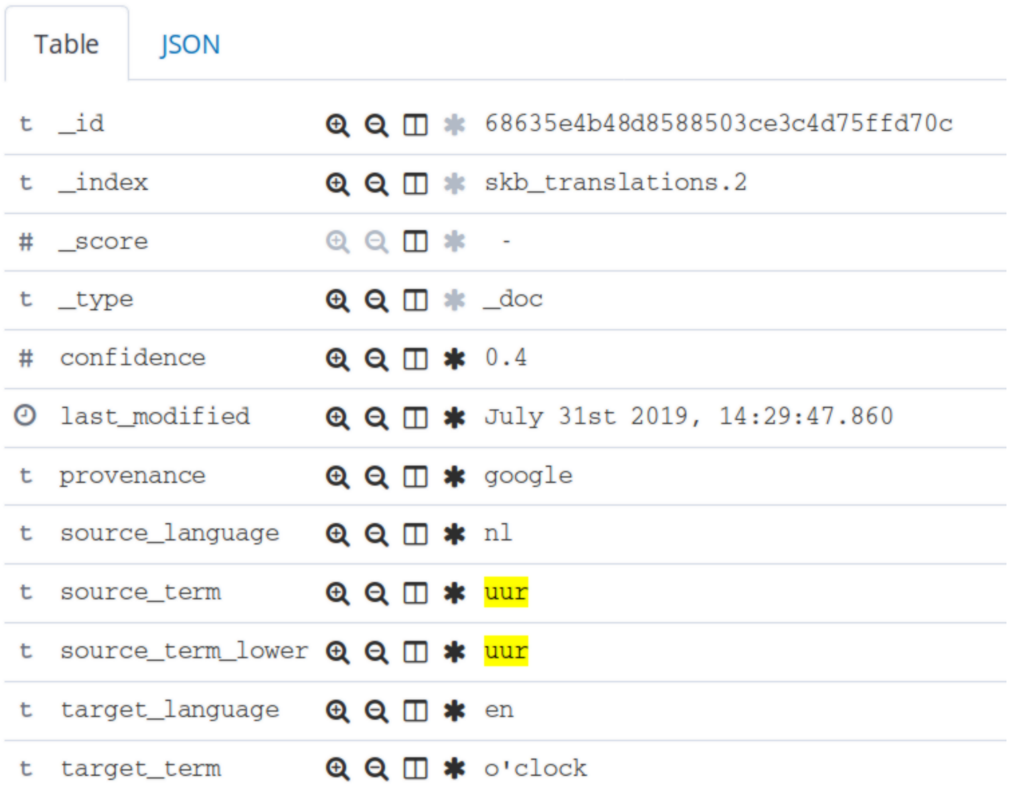
Figure 2: SKBBrowser top results for search 'RBB'

The SKB grows as new keywords or entities are annotated to documents in the metadata repository. Significantly, Dutch was added as a language meaning we had to add the Dutch forms of keywords as a new source-language for each lexeme. Since the TVP Visual Dashboard supports an English or German language user interface, when Dutch content is analyzed, the resulting keywords must be translated into either English or German for display in the UI. Fig. 3 shows an example translation entry in the SKB for the Dutch keyword 'uur' which would be translated in the English UI as 'o'clock'. Direct translations of keywords and tags that are shown in the dashboards are retrieved through the Google Translate API and subsequently cached to provide fast and uniform translation results.

The presence of both keywords and entities in the SKB led us to plan how to use the combination of this information to improve the keyword detection and to address an overlap in annotation when the same text was annotated as both a keyword or an entity. Another case, since keywords are purely lingustic, was that keyword annotations overlapped with the referents of Named Entities, for example we might have the entity 'Donald Trump' and also the keywords 'donald' and 'trump' in a document. This led us to combine keyword and entity annotations, so that in such a case the document should have simply the annotation 'Donald Trump (entity)'. Keywords not aligned to entities are retained, and we refer to them as NEKs (Non-Entity Keywords). This alignment process has significantly improved the document annotation accuracy, and through the SKB we have additional metadata available about both entities and keywords.

3.2.3. Recognyze NER/NEL

Recognyze was not built as a general Knowledge Extraction pipeline, but rather like a focused NEL engine. This means its development was guided by practical considerations, as bad annotations can be taxing on the performance of the tools that use it as an annotation engine



t	_id	68635e4b48d8588503ce3c4d75ffd70c
t	_index	skb_translations.2
#	_score	-
t	_type	_doc
#	confidence	0.4
	last_modified	July 31st 2019, 14:29:47.860
t	provenance	google
t	source_language	nl
t	source_term	uur
t	source_term_lower	uur
t	target_language	en
t	target_term	o'clock

Figure 3: A sample keyword in the SKB with the source term in Dutch and the English translation given as the target term

(e.g., a wrong annotation will carry a double penalty when computing an F1 score, as it will be counted both as an entry that is missing from the gold standard while simultaneously being an added or "extra" entity that is not in the gold). The initial focus was on high-precision extraction of the mentions and links that belong to the three core entity types: Person (PER), Location (LOC) and Organization (ORG). Recognyze itself only provides the following information about an entity mention: (i) the surface form of the mention, (ii) the span (the start and end markers), (iii) the type and (iv) the link. In time, it has become apparent that additional information or other types of content might also be needed. For example, it is not enough to simply link an entity to a DBpedia or Wikidata entry, as this might not help us understand if it is the correct entity, therefore displaying additional information on the discovered entities (e.g. text abstract, thumbnail picture) can often help. Also multiple types of entities might need to be annotated, and entity results might also be important for other tasks like slot filling (e.g., queries like *find all movies by Christopher Nolan* or *show the main stars of The Big Bang Theory*).

Some of the tasks that need to be performed when adapting Recognyze for a new project include:

- building a set of lexicons and profiles for the new languages or entity types;
- building a profile that provides additional information about the new entity types with another tool from the Recognyze ecosystem called Jairo;
- fine-tuning after evaluating the results on one or multiple corpora both from the literature or extracted from production data;

- building a graphical interface for newly added entity types for display into products powered by the webLyzard Platform like the TVP Visual Dashboard.

For ReTV, several new language profiles were built, including English, German, and Dutch. A Recognyze profile contains a series of lexicons built from various KBs (e.g., all German entities from DBpedia, all German entity names from Wikidata) and a reference to the disambiguation algorithm that will be used on those lexicons in order to achieve the desired results. Many existing corpora are based on DBpedia and in order to provide a fair set of evaluations, research profiles of Recognyze provide DBpedia links, as translations between KBs are still prone to errors. Translating between various Knowledge Bases for evaluation purposes can be done with a third-party Python tool[8] if the respective corpora are in the NIF format, but currently many of the good NEL corpora still throw a number of formatting errors when usinng this tool.

3.3. Evaluation of Annotation Quality

In this section, we outline our evaluation of Recognyze for entity-based annotation of general purpose and TV and media related documents.

3.3.1. Recognyze General Evaluation

While working on the Recognyze ecosystem, we have also developed an explainable evaluation tool called Orbis [67], as already mentioned in Section 3.1. This evaluation tool was built around a practical evaluation philosophy we have called **MERA** which stands for:

- (i) **Measurability** - Orbis provides widely recognized metrics (e.g., precision, recall, F1, accuracy, etc).
- (ii) **Explainability** - Most evaluation types from Orbis are also doubled by explained evaluation runs (e.g., we should not only be able to see the results, but also the classification into test results like false positives or false negatives or even into more fine-grained error classes if possible). Explanations can also include visual analysis methods (e.g., drill-down analysis, multiple synchronized charts, etc) for inspecting and debugging the results.
- (iii) **Reproducibility** - Whenever possible (e.g., if published or mentioned in the articles related to the various evaluated tools), Orbis aims to provide reproducible settings for the annotator tools, as well as for the annotation tasks.
- (iv) **Adaptability** - Orbis provides support for updatable resources and versioning (e.g., in order to allow an evaluation to run with a previous or recent version of a KB, for example DBpedia 3.9 or DBpedia 2018-04).

Since Orbis was designed with the idea of also helping developers improve their NEL systems, evaluation results are displayed in parallel with the gold standard results, as it can easily be seen in Figure 4. This often helps discover errors in both gold standards or automated annotators (systems) results.

Data sets. The following data sets were selected for the general: (i) *Reuters128* [75] - a set of texts extracted from the classic Reuters corpora; (ii) *OKE2015 Task 1* [65] and *OKE2016*

[8] https://github.com/wetneb/nifconverter

Table 1: Comparison of the system performance on multiple corpora.

Corpus	System	P	R	F_1
Reuters 128	AIDA	0.53	0.43	0.47
	Babelfy	0.32	0.22	0.26
	Spotlight	0.50	0.49	0.49
	FREME	0.57	0.15	0.24
	Recognyze	**0.58**	**0.58**	**0.58**
OKE 2015	AIDA	0.50	0.41	0.45
	Babelfy	0.40	0.26	0.32
	Spotlight	0.61	0.36	0.45
	FREME	0.60	0.21	0.31
	Recognyze	**0.73**	**0.59**	**0.65**
OKE 2016	AIDA	0.62	0.40	0.49
	Babelfy	0.64	0.47	0.54
	Spotlight	0.61	0.36	0.45
	FREME	0.58	0.26	0.36
	Recognyze	**0.79**	**0.61**	**0.69**

Task 1 [66] - focused on short biographical sentences extracted from Wikipedia. We selected these data sets due to a good mix of old (Reuters) and new entities (OKE), as well as due to the fact that they cover a relatively solid spectrum of categories, from news, to politics, sports and biographies.

Evaluated tools. Table 1 compares Recognyze's performance to several popular NEL services that offer publicly available APIs [9]. AIDA [40], Babelfy [62] and Recognyze [95] use KG disambiguation techniques, while Spotlight [18] and FREME use statistical disambiguation. It has to be noted that each service builds its entity graph differently, therefore, not only the NEL algorithms, but also the differences between KGs can lead to variation in the results. AIDA is based on Wikipedia and, therefore, operates on a substantially different KG than the other tools. Babelfy uses the Babelnet KG and provides DBpedia links via the *owl:sameAs* property. Spotlight, FREME and Recognyze both draw upon DBpedia, although Spotlight and FREME are fine-tuned for knowledge extraction tasks, whereas Recognyze is optimized for NEL and various domain specific extraction tasks. Recognyze has performed well due to our focus on precision. Recall can definitely be improved, as for the OKE data sets it lagged significantly behind precision.

3.3.2. Recognyze Media Annotations

As opposed to the classic entity types in NEL evaluations (Person, Organization and Location), annotation for media-related documents pose specific challenges. In addition to the core entity types like *Person*, *Organization* and *Location*, media-related document annotation also needs to support a fourth large class of entities: *Creative Work* [10] or *Work*. This entity class encompasses a large selection of entities that might be classified as creative works, from books and songs, through games, movies, TV Shows and entire media franchises. It is important to

[9]Since no recommended settings for performing evaluations on Reuters128 and OKE datasets have been published, we have dedicated approximately two days to experimental optimization of the evaluation settings of all evaluated third-party tools.

[10]represented by `http://dbpedia.org/ontology/Work` (abbreviated as dbo:Work) in DBpedia or `https://schema.org/CreativeWork` in the schema.org vocabulary

note that, if we leave the temporal attributes aside (e.g., new positions for a person, key people for a company, new episodes for a TV show), there is a lot of variation when it comes to the main attributes of this entity type as opposed to the three core types (Person, Organization, Location). A TV show might have some executive producers, a production company, some actors starring in it, as well as a set of episodes, each with their own list of directors, writers or stars; a book will have some author(s), publisher, and awards or links to a book series; and a song might have an interpret, author, music producer, and so on. As it can be seen, it is difficult to find common attributes between the various sub-classes, except for the fact that they are all types of creative works that were published in some format or medium in a certain period of time. It can even be argued that all these creative works should be modeled as their own entity types, but in order to perform such a fine-grained extraction, it is important to first identify the large class to which the entities belong. Adding works can also lead to a high number of false positives, as often fictional characters (e.g., *James Bond, Harry Potter*) might share names with real people, as well as with their own media franchises which can encompass different sets of series (e.g., books, tv shows, movies, comics, etc); fictional characters might be based on real people (e.g., see the recent trend of music biopics based on *N.W.A., Queen* or *Elton John* or TV shows like *Narcos* who often fictionalize real characters by changing their names or changing the events in which they participate) or the name of a work is later used for a different work (e.g., again the example of music biopics is relevant).

Another big issue of evaluating the quality of media-related document annotations refers to the fact that there are currently no good corpora for such evaluations. While classic corpora are quite good for identifying people, organizations or locations, there are less good corpora to help with evaluating works (e.g., books, TV shows, music, etc) or events. Due to this we decided to create a corpus[11] focused mostly on works to help us fine-tune our media-related document annotations. We have started by collecting several sentences from the Wikipedia abstracts of 100 media entities. The initial set of entities contained books, TV shows, media companies, YouTube influencers and media franchises. Several entity types were annotated, include Person (PER), Organization (ORG), Location (LOC), Work (WORK) or Other (OTHER). The corpus was annotated by two annotators following an annotation guideline, and was later judged by a third annotator. The resulting corpus was exported into multiple formats, including csv and NIF.

Table 2: Comparison of the system performance on the media corpora.

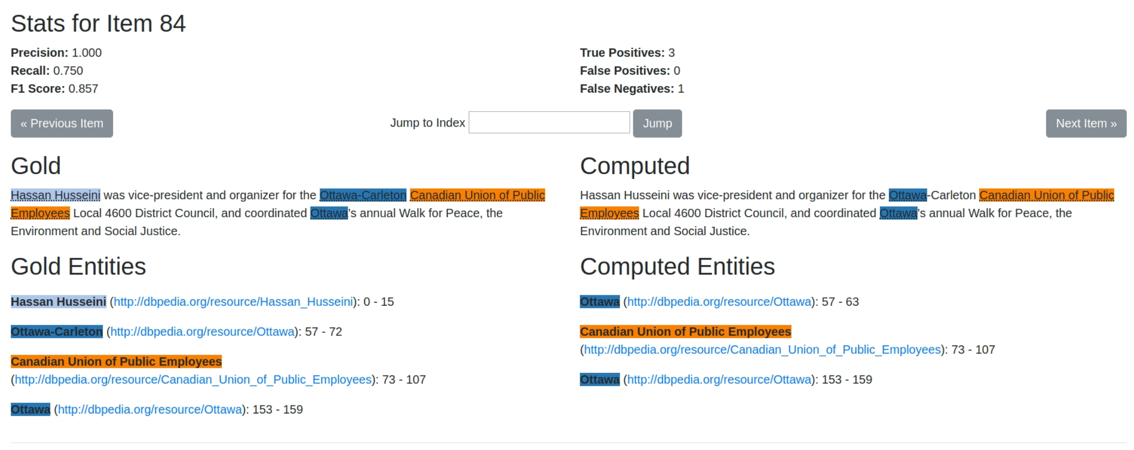
Corpus	System	P	R	F_1
	AIDA	0.46	0.49	0.48
	Spotlight	0.49	**0.59**	0.54
Media	Recognyze	**0.56**	0.55	**0.56**

The evaluation was again performed with Orbis. Babelfy and FREME results were not included as they returned too many errors. The results are somewhat in line with the general corpora, however much lower than the results on OKE data sets. This was expected due to the large amount of error classes introduced by adding works to the corpora. Also as expected, tools were not able to distinguish well between a character and the franchise that bears its name or offer good results on the YouTube influencer section of the corpus. The influencer section is especially difficult due to the fact that some of the works mentioned there (e.g., YouTube channels that were shut down or early gigs for famous influencers) are NIL, therefore not included in Wikipedia or related KBs like DBpedia or Wikidata. While some media franchises

[11] The corpus will be made publicly available through the GitHub page of MODUL Technology

are well-covered by Wikipedia (e.g., *Harry Potter*) and related KBs, others are not. In such cases a good solution to improve coverage and results might to use Linked Data extracted from dedicated wikis like *Memory Alpha* (covering *Star Trek* franchise), *Wookieepedia* (covering *Star Wars*) or *Marvel Database* (covering both *Marvel Comics* and the *Marvel Cinematic Universe*). Most of the fandoms organizes around such wikis and also many of them are published through *Fandom*[12] or similar wiki engines. Some of the information from these wikis is also collected in Linked Data form through DBkwik [39]. We plan to use such a source for both extending the current corpora and creating new profiles for Recognyze in the future, if data quality is good.

3.3.3. Discussion



Stats for Item 84

Precision: 1.000
Recall: 0.750
F1 Score: 0.857

True Positives: 3
False Positives: 0
False Negatives: 1

Gold

Hassan Husseini was vice-president and organizer for the Ottawa-Carleton Canadian Union of Public Employees Local 4600 District Council, and coordinated Ottawa's annual Walk for Peace, the Environment and Social Justice.

Computed

Hassan Husseini was vice-president and organizer for the Ottawa-Carleton Canadian Union of Public Employees Local 4600 District Council, and coordinated Ottawa's annual Walk for Peace, the Environment and Social Justice.

Gold Entities

Hassan Husseini (http://dbpedia.org/resource/Hassan_Husseini): 0 - 15

Ottawa-Carleton (http://dbpedia.org/resource/Ottawa): 57 - 72

Canadian Union of Public Employees (http://dbpedia.org/resource/Canadian_Union_of_Public_Employees): 73 - 107

Ottawa (http://dbpedia.org/resource/Ottawa): 153 - 159

Computed Entities

Ottawa (http://dbpedia.org/resource/Ottawa): 57 - 63

Canadian Union of Public Employees (http://dbpedia.org/resource/Canadian_Union_of_Public_Employees): 73 - 107

Ottawa (http://dbpedia.org/resource/Ottawa): 153 - 159

Figure 4: Explainable evaluation with Orbis. Gold standard ("Gold" panel with OKE2016 data) and system results ("Computed" panel with Recognyze results) are visualized side by side.

Annotation Styles. It has to be noted that in some cases there might not be a correct way to annotate a certain entity as illustrated in Figure 4. In this example from the OKE2015 data set, the text *Ottawa-Carleton Canadian Union of Public Employees* can be annotated as (i) *Ottawa-Carleton*, (ii) *Canadian Union of Public Employees*, (iii) *Ottawa-Carleton Canadian Union of Public Employees*, or (iv) quite possibly with an even more expanded annotation that also includes *Local 4600 District Council*. Similarly it can be argued that *Ottawa's annual Walk for Peace* should be an annotation that identifies a single recurring event. Since the results also depend a lot on the annotation guidelines of each data set, we can argue that these annotation guidelines should be openly accessible in a machine readable format (e.g., NIF, Turtle) in order to standardize evaluations and provide better comparisons between tools.

Overlap. Overlaps tend to appear in cases related to LOC and ORG entities. Quite often, an overlap is identified in long names such as *Chattanooga State Technical College* or *City University of New York Graduate Center*. In similar cases a surface form expansion that will contain the longest possible string should correctly match the entity from the gold standard. Complicated cases like the following: *Loyola's University in Belgium, Economics* (from OKE2016) can be interpreted in multiple ways. This example can either be rendered as (i) one long entity that corresponds to the whole string; (ii) one entity that describes the University

[12]https://www.fandom.com/

(*Loyola's University in Belgium*); (iii) two entities (*Loyola's University* and Belgium); (iv) two entities again (*Loyola's University in Belgium* and a string *Economics*); even (v) three entities (*Loyola's University* and *Belgium* and *Economics*). The phrasing of the examined sentence suggests that the fourth version is the correct one. Without a thorough text analysis such instances are extremely difficult to disambiguate for both humans and machines.

KB evolution. During evaluations we have always used the online versions of the tools. This comes with the disadvantage that some of the tools might not be up to date. Unfortunately current generation of annotators do not provide information related to the KB version or the entity index that was used. This means that in some cases entities that were marked as NIL (only surface forms were identified, but no link), might be available in new KB versions (e.g., entities that were not included in DBpedia 3.9 or 2014-10, might be included in DBpedia 2018-10). Since we have used the latest available KB builds for Recognyze, this partially explains the better results. Since the OKE data sets do not include NIL entities, this issue only affects the Reuters128 evaluation.

Lenses. Partially the problem of KB evolution and copora degradation can be counteracted by publishing a new corpora version every few years. Alternatively, if a KB dissappears (e.g., a problem already experienced by those who used Freebase) translations to newer KBs (e.g., DBpedia or Wikidata) can be provided. If, however, the issues go deeper (e.g., all the previously mentioned issues encountered together) or if there is a need to provide custom annotations focused on a certain domain, it might be a better idea to provide an alternate annotation set which we generically call a lens. We developed a theoretical treatment of lenses that was described in a publication that was accepted at RANLP 2019 [94]. Additionally annotations that show to fix various errors encountered in OKE and Reuters corpora were published on GitHub [13].

3.4. Outlook

Our NLP/NEL pipeline is being continually improved. In NLP, we plan to extend our keyword detection to support other part of speech (POS) classes besides nouns, starting with verbs and adjectives (including their declensions through our SKB). For NER/NEL, new entity types have been added: *Works*, *TVShows (TV)* and Broadcasters. Since we aim to use SKB as our primary source for all Knowledge Extraction activities, from NER/NEL to sentiment or relation extraction, the next development cycle will be focused on building a third set of profiles around SKB entities. Since this graph will merge entities from multiple KBs, it will solve most problems related to entity coverage for the various entity types we currently cover. Recognyze's next development cycle will be focused on: (i) improving recall without sacrificing precision and on providing; (ii) providing fine-grained annotations (e.g., sub-types for the bigger classes); (iii) improving error analysis capabilities in Orbis and using this new features to enhance future Recognyze profiles; and (iv) tighter integration with SKB. Some future Recognyze profiles might also include several upper-level types like *Music*, *Movies* or *Games*, as well as some more fine-grained types like *Fiction* or *Non-fiction* (e.g., especially for TV or Works types).

The SKB sources its initial resource descriptions (entities and keywords) from external sources. Sometimes information is missing, out-of-date or incorrect. Up to now we had to manually correct SKB data when we encounter such issues programmatically, working on the data directly. To make this workflow more efficient, we are developing a Web based SKB Editor to

[13]https://github.com/orbis-eval/corpus_quality_paper

allow direct data correction in the browser, similar to how the information in WikiData may be edited. We keep storage of the original data and modifications, to be able to track, and in the future also provide, the history of changes. The user need only select an entity or keyword and choose an Edit button to access the selected metadata about it and update values as necessary. The prototype extends the previously described SKBBrowser with Edit functionality (this is already visible in Fig. 2) and syncs changes with our core SKB. We use a provenance model to not delete previous triples but capture the information that they have been updated. Every triple has a provenance source and last_modified date associated with it. Edits from the SKBBrowser are labelled as such. Generally, the most recent edit of a triple is the one used in query responses, and edits from the SKBBrowser are given priority over automated updates (e.g. using the latest WikiData dump). This is because we assume edits from the SKBBrowser to be expert user edits (edit capability will only be given to authenticated users from client organisations) and trust their accuracy over the external knowledge sources. This editing layer could also be integrated into the TVP Visual Dashboard so that (trusted and autheticated) users may directly correct underlying entity or keyword information.

4. Concept-Based Video Abstractions

4.1. Video Fragmentation

4.1.1. Updated Problem Statement and State of the Art

The problem of video fragmentation and the relevant literature survey are presented in detail in D1.1 of ReTV [60]. In this section we discuss and address issues that came up after the delivery of D1.1, i.e., in months M11 to M20; specifically, the identification of the video fragmentation step as the bottleneck of the Video Analysis (VA) service of WP1, and how we improved this step.

The VA service implements a pipeline that consists of multiple stages. We have timed each of these stages and calculated the time of each stage as a ratio to the total duration. Specifically, the stages of the VA service's execution are the following: first a video URL is submitted and the video file is downloaded by following the URL. The video is temporally segmented to shots, scenes and sub-shots, and a keyframe is selected for each segment. Then, the feature extraction module analyses the selected keyframes of each sub-shot and extracts: (i) a set of handcrafted features, (ii) the concept probabilities for all adopted concept pools, and (iii) the object detection results (i.e., object class probability and bounding box) for all adopted object detection pools. Subsequently, the ad detection module utilises the object detection results regarding the channel logos pool and certain other features analysed in Section 5.3 to detect if a shot is an advertisement or actual content. Finally, a post-processing stage pools the sub-shots features to the respective shots and scenes the sub-shots belong to, saves the segmentation data as well as the pooled features to compressed binary files and writes the final video analysis results JSON file. In Table 3 we report the timing for each of these stages. The execution times stated here are the average values obtained over more than 20 sessions of video analysis for videos of multiple types, duration and heterogeneous content.

We observe that the most time-consuming stages of the VA service are the video segmentation and the object detection ones. As described in D1.1, for the video fragmentation to shots we adopted the method of [3]. This relies on the comparison of color histograms and local feature descriptors, namely the SURF interest point detector and descriptor [10]. The extraction and the matching of local descriptors is a slow process compared to the more modern image feature extraction methods that are based on Deep Convolutional Neural Networks (DCNN) [101] and take place in specialized hardware, i.e., GPUs. Since the video temporal fragmentation is the slowest process in the VA service's pipeline, in the past months we focused our efforts towards reducing the VA service's execution time, on replacing the used video temporal fragmentation method with one that exhibits faster execution times while achieving the same (or even higher) accuracy. We aimed at replacing the fragmentation to shots and scenes and we continue to use the adopted method of [5] for the fragmentation to sub-shots as described in D1.1, since this relies on the results of shot segmentation and is quite fast, i.e., less than 5% of the analyzed video's duration as reported in [5]. Regarding the object detection stage utilized for the brand detection task of ReTV, we have provided the option to turn-off this stage for a specific session to speed up the processing, since there are scenarios were the results of brand detection are not needed.

For a complete review of the literature on video fragmentation methods, the reader is referred to D1.1 of ReTV. There are numerous new methods that rely on variants of color histograms to assess neighboring frames similarity [25, 92, 43], on local binary pattern features [49], or

Table 3: The stages of the VA service's pipeline and the average execution time of each stage expressed as a ratio to the total processing time for a video

Processing stage	Sub-stage	Processing time (%)
Video download		0.02
Video segmentation		0.35
Handcrafted features extraction		0.05
Concept detection		0.26
	Image pre-loading	0.02
	SIN concept pool	0.12
	ImageNet concept pool	0.02
	Places365 concept pool	<0.01
	Intermediate layers of Places365	<0.01
	SaF concept pool	<0.01
	YT8M concept pool	0.10
Object detection		0.31
	Image pre-loading	0.01
	Brand logo detection	0.21
	Channel logo detection	0.09
Ad detection		<0.01
Features post-processing and saving		0.01

local features [33]; yet, in this deliverable we focus on recent video fragmentation methods that are DCNN-based and thus utilize modern hardware to speed up the processing time. In [32] a simple model architecture with four 3D convolution layers is presented. The model is trained using a dataset that the authors gathered and manually annotated, however, this dataset is not publicly available. In [34] a new spatio-temporal convolutional neural network architecture is presented. The video is divided into segments of 16 frames with an overlap of 8 with three different segments being simultaneously analysed. Each segment is fed to a different 3D CNN. The output of these 3D CNNs serves as input to an SVM and labels are assigned. A filtering process follows in which consecutive segments with the same labeling are merged. The false alarms of gradual transitions are reduced through a histogram-driven temporal differencing. The most recent DCNN-based video fragmentation is presented in [90] where an even more complex cascaded analysis framework is presented. First, a fast method is employed to filter most transitions. This is achieved using an adaptive threshold on the similarity of a center frame to its neighbors, indicating whether the content in the frames exhibits drastic changes. Then, the candidate transitions are further fed into a strong cut transition detector to filter out false cut transitions. Finally, for the remaining center frames which have negative responses to the cut detector, the search area is expanded with frames on both forward and backward temporal directions to form candidate segments. Finally, all these segments are fed to a gradual transition detector to detect and locate the gradual transitions. The authors argue that the whole framework is designed in a cascaded way and thus the computation is light except for the last stage of the gradual transition detector. Another contribution of this work is the compilation of ClipShots, the first large-scale annotated database for shot boundary detection, which is made publicly available. To our knowledge this method presents the best accuracy, with marginal gains compared to the method of [34].

Regarding video fragmentation methods to scenes, most methods rely on clustering features of shots [8, 7]. In [6] clustering is employed too, but on features extracted from a siamese DCNN which simultaneously analyses visual and textual information from transcripts, while in [9] a framework of deep multimodal networks that process semantic, visual, textual and audio features is presented. In Section 4.1.4 we analyse the performance of the previously discussed recent DCNN-based methods for video fragmentation to shots and scenes, in terms of speed and accuracy, and we compare them to our proposed method.

4.1.2. Fast DCNN-based Video Fragmentation

Since our main goal is the speed-up of the video fragmentation stage of the VA service, we adopted and extended the "Ridiculously Fast Shot Boundary Detection with Fully Convolutional Neural Networks" method [32]. The extensions concern three directions: (i) a post-processing stage, (ii) a trick to use a larger temporal window that the network sees without affecting speed, and (iii) the use of transfer learning to also solve the problem of scene segmentation.

In [32] the deep model described labels each frame of the video with a probability of being a shot transition frame. However, little to no information is given in the paper on how the shot transition probabilities are translated to hard decisions on whether a shot-transition exists between the given frames. Instead of relying on simply thresholding the frame transition probabilities, to come up with the shot boundaries, we used a post-processing stage similar to the technique used in [3] to analyse the computed similarity scores between consecutive frames. Specifically, the time series formed by the shot transition probabilities is first smoothed using a moving average filter with a temporal window of 5 frames. Then, the first order derivative of the smoothed time series is calculated to discover the local minima and maxima. Each discovered local maxima is considered a shot transition.

In [32] each frame prediction is based on a context of 10 frames. By using a model that is fully convolutional in time, we can increase the input size and thus make, e.g., 11 predictions by analyzing 20 frames or 91 predictions by analyzing 100 frames, etc., thus minimizing redundant computation. However, it is a reasonable assumption that there are gradual transitions in a video that can be hard to be detected using a temporal content of just 10 frames. Thus, increasing the temporal window may positively affect on the performance of the video fragmentation method. Instead of analysing 10 consecutive frames, we choose to analyse frames using a quadratic incremental step, i.e., the inference of our model for frame with index x is based on the analysis of frames with indices $x-8$, $x-4$, $x-2$, $x-1$, x, $x+1$, $x+2$, $x+4$, $x+8$, $x+16$. This way, we allow our model to look at a larger temporal window while still using 10 frames as input to our model (i.e., the time efficiency remains unaffected).

As a final extension, and drawing inspiration from the transfer learning applications on which DCNNs excel, we tested the approach of using the same method on the scene segmentation problem, i.e., fine-tuning the same model on a dataset with annotated scene transitions. This time, we select one keyframe (the middle frame) from each shot, and use the keyframes of shots as input to the scene model. We also examined the possibility of using as a keyframe of each shot the average image of all its frames. The evaluation of all approaches regarding both the shot and scene segmentation model is discussed in the Section 4.1.4.

4.1.3. Implementation Details and Use

We used the Keras[14] open-source neural-network library version 2.1.2, with the TensorFlow backend[15] version 1.12 for designing and training our DCNN models. All experiments where conducted on an Intel i5 3570K machine equipped with an Nvidia GTX 1080 Ti GPU.

4.1.4. Results

The video shot and scene annotated datasets used in our experiments are the following:

- BBC Planet Earth [6]: contains ground truth shot and scene annotations for each of the eleven episodes of the BBC Planet Earth educational TV Series.
- RAI dataset [8]: Ten randomly selected broadcast videos from the RAI Scuola video archive with shot and scene transitions. The videos are mainly documentaries and talk shows. Shots and scenes have been manually annotated by a set of human experts to define the ground truth.
- ClipShots [90]: ClipShots is the first large-scale dataset for shot boundary detection covering more than 20 categories, including sports, TV shows, animals, etc. In contrast to previous commonly used shot boundary detection datasets, e.g. TRECVID and RAI, which only consist of documentaries or talk shows where the frames are relatively static, this dataset was constructed with short videos from Youtube and Weibo. Many short videos are home-made, with more challenges, e.g., camera shaking and large occlusions. The gradual transitions in this dataset include dissolve, fade in/fade out, and sliding-in/sliding-out.

In our experiments we tested the following methods/combinations:

- RFShot+PostProcess_BBC: We used the BBC Planet Earth dataset for training the model architecture proposed in [32] with the post-processing stage discussed in Section 4.1.2 for shot segmentation.
- RFShot+PostProcess_ClipShots: We used the ClipShots dataset for training the model architecture proposed in [32] with the post-processing stage discussed in Section 4.1.2, for shot segmentation.
- RFShot+PostProcess+QStep_BBC: We used the BBC Planet Earth dataset for training the model architecture proposed in [32] with the post-processing stage extension as well as the quadratic incremental temporal step extension discussed in Section 4.1.2, for shot segmentation.
- RFScene-MID+PostProcess_BBC: We used the scene annotations of the BBC Planet Earth dataset for training the model architecture proposed in [32] with the post-processing stage discussed in Section 4.1.2, for scene segmentation. The input to our model is the middle frame of each shot.
- RFScene-AVG+PostProcess_BBC: We used the scene annotations of the BBC Planet Earth dataset for training the model architecture proposed in [32] with the post-processing

[14] `https://keras.io/`
[15] `https://www.tensorflow.org/`

stage discussed in Section 4.1.2, for scene segmentation. The input to our model is the average of all frames of each shot.

- RFFineScene-MID+PostProcess_BBC: We used the shot annotations of the BBC Planet Earth dataset for training a shot transition model architecture as in [32] with the post-processing stage discussed in Section 4.1.2 (i.e. approach RFShot+PostProcess_BBC), and then we fine-tuned the same model for scene segmentation, using the scene annotations of the same BBC dataset. The input to our model is the middle frame of each shot.

The evaluation of all shot- and scene-segmentation tested approaches was conducted on the RAI dataset, since this is the most commonly used evaluation dataset in the relevant literature.

Table 4: Evaluation of literature methods and proposed approaches to shot segmentation in terms of F-score on the RAI dataset.

Method	Best reported F-score on RAI shots	Speed (fps)
Original ReTV approach (based on [3])	0.84	≈ 125
Ridiculously fast shot segm. [32]	0.88	5895
Spatio-temporal CNN [34]	0.934	382
Deep strctured models [90]	0.935	700
RFShot+PostProcess_BBC	0.832	≈ 1000
RFShot+PostProcess_ClipShots	0.806	≈ 1000
RFShot+PostProcess+QStep_BBC	0.847	≈ 1000

In Table 4 we present the evaluation of the proposed approaches and evaluation of methods found in the literature regarding video shot segmentation. Reported in the second column of this table is the F-score, calculated as the harmonic mean of the Precision and Recall measures which is the standard evaluation measure used in the literature. In the third column we report the speed achieved in terms of frames processed per second, excluding the time for reading the frames of the video. The speed for our proposed methods is from our own measurements on our specific hardware while for the literature methods we have gathered measurements cited in the respective works. Note that, each work utilizes different hardware, therefore these assessments can only taken into consideration as a rough indication of the respective technique's speed. We observe that despite using the architecture of [32] we achieve slightly lower F-score. We argue that this is due to the way that the dataset used by the authors was constructed (which is not made publicly available). Instead, we choose to use publicly available datasets (ClipShots and BBC) so that our results are reproducible. The F-score achieved when training on the BBC dataset (RFShot+PostProcess_BBC) is higher than the one achieved using the ClipShots dataset (RFShot+PostProcess_ClipShots). This is a result of the nature of the ClipShots dataset, which, despite being larger than the BBC dataset, contains much less gradual transitions. On the other hand the BBC dataset and the RAI dataset, using content from TV series, contain much more complex gradual transitions. Finally, using the quadratic incremental temporal step extension we reach a slightly higher F-score than the older used method described in D1.1 of ReTV, while achieving much faster execution times (by almost one order of magnitude), which was our initial goal.

In Table 5 we present the evaluation of the proposed approaches and other literature regarding video scene segmentation. The F-score reported in the 2nd column of this table is calculated on Coverage and Overflow measures, as usually done throughout the scene literature (e.g.

Table 5: Evaluation of literature methods and proposed approaches to scene segmentation in terms of F-score on the RAI dataset.

Method	Best reported F-score on RAI scene
Fast Shot Segm. [3]	0.548
Temporal clustering [7]	0.58
Deep multi-modal networks [9]	0.67
Hierarchical clustering [8]	0.70
RFScene-MID+PostProcess_BBC	0.562
RFScene-AVG+PostProcess_BBC	0.356
RFFineScene-MID+PostProcess_BBC	0.608

in [82]). Comparing the RFScene-MID+PostProcess_BBC and RFScene-AVG+PostProcess_-BBC approaches we observe that using the middle frame of a shot as input to our model yields a better F-score. This is can be explained by the fact that many details of the shot frames are lost during the averaging process. Finally, we notice the significant gains of fine-tuning a model previously trained on shot detection. Regarding speed, most literature works do not cite any measurements, yet since these methods employ clustering algorithms on a multitude of extracted features and our method is based on a very simple and small 3D convolutional model, we can safely assume that our method is considerably faster.

4.2. Concept-based Annotation

4.2.1. Updated Problem Statement and State of the Art

The objective of concept-based annotation is, given a video or a fragment of it, to find which concepts from one or more concept pools best describe it. I.e. it is about enriching the video content with metadata about it, that can be used for finding the right video or fragment within a large collection, and for summarizing this piece of video. Our first focus during months M11 to M20, concerning concept-based annotation, was primarily on developing new deep learning architectures for improving the accuracy of annotation.

The state-of-the-art DNNs for classification tasks consist of a series of weight layers, nonlinear activation functions and downsampling operators and on top of them an output layer typically equipped with a sigmoid or softmax activation function modeling c categorical probability distributions [83, 100]. In [86], it is shown that under the application of the CE loss with gradient descent on separable data the weight vectors of the output layer converge to the max-margin solution with a logarithmic convergence rate. However, as we show here not all weight vectors in the last layers yield a linearly separable problem simultaneously and thus not all class separating hyperplanes converge to the max-margin solution with the same rate. The limitation of DNNs to treat all classes fairly during the training procedure has been mostly studied in the context of class imbalanced learning [79]. Moreover, the identification of classes receiving little attention during training as described above is a relatively unexplored topic. To this end, a new criterion for identifying such neglected classes is proposed. Moreover, in order to turn the attention of the DNN on the identified neglected classes, we resort to a subclass partitioning strategy. Subclass-based classification techniques have been successfully used in the shallow learning paradigm. In [50], learning vector quantization (LVQ) is used to find a set of cluster centers for each class and classification is performed by finding the closest class center. In [35], mixture discriminant analysis (MDA) fits a Gaussian mixture density to each class, extending the linear discriminant analysis (LDA) to the non-normal setting. In [26], nonlinear classification problems are solved by splitting the original set of classes to subclasses and embedding the binary problems in a problem-dependent subclass error-correcting output codes (SECOC) design. In [99, 30], a set of kernel subclass discriminant analysis techniques are proposed in order to deal with nonlinearly separable subclasses, and it is shown that the identification of the optimum kernel parameters can be performed more easily exploiting the subclass partitions. Motivated by the above works, a subclass DNN (SDNN) framework is proposed, where the neglected classes are augmented and partitioned to subclasses, and subsequently a novel subclass CE (SCE) loss, which emphasizes the separation of subclasses belonging to different classes, is applied to train the network. Experimental results in 3 popular benchmarks (CIFAR10, CIFAR100 [51] and SVHN [64]) and in the large-scale YT8M video dataset [1] for the task of multiclass and multilabel classification, respectively, show the efficacy of the proposed approach.

A second focus of our efforts in months M11 to M20 was on expanding the set of supported concept pools for annotation. This is a need that emerged by considering the need: (i) to better support the use-case scenarios, (ii) especially to better characterize viewer segments, (iii) to support ad-related tasks (e.g. text to video matching methods) of WP3 (discussed in D3.2). Therefore, separate from our research on improved concept-detection methods, we also extended the set of concept pools supported by the VA service, with 5 concept pools totalling more than 5.5K concepts. Finally, besides the concept detection, we further extended the VA service to also extract additional features that are needed by the WP3 services. The extraction technique regarding these additional features is discussed in D3.2, since these were developed for and are used in WP3.

4.2.2. Extended Concept Pools for Video Annotation

The newly adopted concept pools are the following:

1. ImageNet [19]: ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a “synonym set” or “synset”. ImageNet provides on average 1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated. This concept pool is particularly useful for text to video matching (discussed in D3.2).

2. EventNet [98]: EventNet is a large scale event-specific concept library that covers as many real-world events and their concepts as possible. WikiHow, an online forum containing a large number of how-to articles on human daily life events was chosen, on which a coarse-to-fine event discovery process was performed and 500 events were finally selected. The event’s name was used as as query to search YouTube and discover event-specific concepts from the tags of returned videos.

3. YouTube-8M [2]: YouTube-8M is a large-scale labeled video dataset that consists of millions of YouTube video IDs, with high-quality machine-generated annotations from a diverse vocabulary of 3800+ visual entities. It comes with pre-computed audio-visual features from billions of frames and audio segments. Each entity represents a semantic topic that is visually recognizable in video, and the video labels reflect the main topics of each video. Furthermore, the entities of the dataset have an hierarchical structure.

Using the (multiple) labels per video as Knowledge Graph entities, the creators organized the 3862 entities into 24 top-level "verticals" (i.e., categories). These categories were found to be of great interest for the ReTV project, especially for characterizing a viewer segment.

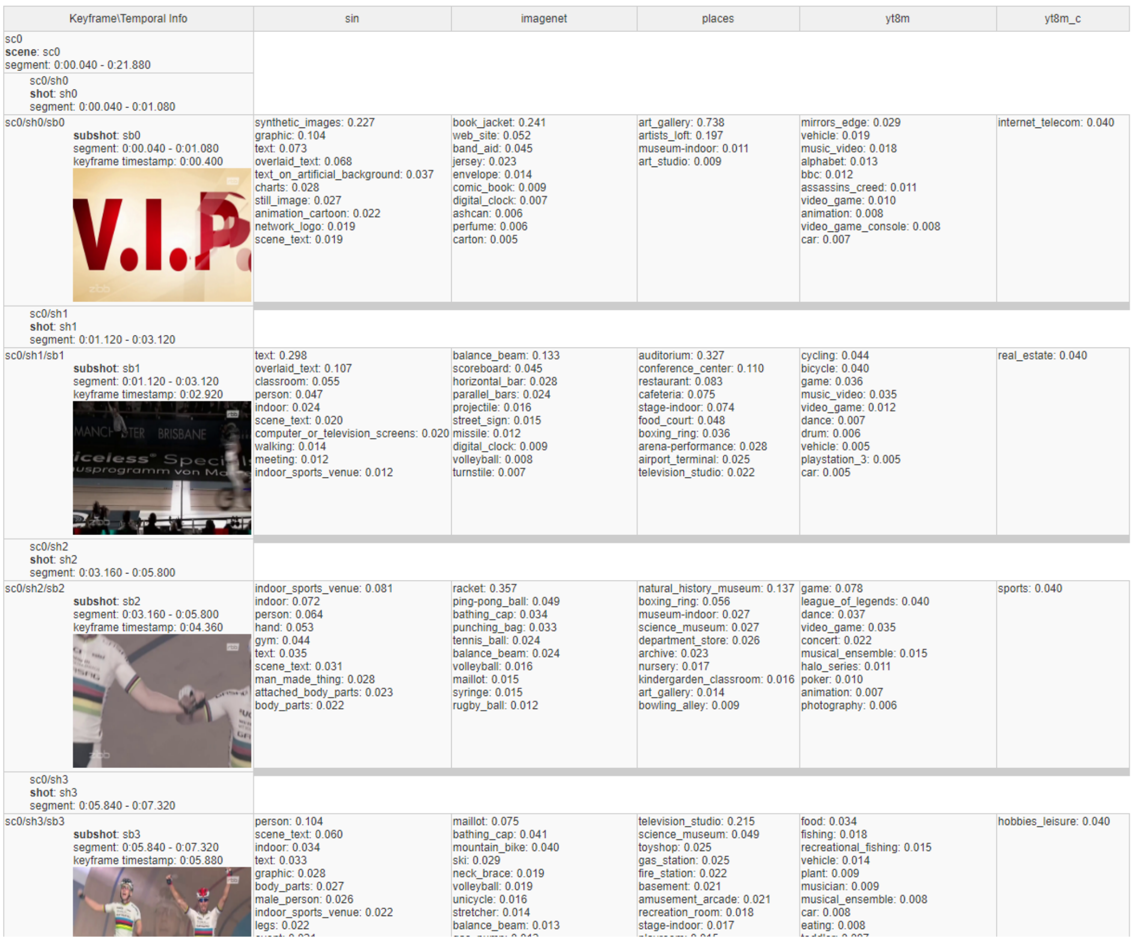
4. Sandmännchen and friends: The inclusion of Sandmännchen and friends TV series in the TVP ingestion pipeline, as requested by RBB, brought the need of annotating the submitted videos' keyframes with the presence of certain characters from this specific TV series.

For the ImageNet and EventNet concept pools we used pre-trained models found in the web. Specifically for the ImageNet concept pools, the Keras deep-learning framework offers the weights for a variety of state-of-the-art model architectures. The accuracy that these achieve in terms of top-1 and top-5 accuracy of the model's performance on the ImageNet validation dataset, as well as the number of parameters they use as a means to measure the computational complexity of each model, are shown in Table 6. Observing the table, we choose to employ the InceptionV3 [89] architecture's pre-trained model, since this seems to be well balanced solution between time-efficiency and effectiveness.

Table 6: Keras Applications of individual pre-trained models. Reported are the top-1 and top-5 accuracy on the ImageNet validation dataset. (The table was taken from `https://keras.io/applications/`)

Model	Top-1 Accuracy	Top-5 Accuracy	Parameters
Xception	0.790	0.945	22.910.480
VGG16	0.713	0.901	138.357.544
VGG19	0.713	0.900	143.667.240
ResNet50	0.749	0.921	25.636.712
ResNet101	0.764	0.928	44.707.176
ResNet152	0.766	0.931	60.419.944
ResNet50V2	0.760	0.930	25.613.800
ResNet101V2	0.772	0.938	44.675.560
ResNet152V2	0.780	0.942	60.380.648
ResNeXt50	0.777	0.938	25.097.128
ResNeXt101	0.787	0.943	44.315.560
InceptionV3	0.779	0.937	23.851.784
InceptionResNetV2	0.803	0.953	55.873.736
MobileNet	0.704	0.895	4.253.864
MobileNetV2	0.713	0.901	3.538.984
DenseNet121	0.750	0.923	8.062.504
DenseNet169	0.762	0.932	14.307.880
DenseNet201	0.773	0.936	20.242.984
NASNetMobile	0.744	0.919	5.326.716
NASNetLarge	0.825	0.960	88.949.818

In order to visually assess the annotations that are produced for a given video, we introduced a tool to visualize the results of the VA service concept detection. A special request in the VA service returns an HTML with all the keyframes of a video and the top detected concepts from each concept pool, for each keyframe. A screenshot of this tool's interface is shown in Fig. 5. Using this tool and after several rounds of submitting videos and empirically evaluating the results of each concept pool, we decided to exclude the EventNet concept pool from the VA service since it was considered to have little value for the applications foreseen in ReTV,



Keyframe\Temporal Info	sin	imagenet	places	yt8m	yt8m_c
sc0 **scene**: sc0 segment: 0:00.040 - 0:21.880					
sc0/sh0 **shot**: sh0 segment: 0:00.040 - 0:01.080					
sc0/sh0/sb0 **subshot**: sb0 segment: 0:00.040 - 0:01.080 keyframe timestamp: 0:00.400	synthetic_images: 0.227 graphic: 0.104 text: 0.073 overlaid_text: 0.068 text_on_artificial_background: 0.037 charts: 0.028 still_image: 0.027 animation_cartoon: 0.022 network_logo: 0.019 scene_text: 0.019	book_jacket: 0.241 web_site: 0.052 band_aid: 0.045 jersey: 0.023 envelope: 0.014 comic_book: 0.009 digital_clock: 0.007 ashcan: 0.006 perfume: 0.006 carton: 0.005	art_gallery: 0.738 artists_loft: 0.197 museum-indoor: 0.011 art_studio: 0.009	mirrors_edge: 0.029 vehicle: 0.019 music_video: 0.018 alphabet: 0.013 bbc: 0.012 assassins_creed: 0.011 video_game: 0.010 animation: 0.008 video_game_console: 0.008 car: 0.007	internet_telecom: 0.040
sc0/sh1 **shot**: sh1 segment: 0:01.120 - 0:03.120					
sc0/sh1/sb1 **subshot**: sb1 segment: 0:01.120 - 0:03.120 keyframe timestamp: 0:02.920	text: 0.298 overlaid_text: 0.107 classroom: 0.055 person: 0.047 indoor: 0.024 scene_text: 0.020 computer_or_television_screens: 0.020 walking: 0.014 meeting: 0.012 indoor_sports_venue: 0.012	balance_beam: 0.133 scoreboard: 0.045 horizontal_bar: 0.028 parallel_bars: 0.024 projectile: 0.016 street_sign: 0.015 missile: 0.012 digital_clock: 0.009 volleyball: 0.008 turnstile: 0.007	auditorium: 0.327 conference_center: 0.110 restaurant: 0.083 cafeteria: 0.075 stage-indoor: 0.074 food_court: 0.048 boxing_ring: 0.036 arena-performance: 0.028 airport_terminal: 0.025 television_studio: 0.022	cycling: 0.044 bicycle: 0.040 game: 0.036 music_video: 0.035 video_game: 0.012 dance: 0.007 drum: 0.006 vehicle: 0.005 playstation_3: 0.005 car: 0.005	real_estate: 0.040
sc0/sh2 **shot**: sh2 segment: 0:03.160 - 0:05.800					
sc0/sh2/sb2 **subshot**: sb2 segment: 0:03.160 - 0:05.800 keyframe timestamp: 0:04.360	indoor_sports_venue: 0.081 indoor: 0.072 person: 0.064 hand: 0.053 gym: 0.044 text: 0.035 scene_text: 0.031 man_made_thing: 0.028 attached_body_parts: 0.023 body_parts: 0.022	racket: 0.357 ping-pong_ball: 0.049 bathing_cap: 0.034 punching_bag: 0.033 tennis_ball: 0.024 balance_beam: 0.024 volleyball: 0.016 maillot: 0.015 syringe: 0.015 rugby_ball: 0.012	natural_history_museum: 0.137 boxing_ring: 0.056 museum-indoor: 0.027 science_museum: 0.027 department_store: 0.026 archive: 0.023 nursery: 0.017 kindergarden_classroom: 0.016 art_gallery: 0.014 bowling_alley: 0.009	game: 0.078 league_of_legends: 0.040 dance: 0.037 video_game: 0.035 concert: 0.022 musical_ensemble: 0.015 halo_series: 0.011 poker: 0.010 animation: 0.007 photography: 0.006	sports: 0.040
sc0/sh3 **shot**: sh3 segment: 0:05.840 - 0:07.320					
sc0/sh3/sb3 **subshot**: sb3 segment: 0:05.840 - 0:07.320 keyframe timestamp: 0:05.880	person: 0.104 scene_text: 0.060 indoor: 0.034 text: 0.033 graphic: 0.028 body_parts: 0.027 male_person: 0.026 indoor_sports_venue: 0.022 legs: 0.022	maillot: 0.075 bathing_cap: 0.041 mountain_bike: 0.040 ski: 0.029 neck_brace: 0.019 volleyball: 0.019 unicycle: 0.016 stretcher: 0.014 balance_beam: 0.013	television_studio: 0.215 science_museum: 0.049 toyshop: 0.025 gas_station: 0.025 fire_station: 0.022 basement: 0.021 amusement_arcade: 0.021 recreation_room: 0.018 stage-indoor: 0.017	food: 0.034 fishing: 0.018 recreational_fishing: 0.015 vehicle: 0.014 plant: 0.009 musician: 0.009 musical_ensemble: 0.008 car: 0.008 eating: 0.008	hobbies_leisure: 0.040

Figure 5: A snapshot of the HTML concept visualization tool. Rows correspond to the extracted key-frames of the video subshots, and there is a separate column for each of the concept pools in the VA service. Each cell contains the top detected concepts for the respective key-frame and concept pool. The thick horizontal gray lines indicate a change of shot (i.e., in this example, there is just one subshot for each of the four depicted shots).

as opposed to the other supported concept pools.

The VA service as of month M18 performs concept annotation of videos with 5 concept pools: the SIN and Places365 concept pools (already introduced since M10 and described in D1.1), as well as the newly introduced ImageNet, YouTube-8M and the Sandmännchen and friends concept pools. In total, submitted videos are annotated with 5582 concepts. In Table 7 we report 10 example concepts from each adopted concept pool. In Table 6 we present the evaluation results on the official ImageNet validation set for the ImageNet concept pool. In Section 4.2.5 we present the evaluation results of the Improved Deep Learning Architecture, which is introduced in the following subsection, on three typical benchmark datasets and on the large-scale YouTube-8M.

Table 7: A set of 10 concept examples from each of the concept pools supported by the VA service

Concept pool	SIN	Places365	Imagenet	YouTube-8M	YouTube-8M verticals	SaF
Example #1	adult female human	airfield	computer mouse	talent show	people & society	(no character)
Example #2	anchorperson	bookstore	digital clock	steel-string acoustic guitar	arts & entertainment	Fuchs, Elster
Example #3	boy	cafeteria	fire engine	mercedes-benz	autos & vehicles	Jan, Henry
Example #4	car	fire station	garbage truck	IPhone	computers & electronics	Kalli
Example #5	cheering	hospital	studio couch	ABS-CBN News	news	Konig
Example #6	studio expert	lecture room	electric guitar	X-Men	hobbies & leisure	Moffels
Example #7	female human face	office cubicles	hammer	nail art	beauty & fitness	Rita
Example #8	military airplane	physics laboratory	car wheel	bowling ball	sports	Sandmänn
Example #9	skier	skyscraper	coffeepot	PlayStation 4	Games	
Example #10	speaking to camera	television studio	wine bottle	outline of meals	food & drink	
Total concepts	323	365	1000	3682	24	8

4.2.3. Improved Deep Learning Architecture

During minibatch gradient-based optimization, the contribution of observations to the updating of the deep neural network's (DNN's) weights for enhancing the discrimination of certain classes can be small, despite the fact that these classes may still have a large generalization error. This happens, for instance, due to overfitting, i.e. to classes whose error in the training set is negligible, or simply when the contributions of the misclassified observations to the updating of the weights associated with these classes cancel out. To alleviate this problem, in the following, a novel subclass DNN framework is proposed, where the neglected classes, i.e. the classes which stop to optimize early in the training procedure, are augmented and partitioned to subclasses, and subsequently a novel subclass CE (SCE) loss, which emphasizes the separation of subclasses belonging to different classes, is applied to train the network. Suppose a DNN with a sigmoid output layer (SG)

$$\mathbf{h}_\kappa = \mathbf{W}^T\mathbf{x}_\kappa + \mathbf{b}, \tag{1}$$

$$q_{i,\kappa} = \frac{1}{1+\exp(-h_{i,\kappa})}, \tag{2}$$

where $\mathbf{W} = [\mathbf{w}_1,\ldots,\mathbf{w}_c] \in \mathbb{R}^{f\times c}$, $\mathbf{b} = [b_1,\ldots,b_c]^T$ are the weight matrix and bias vector of the SG layer, and c is the number of classes. Moreover, assuming a batch of n training observations, the vectors $\mathbf{x}_\kappa = [x_{1,\kappa},\ldots,x_{f,\kappa}]^T$, $\mathbf{h}_\kappa = [h_{1,\kappa},\ldots,h_{c,\kappa}]^T$, $\mathbf{q}_\kappa = [q_{1,\kappa},\ldots,q_{c,\kappa}]^T$, $\mathbf{y}_\kappa = [y_{1,\kappa},\ldots,x_{c,\kappa}]^T$, are associated with the κth training observations in the batch, and are the input and output vector of the linear part of the SG layer, the output vector of the SG layer, and the class indicator vector, respectively. That is, the ith component $y_{i,\kappa}$ of $\mathbf{y}_\kappa$ is the label of the κth observation with respect to the ith class

$$y_{i,\kappa} = \begin{cases} 1 & \text{if } \mathbf{x}_\kappa \in \omega_i, \\ 0 & \text{else,} \end{cases} \tag{3}$$

and ω_i denotes the ith class. For training the DNN, the minibatch stochastic gradient descent (SGD) and the CE loss are used

$$L = -\frac{1}{n}\sum_{\kappa=1}^{n}\sum_{i=1}^{c}(y_{i,\kappa}\ln(q_{i,\kappa}) + (1-y_{i,\kappa})\ln(1-q_{i,\kappa})). \tag{4}$$

Under this framework, the weight vector associated with the ith class is updated at each iteration as below

$$\begin{aligned} \mathbf{w}_i &= \mathbf{w}_i - \eta\mathbf{g}_i, & (5)\\ \mathbf{g}_i &= \frac{1}{n}\sum_{\kappa=1}^{n}\zeta_{i,\kappa}\mathbf{x}_\kappa, & (6)\end{aligned}$$

where, $\mathbf{g}_i$ is the gradient of L with respect to $\mathbf{w}_i$, η is the learning rate and $\zeta_{i,\kappa} = q_{i,\kappa} - y_{i,\kappa}$. Noting that $q_{i,\kappa} \in [0,1]$ we observe that $\zeta_{i,\kappa} \in [-1,1]$, with $\zeta_{i,\kappa} \approx 0$ when the right answer for $\mathbf{x}_\kappa$'s label is provided from SG layer's unit i, and $\zeta_{i,\kappa}$ moving towards $|1|$ as the likelihood of unit i to provide a wrong answer increases

$$\zeta_{i,\kappa} = \begin{cases} 1 & \text{if } q_{i,\kappa} = 1, y_{i,\kappa} = 0, \\ -1 & \text{if } q_{i,\kappa} = 0, y_{i,\kappa} = 1, \\ 0 & \text{if } q_{i,\kappa} == y_{i,\kappa}. \end{cases} \tag{7}$$

These properties of $\zeta_{i,\kappa}$ can aid the correct operation of the gradient-based learning approach, i.e., shrinking the gradient in (5) when the right answer is obtained, and providing a strong gradient otherwise, forcing the overall network to act quickly in order to correct the mislabeled observations. However, this is not always the case. For instance, considering that the contribution to the summand in (5) of different observations may cancel out, the gradient may shrink despite the fact that many observations are misclassified. To see this, we rewrite the gradient as

$$\begin{aligned} \mathbf{g}_i &= \frac{1}{n}(\tilde{\boldsymbol{\delta}}_i - \hat{\boldsymbol{\delta}}_i) = \frac{1}{n}\boldsymbol{\delta}_i, & (8)\\ \hat{\boldsymbol{\delta}}_i &= \sum_{\mathbf{x}_\kappa \in \omega_i} -\zeta_{i,\kappa}\mathbf{x}_\kappa, & (9)\\ \tilde{\boldsymbol{\delta}}_i &= \sum_{\mathbf{x}_\kappa \notin \omega_i} \zeta_{i,\kappa}\mathbf{x}_\kappa, & (10)\end{aligned}$$

where $\hat{\boldsymbol{\delta}}_i$, $\tilde{\boldsymbol{\delta}}_i$ equal zero when the positive and negative observations, respectively, are classified correctly. Note that $-\zeta_{i,\kappa}, \mathbf{x}_\kappa \in \omega_i$ and $\zeta_{i,\kappa}, \mathbf{x}_\kappa \notin \omega_i$ are less than one and always positive, and thus $\hat{\boldsymbol{\delta}}_i$, $\tilde{\boldsymbol{\delta}}_i$ are the weighted means of the target and non-target class, respectively, weighted with the likelihood derived from the DNN that this observation belongs to the respective category or not. When $\hat{\boldsymbol{\delta}}_i$, $\tilde{\boldsymbol{\delta}}_i$ are close to each other, the overall gradient $\boldsymbol{\delta}_i$ approaches zero and $\mathbf{w}_i$ remains relatively unchanged, despite the fact that many observations are still not classified correctly from unit i. When this undesired effect appears, the network gradually stops to optimize the weights of the different layers below for extracting discriminant features associated with such "neglected" classes, paying more attention on improving the training classification rates of classes which still produce a strong gradient at each iteration. A unit i with large $\|\hat{\boldsymbol{\delta}}_i\|$, $\|\tilde{\boldsymbol{\delta}}_i\|$ and at the same time small difference between these two quantities reflects a high likelihood that the associated class is not getting the required attention and is going to be neglected in subsequent iterations. Based on the analysis above, every τ minibatch iterations we compute the following measure for estimating how likely a class is to be neglected

$$\theta_i = \frac{1}{n\tau} \sum_{l=p-\tau+1}^{\tau} \frac{\|\hat{\boldsymbol{\delta}}_{i,l}\| + \|\tilde{\boldsymbol{\delta}}_{i,l}\|}{\|\boldsymbol{\delta}_{i,l}\|}, \tag{11}$$

where $\hat{\boldsymbol{\delta}}_{i,l}$, $\tilde{\boldsymbol{\delta}}_{i,l}$ are the gradient terms (9), (10) at the lth minibatch iteration, $\|\|$ is the vector norm operator and p is the current iteration. The identification of the most neglected class $\imath$ is then performed by using a simple argmax rule

$$\imath = \underset{i}{\operatorname{argmax}}(\theta_i). \tag{12}$$

The major consequence of neglecting a class during the optimization procedure is that the trained DNN will fail to learn an appropriate feature mapping where the neglected classes are linearly separable. To alleviate this unwanted behavior we propose the use a clustering algorithm to derive a subclass partition for those classes that are prone to be neglected. By exploiting this partition it is expected that it will be generally easier for the DNN to learn a nonlinear mapping where the subclasses are linearly separable. Under this framework, the easiest way to extend the CE criterion would be to treat each subclass as a class. However, this loss will treat equivalently the costs associated with misclassifying an observation to the non-target subclasses without examining which non-target subclasses are associated with the target class of the observation and which not. To this end, we propose the following loss in order to favor the separability of those subclasses that correspond to different classes

$$L = -\frac{1}{n} \sum_{\kappa=1}^{n} \sum_{i=1}^{c} \sum_{j=1}^{H_i} (y_{i,j,\kappa} \ln(q_{i,j,\kappa}) + (1 - y_{i,\kappa}) \ln(1 - q_{i,j,\kappa})), \tag{13}$$

where, $y_{i,j,\kappa}$ is the label of the κth training observation in the batch associated with jth subclass of class i,

$$y_{i,j,\kappa} = \begin{cases} 1 & \text{if } \mathbf{x}_\kappa \in \omega_{i,j}, \\ 0 & \text{else,} \end{cases} \tag{14}$$

and $h_{i,j,\kappa}$, $q_{i,j,\kappa}$ are the input and output to the activation function of the (i,j) unit associated with $\mathbf{x}_\kappa$. Note, that in the second summand of (13) the class label $y_{i,\kappa}$ is utilized instead of the subclass label $y_{i,j,\kappa}$ in order to emphasize the separation of subclasses belonging to different classes as explained above.

Any clustering algorithm and augmentation approach can be applied to derive a subclass division of the neglected classes. However, for large-scale datasets such as the YT8M, it may be infeasible to use computationally demanding clustering approaches such as k-means. To this end, the lightweight approach described in Algorithm 1 for partitioning the observations of the $\imath$th class into two subclasses is proposed. It is based on the computation of the distance of each class observation to $\mathbf{m}$, which is the mean along all observations in the training set and used as a representation of the rest-of-world class. Moreover, data augmentation can be performed to the neglected classes by applying extrapolation in the feature space for each observation as proposed in [20]

$$\acute{\mathbf{x}}_{i,j,\kappa} = \lambda(\mathbf{x}_{i,j,\kappa} - \check{\mathbf{x}}_{i,j}) + \mathbf{x}_{i,j,\kappa}, \tag{15}$$

Algorithm 1 Subclass partitioning algorithm

```
Input: $\{\mathbf{x}_{i,1},\ldots,\mathbf{x}_{i,n_i}\}$, $\mathbf{m}$
Output: $\mathbf{x}_{i,j,\kappa}$, $\check{\mathbf{x}}_{i,j}$,   $\kappa = 1,\ldots,n_{i,j}$, $j = 1,2$
1: Compute $d_j = \|\mathbf{x}_{i,j} - \mathbf{m}\| \; \forall j$
2: Sort $\mathbf{x}_{i,j}$'s in descending order according to the $d_j$'s: $\{\tilde{\mathbf{x}}_{i,1},\ldots,\tilde{\mathbf{x}}_{i,n_i}\}$
3: Compute $n_{i,1} = \lfloor n_i/2 \rfloor$, $n_{i,2} = n_i - n_{i,1}$
4: Set $\{\mathbf{x}_{i,1,1},\ldots,\mathbf{x}_{i,1,n_{i,1}}\} = \{\tilde{\mathbf{x}}_{i,1},\ldots,\tilde{\mathbf{x}}_{i,n_{i,1}}\}$
5: Set $\{\mathbf{x}_{i,2,1},\ldots,\mathbf{x}_{i,2,n_{i,2}}\} = \{\tilde{\mathbf{x}}_{i,n_{i,1}+1},\ldots,\tilde{\mathbf{x}}_{i,n_i}\}$
6: Set $\check{\mathbf{x}}_{i,1} = \tilde{\mathbf{x}}_{i,1}$, $\check{\mathbf{x}}_{i,2} = \tilde{\mathbf{x}}_{i,n_i}$
```

where, $\lambda \in [0,1]$ and $\check{\mathbf{x}}_{i,1}$, $\check{\mathbf{x}}_{i,2}$ are the observations of class i with the largest and smallest distance from $\mathbf{m}$, respectively. Using the approach described in this section, both class partitioning and augmentation can be performed very efficiently on-line without the need to load the whole dataset or large parts of it in memory. This work has been submitted for publication to a scientific conference.

4.2.4. Implementation Details and Use

Regarding our efforts on expanding the set of used concept pools, we used the Keras[16] open-source neural-network library version 2.1.2, with the TensorFlow backend[17] version 1.12. All experiments where conducted on an Intel i5 3570K PC with 32 GB RAM, running Windows 10, equipped with an Nvidia GeForce GPU (GTX 1080 Ti).

Particularly for developing the new deep learning architecture of Section 4.2.3, and due to the size of the YT8M dataset together with the amount of experiments we needed to run on this dataset, we used the TensorFlow backend as above, version 1.13.1, and the experimental evaluation was performed in an Intel i7 3770K PC with 32 GB RAM, running Windows 10, again equipped with an Nvidia GeForce GPU (GTX 1080 Ti).

4.2.5. Results

The newly introduced concept-detection method was initially evaluated for the task of multi-class classification in 3 publicly available datasets, namely, the CIFAR-10, CIFAR-100 and the SVHN dataset [51, 64]. These are relatively small-scale datasets, but they are standard benchmark datasets of the literature and thus useful for documenting the merits of our proposed method against the state-of-the-art. All the above datasets are already split to a training and a testing partition. For the SVHN dataset an extra partition is also provided. Following the standard procedure for this dataset, the training and extra partitions are combined in our experiments to form a new training partition.

The VGG16 [83] and two variants of the wide residual networks (WRN) [100] are used for the evaluation, specifically, a WRN with depth 28, widening factor 10 (WRN-28-10) and dropout rate of 0.4 is used for the CIFAR datasets, and the WRN-16-8 with 0.3 dropout rate is employed for the SVHN benchmark [21]. All networks are trained for 200 epochs using the CE loss (4), minibatch SGD with Nesterov momentum of 0.9, batch size of 128, weight decay of 0.0005,

[16]`https://keras.io/`

[17]`https://www.tensorflow.org/`

Table 8: Accuracy rates and training times (hours) in 3 datasets

	VGG16 [83]	SVGG16	WRN [21]	SWRN
CIFAR10	93.5% (2.6h)	94.8% (2.7h)	96.92% (7.1h)	**97.14%** (8.1h)
CIFAR100	71.24% (2.6h)	73.67% (2.6h)	81.59% (7.1h)	**82.17%** (7.5h)
SVHN	98.16% (29.1h)	98.35% (34.1h)	98.70% (33.1h)	**98.81%** (42.7h)

and an exponential learning rate schedule set to decrease at the 60th, 120th and 160th. For the CIFAR datasets, the initial learning rate is set to 0.1 and reduced by a factor 0.1 according to the learning rate schedule above, while for the SVHN dataset an initial learning rate and reduction factor of 0.01 and 0.2 are used, respectively. The images are normalized per-channel to zero mean and unit variance, and data augmentation with cutout regularization is performed during training following [21]. The subclass VGG16 (SVGG16) and WRN (SWRN) are created as explained in the following. The original VGG16 and WRN are executed for 30 epochs in order to compute a reliable neglection score θ_i for each class. In this way, 2 classes from the CIFAR10 and SVHN (20% of the total classes) and 10 classes from the CIFAR100 (10% of the total classes) with the highest θ_i's are selected. In order to alleviate any class imbalance problems resulting from the partitioning to subclasses, the selected classes are first doubled in size using the augmentation method described in [44], and then the k-means algorithm is applied to create two new subclasses from each class. The performance of each method is measured using the correct classification rate (CCR) along all classes achieved by the trained network in the test set.

The evaluation results in terms of CCR and training times in hours are shown in Table 8. The testing times are only a few seconds in all cases. From the obtained results we can see that the proposed SVGG16 and SWRN outperform the conventional networks in all datasets, with differences in performance from 0.21% (SWRN over WRN in SVHN) to $\approx 2.5\%$ (SVGG16 over VGG16 in CIFAR100). Considering that the CCR rates obtained with the WRN combined with cutout regularization [21] are currently among the state-of-the-art performances, even the small improvements obtained with the proposed approach are considered significant.

The very large-scale, and much more interesting for ReTV YT8M [1] was then utilized to further evaluate the proposed approach for the task of multilabel classification. This is the largest publicly available multilabel video dataset consisting of approximately 6.1 million videos annotated with one or more labels from 3862 classes. Visual and audio feature vectors in $\mathbb{R}^{1024}$ and $\mathbb{R}^{128}$, respectively, are already provided at video-level as well as at frame-level granularity.

For the evaluation, a rather simple convolutional neural network (CNN) architecture is utilized with a convolutional, a max-pooling, a dropout and a SG layer of c outputs. The convolutional layer consists of 64 one-dimensional (1D) filters and is equipped with a rectification (ReLU) nonlinearity. Each filter has a receptive field of size 3 and stride 1, and zero padding is applied in order to preserve the spatial size of the input signals. The max-pooling layer employs a filter of size 2 and stride 2, while a keep-rate of 0.7 is used for the dropout layer. The CE loss combined with the minibatch SGD-BP algorithm and weight decay of 0.0005 is used for training the CNN. The training is performed over 5 epochs with an exponential learning rate schedule, initial learning rate of 0.001 and learning rate decay 0.95 every epoch. For the construction of the subclass CNN (SCNN), the CNN above is initially applied in the training set for $\frac{1}{3}$ of an epoch in order to obtain a neglection value θ_i (11) for each YT8M class and the 386 classes with the highest θ_i are selected, i.e., 10% of the total number of classes. The selected classes are then partitioned to $H_i = 2$ subclasses using the efficient on-line algorithm described in Algorithm 1, avoiding the loading of the whole dataset or large parts of it in memory, which would be infeasible for the YT8M dataset. Moreover, data augmentation is performed to the neglected classes using the extrapolation technique described in Section 4.2.3, setting $\lambda = 0.05$. In this way the number of observations in each subclass partition is doubled. The resulting SCNN is trained using the proposed SCE loss and the training procedure described for the conventional CNN. For completeness, a standard logistic regression (LR) classifier is also evaluated using the same training procedure with initial learning rate of 0.01.

Table 9: Evaluation results in YT8M

	Visual			Visual + Audio		
	LR	CNN	SCNN	LR	CNN	SCNN
Hit@1	82.4%	82.5%	**83.2%**	82.3%	85.2%	**85.7%**
PERR	71.9%	72.2%	**72.9%**	71.8%	75.4%	**75.9%**
mAP	41.2%	42.3%	**45.2%**	40.1%	45.6%	**47.9%**
GAP@20	77.1%	77.6%	**78.6%**	77%	80.7%	**82.2%**
T_{tr} (min)	**18.7**	59.2	66.2	**18.9**	60.3	67.1

Table 10: Comparison with the best single-model approaches in YT8M

	[42]	[63]	[12]	[85]	SCNN
GAP@20	82.15%	80.9%	**82.5%**	82.25%	82.2%
Parameters (millions)	384	**54**	356	229	142

The evaluation results in terms of Hit@1, PERR, mAP, GAP@20 and training time (T_{tr}) in minutes for each method are shown in Table 9. Moreover, in Table 10 we show state-of-the-art results achieved from single-model approaches in terms of GAP@20 and number of network parameters (in millions) for YT8M. From the analysis of the obtained results we observe the following: i) The SCNN attains the best results, outperforming the conventional CNN by 1% and 1.5% GAP using the visual and audio-visual features, respectively. Both networks outperform the standard LR. ii) By exploiting the audio information both CNN and SCNN attain a significant performance gain of more than 3%. On the other hand, a degradation in performance is observed for the LR model, which most likely does not have the capacity to exploit the additional discriminant information provided by the audio modality. iii) As shown in Table 10, our SCNN method achieving a GAP of 82.2% performs in par with the best single-model approaches reported in [42, 12, 85]. This is an excellent performance considering that our SCNN exploits only the video-level feature vectors provided by the YT8M dataset in contrast to the top-performers in the competition, which additionally exploit the frame-level visual features and build upon stronger and much more computationally-demanding feature vector descriptors such as Fisher Vectors, VLAD, BoW, and other [12, 85]. For instance, a training time of around 8 hours in an Amazon Web Services p2-2xlarge instances (providing GPU-based parallel compute capabilities with 16 NVIDIA K80 GPUs, 64 vCPUs, etc.) is reported in [12], whereas our model requires a little more than one hour in a conventional PC. Even more higher training times are required for methods such as [84], which exploit learnable pooling and much more complex architectures. Moreover, in terms of model size our model is the second smallest behind [63] (which however has the worst GAP performance among all the other approaches shown in Table 10); our model has only about 60% of the parameters of the next smallest model exhibiting similar GAP performance [85].

5. Brand Detection

5.1. Updated Problem Statement and State of the Art

Updated State of the Art for Brand detection

For a complete state-of-the-art analysis regarding object detection, the reader is referred to D1.1 of ReTV [60]. In this deliverable we expand our previous literature survey by discussing a selection of the most recent DCNN-based object detection frameworks.

Conducting a fair comparison among different object detection frameworks is a difficult task, since there is no definite answer on which one is the “best” from every aspect. One must take into account the specific merits of each model architecture but usually for real-life applications, the choice to be made is based on a balance between accuracy and speed. Having said this we can distinguish modern object detection frameworks in two main categories: (i) two-stage detectors and (ii) one-stage detectors. Regarding two-stages detectors, first, the model proposes a set of regions of interest coming from, e.g., a region proposal network. Then, a classifier only processes the region candidates. All models of the Region-based Convolutional Neural Network (R-CNN) family, i.e., the original R-CNN [29] and its recent improved and faster variants Fast R-CNN [28] and Faster R-CNN [74] are two-stage detectors. In the same family, the Mask R-CNN [38] extends Faster R-CNN to pixel-level image segmentation. Based on the framework of Faster R-CNN, it adds a third branch for predicting an object mask in parallel with the existing branches for classification and localization. The mask branch is a fully-connected network applied to each region of interest, predicting a segmentation mask in a pixel-to-pixel manner.

The other category of approaches, the one-stage detection, skips the region proposal stage and runs detection directly over a dense sampling of possible locations. This is faster and simpler, but may have a negative impact on performance, since all regions must be analysed. For many years the two-stage detector were the obvious choice when selecting a object detection framework due to its higher inference speed. However, more recently, many one-stage frameworks were proposed to alleviate this problem. The YOLO model (“You Only Look Once” [71]) is the first attempt at building a fast real-time object detector. Because YOLO does not undergo the region proposal step and only predicts over a limited number of bounding boxes, it is able to do inference very fast. In [71] it is noted that a neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Several performance upgrades resulted in enhanced versions, namely the YOLOv2 (a.k.a. the YOLO9000 [72]) and YOLOv3 [73]. The Single Shot Detector (SSD; [58]) is another one-stage detector and one of the first attempts at using convolutional neural network’s pyramidal feature hierarchy for efficient detection of objects of various sizes. The RetinaNet [54] is a one-stage dense object detector. Two crucial building blocks are the Feature Pyramid Network (FPN) and the use of Focal Loss. The FPN [53] is the backbone network for RetinaNet, following the same approach by image pyramid in SSD, making RetinaNet able to provide object detection at different scales. However, the main contribution of RetinaNet is the Focal Loss. Methods like SSD or YOLO suffer from an extreme class imbalance: The detectors evaluate roughly between ten to hundred thousand candidate locations and most of these regions do not contain an object. The authors of [54] argue that the problem of one-stage detection frameworks is that many of these “easy” negative examples (i.e., the background class, where no object is detected) are used in the training process. Using the proposed Focal loss, the contribution of these “easy” negative samples to the learning of the model is weighed down. In this way, the large number of easily classified examples (mostly of the background class) does not dominate the loss and learning can concentrate on the few interesting cases.

We selected the RetinaNet framework after collecting from the literature the evaluation results on standard object detection benchmark for most of the aforementioned models, which are discussed 5.5. The object detection module of the VA service was updated to use the new framework on an expanded set of brands pools supported by the service, discussed in Section 5.2. Comparisons between the older and the new used object detection framework are reported in Section 5.5.

Updated requirements of the Brand Detection task (T1.3) for an Ad detector

One of the newly identified requirements of ReTV in the past months is the development of an advertisements detection module. Advertisements shown in TV is an important marketing method that can boost the popularity of a product. There are many reasons why being able to detect commercial segments (i.e. segments comprising of numerous advertisements) within television broadcasts is of interest. The first and foremost reason to detect advertisements can be their verification and analysis process. Advertisement companies seek a safe way to track if an advertisement has been aired the number of times and in the time slots that it was agreed, since the price of a TV advertisement heavily depends on the popularity of the TV program it is inserted in and the cost is correspondingly high. Marketing companies employ such tools in the context of competitive marketing analysis (i.e. to observe competitors behavior) and help their customers make an informed decision regarding their strategy. The embedding of advertisements within regular broadcast TV program can some times be informing, and other times annoying which rationalizes the wish of some online TV users to detect commercials for the purpose of eliminating them. In digital video indexing systems that enable later browsing and playback of TV program (i.e. after it has been officially transmitted in more traditional channels, like ordinary TVs) viewers would appreciate the ability to watch their recorded television shows with annoying commercials automatically edited out.

Another scenario where ad detection is crucial, is advertisement replacement in digital TV frameworks: the original advertisements must be detected so that they can be removed and replaced with new advertisements that are more appropriately targeted to a specific viewer segment. On the one hand, since the selection of the new advertisement will be done based on (implicitly or explicitly collected) viewer’s preferences, it is expected that this replacement will be at least interesting to the user. On the other hand, the higher buying potential is a common pursuit for the brand being advertised, therefore this scenario is actually beneficial for the viewer, the content distributor as well as the brand. Finally, for digital/internet TV distributors, it is often the case that the Electronic Program Guide (EPG) is not accurate enough with respect to the actual broadcast time. This can create a multitude of problems, e.g., erroneous segmentation of specific shows that is needed for later on-demand streaming, advertisements replacements with a temporal offset that “steps on” a TV show and might disturb the storyline of the show, inaccurate recording (and therefore further analysis) of viewing statistics. Having a way to align the broadcasted content to the information provided by the EPG, can also help alleviate such problems.

Some of the aforementioned applications of ad detection methods relate to the requirements of ReTV, namely, ad-replacement based on viewer segments data and accurate time alignment of EPG (for the ContentSwitch use-case scenario). Therefore a need for an ad detection method emerged in ReTV. The developed method is proposed in Section 5.3 and evaluated in Section 5.5. Following is an analysis of the state-of-the-art in ad detection methods.

State of the Art for Ad detection

The existing methods for advertisement detection can be categorized [80] to (i) feature-based classification methods, (ii) known-ad recognition methods, and (iii) duplicate segments detection methods. Feature-based classification methods attempt to classify video segments to the ad/non-ad classes using content features that are inspired by certain observations [59], e.g. the observation that most advertisements have rapidly changing visual content. To achieve this, low-level visual features are used, due to fact that these can be computed fast, such as edge detection techniques to measure motion activity [24], the edge-change-ratio (ECR) measure [80, 52], or statistics of these features, for example the use of the average and variance of ECR in [41]. Others utilize the encoding of the TV-stream to MPEG bitstream [78, 22, 59] to quickly infer features similar to the aforementioned. In terms of classification algorithms, the authors of [96] proposed to use adaptive threshold techniques on inter-frame difference since several earlier methods relied on thresholds that need re-adjusting to be able to detect advertisements in varying scenarios (e.g. different channels). On the other hand, [41] proposed a learning-based method that employs an SVM and [70] introduced statistical modeling using a Hidden Markov Model, however their method focuses on advertisement detection in TV news videos. Finally, a notably different method is [37] that utilized the TV program transcripts to detect the advertisements with a focus on precisely segmenting the input video.

Detecting the boundaries of different TV program material is often used as a pre-processing step. The most used technique for this is the detection of black frames [59, 70, 78, 24] since the most common delimiter for TV-content and an advertisement is six or more black frames [80]. In [57] the audio signal is utilized for the segmentation of the TV-stream, i.e. detecting silence segments.

Known-ad recognition-based methods [16, 80, 15, 52] rely on encoding a segment of TV stream and checking if there are any matches to items of a pre-computed database of known advertisements. Some of this kind of methods encode the visual content of advertisements using hashing techniques [81, 56] to speed-up the matching speed, while [91] uses the SIFT descriptor to perform the matching instead. Once again, there are methods [16] that use the audio stream to compute audio fingerprints, which are typically smaller than the visual fingerprints.

An advantage of recognition-based detection methods is that they achieve to not only detect but also recognize advertisements, thus inferring more information and making possible the computation of brand analytics for scenarios that this is needed. However, such methods can only detect a limited set of previously analyzed advertisements. The expansion of the set of known-ads, in order to be able to detect new advertisements, requires constant effort. Additionally, as the set of previously analyzed advertisements grows, the retrieval speed may be reduced. To our knowledge, the only recognition-based work that has a specific focus on efficiency is [15] in which a framework for fingerprint matching with GPU utilization is described.

To alleviate the need of pre-constructing a database of advertisement fingerprints, there are methods that try to detect duplicate segments in the TV stream based on the fact that an advertisement is shown numerous times in a day in a TV program. The authors of [24] propose computing color histograms for this purpose. In [17] a three-stage approach to localize repeated content, a repetition candidate detection stage that utilizes the audio track, a validation stage that uses very compact visual descriptors and a refinement stage to find the starting and ending points of the repeated, segments while [27] uses hashing techniques.

A method that falls out of the scope of this categorization is [61] in which a deep learning system for ad detection is introduced. This system comprises of a two-stream audio-visual convolutional neural network. The audio and visual embeddings are fused together to classify a video segment as an advertisement or tv-content. To our knowledge, this is the first method that deals with the problem of ad detection using a deep learning based approach.

Regarding the evaluation datasets used throughout the literature, we should mention that all the aforementioned methods utilize hand-selected and manually annotated content ranging from approximately 4 hours (in [78]) to 215 hours (in [59]). However, most of this content is subject to copyright and thus cannot be made publicly available. The absence of a common dataset for evaluation and comparison of different methods is a big setback for ad detection methods. Only in [48] such a dataset is provided, yet again, this dataset is made of just the features extracted from the TV program; the original videos are not available for the same aforementioned copyright reasons. Therefore the evaluation results of our developed method in Section 5.5 concern the comparison between different tested approaches.

5.2. Extended Brand Pools for Video Annotation

After discussion with ReTV partners the decision to expand the set of detected brands was taken, mainly to support the use-case scenarios. Partners from GENISTAT gathered a set of Swiss brands, while partners from RBB gathered a set of German football teams whose logo should be detectable by the VA service. CERTH built a web-crawler that fetches the top results from the Google Image search web-page, for any submitted query. The web-crawler was employed to fetch the top 20 results when searching for each name of the brands under consideration. The retrieved images were manually checked and irrelevant images were excluded. A data augmentation scheme similar to [88] was then applied, where each image undergoes several random distortions (brightness and saturation adjustments, affinity distortions, noise addition, blurring) and then pasted to an image that serves as background (i.e., does not contain any object). Using this data augmentation scheme we formed a dataset of 1000 images for each brand logo, of which 900 images were used for training the object detector and the rest for the validation of the model during the training procedure.

Furthermore, for the needs of the ad detection method (described in Section 5.3) we trained an object detection model to detect 16 channel logos. The channels were selected after discussion with the ReTV use-case partners and in accordance with the content that ReTV ingests. Besides the 16 gathered logos, we added two more classes, namely: (i) the *HD* class which denotes whether the current content is of high definition, which is usually indicated by the including the "HD" text near the channel logo, and (ii) the *live* class, which denotes whether the current content is a live transmission and which is usually indicated by including the "live" text near the channel logo. Therefore, the channel logo detection model can detect in total 18 objects. The videos for each channel were provided by GENISTAT and the manual annotation of the position of the channel logo in the video frame was conducted by CERTH. The training data were formed by sampling one frame each second from the video.

In Table 11 we report 10 example object names from each adopted brand pool. The VA service, as of month M18 can detect 322 classes of different objects which include 251 brand logos, 53 football team logos, and 18 channel logos. The object detection is performed in each sub-shot and the probability of correct detection, as well as the bounding box for each detected brand's instance, are recorded in the JSON-formatted results of the VA service. We also introduce a brand-density metric for each scene and shot, by counting the number of different brands detected with high confidence in the keyframes of a specific scene/shot. This measure is used by the ad detection method and is discussed in Section 5.3.

Table 11: 10 object class examples from each of the brand pools supported by the VA service

Object detection pools	CERTH brands	Swiss brands	football clubs	channel logos
Example #1	nike	Adecco	brandenburger	3plus
Example #2	nbc	Coop	fc_frankfurt	ard
Example #3	kfc	Lindt	fc_ludwigsfelder	br
Example #4	google	Migros	fsv_gluckauf	prosieben
Example #5	pepsi	Nescafe	mtv_altlandsberg	rbb
Example #6	milka	Rivella	sc_potsdam	rsi-la2
Example #7	bershka	Rolex	sv_babelsberg	rtl
Example #8	firefox	SBB	sv_turbine	rts1
Example #9	bbc	UBS	union_klosterfelde	srf-1
Example #10	espn	Vogele	vfb_krieschow	srf-info
Total objects	225	26	53	18

5.3. ReTV method for Ad Detection

In this section we will refer to every show in a TV program (e.g. news, series, movies, talk shows, show in general) that is not an advertisement as *content*. Also we will use the term *TV* for every kind of TV (i.e. traditional TV, internet TV, streaming TV). All kinds of commercials that can be found in a TV program that concern a single brand will be referred to as an *ad*. Finally, commercials that deal with advertising the future broadcast of content in the channel will be referred to as *promo*.

Advertisement breaks are usually flagged by a series of black, silent frames at the beginning and end of each commercials block. However, there are TV stations which do not use black or silent frames to flag commercial segments [59] or these separators are of varying length. Advertisements often present quickly changing visual content, but this may also be the case for certain movie scenes that present high levels of action. Text overlaid regions are also common in advertisements, yet this also happens in TV news (e.g. top and bottom banners). Therefore, multiple features need to be taken into consideration for classifying a piece of video in the *content*, *ad* and *promo* classes. The method proposed in this deliverable is based on a direction that is similar to the general idea of [31] in the sense that it tries to identify channel TV logos, and also related to [96], since they also base the fragmentation of the TV stream on video shot segmentation techniques; but differs from both of these works in the set of features used (which is much richer in the proposed ReTV approach), and in the classes it can classify the TV content to; namely, our method can distinguish between the *content*, *ad* and *promo* classes (whereas all previous works considered only two classes *content* and *ad*).

Distinguishing between a advertisement and actual TV content might be seemingly easy for a human to do, but its a not a trivial task for a computer. To better understand this, one can imagine being in a foreign country whose language is unknown and switching on the TV in the middle of an ongoing program. For a certain amount of time you don't have any information about what type of content you are watching. The conclusion of watching TV content or an advertisement might be formed based on how quickly the visual content changes, on the overlay of text/logos of the screen, or what kind of background music the current content is decorated with. You may be able to remember commercials that were repeated in the day or even detect new commercials of the same brand (provided they don't change their logo). Some patterns may be perceived sooner or later, however it is easy to understand that one has to base their decision regarding this on a plethora of characteristics.

We present an ad detection framework based on the analysis of a multitude of features. Initially, the problem definition from our view is as follows: We wish to distinguish between three classes of TV program: (i) *content*, (ii) *ad*, which is a single advertisement about a brand, and (iii) *promo*. Therefore, given a stream of TV content frames f_i with $i \in [1 \dots n]$ an ad detection system must first find the boundary frame indices b_j with $j \in [1 \dots m], m \ll n$ in which the content class changes, and then classify each of the formed segments to a single class, e.g. $s_k = [f_{b_i} \dots f_{b_{i+1}}]$ and $\mathrm{typeof}(s_k) \in \{\mathrm{content}, \mathrm{ad}, \mathrm{promo}\}$.

Regarding the identification of the boundary frame indices, as previously mentioned, a commonly used characteristic for the segmentation is the high cut rate observed in commercials. The problem of determining the boundaries of TV advertisements can be based on determining shot changes, and thus, methods for video temporal segmentation (where the video switches from one shot to another) can be utilized. We employ the method of [4] for the experiments reported in this section, while in the implementation of ad detection in the VA service the newer method of Section 4.1.2 is used.

Regarding the class assignment, we use a decision tree based on a multitude of extracted features. Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression [23]. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. They are non-linear classifiers like neural networks, i.e., they are generally used for classifying non-linearly separable data. The pool of features we choose to test consists of numerous heterogeneous computed measures. In the enumerated list that follows, we describe each one of them and the intuition behind using them.

1. *channel_id*: The id of the channel logo detected. We used the channel logo detector previously described in the current section (and, since the test dataset for ad detection involved only 7 channels as explained in the sequel of this section, we only considered the detection results for these 7 channel logos. After discussions with digital TV professionals, we learned that most often the channel logo is not depicted in the frame of an advertisement. If no channel logo is found then the value of this feature is -1.
2. *channel_loc*: Again, discussion with digital TV professionals indicated that in TV station promos the channel logo is often missing, or in a different position than in the regular program. To capture this, we split the frame in four equal-sized regions and establish a categorical measure with with four possible values. A value of 1, 2, 3 or 4 denotes that a channel logo is located inside the top-left, top-right, bottom-left, bottom-right region, respectively. If no channel logo is found then the value of this feature is 0.
3. *channel_hd*: Denotes the presence or not of an "HD" logo close to the channel logo. Most often an "HD" logo is included in the TV frame only for movies and thus can be an indication that the class of the current segment is not "ad" or "promo".

4. *channel_live*: Denotes the presence or not of a "live" logo close to the channel logo. A "live" logo is included in the TV frame only for live reportage of TV news and thus is a strong indication that the class of the current segment is not “ad” or “promo”. For the inference of the *channel_hd* and *channel_live* features we used the same object detector that recognizes the 7 channel logos.
5. *shot_temporal_density*: We compute the average duration of the next 5 shots as a means to measure how quickly the visual content changes.
6. *max_ECR*, *avg_ECR*, *var_ECR*: Drawing inspiration from [41] we compute the Edge Change Ratio (ECR) measure for each frame and calculate the maximum, average and variance of the ECR values for frames of the shot.
7. *max_blackness*, *avg_blackness*, *avg_blackness*: We convert the video frame to grayscale, we compute the range of all pixel brightness values as a measure of blackness. Then, for each shot we calculate the maximum, average and variance of the blackness values for the frames of the shot.
8. *max_blurriness*, *avg_blurriness*, *var_blurriness*: We use the variance of the Laplacian of each frame. We then calculate the maximum, average and variance of blurriness values of all frames in a shot.
9. *starts_w_black*, *ends_w_black*: Binary variables, indicating if the shot starts or ends, respectively, with 6 or more black frames.
10. *sin+pca*, *places+pca*: Attempting to incorporate features of semantic nature in the ad-detector decision, we used the concept scores of the SIN and Places365 concept pools, respectively (as discussed in Section 4.2) max-pooled for each shot. The dimensionality of the DCNN-based feature vector is reduced, using principal components analysis (PCA) and forcing the algorithm to select the number of components so that the percentage of the total variance explained by the selected components is greater that 99%. The eigenvector and eigenvalues of PCA are computed on the training dataset, and in the testing phase the same ones are used to infer the lower-dimensionality feature vectors.

Table 12: Specifications of the dataset for the ad detection.

Channel ID	*content* shots/min.	*ad* shots/min.	*promo* shots/min.
1	877/ 77	525/ 26	853/ 31
2	661/ 35	98/ 2	1116/ 32
3	1596/ 89	1398/ 61	516/ 27
136	497/ 23	60/ 1	258/ 20
137	246/ 25	259/ 9	230/ 13
166	1567/111	10/ 0	2296/ 75
167	1202/ 91	6/ 0	2795/ 89
Total	6646/454	2356/102	8064/290

In Table 12 we provide the specifications of the dataset that GENISTAT provided for the purpose of ad detection experiments. This comprises of program from 7 channels, and there are 6646 annotated shots for *content*, 2356 annotated shots for *ad* and 8064 annotated shots for *promo*. As discussed in Section 5.1, it is often the case that EPG data is not perfectly aligned to the actual material. Since the annotation for this dataset was inferred from EPG, we notice that this is also the case for this dataset, namely, in segments annotated as *Ad* there were often some seconds of the *content* class in the beginning. To alleviate this, we excluded the first two shots from the material every time the ground truth annotation changes. The specifications of the filtered dataset that was finally used in our experiments are reported in

Table 13.

Table 13: Specifications of the filtered dataset for the ad detection.

Channel ID	*content* shots/min.	*ad* shots/min.	*promo* shots/min.
1	536/ 41	345/ 11	699/ 16
2	434/ 18	76/ 1	906/ 23
3	1084/ 66	1284/ 56	242/ 15
136	271/ 11	39/ 1	89/ 9
137	87/ 4	215/ 7	74/ 2
166	1121/ 59	2/ 0	2034/ 63
167	661/ 35	2/ 0	2507/ 76
Total	4194/236	1963/ 79	6551/209

For the labeling the shots of TV program with the content, ad and promo labels we tested various classifiers:

1. Decision trees [36]: A decision tree is a supervised learning methods that uses a tree-like graph or model of decisions and their possible consequences. In a decision tree each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from root to leaf represent classification rules.
2. SVM [69]: Support-vector machines are supervised learning models. The multiclass support is handled according to a one-vs-one scheme.
3. KPCA+LDA [97]: A common non-linear version of the literature to overcome the weakness of “linearity” of the Fisher linear discriminant analysis (LDA) method.
4. Random forests [14]: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

The evaluation results on experiments using different classifiers and different subsets of the proposed features in this section, are reported in Section 5.5.

5.4. Implementation Details and Use

We used the Keras[18] open-source neural-network library version 2.1.2, with the TensorFlow backend[19] version 1.12. We used the Keras implementation of the RetinaNet[20] object detection framework for all detection-related features. For the implementation of decision trees, random forest, Kernel PCA and LDA methods we used the scikit-learn[21] machine learning package. The multi-class SVM implementation we use comes from the libSVM library [22]. All experiments where conducted on an Intel i5 3570K PC equipped with an Nvidia GTX 1080 Ti GPU, which was utilized only for the DCNN-based feature extraction.

[18] `https://keras.io/`

[19] `https://www.tensorflow.org/`

[20] `https://github.com/fizyr/keras-retinanet`

[21] `https://scikit-learn.org/stable/`

[22] `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

5.5. Results

We gathered evaluation results from [72, 73, 74, 58] of the recent one-stage object detection frameworks discussed in Section 5.1 on the most recent standard object detection benchmark dataset, the MS COCO [55]. Our findings are reported in Table 14. We observe that YOLOv3 is the fastest, with inference times as low as 22ms per image (when using input images of 320×320 size), while RetinaNet achieves the best performance on this dataset, namely 37.8% mAP (when using the ResNet-101 network as the backbone and input images of 800×800 size). The RetinaNet with the ResNet-101 backbone and 500×500 size for the input images seems to represent a good compromise between time-efficiency and effectiveness. We choose to use this framework for the brand detection, channel logo detection and ad detection parts of the VA service.

Table 14: State-of-the-art one-stage object detection frameworks, and comparison with the two-stage Faster-RCNN detector.

Method	Backbone	Input image height (pixels)	mAP (%)	inference time for an image (ms)
Faster R-CNN	Inception-ResNet-v2	600	34.7	>200
SSD	ResNet-101-SSD	512	28.0	61
YOLOv2	DarkNet-19	416	21.6	25
YOLOv3	Darknet-53	320	28.2	**22**
		416	31.0	29
		608	33.0	51
RetinaNet	ResNet-50	500	32.5	72
		600	34.3	98
		800	35.7	153
	ResNet-101	500	34.4	90
		600	36.0	122
		800	**37.8**	198

The CERTH brands set is the first collection of brand logos that was collected by CERTH. On this brands set, the Faster-RCNN object detection framework was used to establish a first version of the brand detection module in the VA service (as discussed in D1.1). We trained a new model for this brands set, this time using the RetinaNet object detection framework. To compare the performance of the new model to the older one, as well as to other literature methods on various datasets, we evaluated the new model using the respective dataset's test set. In Table 15 we report the results. We observe that the new model outperforms the older by a large margin and, at the same the time, the execution time is considerably lower as can be ascertained by the fifth column of Table 14.

Regarding the newly adopted brand pools, Figures 6 and 7 illustrate some example results for the *Swiss* brand and the *football* logo pools, respectively. As discussed in 5.2, a data augmentation scheme similar to [88] was utilized to construct the training and testing dataset. The background images set is a selection of images from the California ND [45] and Oxford Buildings [68] datasets. The images in these figures are samples from the testing datasets of the corresponding pools. In Fig. 6, sample images for the Coop (Fig. 6a), Zweiffel (fig. 6b), Nescaffe (fig. 6c), ABB (fig. 6d), Navyboot (fig. 6e), Alprausch (fig. 6f) and Freitag (fig. 6g) brands were used, while in Fig. 7, the selected images are of the BSV Cottbus (fig. 7a), Stahl Brandenburg (fig. 7b), Askania Schipkau (fig. 7c) and Blau Weiss Gross (fig. 7d) football

Table 15: Evaluation of the first and second version of logo detection models for the CERTH brands set supported by the VA service.

Dataset	Max detection score reported in the literature	Detection score of the ReTV method in D1.1	Detection score of the current ReTV method
LogosInTheWild [93]	84.2% mAP at closed set, 46.4% at open set [93]	79.8% mAP at closed set	**92.2%** mAP at closed set
WebLogo-2M [87]	34.37% mAP [87]	N/A (only test set is provided and is used for training in our case)	N/A (only test set is provided and is used for training in our case)
TopLogo10 [88]	41.8% mAP [88]	53.3% mAP	**83.3%** mAP
Logos-32plus [11]	94.5% F-score, 95.8% accuracy [11]	94.13% F-score, 92% accuracy	**97.25%** F-score, **98.83%** accuracy
FlickrLogos-47 [23]	48.1% mAP [71]	27.84% mAP	**58.57**% mAP
FlickrLogos-32 [77]	90.3% F-score [76]	94.13% F-score	**98.26%** F-score
FlickrLogos-27 [47]	53% accuracy [47]	81.5% accuracy	**91.5%** accuracy
BelgaLogos [46]	34.11% mAP [46]	23.11% mAP	**41.11%** mAP

teams. We observe that the developed method can detect logos with perspective distortions (e.g., fig. 7a and fig. 6d), logos that are partially depicted in the image (e.g., fig. 6c and fig. 7d), and logos of small size with respect to the background image (e.g., fig. 6f and fig. 6g - note that for presentation purposes, these images are shown cropped in this figure).

For the needs of ad detection we trained an object detector to detect the logos of 16 channels which were decided after discussions with ReTV partners. We used the RetinaNet object detection framework for this task. The channel logo detector was trained on a dataset provided by GENISTAT for this specific reason. We split the dataset to 80% for training the detector and the remaining 20% for testing. There is no overlap between the training and testing dataset. In Table 16 the evaluation results are reported. We observe that the trained model presents near-perfect precision and very good recall. The lowest recall is obtained for channels with logos that exhibit high transparency (e.g, channels *br*, *srf-1* and *ard*). Especially for the *srf-1* channel, there are cases (i.e., key-frames from the videos in the testing set) where even for a human observer it is hard to distinguish the channel logo.

The accuracy of the ad detection method is evaluated in terms of the harmonic mean (F-scores) of the Precision and Recall measures, for each of the content, ad and promo classes. We conducted a multitude of preliminary experiments, testing different parameter values for each classifier. For these preliminary experiments all proposed features were utilized. For the case of polynomial SVM and RBF kernel SVM we performed a grid search for parameters C polynomial degree and parameters C, RBF kernel gamma, respectively. All results reported regarding ad detection experiments were carried out using 10-fold cross validation. For space economy in this deliverable we report in Table 17 the best achieved score of each classifier taking into account all the various experiments conducted. Based on these experiments and observing the poor performance of SVMs and KPCA+LDA, as well as taking into account computational complexity issues, we rejected the use of these methods. Finally, we decided to use the random forest classifier since it scored much higher and the toll of using an ensemble method (compared to using the decision tree classifier) on the performance of the ad detection module is negligible.

We continued the evaluation by assessing the impact of using different set of features as input

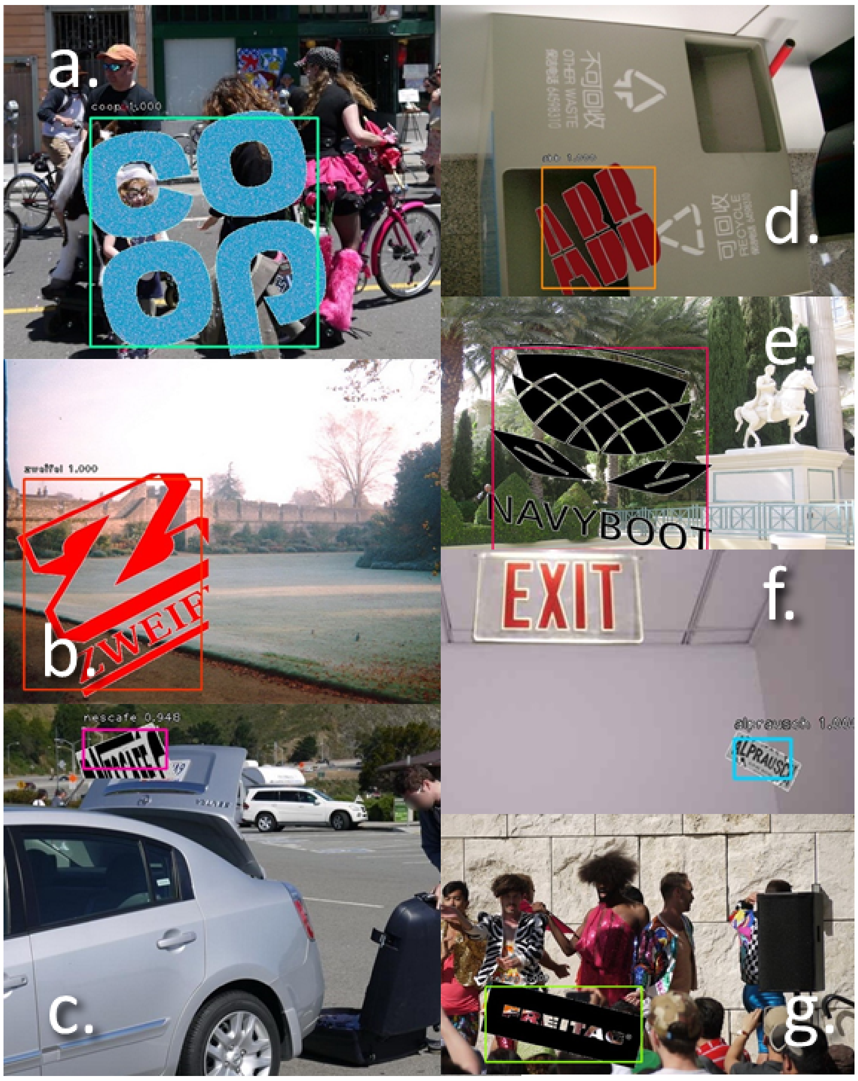


Figure 6: Detection examples for the *Swiss* brands.

to the random forest classifier. In Table 18 we report the results of these experiments. *all* refers to all the non-DCNN-based features. *max* refers to all the non-DCNN-based features except the variances and averages for the features for which we calculate maximum, average and variance values. The feature sets *avg* and *var* are defined correspondingly. *sin_pca* and *places_pca* refer to the usage of dimensionality reduced DCNN-based features of SIN and Places365 concepts pools, respectively. We make the following observations:

- Regarding feature sets that do not utilize DCNN-based features and their combination, the *avg+max* performs the best, yet the performance is slightly lower than using all features (*all* feature set).
- Regarding DCNN-based features, using the *sin_pca* features always boosts the performance while using the *places_pca* features always reduces the performance of the classifier.

Finally, we notice that the best performance is achieved using the *avg+var* non-DCNN-based feature combination along with the dimensionality-reduced features from the SIN concept pool (approach *avg+var+sin_pca*) achieving a 92.75% mean F-score.

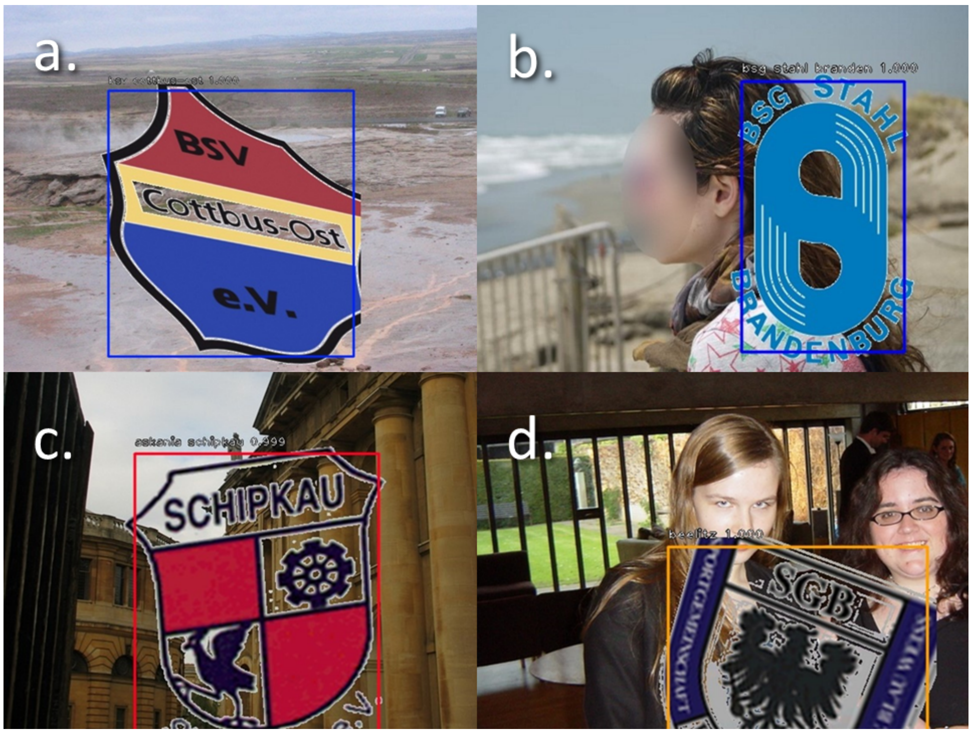


Figure 7: Detection examples for the *football* team logos.

6. Updated Video Analysis Component, Workflow and Extended API

6.1. Video Analysis Component Functionalities and Outputs

The video fragmentation techniques discussed in Section 4.1, the concept-based video abstractions discussed in Section 4.2, as well as the brand detection method discussed in Section 5, have all been incorporated into a single Video Analysis component. The component was deployed as a REST service hosted by CERTH servers. The VA REST service:

1. Retrieves a video file from a web page or a provided zip file.
2. Segments the video file to scenes, shots, and subshots.
3. Selects a number of representative frames (keyframes) from each segmented scene, shot and subshot.
4. Performs concept detection on all selected keyframes.
5. Performs brand logo detection on the selected keyframes of subshots.
6. Performs channel logo detection on the selected keyframes of subshots.
7. Performs ad detection.
8. Extracts and stores features used in the video summarization (WP3) service.

6.2. Component API and Usage Instructions

The service works in an asynchronous way, thus, to use the service, there are three types of call:

Table 16: Evaluation results of the channel logo detection model trained using the RetinaNet framework.

Channel Names	P (%)	R (%)	F (%)
3plus	100	98.9	99.5
4plus	100	99.2	99.6
ard	100	90.2	94.8
br	100	89.2	94.3
prosieben	100	95.3	97.6
rbb	100	96.1	98.0
rsi-la1	100	97.5	98.7
rsi-la2	100	93.4	96.6
rtl	100	93.6	96.7
rtl-2	100	96.2	98.1
rts1	100	95.9	97.9
rts2	100	96.4	98.2
srf-1	100	88.3	93.8
srf-2	100	92.1	95.9
srf-info	100	93.5	96.6
zdf	100	94.5	97.2
hd	95.2	86.4	90.6
live	98.1	85.2	91.2
Total	99.6	93.4	96.4

Table 17: Ad detection results: best score achieved for each classifier in our preliminary experiments.

Classifier	*content*			*ad*			*promo*			Mean F-score
	P	R	F	P	R	F	P	R	F	
Decision tree	82.39	81.76	82.07	83.40	84.24	83.77	87.86	88.01	87.93	85.36
SVM (poly)	92.73	48.05	63.27	83.85	78.88	81.24	75.88	96.47	84.93	80.86
SVM (rbf)	94.85	54.08	68.87	80.66	84.10	82.26	79.06	95.80	86.62	82.87
KPCA+LDA	98.81	80.86	88.93	67.38	78.51	71.96	65.77	71.95	67.70	79.56
Random forests	93.92	81.59	87.30	88.84	92.71	90.66	90.99	95.99	93.42	**91.39**

1. Start call: an HTTP POST call (i.e. has a BODY) which submits the input video and initiates a new session.
2. Status call: an HTTP GET call that queries the status of a session.
3. Results calls: an HTTP GET call that retrieves various information about a successfully completed session.

Start call

```
HTTP POST http://retv.iti.gr:8090/vs
```

Obligatory JSON structured arguments in the POST call body:

- "video_url": a video URL that serves as input to the service. The video is downloaded and analyzed[24]. This can also be a Google Drive link or any link that directly points to

[24]For a complete list of supported sites, see `https://ytdl-org.github.io/youtube-dl/supportedsites`.

Table 18: Ad detection experiment: performance of different feature combinations

	content			*ad*			*promo*			
	P	R	F	P	R	F	P	R	F	mean F-score
all	93.9	81.6	87.3	88.8	92.7	90.7	91.0	96.	93.4	91.4
all+sin-pca	94.5	84.1	89.0	91.6	94.5	93.0	92.0	96.4	94.2	92.6
all+places-pca	92.5	82.2	87.0	90.2	92.7	91.4	91.1	95.6	93.3	91.3
all+sin_pca+places-pca	93.4	84.3	88.6	92.9	93.1	92.9	91.7	96.3	93.9	92.4
max	91.8	78.8	84.7	87.1	92.1	89.5	90.0	95.1	92.5	90.0
max+sin-pca	94.5	83.4	88.6	91.5	94.4	92.9	91.7	96.5	94.0	92.5
max+places-pca	92.5	81.2	86.4	89.2	91.7	90.4	90.4	95.4	92.9	90.8
avg	92.3	79.4	85.3	87.7	91.6	89.5	89.7	95.0	92.3	90.0
avg+sin-pca	94.6	83.6	88.7	92.5	93.8	93.1	91.7	96.9	94.2	92.6
avg+places-pca	92.4	81.3	86.5	89.3	91.0	90.1	90.1	95.3	92.6	90.6
var	92.7	79.7	85.7	87.9	92.3	90.0	90.2	95.3	92.7	90.4
var+sin-pca	93.9	83.9	88.6	92.0	94.1	93.0	92.0	96.5	94.2	92.5
var+places-pca	92.5	81.4	86.6	89.4	91.4	90.4	90.4	95.4	92.8	90.8
max+var	94.3	80.6	86.9	88.4	93.4	90.8	90.9	96.1	93.4	91.3
max+var+sin-pca	94.9	83.5	88.8	91.6	94.6	93.0	91.8	96.7	94.2	92.6
max+var+places-pca	92.6	82.0	87.0	90.0	92.4	91.1	90.8	95.5	93.1	91.2
avg+var	93.4	81.1	86.8	88.6	92.5	90.4	90.7	95.7	93.1	91.0
avg+var+sin-pca	94.8	84.2	89.2	92.2	94.4	93.2	91.9	96.6	94.2	**92.8**
avg+var+places-pca	92.1	82.0	86.7	90.1	91.7	90.8	90.7	95.4	93.0	91.0
avg+max	94.3	80.6	86.9	88.4	93.4	90.8	90.9	96.1	93.4	91.3
avg+max+sin-pca	94.9	83.5	88.8	91.6	94.6	93.0	91.8	96.7	94.2	92.6
avg+max+places-pca	92.6	82.0	87.0	90.0	92.4	91.1	90.8	95.5	93.1	91.2

a downloadable video file.

Optional JSON structured arguments in the POST call body:

- "brands_detection" (int, [0,1]. default=1): Set to 0 to disable brand detection for the current session (resulting in faster execution times).
- "channels" (int, [0,1]. default=1): Set to 0 to disable channel logo detection for the current session (resulting in faster execution times). Note that disabling channel logo detection will also disable the ad detection.
- "saf" (int, [0,1]. default=0): Set to 1 to enable concept detection on the SaF concept pool.

The start POST call returns a JSON file. If the call is successful, the JSON file contains the following fields:

- "message": "The call has been received"
- "session": a unique id of the call (used later to get status or results)

If the call is NOT successful, the JSON file contains the following field:

- "message": "No video or zip URL provided"

Each call analyses one video file (i.e. the session ID returned concerns the analysis of a single video).

html

Status call

```
HTTP GET http://retv.iti.gr:8090/<session>/status
```

The GET call returns a JSON file. If the call is successful, the JSON file contains the "status" and "message" fields:

If the message field equals "The status you requested does not exist", please check that you provided a valid session ID. The status field will contain various messages throughout the procedure of the summarization. If a message containing the word "FAILED" has been received then there was an error during the summarization process. If the "VIDEO ANALYSIS COMPLETED" message has been received, you can proceed to make the results GET calls.

Results calls

There are various GET calls to retrieve information about a completed session. These are the following:

-
  ```
  http://retv.iti.gr:8090/<session>/va_results
  ```
  Retrieves the JSON file with the video analysis results (va.json).

-
  ```
  http://retv.iti.gr:8090/<session>/segm
  ```
  Retrieves the segmentation data in a pickle file (segm.pkl)

-
  ```
  http://retv.iti.gr:8090/<session>/feats
  ```
  Retrieves the features extracted in a pickle file (feats.pkl).

-
  ```
  http://retv.iti.gr:8090/<session>/va_time
  ```
  Retrieves the duration of the video analysis process in seconds as well as a ratio to the original video duration.

-
  ```
  http://retv.iti.gr:8090/<session>/vis
  ```
  Retrieves an html visualization of the concept detection results for each keyframe. You must use this call in a browser.

-
  ```
  http://retv.iti.gr:8090/<session>/log
  ```
  Retrieves a compact log of the summarization process (mostly used by CERTH for debugging and error tracking purposes).

-
  ```
  http://retv.iti.gr:8090/<session>/full_log
  ```
  Retrieves the full log of the summarization process (mostly used by CERTH for debugging and error tracking purposes).

-
  ```
  http://retv.iti.gr:8090/<session>/type
  ```
  Retrieves a JSON formatted response with the *session_type* key which states the type of session. The session type can be "analysis", "summarization", "render" or a combination of the above (mostly used by CERTH for debugging and error tracing purposes).

All sessions that are older than 48 hours are automatically deleted. After that time, any status or results GET calls for these sessions will return the message "The status/results you requested does/do not exist".

Module reloading calls

In an effort to have the VA service ready towards a production-ready state, we implemented HTTP calls that can reload individual modules while the service is in use. The implemented calls are the following:

-
  ```
  http://retv.iti.gr:8090/reload/concept_detection_tools
  ```
  Reloads the module that performs the concept detection for the most concept pools supported by the VA service.

-
  ```
  http://retv.iti.gr:8090/reload/yt8m_tools
  ```
  Reloads the module that performs the concept detection for the YouTube concept pools.

-
  ```
  http://retv.iti.gr:8090/reload/object_detection_thresholds
  ```
  Reloads the module that performs object detection and is used in brand detection and channel logo detection tasks.

-
  ```
  http://retv.iti.gr:8090/reload/merge_pickles_tools
  ```
  Reloads the module that deals with the reading, processing and saving of binary compressed files.

-
  ```
  http://retv.iti.gr:8090/reload/html_aux_tools
  ```
  Reloads the module that deals with the writes the HTML result of the concept visualization tool.

-
  ```
  http://retv.iti.gr:8090/reload/ad_detector
  ```
  Reloads the module that performs procedures related to ad detection.

-
  ```
  http://retv.iti.gr:8090/reload/video_retriever
  ```
  Reloads the module that downloads videos from the web.

-
  ```
  http://retv.iti.gr:8090/reload/json_aux_tools
  ```
  Reloads the module that deals with the reading, processing and saving of JSON-formated files.

-
  ```
  http://retv.iti.gr:8090/reload/retine_aux_tools
  ```
  Reloads the module with auxiliary methods for the object detection using the RetinaNet framework.

-
  ```
  http://retv.iti.gr:8090/reload/logging_tools
  ```
  Reloads the module that deals with the reading, processing and saving of log files.

-
  ```
  http://retv.iti.gr:8090/reload/aux_tools
  ```
  Reloads a collection of various auxiliary methods.

-
  ```
  http://retv.iti.gr:8090/reload/gen_rep_url
  ```

Reloads the module that deals with fetchin data from GENISTAT and various parameters regarding the same procedure.

6.3. Component Workflow and Integration

Several discussions took place among the ReTV partners on how to optimize the workflow and the exchange of data between the ReTV services. Our goal is to avoid duplicating costly (both in the terms of time and storage) procedures and thus a updated, refined framework was decided. An indicative example is the process regarding the ingestion of content from RBB. Specifically, it was decided that the following procedure would be followed:

- GENISTAT accesses the RBB EPG data.
- GENISTAT extracts the desired HLS stream for a program. The HLS format is used because it is the "de facto" standard used to handle video streams, and is also used by RBB.
- GENISTAT issues a *start* call to the VA service of CERTH to analyse the specific program.
- GENISTAT with a regular time interval issues *status* GET calls to the CERTH server to ask whether the analysis was completed.
- Once the VA service finishes the analysis it stores the results, the features extracted and the segmentation data to a number of binary compressed files. A corresponding response is given to the *status* GET calls of GENISTAT.
- GENISTAT retrieves the binary compressed files and stores them to their repository. These files are later deleted from the CERTH server.

The content collection from other partners takes place in a similar manner. This way the results of the VA service of WP1 are available for any other service to use, e.g. the Video Summarization (VS) service of WP3. For example, issuing a call to the VS service to generate a summary for an already analysed RBB video will force the VS service to retrieve the VA service's analysis results from the GENISTAT repository, avoiding re-analyzing the video, thus swiftly performing the actual summarization process.

6.4. Component Testing and Software Quality Assessment

The VA REST service and its constituent methods have undergone extensive testing, both by their developers and by other ReTV partners that use the service for receiving video analysis results. In terms of benchmark testing of the methods implemented in the service, these results have been reported in the respective results sections of the present deliverable (Sections 4.1.4, 4.2.2, 4.2.5, and 5.5). In terms of unit testing, several hundred such tests were conducted to cover all the different possible call configurations of the service (given the number of either mandatory or optional parameters of the REST service API, as detailed in Section 6.2), and also cover edge cases (such as very short or very long videos submitted for analysis; corrupt video files submitted, unsupported formats etc.). In terms of stress testing, as our REST service implements a queuing mechanism, any number of requests that come in a burst or in general while the service is processing another request will be queued, in a first-in-first-out manner. The queuing mechanism was tested with a few hundred analysis requests (videos) submitted in bursts, and behaved as expected, i.e. all the requests were added to the queue and were processed sequentially. Overall, the Video Analysis REST service is a mature software component for use in ReTV, although the possibility of unforeseen edge cases arising in the future during the use of the service cannot possibly be excluded, as with any piece of software. Such problems may also arise as a result of further updates of the service with new processing algorithms or the introduction of additional parameters for controlling the operation of such algorithms, and for this reason the testing and software quality assessment of the Video Analysis REST service will continue alongside the further updates of the service throughout the life of WP1.

7. Conclusion and Outlook

In this deliverable, we described the second version of the ReTV data ingestion, analysis and annotation components. Data collection is active across multiple vectors and languages, feeding annotated documents into the ReTV metadata repository for subsequent analysis tasks (WP2). The NLP/NEL pipeline has been extended and improved, particularly in terms of adding Dutch language support, aligning the keyword and entity annotations and improving the accuracy of our Named Entity annotations. Our efforts to improve the processing speed of the Video Analysis (VA) service led to the implementation of a new and faster DCNN-based video fragmentation method. A novel concept detection method was also introduced, while the concept pool set that is supported by the VA service was expanded. The set of brand pools supported by the VA service was also expanded and new functionalities regarding the Brands Detection task were added, namely, the detection of a set of channel logos and a method for ad detection.

As future outlook, in terms of content collection across vectors, and annotation and knowledge graph alignment, we plan to continually evaluate data quality and correct the data collection pipeline when needed. We will add vectors or languages as needed by partners or stakeholders. Keyword extraction will be extended to other POS classes (verb/adjective) and Named Entity Linking will be improved by full integration with our SKB and extended to better support entities of type Works (e.g. TV series), also through the development of a larger training corpora for ReTV.

Concerning concept-based video abstractions, we plan to continue working on improved deep learning architectures for making concept detection more accurate while at the same time keeping its complexity low enough to enable efficient processing of large amounts of video data, and efficient learning from large-scale annotated datasets; these are necessary for making concept detection more useful in everyday practice. We will also continue to work on the improvement of advertisement detection, which includes documenting its effectiveness on larger-scale datasets, and we will employ it for the accurate alignment of video and the corresponding EPG data - a task that is also of great interest for ReTV. On the software development and implementation side, there is also room for improving the speed of the Video Analysis service; for this, future developments will include unifying the video fragmentation for all granularities in a single, end-to-end framework, and merging the different logo detection models in a single one.

The above and all other future developments in all tasks of WP1 will be reported in D1.3 of ReTV, which will be submitted in June 2020.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, A. P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. In *arXiv:1609.08675*, 2016.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[3] E. Apostolidis and V. Mezaris. Fast shot segmentation combining global and local visual descriptors. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 6583–6587, May 2014. Software available at http://mklab.iti.gr/project/video-shot-segm.

[4] E. Apostolidis and V. Mezaris. Fast shot segmentation combining global and local visual descriptors. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6583–6587. IEEE, 2014.

[5] K. Apostolidis, E. Apostolidis, and V. Mezaris. A motion-driven approach for fine-grained temporal segmentation of user-generated videos. In *International Conference on Multimedia Modeling*, pages 29–41. Springer, 2018.

[6] L. Baraldi, C. Grana, and R. Cucchiara. A deep siamese network for scene detection in broadcast videos. *arXiv preprint arXiv:1510.08893*, 2015.

[7] L. Baraldi, C. Grana, and R. Cucchiara. Scene segmentation using temporal clustering for accessing and re-using broadcast video. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015.

[8] L. Baraldi, C. Grana, and R. Cucchiara. Shot and scene detection via hierarchical clustering for re-using broadcast video. In *International Conference on Computer Analysis of Images and Patterns*, pages 801–811. Springer, 2015.

[9] L. Baraldi, C. Grana, and R. Cucchiara. Recognizing and presenting the storytelling video structure with deep multimodal networks. *IEEE Transactions on Multimedia*, 19(5):955–968, 2016.

[10] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[11] S. Bianco, M. Buzzelli, D. Mazzini, and R. Schettini. Deep learning for logo recognition. *Neurocomputing*, 245:23–30, 2017.

[12] M. Bober-Irizar, S. Husain, E.-J. Ong, and M. Bober. Cultivating DNN diversity for large scale video labelling. In *CVPR Workshops*, 2017.

[13] A. Brasoveanu, G. Rizzo, P. Kuntschik, A. Weichselbraun, and L. J. B. Nixon. Framing named entity linking error types. In N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA), 2018.

[14] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[15] P. Cardinal, V. Gupta, and G. Boulianne. Content-based advertisement detection. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[16] J. R. Cerquides. A real time audio fingerprinting system for advertisement tracking and reporting in fm radio. In *2007 17th International Conference Radioelektronika*, pages 1–4. IEEE, 2007.

[17] M. Covell, S. Baluja, and M. Fink. Advertisement detection and replacement using acoustic and visual repetition. In *2006 IEEE Workshop on Multimedia Signal Processing*, pages 461–466. IEEE, 2006.

[18] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems, ISEM '13, Graz, Austria, September 4-6, 2013*, pages 121–124. ACM, 2013.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[20] T. DeVries and G. W. Taylor. Dataset augmentation in feature space. In *ICLR Workshops*, Toulon, France, Apr. 2017.

[21] T. Devries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv:1708.04552*, 2017.

[22] N. Dimitrova, S. Jeannin, J. Nesvadba, T. McGee, L. Agnihotri, and G. Mekenkamp. Real time commercial detection using mpeg features. In *Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowlwdge-based Systems (IPMU2002)*, pages 481–486, 2002.

[23] M. Dumont, R. Marée, L. Wehenkel, and P. Geurts. Fast multi-class image annotation with random subwindows and multiple output randomized trees. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 2, pages 196–203, 2009.

[24] P. Duygulu, M.-y. Chen, and A. Hauptmann. Comparison and combination of two novel commercial detection methods. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 2, pages 1267–1270. IEEE, 2004.

[25] A. C. S. e Santos and H. Pedrini. Adaptive video shot detection improved by fusion of dissimilarity measures. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002948–002953. IEEE, 2016.

[26] S. Escalera, D. M. J. Tax, O. Pujol, P. Radeva, and R. P. Duin. Subclass problem-dependent design for error-correcting output codes. 30(6):1041–1054, June 2008.

[27] J. M. Gauch and A. Shivadas. Identification of new commercials using repeated video sequence detection. In *IEEE International Conference on Image Processing 2005*, volume 3, pages II–1252. IEEE, 2005.

[28] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[30] N. Gkalelis, V. Mezaris, I. Kompatsiaris, and T. Stathaki. Mixture subclass discriminant analysis link to restricted Gaussian model and other generalizations. *IEEE Trans. Neural Netw. Learn. Syst.*, 24(1):8–21, Jan. 2013.

[31] A. Gomes, M. P. Queluz, and F. Pereira. Automatic detection of tv commercial blocks: A new approach based on digital on-screen graphics classification. In *2017 11th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–6, Dec 2017.

[32] M. Gygli. Ridiculously fast shot boundary detection with fully convolutional neural networks. In *2018 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–4. IEEE, 2018.

[33] R. Hannane, A. Elboushaki, K. Afdel, P. Naghabhushan, and M. Javed. An efficient method for video shot boundary detection and keyframe extraction using sift-point distribution histogram. *International Journal of Multimedia Information Retrieval*, 5(2):89–104, 2016.

[34] A. Hassanien, M. Elgharib, A. Selim, S.-H. Bae, M. Hefeeda, and W. Matusik. Large-scale, fast and accurate shot boundary detection through spatio-temporal convolutional neural networks. *arXiv preprint arXiv:1705.03281*, 2017.

[35] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society. Series B*, 58(1):155–176, July 1996.

[36] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: prediction, inference and data mining. *Springer-Verlag, New York*, 2009.

[37] A. G. Hauptmann and M. J. Witbrock. Story segmentation and detection of commercials in broadcast news video. In *Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries-ADL'98-*, pages 168–179. IEEE, 1998.

[38] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[39] S. Hertling and H. Paulheim. Dbkwik: A consolidated knowledge graph from thousands of wikis. In X. Wu, Y. Ong, C. C. Aggarwal, and H. Chen, editors, *2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17-18, 2018*, pages 17–24. IEEE Computer Society, 2018.

[40] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 782–792, 2011.

[41] X.-S. Hua, L. Lu, and H.-J. Zhang. Robust learning-based tv commercial detection. In *2005 IEEE International Conference on Multimedia and Expo*, pages 4–pp. IEEE, 2005.

[42] P. Huang, Y. Yuan, Z. Lan, L. Jiang, and A. G. Hauptmann. Video representation learning and latent concept mining for large-scale multi-label video classification. *arXiv:1707.01408*, 2017.

[43] Y. Huo, Y. Wang, and H. Hu. Effective algorithms for video shot and scene boundaries detection. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[44] H. Inoue. Data augmentation by pairing samples for images classification. *arXiv:1801.02929*, 2018.

[45] A. Jinda-Apiraksa, V. Vonikakis, and S. Winkler. California-nd: An annotated dataset for near-duplicate detection in personal photo collections. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 142–147. IEEE, 2013.

[46] A. Joly and O. Buisson. Logo retrieval with a contrario visual query expansion. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 581–584. ACM, 2009.

[47] Y. Kalantidis, L. G. Pueyo, M. Trevisiol, R. van Zwol, and Y. Avrithis. Scalable triangulation-based logo recognition. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 20. ACM, 2011.

[48] R. Kannao and P. Guha. Tv commercial detection using success based locally weighted kernel combination. In *International Conference on Multimedia Modeling*, pages 793–805. Springer, 2016.

[49] T. Kar and P. Kanungo. A texture based method for scene change detection. In *2015 IEEE Power, Communication and Information Technology Conference (PCITC)*, pages 72–77. IEEE, 2015.

[50] T. Kohonen. Learning vector quantization. In Springer, editor, *Self-Organizing Maps*, chapter 6. Springer Series in Information Sciences, Berlin, Heidelberg, 1995.

[51] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[52] R. Lienhart, C. Kuhmunch, and W. Effelsberg. On the detection and recognition of television commercials. In *proceedings of IEEE international Conference on Multimedia Computing and Systems*, pages 509–516. IEEE, 1997.

[53] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[54] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[56] N. Liu, Y. Zhao, and Z. Zhu. Commercial recognition in tv streams using coarse-to-fine matching strategy. In *Pacific-Rim Conference on Multimedia*, pages 296–307. Springer, 2010.

[57] N. Liu, Y. Zhao, Z. Zhu, and H. Lu. Exploiting visual-audio-textual characteristics for automatic tv commercial block detection and segmentation. *IEEE Transactions on Multimedia*, 13(5):961–973, 2011.

[58] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[59] S. Marlow, D. A. Sadlier, K. McGeough, N. E. O'Connor, and N. Murphy. Audio and video processing for automatic tv advertisement detection. 2001.

[60] V. Mezaris, K. Apostolidis, and L. Nixon. ReTV Deliverable 1.1: Data Ingestion, Analysis and Annotation. Technical report, 2018.

[61] S. Minaee, I. Bouazizi, P. Kolan, and H. Najafzadeh. Ad-net: Audio-visual convolutional neural network for advertisement detection in videos. *arXiv preprint arXiv:1806.08612*, 2018.

[62] A. Moro, A. Raganato, and R. Navigli. Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244, 2014.

[63] S. Na, Y. Yu, S. Lee, J. Kim, and G. Kim. Encoding video and label priors for multi-label video classification on YouTube-8M dataset. In *CVPR Workshops*, 2017.

[64] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshops*, pages 2999–3007, Venice, Italy,, Oct. 2017.

[65] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, D. Garigliotti, and R. Navigli. Open knowledge extraction challenge. In *Semantic Web Evaluation Challenges - Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, volume 548 of *Communications in Computer and Information Science*, pages 3–15. Springer, 2015.

[66] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, R. Meusel, and H. Paulheim. The second open knowledge extraction challenge. In H. Sack, S. Dietze, A. Tordai, and C. Lange, editors, *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, volume 641 of *Communications in Computer and Information Science*, pages 3–16. Springer, 2016.

[67] F. Odoni, P. Kuntschik, A. M. P. Brasoveanu, and A. Weichselbraun. On the importance of drill-down analysis for assessing gold standards and named entity linking performance. In A. Fensel, V. de Boer, T. Pellegrini, E. Kiesling, B. Haslhofer, L. Hollink, and A. Schindler, editors, *Proceedings of the 14th International Conference on Semantic Systems, SEMANTICS 2018, Vienna, Austria, September 10-13, 2018*, volume 137 of *Procedia Computer Science*, pages 33–42. Elsevier, 2018.

[68] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[69] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[70] P. V. Rangarajan. A pseudo-statistical approach to commercial boundary detection.

[71] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[72] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[73] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[74] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[75] M. Röder, R. Usbeck, S. Hellmann, D. Gerber, and A. Both. N^3 - A collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pages 3529–3533, 2014.

[76] S. Romberg and R. Lienhart. Bundle min-hashing for logo recognition. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 113–120. ACM, 2013.

[77] S. Romberg, L. G. Pueyo, R. Lienhart, and R. Van Zwol. Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 25. ACM, 2011.

[78] D. A. Sadlier, S. Marlow, N. O'Connor, and N. Murphy. Automatic tv advertisement detection from mpeg bitstream. *Pattern Recognition*, 35(12):2719–2726, 2002.

[79] N. Sarafianos, X. Xu, and I. A. Kakadiaris. Deep imbalanced attribute classification using visual attention aggregation. In *ECCV*, Munich, Germany, Sept. 2018.

[80] B. Satterwhite and O. Marques. Automatic detection of tv commercials. *IEEE Potentials*, 23(2):9–12, 2004.

[81] A. Shivadas and J. M. Gauch. Real-time commercial recognition using color moments and hashing. In *Fourth Canadian Conference on Computer and Robot Vision (CRV'07)*, pages 465–472. IEEE, 2007.

[82] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, H. Meinedo, M. Bugalho, and I. Trancoso. Temporal video segmentation to scenes using high-level audiovisual features. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(8):1163–1177, 2011.

[83] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, San Diego, CA, USA, May 2015.

[84] M. Skalic and D. Austin. Building a size constrained predictive model for video classification. In *CVPR Workshops*, 2017.

[85] M. Skalic, M. Pekalski, and X. E. Pan. Deep learning methods for efficient large scale video labeling. In *CVPR Workshops*, 2017.

[86] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro. The implicit bias of gradient descent on separable data. *JMLR*, 19(70):1–57, Jan. 2018.

[87] H. Su, S. Gong, and X. Zhu. Weblogo-2m: Scalable logo detection by deep learning from the web. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 270–279, 2017.

[88] H. Su, X. Zhu, and S. Gong. Deep learning logo detection with data expansion by synthesising context. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 530–539. IEEE, 2017.

[89] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[90] S. Tang, L. Feng, Z. Kuang, Y. Chen, and W. Zhang. Fast video shot transition localization with deep structured models. In *Asian Conference on Computer Vision*, pages 577–592. Springer, 2018.

[91] R. M. Thool and P. P. Gundewar. Detection of tv commercials using sift and phase correlation. 2014.

[92] D. M. Thounaojam, T. Khelchandra, K. M. Singh, and S. Roy. A genetic algorithm and fuzzy logic approach for video shot boundary detection. *Computational intelligence and neuroscience*, 2016:14, 2016.

[93] A. Tüzkö, C. Herrmann, D. Manger, and J. Beyerer. Open set logo detection and retrieval. *arXiv preprint arXiv:1710.10891*, 2017.

[94] A. Weichselbraun, A. Brasoveanu, P. Kuntschik, and L. J. B. Nixon. Improving Named Entity Linking Corpora Quality. In *RANLP 2019*, page 77, 209.

[95] A. Weichselbraun, P. Kuntschik, and A. M. P. Brasoveanu. Name variants for improving entity discovery and linking. In M. Eskevich, G. de Melo, C. Fäth, J. P. McCrae, P. Buitelaar, C. Chiarcos, B. Klimek, and M. Dojchinovski, editors, *2nd Conference on Language, Data and Knowledge, LDK 2019, May 20-23, 2019, Leipzig, Germany.*, volume 70 of *OASICS*, pages 14:1–14:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[96] Q. Xia and G. Tang. Research on tv advertisement detection base on video shot. In *2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization*, volume 2, pages 245–248. IEEE, 2012.

[97] J. Yang, Z. Jin, J.-y. Yang, D. Zhang, and A. F. Frangi. Essence of kernel fisher discriminant: Kpca plus lda. *Pattern Recognition*, 37(10):2097–2100, 2004.

[98] G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 471–480. ACM, 2015.

[99] D. You, O. C. Hamsici, and A. M. Martinez. Kernel optimization in discriminant analysis. 33(3):631–638, Mar. 2011.

[100] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, York, UK,, Sept. 2016.

[101] L. Zheng, Y. Yang, and Q. Tian. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244, 2017.