Enhancing and Re-Purposing TV Content for Trans-Vector Engagement (ReTV)
H2020 Research and Innovation Action - Grant Agreement No. 780656

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

Deliverable 3.3 (M34)

Content Adaptation, Re-Purposing and Scheduling

Final Version

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	3.3
Deliverable Title	Content Adaptation, Re-purposing and Scheduling
Due Date	M34
Submission Date	October 30, 2020
Work Package	WP3
Partners	GENISTAT, CERTH, MODUL TECHNOLOGY, WEBLYZARD
Author(s)	Basil Philipp, Krzysztof Ciesielski (GENISTAT), Konstantinos Apostolidis, Evlampios Apostolidis, Damianos Galanopoulos, Eleni Adamantidou, Alexandros Metsai, Vasileios Mezaris (CERTH), Lyndon Nixon (MODUL), Arno Scharl (WEBLYZARD)
Reviewer(s)	Rasa Bocyte (NISV)
Keywords	Content Adaptation, Content Re-purposing, Video Summarization, Content Scheduling, Text to Video, Text Summarization
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	2020-05-06	Basil Philipp, Konstantinos Apostolidis, Vasileios Mezaris	Initial structure
0.2	2020-06-06	Evlampios Apostolidis, Eleni Adamantidou, Alexandros Metsai	Added text on Video Summarization
0.3	2020-06-05	Krzysztof Ciesielski	Initial text on recommendation and scheduling
0.4	2020-06-11	Basil Philipp	Extended sections on recommendation and scheduling
0.5	2020-06-12	Damianos Galanopoulos	Added text to Text to Video Matching section
0.6	2020-06-12	Konstantinos Apostolidis	Reviewed CERTH's inputs
0.7	2020-06-15	Krzysztof Ciesielski	Reviewed inputs
0.8	2020-06-16	Arno Scharl	Storypact section and restructuring
0.9	2020-06-16	Basil Philipp	Reviewed section 3 and added the Introduction and Conclusion
0.10	2020-06-17	Vasileios Mezaris	Reviewed CERTH's inputs
0.11	2020-06-17	Basil Philipp	Minor restructuring of Section 3
0.12	2020-06-26	Rasa Bocyte	Review
0.13	2020-10-11	Krzysztof Ciesielski	Extend sections on recommendation
0.14	2020-10-16	Basil Philipp	Restructuring of Section 3
0.15	2020-10-17	Arno Scharl	Text summarization and editor mode revision
0.16	2020-10-17	Konstantinos Apostolidis, Damianos Galanopoulos	Added text to Smart Cropping section, and updated the Text to Video Matching section
0.17	2020-10-17	Evlampios Apostolidis, Eleni Adamantidou, Vasileios Mezaris	Updated the Video Summarization section, and added text to the Evaluating Video Summarization section
0.18	2020-10-20	Krzysztof Ciesielski	Report on evaluation results of recommendation algorithm
0.19	2020-10-21	Basil Philipp	Review before QA
0.20	2020-10-22	Rasa Bocyte	QA review
0.21	2020-10-24	Arno Scharl	Review and minor corrections
0.22	2020-10-26	Basil Philipp	Review and minor corrections
1.0	2020-10-27	Lyndon Nixon	Post QA check by coordinator

Statement of Originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

This deliverable reflects only the authors' views and the European Union is not liable for any use that might be made of information contained therein.

Contents

EXECUTIVE SUMMARY

The objective of WP3 is to deliver components for content-adaptation, summarization, recommendation and scheduling. This deliverable is an update of D3.2 and the final deliverable of WP3. It reports on our improved results in content adaptation, summarization, recommendation and scheduling that are employed in ReTV's Content Wizard and 4u2 use case applications.

We present a new architecture for unsupervised video summarization which works together with the previously developed, non-learning method of video summarization.

We present the state-of-the art results of our Text to Video matching service and how we built a search API on top of it. This search API is used in the Content Wizard for improved scheduling of content. We also use Text to Video matching in our extended approach to recommendation in the 4u2 use case applications.

We further present a text editing environment, which assists users in creating and fine-tuning the wording of postings and other dissemination texts. The editor's sidebar provides access to various content recommendation and summarization tools.

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.
DTR	Dilated Temporal Relational unit: a convolutional module for capturing temporal dependencies on various time windows.
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.
GAN	Generative Adversarial Network: a deep learning architecture where two separate neural networks compete against each other.
GRU	Gated Recurrent Unit: a type of recurrent neural network.
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.
KTS	Kernel Temporal Segmentation: a video fragmentation algorithm.
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.
VAE	Variational Auto-Encoder: a generative neural network that models a data distribution.

1. Introduction

This deliverable reports on the work done in WP3 during months 20 to 34. It covers work done in T3.3 "Content Adaptation and Re-Purposing" and T3.4 "Content Recommendation and Scheduling". The tasks T3.1 "Metadata and Vocabulary Interoperability" and T3.2 "Viewer Profiling" ended in M20 and are covered in D3.2.

Section 2 summarizes the work done in T3.3 and explains our updated approaches to video summarization and Text to Video matching. Both video summarization and Text to Video matching are deployed as services and integrated into the use cases.

Section 3 summarizes the work done in T3.4. We present our extended approach to recommendation and scheduling which is specifically targeted at increasing user retention in the 4u2 Messenger. It builds on top of the work present in section 2, resulting in an innovative approach to video recommendation. We also describe and evaluate our newly developed Text to Video search API which is at the core of a novel solution to surface content to editors in the Content Wizard. Finally, content publishers need support not only in the re-purposing of the video to be posted but also benefit from guidance regarding the accompanying text. The Storypact text editing and summarization tool of ReTV described in Section 3.4 provides such guidance. It assists users in creating and fine-tuning the wording of postings and other dissemination texts to maximize their impact. The editor's sidebar provides access to various content recommendation and summarization tools.

2. Video Adaptation and Re-Purposing

2.1. Video Summarization

2.1.1. Updated Problem Statement and State of the Art

A first description of the video summarization task (used in ReTV for content adaption and re-purposing) was provided in Section 4.1 of D3.2; this description is still completely valid. As a brief reminder, video summarization aims to generate a concise synopsis that conveys the important parts of the full-length video; based on this, viewers can have a quick overview of the whole story without having to watch the entire content. Several methods were developed to automate video summarization, and a first overview of the relevant bibliography was given in Section 4.2.1 of D3.2. In the following sections, we provide an update of our first report, focusing on deep-learning-based methods that represent the current state of the art. As a short reminder, a coarse division of these methods can be made between supervised and unsupervised approaches. The former rely on datasets with ground-truth data and try to discover the underlying criterion for video summarization from human-generated summaries. The latter aim to learn video summarization without any use of ground-truth data.

Regarding the class of supervised video summarization techniques, most new algorithms adopt one of the following general directions: i) model the temporal dependency among frames to estimate their importance and select the most important frames/fragments to form the summary, ii) model the spatiotemporal structure of the video to estimate the frames' importance and select the most important frames/fragments to form the summary, or iii) exploit the available textual metadata to perform semantic- or video-category-driven summarization.

In the first direction, [53] builds on [108] (presented in D3.2) and introduces an attention mechanism to model the temporal evolution of the users' interest. Following, it uses this information to estimate frames' importance and select the video key-frames to build a video storyboard. [23] presents a seq2seq network made of a soft self-attention mechanism and a two-layer fully connected network for regression of the frames' importance scores. [64] proposes a hierarchical approach which uses a generator-discriminator architecture (similarly to [65] which was reported in D3.2) as an internal mechanism to estimate the representativeness of each shot and define a set of candidate key-frames. Then, it employs a multi-head attention model to further assess candidates' importance and select the key-frames that form the summary. Finally, [89] stacks multiple LSTM (Long Short-Term Memory; types of Recurrent Neural Networks, as discussed in D3.2) and memory layers hierarchically to derive long-term temporal context, and uses this information to estimate the importance of video frames.

In the second direction, most algorithms rely on advanced network architectures to extract spatiotemporal data about the video. [49] utilizes convolutional LSTMs under an encoder-decoder architecture to identify the spatiotemporal relationship among different parts of the video, and enhances the diversity of the video summary via next frame prediction and next scene detection mechanisms. [105] uses 3D-CNNs (3-Dimensional Convolutional Neural Network) for representing the video content, in combination with convolutional LSTMs to model the spatial and temporal structure of the video, and select the video key-frames. [12] extracts spatial and temporal information by processing the raw frames and their optical flow maps with CNNs, and estimates the importance of each frame using an LSTM and based on human annotations, after training the entire model via a label distribution learning process. [36] trains a neural network for spatiotemporal data extraction, and uses this information to create an inter-frames motion curve. Then, shot segmentation is performed via shot transition detection,

and a self-attention model is utilized to select the key-frames/fragments of the video and form a static/dynamic video summary.

In the third direction, a new method that follows the idea of early supervised approaches and focuses on the high-level semantics of the visual content, was presented in [106]. This method defines a video summary by maximizing its relevance with the available metadata about the video (i.e., title, abstract, keywords), after representing both the visual and the textual information in a common semantic/latent space. [118] follows the paradigm of [101] (presented in D3.2) and learns category-driven summarization by rewarding the preservation of the core parts found in video summaries from the same category (e.g., the main parts of a wedding ceremony when summarizing a wedding video). Similarly, [54] trains action classifiers with video-level labels for action-driven video fragmentation and labeling; then, it extracts a fixed number of key-frames and applies reinforcement learning to select the ones with the highest categorization accuracy, thus performing category-driven summarization. Finally, [96] applies a visual-to-text mapping and a semantic-based video fragment selection according to the relevance between the automatically-generated and the original video description, with the help of semantic attended networks.

By contrast to the aforementioned three general approaches, [9] presents a weakly-supervised approach that uses the principles of reinforcement learning to learn summarization based on a limited set of human annotations and a set of hand-crafted rewards. The later relate to the similarity between the machine- and the human-selected fragments, as well as to specific characteristics of the created summary (e.g., its representativeness). This method applies a hierarchical key-fragment selection process that is divided into sub-tasks. Each task is learned through sparse reinforcement learning (thus avoiding the need for exhaustive annotations about the entire set of frames, and using annotations only for a subset of frames) and the final summary is formed based on rewards about its diversity and representativeness.

With respect to unsupervised video summarization, new methods build on the success of previous approaches, that mainly rely on the use of Generative Adversarial Networks or the principles of reinforcement learning. [41] extends the architecture of [65] (which was discussed in D3.2) with a chunk and stride network (CSNet) and a tailored attention mechanism for assessing the frames' temporal dependency at different granularities to select the video key-frames. [104] tries to maximize the mutual information between the summary and the video using a trainable couple of discriminators and a cycle-consistent adversarial learning objective. [34] presents a self-attention-based conditional GAN to simultaneously minimize the distance between the generated and raw frame features, and focus on the most important fragments of the video. [26] utilizes Temporal Segment Networks (proposed in [91] for action recognition in videos) to extract spatial and temporal information about the video frames, and trains the summarizer through reinforcement learning and a reward function that assesses the maintenance of the video's main spatiotemporal patterns in the produced summary. [115] presents a mechanism for both video summarization and reconstruction. Video reconstruction aims to estimate the extent to which the summary allows the viewer to infer the original video (similar to the GAN-based methods), and video summarization is learned via reinforcement learning, based on the reconstructor's feedback and the output of models assessing the representativeness and diversity of the generated summary.

2.1.2. Video Summarization Updated Approach

In D3.2, we presented our first release of methods for unsupervised video summarization. Both of these algorithms rely on a GAN-based architecture to train a mechanism for key-

frame/fragment selection. The first (termed "SUM-GAN-sl" and presented in Section 4.4.1 of D3.2) uses a Variational Auto-Encoder to reconstruct the original video given the set of weighted feature vectors according to the importance of the corresponding video frames. The second (called "SUM-GAN-AAE" and described in Section 4.4.2 of D3.2) replaces the Variational Auto-Encoder with a deterministic Attention Auto-Encoder (AAE) to better identify the most important frames/fragments of the video. The evaluation of these algorithms on two benchmark summarization datasets (SumMe [30] and TVSum [83]) showed that both of them are highly-competitive against the current state of the art on unsupervised video summarization. Nevertheless, further experimentation with these methods, as well as with other video summarization techniques (dppLSTM [108], DR-DSN [117]) that use similar frame scoring mechanisms, resulted in findings that are consistent with the claims in [41] about the low variation of the computed frame-level importance scores by LSTMs. As a consequence, the selections made by the trained LSTM seem to have a limited impact in summarization; the latter is mainly affected by factors such as the video fragmentation, or the approach used for fragment selection given a target summary length (such as the Knapsack algorithm).

Aiming to tackle the identified limitation, we developed a new architecture for unsupervised video summarization, that is based on a new formulation of the video summarization task. This architecture introduces an Actor-Critic (AC) model into a Generative Adversarial Network, to automatically learn a policy for selecting the most important parts of the video. The learned policy amplifies the differences of the produced importance scores, thus forcing the key-fragment selection process (i.e., the utilized Knapsack method) to choose parts of the video that have been clearly indicated as the most important ones.

The building blocks for defining a new formulation of the video summarization task were the works of [27] and [65]. The former discusses a connection between Generative Adversarial Networks and Actor-Critic models, as the core part of an algorithm that deals with language modelling tasks. The latter, (as reported in Section 4.1 of D3.2) is the first to utilize generative adversarial learning for unsupervised video summarization. As reported in D3.2, [65] introduces the use of a trainable discriminator to automatically define a similarity threshold between the original video and a reconstructed version of it based on a sparse set of selected keyframes (i.e., the video summary). So, we transferred the idea of [27] to the visual domain and formulated the selection of important parts of the video (that will be used to define the video key-fragments and produce the summary using the Knapsack algorithm) as a "visual sentence" generation process. In most existing approaches for real-valued data sequence generation (e.g., text, speech or music synthesis [103]) the used vocabulary of tokens for synthesizing the data sequence is a predefined collection of e.g., letters, words, or music notes. In our conceptualized "visual sentence" generation process this vocabulary is created on-the-fly according to the visual content of the submitted video for summarization. In particular, the tokens of the created vocabulary when summarizing a video, correspond to video fragments of roughly the same length, where each fragment presents a different part of the story. Based on the above, we formulated video summarization as a sequential process that aims to progressively select a set of visual tokens and produce a "visual sentence" that conveys the essential parts and the flow of the story.

To materialize this formulation, we started from the unsupervised summarization algorithm of [65] and built a new architecture (called "AC-SUM-GAN") that embeds an Actor-Critic model into a Generative Adversarial Network to learn the optimal policy for selecting the most important video fragments and form the summary. The Actor has the role of the sequence generator and the generation is performed incrementally based on a set of discrete sampled

actions over a group of video fragments. These actions indicate the selection or not of a fragment and affect the state of the action-state space that is essential for training the AC model. The number of actions N is a hyper-parameter of the architecture, which relates to the duration of the generated summary. The Critic has the role of the evaluator of the Actor's choices and returns a value for scoring each choice according to its impact on the action-state space. Finally, the Discriminator acts as the AC environment and returns a reward that is used to train the AC model, which learns a value function (Critic) and a policy for video fragment selection (Actor). This reward relates to the appropriateness of the Actor's choices that define the video summary, for eventually reconstructing a video that is indistinguishable from the original one. In the sequel we describe in more detail the overall network architecture and the learning objectives and pipeline. With respect to the used notation: capital bold letters denote matrices, small bold letters denote vectors and non-bold letters (either capital or small) denote scalar values.

Figure 1 shows the architecture of the developed AC-SUM-GAN model. The sub-figure on the left side provides details about the building blocks of the architecture and shows how these blocks are connected and interact. Blue coloured rectangles indicate parts related to the Actor-Critic model. The sub-figure on the right side presents the data flow in the architecture. These illustrations show the input and output of each different part of the architecture, thus explaining the role of each part of the architecture and the way that the AC model is used to incrementally select the key fragments of the video and form the summary. On both sides of Fig. 1, dashed lines represent iterative processes during the training of the AC part.

The developed AC-SUM-GAN architecture extends the previous ReTV method, reported in [65] and D3.2, by: i) introducing an AC model for key-fragment selection, ii) adding a new component (called State Generator) that integrates the Frame Selector of [65] (bi-directional LSTM) and produces a state of a fixed length which is essential for training the AC model, and iii) using the Discriminator's feedback to automatically learn a value function (Critic) and a policy for video fragment selection (Actor). In the sequel we present the different parts of the architecture by describing the training workflow.

Given a video of T frames and a linear compression layer that reduces the size of the deep feature vectors, the processing pipeline for training AC-SUM-GAN comprises of:

A **State Generator** that consists of a bi-directional LSTM followed by an average pooling operator. The former captures the temporal dependency over the sequence of frames in both forward and backward direction and assigns a weight to each video frame that represents its importance (frame-level scores $\boldsymbol{s} = \{s_t\}_{t=1}^{T}$ with $s_t \in \mathbb{R}$ and $0 \leq s_t \leq 1$). The latter takes the computed frame-level scores $\boldsymbol{s}$ and produces the initial state $\boldsymbol{f}$ of the AC action-state space by calculating scores at a coarser fragment-level; for this the video is segmented into M non-overlapping fragments of duration d, and a score is computed for each fragment by averaging the weights of the frames included in the fragment ($\boldsymbol{f} = \{f_j\}_{j=1}^{M}$ with $f_j \in \mathbb{R}$ and $f_j = (\sum_{t=(j-1)d+1}^{jd} s_t)/d$).

An **Actor** (fully connected network), who plays an "N-picks" game to explore the action-state space, and in every step i (with $1 \leq i \leq N$) of this game: i) gets the current state ($\boldsymbol{f_i} = \{f_j\}_{j=1}^{M}$), ii) produces a distribution of actions $\boldsymbol{c_i} = \{c_j\}_{j=1}^{M}$, and iii) takes an action p_i by sampling the computed distribution, and picks a video fragment k. This action leads to the next state $\boldsymbol{f_{i+1}}$ of the action-state space, which is produced by zeroing its k^{th} element ($f_k = 0$) to minimize the probability of having the k^{th} fragment re-selected in a subsequent step of the game. Moreover, it affects the computed frame-level weights $\boldsymbol{s}$ by increasing

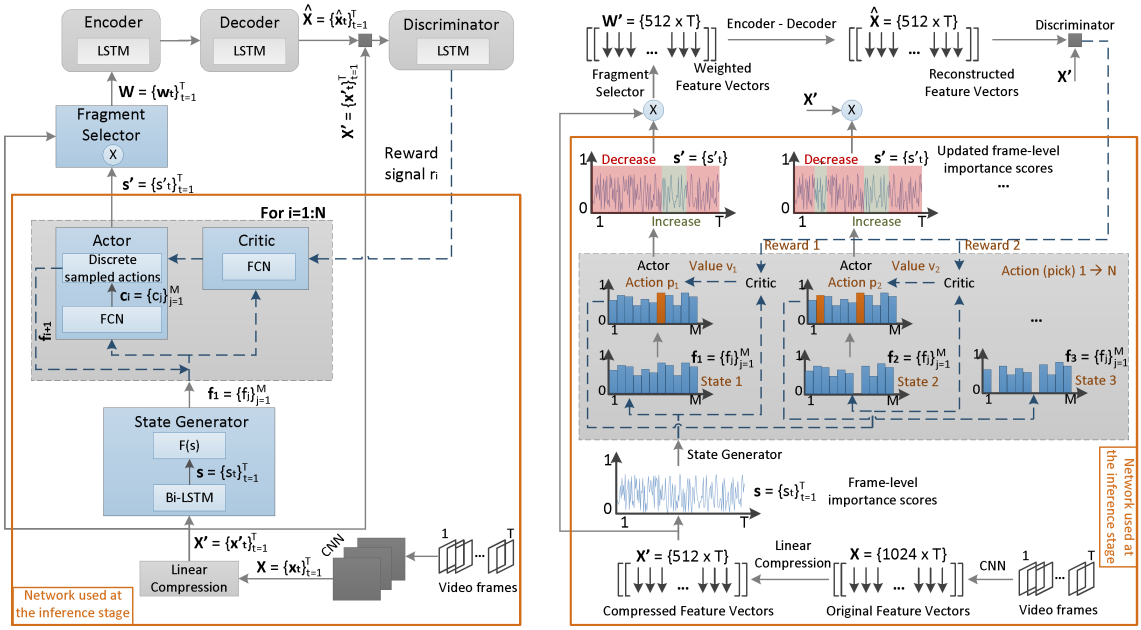

Figure 1: The AC-SUM-GAN architecture. On the left side we show the building blocks of the architecture and their connections. Blue colored rectangles indicate parts related to the Actor-Critic models. On the right side we give an example of the data flow by presenting the input and output of each different part of the architecture. On both sides of the figure, dashed lines represent iterative processes during the training of the AC part. The orange box shows the part of the architecture that is used for inference; at the training stage, the entire architecture is used.

the ones associated to the frames within the selected fragment using action-weighting factors and reducing the ones that correspond to frames of fragments that have not been selected to any step of the game, resulting in a new set of frame-level weights $\boldsymbol{s'}$. For the i^{th} step, these action-weighting factors (AwF) for promoting the selected fragments are computed as follows:

$$AwF_i = \frac{N-(i-1)}{M-(i-1)} + 1,\ i \in [i, N] \tag{1}$$

The reasoning behind the computation of the action-weighting factors is that the model needs to pay more attention to the first-selected fragments, thus the action-weighting factor in step i is larger than the one in step $i+1$.

The reduction factor (RF) is applied to the non-selected fragments only once, at the end of the game, and is computed as follows:

$$RF = (M-N)/M \tag{2}$$

A **Critic** (fully connected network), who is also involved in the "N-picks" game and in every step i (with $1 \leq i \leq N$) of this game: i) gets the current state $\boldsymbol{f_i}$ (generated either at the beginning of the game by the State Generator, or as a result of the Actor's choices in every step of the game) and ii) computes a value ν_i about this state, as an assessment of the Actor's choice.

A **Fragment Selector** (matrix multiplication operator), which uses the updated frame-level scores after each step of the game $\boldsymbol{s'}$, that carry information about the Actor's preferences

w.r.t. the most important (key) fragments of the video, to assign scores to the compressed features of the video frames ($\boldsymbol{X'} = \{\boldsymbol{x'_t}\}_{t=1}^{T}$) and produce a weighted version of them ($\mathbf{W} = \{\boldsymbol{w_t}\}_{t=1}^{T}$).

A **Variational Encoder-Decoder** (LSTMs), which tries to discover the underlying structure of the weighted data after the Actor's choices and reconstruct the original video frames ($\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}_{\boldsymbol{t}}\}_{t=1}^{T}$). The goal of this encoding-decoding process is to minimize the reconstruction error and produce a highly realistic representation of the original video that fools the Discriminator.

A **Discriminator** (LSTM), which forms the AC environment and in every step i (with $1 \leq i \leq N$) of this game: i) gets the compressed feature vectors of the original video $\boldsymbol{X'}$ and the feature vectors of its reconstructed version, based on the Actor's choices and the subsequent encoding-decoding process, $\hat{\boldsymbol{X}}$, ii) defines a new latent representation for each of the aforementioned versions of the video, iii) computes a reconstruction loss (scalar value) based on the proximity of these representations, and iv) returns a reward to the Critic that is calculated as follows:

$$r_i = 1 - L_{recon}, \;\; r_i \in \mathbb{R}, \; i \in [i, N] \tag{3}$$

When the action sampled by the Actor leads to the selection of an already selected fragment, then the returned reward equals to zero to penalize the fragment's re-selection.

The different components of the architecture are trained based on a set of learning objectives and through a 4-step process. The learning objectives for training the State Generator, Encoder, Decoder and Discriminator of the developed AC-SUM-GAN architecture include: a regularization loss ($L_{sparsity}$), a prior loss (L_{prior}), a reconstruction loss (L_{recon}), the "original" (L_{ORIG}) and "summary" (L_{SUM}) losses, and the generator loss (L_{GEN}). For sake of space we provide a short explanation of these losses and refer the reader to Section 4.4.1 of D3.2 for a more detailed description.

- $L_{sparsity}$ aims to force the State Generator to produce a sparse and diverse set of scores based on a regularization factor σ.
- L_{prior} measures how much information is lost when using the Encoder's latent space to represent the VAE's prior distribution.
- L_{recon} estimates the distance between the original and the reconstructed feature vectors.
- L_{ORIG} and L_{SUM} relate to a label-based training approach (labels "1" and "0" denote the original and the reconstructed feature vectors for the adversarial part of our method) and used to train the Discriminator; L_{ORIG} is used to minimize the difference between the computed probability and the "video" label when the Discriminator gets the original video, and L_{SUM} is used to minimize the difference between the computed probability and the "summary" label when the Discriminator gets the summary-based reconstructed video.
- L_{GEN} is used to minimize the difference between the probability computed by the Discriminator when the latter is fed with the reconstructed video and the "video" label, thus forcing the generator to reconstruct a video that is indistinguishable from the original.

To train the introduced AC model, the Actor uses the received feedback from the Critic after each step of the "N-picks" game, and aims to learn a policy that maximizes the probability of an important fragment to be used during the summary generation. This goal is captured by

the following loss:

$$L_{actor} = -\left(\frac{1}{N}\sum_{i=1}^{N} ln\boldsymbol{c_i}\,\alpha_i + \delta\sum_{i=1}^{N} H(\boldsymbol{c_i})\right) \tag{4}$$

where $ln\boldsymbol{c_i}$ and $H(\boldsymbol{c_i})$ represent the logarithm and the entropy of the calculated probability density function $\boldsymbol{c_i}$ at each step of the game, a_i is the advantage that indicates how much better it is to take a specific action compared to the average action at the i^{th} state of the game, and δ is an entropy regularization coefficient. The advantage is defined as the difference between the returns z_i and the values ν_i computed by the critic:

$$\alpha_i = z_i - \nu_i,\ \ i \in [1, N] \tag{5}$$

The return is the discounted cumulative reward of all steps and is computed by the following formula:

$$z_i = \sum_{k=i}^{N} \gamma^{k-i} r_i \tag{6}$$

where r_i is the Discriminator's reward at the i^{th} step of the game, and γ is the discount factor that shows how important future rewards are to the current state ($\gamma \in \mathbb{R}$, $0 \leq \gamma \leq 1$).

Finally, the Critic tries to learn how to evaluate the Actor's choice at the i^{th} step of the game by computing a scalar value ν_i. Its training is based on the following loss:

$$L_{critic} = \frac{1}{N}\sum_{i=1}^{N} \alpha_i^2 \tag{7}$$

The training process is comprised of 4 distinct steps (4 pairs of forward and backward passes), in each of which a different part of the AC-SUM-GAN architecture is trained (Figs. 2 and 3). Specifically, in the 1^{st} step the algorithm performs a forward pass through the entire network, computes L_{prior} and L_{recon} and makes a backward pass to update the Encoder. In the 2^{nd} step, after a forward pass of the partially updated architecture, it computes the L_{recon} and L_{GEN} and uses their sum to update the Decoder. The 3^{rd} step is implemented in 2 sub-steps. In particular, a forward pass of the (once again) partially updated model leads to the creation of the reconstructed feature vectors $\hat{\boldsymbol{X}}$, which are then used for calculating L_{SUM}. Subsequently, the compressed feature vectors of the video frames $\boldsymbol{X'}$ are fed to the Discriminator and L_{ORIG} is calculated. The gradients computed from the losses after two individual backward passes are accumulated and used to update the Discriminator and the linear compression layer that affects the compressed feature vectors.

The training of the remaining components, namely the State Generator, the Actor and the Critic is carried out in the 4^{th} step of this incremental process (see Fig. 3). More precisely, the original feature vectors $\boldsymbol{X}$ pass through the first three components of the partially updated model and produce the initial state ($\boldsymbol{f_1} = \{f_j\}_{j=1}^{M}$) of the action-state space. The latter is given as input to the Actor and Critic which then play the "N-picks" game. In every step i of this game (this iterative process is denoted by the "For loop" and the dashed-line bounding box in Fig. 3) the Critic computes a scalar value ν_i to assess the current state, while the Actor takes an action by generating and sampling the distribution $\boldsymbol{c_i}$. This action affects the computed frame-level weights $\boldsymbol{s}$, resulting in $\boldsymbol{s'}$. As explained before, these scores pass through the remaining components of the architecture that also take part in the game during

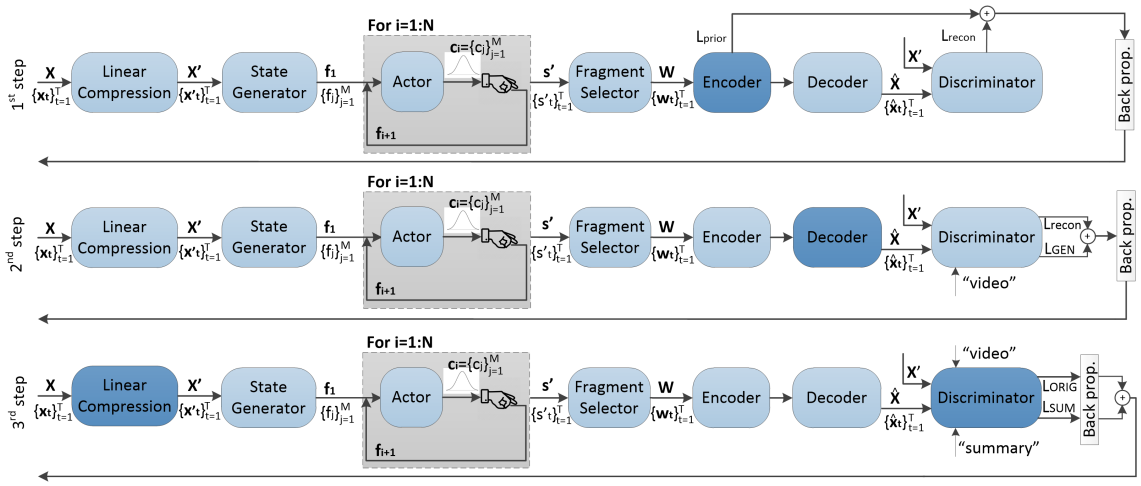

Figure 2: The first three steps of the incremental training procedure. Dark-coloured boxes denote parts updated in each step.

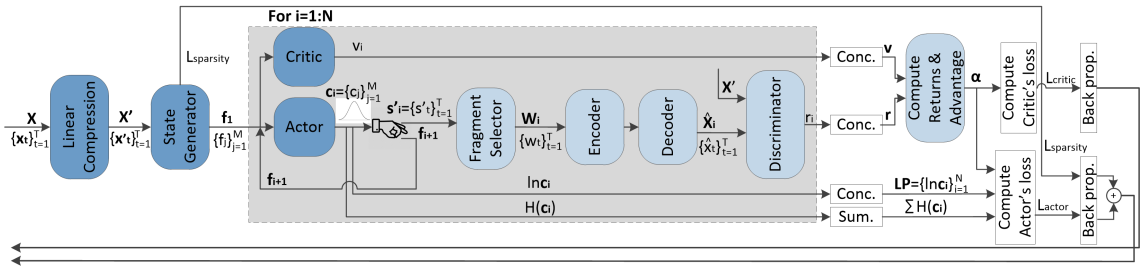

Figure 3: The 4^{th} step of the incremental training procedure. Dark-coloured boxes denote the parts updated in this step.

this 4^{th} step. The reconstructed video is finally assessed by the Discriminator, which computes a reward r_i at each step of the game.

At the end of the game, the architecture produces the vectors $\boldsymbol{v} = \{\nu_i\}_{i=1}^{N}$, $\boldsymbol{r} = \{r_i\}_{i=1}^{N}$, $\boldsymbol{LP} = \{ln\boldsymbol{c_i}\}_{i=1}^{N}$, and the scalar value $En = \sum_{i=1}^{N} H(\boldsymbol{c_i})$, whose elements have been previously described. The former two are used to compute the maximum expected returns and subsequently the advantage of taking a specific action compared to the average, general action at each given state. The computed advantages contribute to the training of the Critic. The training of the Actor is performed simultaneously with the training of the State Generator in a step-wise manner, similar to the Discriminator's training process. It uses the computed advantages $\boldsymbol{a} = \{a_i\}_{i=1}^{N}$, $\boldsymbol{LP}$ and En values to form the L_{actor} and train the Actor, and the $L_{sparsity}$ that trains the State Generator. In this update step, the linear compression layer is also trained.

The added complexity with regards to [65] is the introduction of the AC model (composed of fully-connected networks) for key-fragment selection and the design of a training process that uses the Discriminator's feedback as a reward. However, as shown in Fig. 4, the applied step-wise learning process allows all the different components to be trained effectively, and the AC-SUM-GAN model gets higher rewards as the training proceeds (see the bottom-right sub-figure of Fig. 4).

After the end of the training time, the model can be used for summarizing a new (unseen during

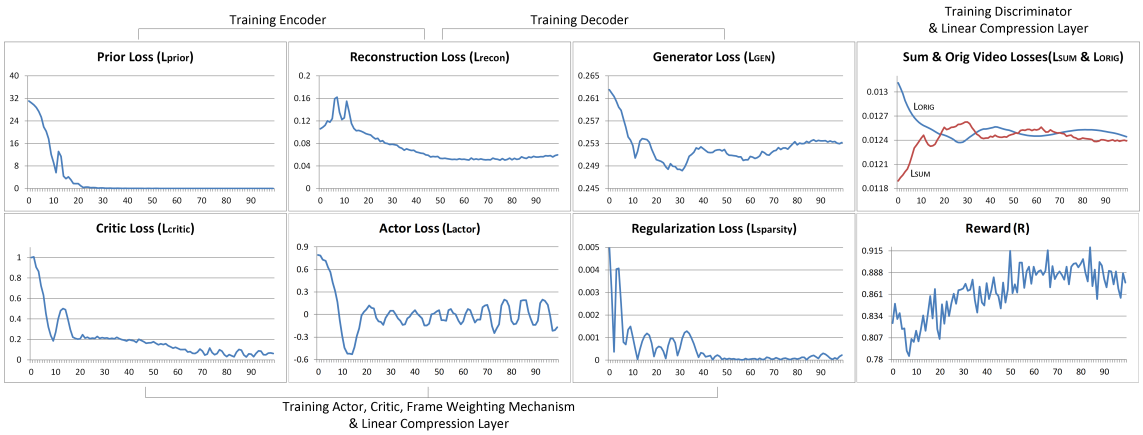

Figure 4: Loss and reward curves for the different components of the developed model. The horizontal axis in all plots indicates the epoch number. These curves indicate the successful training of Encoder, Decoder, Actor, Critic, Frame Weighting Mechanism and Discriminator, and the model's ability to get higher rewards as the training proceeds.

training) video. The utilized components at this stage (inference) are the ones surrounded by the orange boxes in Fig. 1. In particular, at inference time, given a video of T frames, the model gets as input the CNN-based deep feature representations of the video frames ($\boldsymbol{X} = \{\boldsymbol{x_t}\}_{t=1}^{T}$) and produces a sequence of frame-level scores ($\boldsymbol{s'} = \{s'_t\}_{t=1}^{T}$) that signify each frame's importance and thus, its suitability to be included in the summary. This process starts by passing the deep feature vectors through a linear compression layer (fully connected layer for dimensionality reduction) that reduces their size. Then, the State Generator gets the compressed feature vectors and produces the initial state of the action-state space for training the AC model. For this, it assigns an importance score to every video frame according to its temporal dependency with the other frames of the video, and computes fragment-level importance scores via an average pooling operation. Given this state, the trained Actor plays an "N-picks" game and selects N non-overlapping, roughly equal in length, fragments of the video. The Actor's choices result to an update of the initially computed weights, by increasing the scores of the frame sequences corresponding to the selected fragments and reducing the scores of the remaining ones, according to the predefined scaling factors (see Equations 1,2). The updated sequence of frame-level scores - with the selected fragments being clearly indicated by greater importance scores - forms the output $\boldsymbol{s'}$ of the inference stage. This output $\boldsymbol{s'}$ is finally used to define a video summary that does not exceed the target summary duration (in most SoA summarization works this is typically set to 15% of the original video duration, a condition adopted also here to allow direct comparisons). For this, importance scores are computed at the level of video fragments defined using the KTS method [75], and the key-fragments of the video are selected and form the summary using the Knapsack algorithm.

The above-described method does not incorporate any of the editor-specific rules that are also important for the ReTV applications. As a brief reminder, these rules, which were originally reported in Section 4.3 of D3.2, are:

- avoid the selection of segments that include:
  - blurry content
  - visual effects
  - content related to specific parts of a TV news program

  - extreme camera movement
- avoid including two or more visually similar segments
- support taking into consideration a list a concept IDs from the adopted concept pools of ReTV, i.e., query-based summarization
- support taking multiple videos as input and produce a single summary, i.e., multi-video summarization
- support a user-defined parameter to adjust the generated summary's length
- support a user-defined parameter that adjusts the pace of segments' changing, i.e., the rhythm of the summary

In order to incorporate these editorial rules, we further extend the method described thus far in this section, as follows. First, the frame-level importance scores are calculated using the above described method. Taking into account the shot segmentation that the Video Analysis service of WP1 computes, we calculate the shot-level scores by averaging the scores of the frames of each shot. We continue to rank the shots based on this shot-level score, producing a first shot ranking. We also compute a second shot ranking by employing the editor-specific rules, as described in Section 4.3 of D3.2. We combine these two rankings by averaging the rank of each shot, producing a final, combined shot ranking that does incorporate the ReTV editorial rules. The summary script is then generated using this final ranking, by applying to it the exact same procedure that was described in the 4^{th} paragraph in Section 4.3 of D3.2. In this way, the adjustment of the summary length and summary rhythm parameters, as well as the multi-video and query-based summarization functionalities, are still supported by the new video summarization method. It should be stressed that the target summary duration is user-defined and can be freely adjusted, as in our previous ReTV methods; the "15% of the original video duration" that is mentioned earlier in this paragraph is merely a training parameter for the learning-based method, and is also used in order to allow the direct comparison of the learning-based video summarization method's results to those of other literature methods (15% is the commonly-used parameter for summary length in the relevant literature).

Complying with the agreed ReTV architecture, all features necessary to apply the editor-specific rules as well as the learning-based summarization scores are extracted in the Video Analysis service of WP1 (see Section 6.2 of D1.3), are stored by GENISTAT and are readily available for the Video Summarization service of WP3.

2.1.3. Evaluating Video Summarization: A Study on the Established Evaluation Protocol and a New Robust Evaluation Measure

In parallel to the development of the AC-SUM-GAN method, we studied in more details the established protocol for evaluating video summarization, which relies on the use of the SumMe [30] and TVSum [83] datasets. As a brief reminder of our reportings in Section 4.6.1 of D3.2, these datasets provide a set of videos along with multiple human annotations for each video; in SumMe the annotations indicate the selected video fragments that form the video summary, while in TVSum they correspond to values signifying the importance of each frame of the video. To enable matching between key-fragment-based summaries (i.e., to compare the user-generated with the automatically-defined summary), videos are first segmented into consecutive and non-overlapping fragments. Then, based on the determined scores for the fragments of a given video (through the analysis), an optimal subset of them (key-fragments) is selected and forms the summary. The alignment of this summary with the user summaries for this video is evaluated by computing F-Score in a pairwise manner. In particular, the F-Score

for the summary of the i^{th} video is computed as follows:

$$F_i = \frac{1}{W_i} \sum_{j=1}^{W_i} 2 \frac{P_{i,j} R_{i,j}}{P_{i,j} + R_{i,j}} \quad (8)$$

where W_i is the number of available user-generated summaries for the i^{th} test video, $P_{i,j}$ and $R_{i,j}$ are the Precision and Recall against the j^{th} user summary, and they are both computed on a per-frame basis. Most commonly, the evaluation involves the use of a small set of randomly-created training/testing splits of the utilized dataset and the typical experimental setting found in the literature can be summarized as follows: 80% of data is used for training and the remaining 20% for testing; the length of the generated summary should not exceed 15% of the original video duration; experiments are usually conducted on 5 different randomly-created data splits and the average performance is reported. The above described evaluation protocol - with slight variations that mainly relate to the number of experiments using different randomly created splits of the data (5-splits; 10-splits; "few"-splits; 5-fold cross validation) - has been adopted by the vast majority of the SoA works on video summarization (e.g., [66, 38, 69, 111, 117, 97, 109, 118, 113, 23, 21, 42, 104, 5, 3, 116, 13, 35, 77, 90, 50, 57]).

We examined the aforementioned evaluation approach from a perspective that is aligned with our view regarding the characteristics of an optimal evaluation protocol for video summarization. More specifically, such a protocol should be applicable to a small set of data splits and provide results that are highly representative of the method's performance. In this way, the evaluation outcomes on a few data splits (e.g., on a set of 5 splits, as is commonly the case in the literature) would be generalizable to any large set of data splits, that typically enables more safe conclusions about a method's performance. This would also allow reliable comparisons among algorithms that have not been assessed on the exact same set of data splits. To our knowledge, whether the established evaluation approach has these properties or not, has not been investigated thus far. Nevertheless, such a study is particularly important for assessing the reliability of the reported comparisons in video summarization works.

Our study started with the evaluation of five video summarization methods (two supervised: dppLSTM [108], VASNet [23]; and three unsupervised: DR-DSN [117], SUM-GAN-sl [5], SUM-GAN-AAE [3]) using the established protocol and a fixed set of 5 randomly-generated data splits of the SumMe and TVSum datasets (that simulates the evaluation conditions of most SoA works). Besides our SUM-GAN-sl and SUM-GAN-AAE methods, the other three utilized methods are, to our knowledge, the only ones for which implementations are publicly available, and thus allow us to run our experiments. Then, we examined the extent to which the evaluation outcomes are generalizable on a significantly larger set of 50 splits. Finally, we compared our findings with the performances reported in the corresponding papers and assessed the reliability of comparisons that do not consider common evaluation conditions (i.e., the exact same data splits) for all methods, but simply rely on the reported results. In both cases, splitting into training and testing data was based on the typical approach in most SoA works; i.e., 80% of data was used for training and the remaining 20% for testing.

The results of these evaluations, along with the reported performances in the relevant papers (see column "Rep.") are presented in Table 1. In most cases there is a noticeable difference between the results obtained using the small and the large set of splits. These differences do not necessarily indicate performance reduction in the large set, and are often larger than differences between the methods. Furthermore, the methods' rankings are quite different on the small and large set of splits, and do not match the ranking based on the reported results.

	SumMe			TVSum		
Splits	5	50	Rep.	5	50	Rep.
dppLSTM [108]	40.8	<u>41.7</u>	38.6	<u>59.6</u>	<u>57.4</u>	54.7
VASNet [23]	**44.2**	**43.1**	**49.7**	**63.1**	**59.6**	**61.4**
DR-DSN [117]	38.7	41.1	41.4	57.6	56.0	57.6
SUM-GAN-sl [5]	<u>43.9</u>	40.9	47.3	59.2	57.2	58.0
SUM-GAN-AAE [3]	41.0	41.4	<u>48.9</u>	58.7	56.2	<u>58.3</u>

Table 1: Comparison (F-Score (%)) of five publicly-available video summarization approaches in SumMe and TVSum datasets, using 5 and 50 randomly-generated splits. Column "Rep." reports the score from the relevant paper. Best score shown in bold, second-best is underlined.

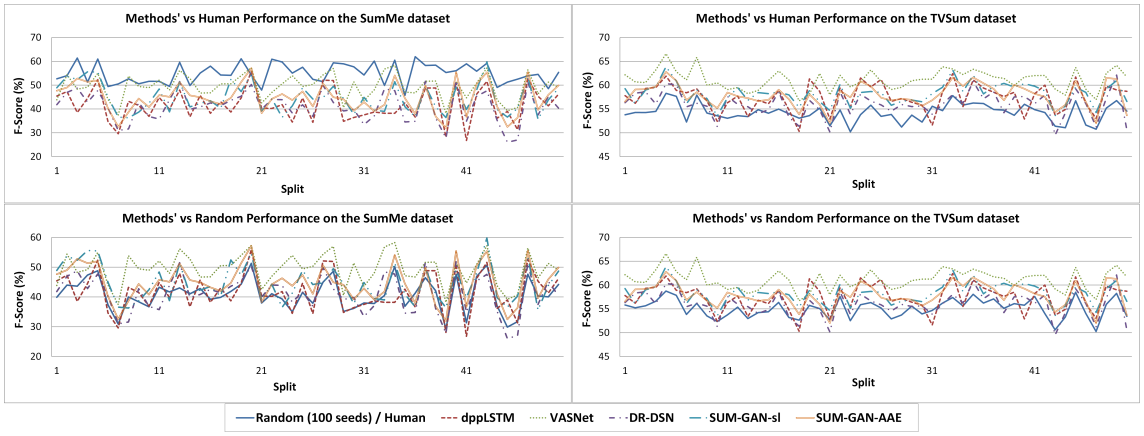

Figure 5: Visualized performance for the tested summarization methods, the random and the human summarizer in the SumMe (left side) and TVSum (right side) datasets. Many similarities can be observed between the performance curves.

All these remarks point out a serious lack of reliability of comparisons that do not use the exact same set of data splits.

To identify the reasons for the varying performance of all tested algorithms in the different evaluation settings, we grouped the recorded values on a per-split basis. The result is depicted in Fig. 5 and makes it obvious that there is a noticeable variability in the performance of the examined algorithms over the set of splits. Moreover, this variability follows a quite similar pattern for all methods, i.e., the performance curve of a summarization method is similar to the curves of the other algorithms. These observations point to different levels of difficulty for the used splits, a fact that clearly affects the outcomes of the performance evaluation.

Aiming to reduce the impact of the utilized data splits, we investigated the existence of a potential association between the methods' performance and a measure of how challenging each split is. Our study resulted in the design of a new evaluation protocol, called "Performance over Random" (PoR), that is presented in [2]. This protocol makes estimates about the difficulty of each used data split and utilizes this information during the evaluation process. The experiments reported in [2], document the ability of the proposed PoR evaluation protocol to provide more representative results about the performance of a summarization method, and more reliable results when comparing methods run on different data splits.

For the needs of the ReTV project, we focused on the main conclusion of the aforementioned study, that is the existence of different levels of difficulty among the randomly created data

splits, which affect the results of the performance comparisons. Based on this finding, we used the exact same set of data splits when comparing the developed AC-SUM-GAN method with: our previous SUM-GAN-sl and SUM-GAN-AAE methods (see Table 3); its implemented variants as part of the conducted ablation study (see Table 6); as well as other publicly-available summarization methods (marked with asterisk in Tables 4 and 5). These comparisons allow us to document the achieved improvement over the last 14 months of the ReTV project, the positive impact of the AC model when it is used as suggested, to select the most important parts of the video, and the competitiveness of the developed AC-SUM-GAN method against other SoA summarization approaches.

2.1.4. Implementation Details and Use

Video summarization was implemented and tested using the PyTorch[1] open-source neural-network library version 1.5. Most of the experiments were conducted on an Intel i7 3770K PC with 32GB of RAM, running Ubuntu 18, equipped with an Nvidia GeForce GPU (GTX 1080 Ti) or PC's with similar specifications.

2.1.5. Results

Video Summarization Experimental Setting

For evaluating our new AC-SUM-GAN algorithm we used the established benchmark datasets and evaluation protocols in the relevant bibliography (discussed briefly in Section 2.1.3 of the deliverable and described in more details in Section 4.6.1 of D3.2). The performance of our unsupervised AC-SUM-GAN method was assessed on the SumMe [30] and TVSum [83] datasets. We evaluated the generated summaries with the F-score measure, which, as mentioned before, expresses the overlap between a machine-generated and a user-defined ground-truth summary. Concerning the split of data for training and testing, we again followed the established approach (e.g., [108] and most literature works) of using 80% of the videos of each dataset for training and the remaining 20% for testing. We run experiments on 5 different randomly-generated splits for each dataset and in the following we report the average performance.

With respect to the different adopted settings and made configurations when training and testing our method, we did the following. Similarly to other SoA summarization works, videos were downsampled to 2 fps. Then M, the number of non-overlapping and temporally equal video fragments, is dictated by the shortest video in our datasets, which is represented by 60 frames. So, $M = 60$ is the most fine-grained video representation possible, and this hyper-parameter needs to be the same for all videos for training the AC model. The duration d of each video fragment equals to the number of frames of a video divided by M. The target summary length must not exceed 15% of the original video duration, a convention adopted by most video summarization approaches. With regards to the number of steps N, given the target summary length (as stated before, 15% of the original video duration is the common choice of most SoA works and adopted here to allow direct comparisons), $N = 15\% \cdot M = 9$. Deep representations of frames were obtained by taking the output of the pool5 layer of GoogleNet [86] trained on ImageNet (similar deep features are used in most SoA works). The linear compression layer reduces the size of feature vectors from 1024 to 512. The State Generator, Encoder, Decoder and Discriminator components are composed of 2-layer LSTMs with 512 hidden units, while the State Generator's LSTM is a bi-directional one. Actor and Critic consist of 4 and 5 fully connected layers respectively (see Fig. 6). The output of the last layer of

[1] https://pytorch.org/

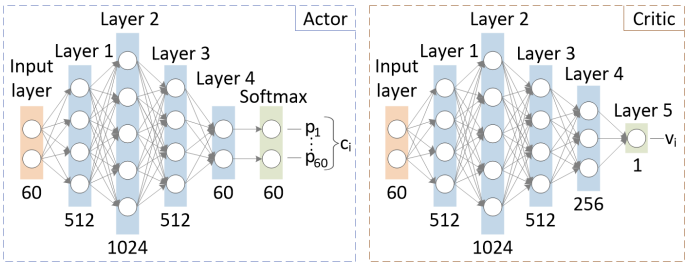

Figure 6: The architecture of Actor and Critic models. The values below each layer's sketch represent the size of the layer (number of nodes).

the Actor is fed to a softmax layer, to form a categorical distribution of probabilities. The output of the last layer of the Critic is a scalar value between 0 and 1. The value of the discount factor γ is set to 0.99 in order to assign high importance to future rewards. Finally, the value of the entropy regularization coefficient δ is set to 0.1, following the example of other publicly-available implementations of the Actor-Critic model [2]. The AC-SUM-GAN model is trained in a full-batch mode (i.e., batch size is equal to the number of training samples) using the Adam optimizer. The learning rate for all components but the discriminator is 10^{-4} and for the latter one is 10^{-5}. Training stops after a maximum number of epochs (100 in our case), and a well-trained model is selected according to a designed criterion which targets the maximization of the received rewards and the simultaneous minimization of the Actor's loss. To promote reproducibility of our reportings, the implementation of the AC-SUM-GAN model will be made publicly available on GitHub.

Selecting the Trained Model

We started our experimentation by studying different criteria for selecting a well-trained model after the end of the unsupervised training process. In particular, we evaluated the performance of the developed AC-SUM-GAN architecture when the trained model is selected based on the training set only and according to:

- The maximization of the overall received reward, computed as the mean of the received rewards r_i after each step of the "N-picks" game (so $i \in [1, N]$) that guide the training of the Actor-Critic model (the reward is a typical factor for early stopping when training relies on reinforcement learning; such a criterion is used in [117]).
- The maximization of the overall received reward and the simultaneous minimization of the Actor's loss L_{actor}, which is the main component of the AC-SUM-GAN model that is involved in the key-fragment selection process during the inference stage.
- The minimization of the reconstruction loss L_{recon} that signifies a maximum alignment between the original and the summary-based reconstructed video, and thus a representative summary.
- The simultaneous minimization of the reconstruction L_{recon} and sparsity losses $L_{sparsity}$; the latter is used (in combination with L_{actor}) for training the model's components used at the inference stage (i.e., the linear compression layer, the State Generator and the Actor).
- The maximization of the overall received reward and the simultaneous minimization

[2] https://github.com/dennybritz/reinforcement-learning/tree/master/PolicyGradient

Criterion / Dataset	Reward	Reward & Actor loss	Recon. loss	Recon. & Sparsity loss	Reward & Recon. loss
SumMe	49.0	**50.8**	50.1	49.8	49.0
TVSum	60.5	60.6	60.7	**60.8**	60.0

Table 2: Performance comparison for different model selection criteria. Values represent F-Score (%).

of the reconstruction loss L_{recon}, that both indicate maximum similarity between the original and the summary-based reconstructed video, and thus a representative summary.

Driven by the remarks in [65] about the impact of the regularization factor σ on the summarization performance, we considered several values for this parameter (i.e., σ ranges in $[0.1, 1]$ with a step equal to 0.1). Instead of manually choosing a value, the best value for σ is also selected based on the used criterion for model selection. So, this criterion is responsible for selecting a well-trained model by indicating both the training epoch and the value of the regularization factor σ.

The results reported in Table 2 show that the impact of the employed criterion is much more pronounced on the SumMe dataset, whereas on the TVSum dataset different criteria lead to much smaller variation. Based on these results, we selected and used in all subsequent experiments as criterion for model selection, the maximization of the overall received reward and the simultaneous minimization of the Actor's loss, which leads to the highest performance on SumMe and a near-optimal performance on TVSum.

Video Summarization Evaluation Outcomes

The developed method was initially compared against our earlier approaches presented in D3.2 (SUM-GAN-sl and SUM-GAN-AAE). The results in Table 3 document the achieved progress towards the improvement of learning-based video summarization during the last 14 months of the project. The performance of the AC-SUM-GAN algorithm surpasses the performance of both SUM-GAN-sl and SUM-GAN-AAE techniques. More specifically, the replacement of the Frame Selector component of SUM-GAN-sl, by the Actor-Critic model and the State Generator - that is mainly needed for supporting the training of the Actor and Critic - leads to an increase in summarization performance equal to 3.0 percentage points on SumMe and 2.2 percentage points on TVSum. Moreover, our latest method AC-SUM-GAN exceeds our attention-based approach (SUM-GAN-AAE) by 1.9 percentage points on SumMe and 2.3 percentage points on TVSum. So, 14 months after the initial release of technologies for automatic video summarization, we have now made available a new method that further advances the performance of the learning-based video summarization component of the ReTV platform, and thus, improves the efficiency of the ReTV content adaptation and re-purposing workflow.

	SumMe	TVSum
SUM-GAN-sl (M20 of ReTV)	47.8	58.4
SUM-GAN-AAE (M20 of ReTV)	48.9	58.3
AC-SUM-GAN	**50.8**	**60.6**

Table 3: Comparison of the different developed learning-based video summarization methods in ReTV, in terms of F-Score (%).

	SumMe		TVSum		AVG
	F-Score	Rank	F-Score	Rank	Rank
Random summary	40.2	10	54.4	9	9.5
Online Motion-AE [110]	37.7	12	51.5	11	11.5
SUM-FCN$_{unsup}$ [78]	41.5	9	52.7	10	9.5
DR-DSN [117]	41.4	–	57.6	–	–
*DR-DSN [117]	38.7	11	57.6	6	8.5
EDSN [26]	42.6	8	57.3	7	7.5
UnpairedVSN [77]	47.5	5	55.6	8	6.5
PCDL [115]	42.7	7	58.4	4	5.5
ACGAN [34]	46.0	6	58.5	3	4.5
SUM-GAN-sl [5]	47.8	4	58.4	4	4
SUM-GAN-AAE [4]	48.9	3	58.3	5	4
CSNet [41]	**51.3**	1	58.8	2	**1.5**
AC-SUM-GAN	50.8	2	**60.6**	1	**1.5**

Table 4: Comparison with different unsupervised video summarization approaches, on SumMe and TVSum. Rank denotes the relative ranking of the compared methods. Methods marked with asterisk (*) have been evaluated also using the same data splits with AC-SUM-GAN.

The AC-SUM-GAN model was then compared against a random summarizer and a set of SoA unsupervised video summarization methods, on the SumMe and TVSum datasets. To estimate the performance of a random summarizer, importance scores for each frame are randomly assigned based on a uniform distribution of probabilities. The corresponding fragment-level scores are then used to form video summaries using the Knapsack algorithm and a length budget of maximum 15% of video duration. Random summarization is performed 100 times for each video, and the overall average score is reported. The results in Table 4 show that: i) the use of GANs for unsupervised learning of the video summarization task is a good choice, as the five top-performing methods (AC-SUM-GAN, CSNet, SUM-GAN-AAE, SUM-GAN-sl, ACGAN) rely on this learning framework; ii) algorithms that use reinforcement learning and tailored reward functions (DR-DSN, EDSN) are less competitive than the GAN-based approaches, especially on SumMe; iii) a few methods (placed at the top of the Table) perform approximately equally to the random summarizer in at least one of the used datasets; finally, iv) the top-performing methods (AC-SUM-GAN, CSNet) try to tackle the limitation of the LSTM-based models that relates to the low variance of the predicted importance scores for the video frames. Concerning the top-performing methods, we see that AC-SUM-GAN is the best on TVSum and the second best on SumMe, while the opposite is observed for CSNet; so, practically we have a tie between these two methods. The competitive performance of CSNet is mainly affected by the use of a tailored variance loss function which aims to increase the variance of the estimated frame-level importance scores. In our AC-SUM-GAN method the boost in performance is gained by the use of a trained AC model that uses the Discriminator's feedback to learn a policy for key-fragment selection.

Our unsupervised AC-SUM-GAN model was also compared with SoA supervised video summarization approaches, despite the fact that this is a rather unfair comparison for our method. The data presented in Table 5 shows that: i) once again a few methods (placed at the top of the Table) exhibit random performance in at least one of the used datasets; ii) a number of summarization techniques (Tessellation, MAVS) that exhibit high performance on one dataset perform very poorly on the other; iii) the developed unsupervised AC-SUM-GAN model performs consistently well on both datasets and, based on the average ranking after consid-

ering both datasets, is the 3^{rd} top-performing method among a large set of SoA supervised techniques; finally, iv) the best-performing approaches utilize tailored attention mechanisms (VASNet, H-MAN, CSNet$_{sup}$) or memory networks (SMN) to capture variable- and long-range temporal dependencies respectively, and we attribute their good performance on these mechanisms.

Ablation Study

To assess the contribution of each of the major components of our model, we conducted an ablation study. This study involves the following variants of the AC-SUM-GAN model:

- **AC-SUM-GAN w/o VAE**. This variant excludes the Variational Auto-Encoder, and the weighted feature vectors at the output of the Fragment Selector are directly forwarded to the Discriminator (i.e., $\hat{\mathbf{X}} = \mathbf{W}$). Therefore, the incremental training of this variant involves only the 3^{rd} and 4^{th} step of the entire process (see Fig. 2 and 3).
- **AC-SUM-GAN w/o Discriminator**. This variant leaves out the Discriminator. Hence, the model is not trained under an adversarial manner and the similarity between the original and summary-based reconstructed version of the video (expressed by the reconstruction loss) is estimated through the direct comparison of the corresponding feature vectors. As a consequence, the 3^{rd} step of the incremental training process of Fig. 2 is omitted.
- **AC-SUM-GAN w/o Actor-Critic**. This variant does not contain the Actor-Critic model and the State Generator's function $F(s)$ that is essential only for training the Actor-Critic model. Consequently, the 4^{th} step of the applied training process (Fig. 3) updates only the State Generator and the linear compression layer using the sum of $L_{sparsity}$ and L_{recon}.

To eliminate the impact of the model selection criterion, in this set of experiments we considered a fixed σ value equal to 0.5 (which is the median of the σ values considered in our experiments) and manually selected the best trained model according to its performance on the test set (thus, a performance higher to the reported one in Tables 4 and 5 can be recorded). Once again, we ran this experiment on the same group of 5 randomly-created data splits and we report the average performance. The results in Table 6 show that the introduction of the Actor-Critic model has a clearly positive impact on the summarization performance on both datasets, which is more pronounced on SumMe. Moreover, the other two major components of the developed architecture, i.e., the Variational Auto-Encoder and the Discriminator, are also shown to have a positive impact on performance.

In order to investigate what is the computational complexity of embedding an AC model into GAN-based summarization architectures (such as the SUM-GAN model and its existing variations), we measured the training and inference times for AC-SUM-GAN against its variation without AC. Results averaged over 5 data splits of the SumMe and TVSum datasets show that the training time is increased by 55% - this is expected given the additional parameters that need to be learned; however, there is no noticeable difference at the inference stage - in both cases, video summarization takes less than 0.2 seconds.

[3] This literature work uses a different evaluation protocol; for this reason we do not present this result here.

	SumMe		TVSum		AVG
	F-Score	Rank	F-Score	Rank	Rank
Random summary	40.2	27	54.4	22	24.5
vsLSTM [108]	37.6	29	54.2	23	26
SASUM [96]	40.6	25	53.9	24	24.5
ActionRanking [21]	40.1	28	56.3	20	24
APDVS [54]	41.2	21	51.3	25	23
ESS-VS [107]	40.9	23	–	–	23
H-RNN [112]	41.1	22	57.7	17	19.5
vsLSTM+Att [53]	43.2	18	– [3]	–	18
DSSE [106]	–	–	57.0	18	18
DR-DSN$_{sup}$ [117]	42.1	19	58.1	15	17
dppLSTM [108]	38.6	–	54.7	–	–
*dppLSTM [108]	40.8	24	59.6	8	16
dppLSTM+Att [53]	43.8	15	– [3]	–	15
WS-HRL [10]	43.6	17	58.4	13	15
UnpairedVSN$_{psup}$ [77]	48.0	6	56.1	21	13.5
SUM-FCN [78]	47.5	8	56.8	19	13.5
SF-CVS [36]	46.0	11	58.0	16	13.5
MAVS [24]	40.3	26	**66.8**	1	13.5
SASUM$_{fullysup}$ [96]	45.3	12	58.2	14	13
PCDL$_{sup}$ [115]	43.7	16	59.2	10	13
CRSum [105]	47.3	9	58.0	16	12.5
Tessellation [43]	41.4	20	64.1	3	11.5
DQSN [118]	–	–	58.6	11	11
HSA-RNN [114]	44.1	14	59.8	7	10.5
ACGAN$_{sup}$ [34]	47.2	10	59.4	9	9.5
SUM-DeepLab [78]	48.8	4	58.4	13	8.5
CSNet$_{sup}$ [41]	48.6	5	58.5	12	8.5
VASNet [23]	49.7	–	61.4	–	–
*VASNet [23]	44.2	13	63.1	4	8.5
SMLD [12]	47.6	7	61.0	5	6
H-MAN [64]	51.8	2	60.4	7	4.5
SMN [89]	**58.3**	1	64.5	2	**1.5**
AC-SUM-GAN	50.8	3	60.6	6	4.5

Table 5: Comparison of our unsupervised method with supervised video summarization approaches on SumMe and TVSum. Rank denotes the relative ranking of the compared methods. Methods marked with asterisk (*) have been evaluated also using the same data splits with AC-SUM-GAN.

	SumMe	TVSum
AC-SUM-GAN w/o VAE	53.0	61.1
AC-SUM-GAN w/o Discriminator	53.3	60.7
AC-SUM-GAN w/o Actor-Critic	50.4	60.7
AC-SUM-GAN	**54.5**	**61.4**

Table 6: Ablation study based on the performance (F-Score (%)) of three variations of the developed model, on SumMe and TVSum.

Video Summarization Qualitative Analysis - A Summarization Example

In addition to the above reported findings, we illustrate the quality of the produced summaries by the developed AC-SUM-GAN method with an example. For this, we used video #15 of the TVSum dataset (titled as "How to Clean Your Dog's Ears - Vetoquinol USA") that is used for the same purpose in a few other SoA works (e.g., [108, 41, 65, 104, 10, 34]), and we compared the performance of AC-SUM-GAN against our previous methods (SUM-GAN-sl and SUM-GAN-AAE) and the three approaches that were used when studying the established evaluation protocol (dppLSTM [108], VASNet [23], DR-DSN [117]). Fig. 7 gives an overview of the video after selecting one frame per shot (shot segmentation performed by KTS) and presents the results for the examined techniques. In each case, the gray bars denote the averaged human-annotated importance scores for the frames of the video, the black vertical lines within these bars correspond to the shot boundaries, and the coloured bars indicate the selected key-shots for creating the summary. Moreover, for each method we provide an illustration of the generated summary by selecting one representative keyframe from each one of the major key-shots of the summary.

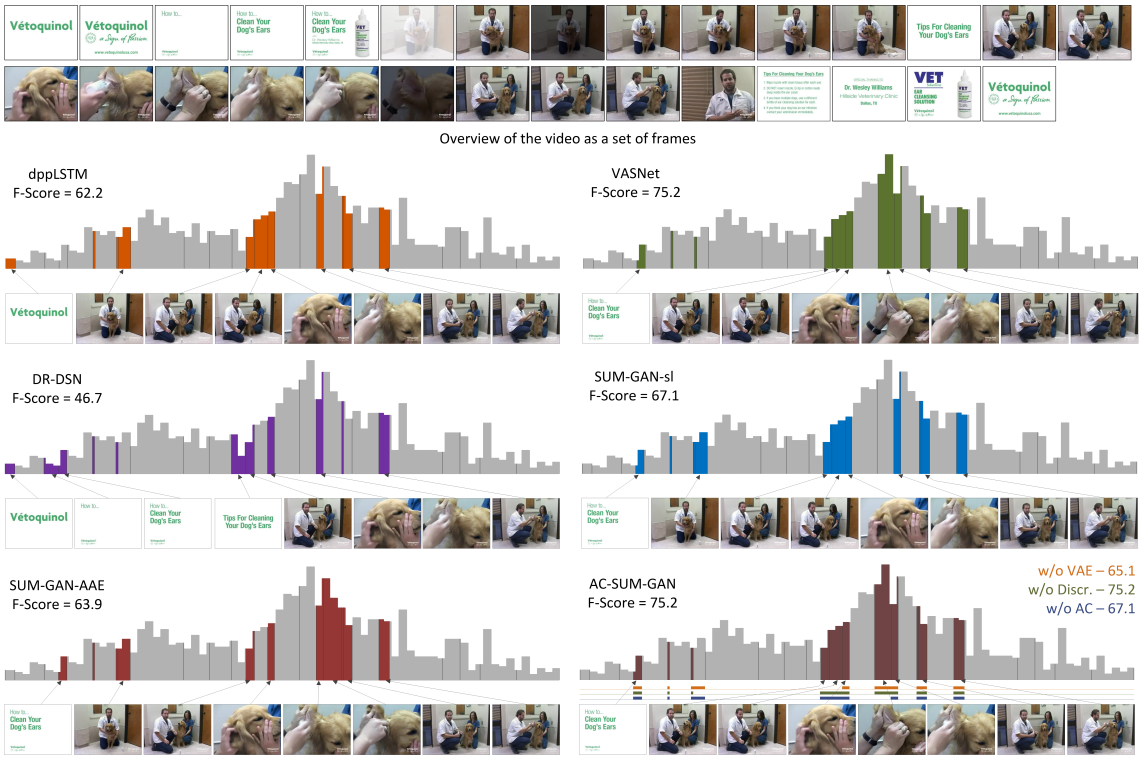

Figure 7: A key-frame-based overview (using one key-frame per shot), and example summaries of six summarization methods on video #15 of the TVSum dataset (the first two methods, dppLSTM and VASNet, are supervised, while the rest are unsupervised). For AC-SUM-GAN, we also illustrate with coloured horizontal line segments under the corresponding bar-chart, the result of each of the three variations of it discussed in the ablation study.

These results show that the developed unsupervised AC-SUM-GAN method generates the exact same summary with the VASNet algorithm, which is one of the best-performing supervised summarization approaches on TVSum. And the superiority of these two algorithms is proven also in terms of F-Score (see values plotted under each method's name). The generated summary focuses on the main event of the video (i.e., the cleaning of the dog's ears), but it

also contains shots with diverse visual content from other parts of the video. In this way, it provides a comprehensive presentation of the entire story, with a special focus on its main event. With regards to other techniques, our previous attention-based SUM-GAN-AAE algorithm also selects some fragments of top importance, ending up to a similar visual result with AC-SUM-GAN and VASNet (the difference in terms of F-Score is due to the imperfection of the KTS segmentation algorithm, which erroneously splits one shot in more shots; and, in this example, such a fragment that is visually similar with the best selection ended up in the summary). The three remaining methods focus less on the main event, with DR-DSN losing the point of the video and choosing many frames that mainly contain graphics.

Finally, to examine the impact of each of the main components of the AC-SUM-GAN architecture on the summarization outcome, at the bottom-right part of Fig. 7 we illustrate also the selected fragments by each different variation of the AC-SUM-GAN model. The coloured line segments right below the bar-chart show that the variation without the Discriminator produces the exact same summary with the AC-SUM-GAN method. The other two variations lead to different and slightly worse summaries. The model without the AC misses the selection of the most important part of the video, while the model without the VAE also misses some important part of the main story by instead selecting a video part that is of lower importance according to the ground-truth annotations. These findings are consistent with the findings of the conducted ablation study and indicate the positive impact of the introduced AC model in the summarization performance. Concluding, the above discussed findings of the conducted qualitative analysis confirm the competitiveness of the AC-SUM-GAN method, that was documented through the quantitative evaluations. And once again, they highlight the achieved improvement on automatic video summarization over the last 14 months of the ReTV project.

2.2. Smart Cropping

2.2.1. Problem Statement and State of the Art

Videos created for traditional TV and desktop computer monitors are typically consumed in landscape aspect ratios (16:9 or 4:3). With the rise of mobile devices (mobile phones and tablets), these historical aspect ratios do not deliver the best user experience. Due to the widespread usage of such devices many video sharing platforms now dictate the use of certain aspect ratios for videos that are to be published on their platform. ReTV's Content Wizard application considers multiple target vectors for publishing and needs to comply with the standards of each of these vectors. A straight-forward approach for transforming a video to a different aspect ratio would involve either static cropping of content or padding the frames with black borders to reach the target aspect ratio. However, static cropping can often lead to significant loss of visual content which in certain cases might even be the center of attention, while padding shrinks the original video content by introducing large borders in the output video. In both cases, ultimately the results are often unsatisfactory.

The video aspect ratio transformation algorithms can be divided in three main categories: 1) warping, 2) cropping, and 3) seam carving. Warping methods [61, 98, 95, 29, 93, 94, 59], instead of resizing the entire image uniformly, determine scaling factors in a content-adaptive way: the image is divided using a grid and important image regions are left untouched while scale factors are applied to other less important areas. Cropping techniques [84, 81, 63, 62, 88, 44] select a rectangular area in the image/frame and discard visual content outside of it. Seam carving algorithms [1, 6, 79, 80, 28, 76, 46, 45] remove seams of uninteresting pixels, i.e., connected paths of pixels inside the image are discarded with the most transparent way to

the human eye. We should highlight here that, to our knowledge, there is not an established benchmark dataset publicly available for testing video aspect ratio transformations.

It is easily understood that when applying warping or seam cropping to the frames of a video, apart from undesirable artifacts, the original video content is distorted significantly. For example, let us consider a video frame depicting two persons at the edges of the image and the content between being a uniform background. A warping method would shrink the uniform area while a seam cropping method most probably entirely remove this area. Such strong semantic distortions are unacceptable for ReTV applications, therefore we choose to implement a cropping method for video aspect ratio transformation.

Most cropping methods rely on modelling human's eye visual attention to decide the size and location of the final crop window. Older visual saliency techniques [32, 37, 85, 52, 40] utilize low-level image features such as intensity, contrast, color, edges. With the advent of deep neural networks (DNN) and their successful application on many computer vision tasks, most recent visual saliency methods [87, 48, 73, 19] employ some sort of DNN.

For ReTV applications we need a method with two main characteristics: 1) the algorithm must be faster than real-time, i.e., the return times must be well below the duration of the input video, and 2) all regions of interest in the video frames must be included. Since none of the discussed literature methods or off-the-self application solutions (such as Google's AutoFlip[4]) offer a way to rate the quality of the cropped version, we decided to construct our own technique that satisfies the aforementioned characteristics and fits well within the already mature framework of ReTV components. Section 2.2.2 details our algorithm. Acknowledging that a cropped version of a video cannot always retain all regions of interest of the original video, and in order to achieve the second desired characteristic of our method, we also discuss a way to quantify the quality of the cropped version and resort to alternative approaches for transforming a video to a different aspect ratio, in the cases where the results of cropping are unsatisfactory.

2.2.2. Smart Cropping Approach

We developed a smart-cropping method that can adjust the video summaries to a target aspect ratio different than that of the original video. This is conducted by detecting the area of the main interest/attention and following it throughout the whole video. Our algorithm can also quantify the quality of the resulting smart-cropped video summary and can automatically decide when a padded version of the original summary video would be preferred.

The smart-cropping module receives as input the set of frames comprising the segments of the original video that the video summarization method has selected for forming a video summary. We start by excluding the rows and columns of video frames of which the variance of all frame pixels throughout the whole video is below a threshold t_b. This way we detect and exclude regions at the edges of the video frames which are either a black border or where the visual content is changing little to none throughout the video. We set $t_b = 30$, to compensate for black or still areas that may have an amount of noise, e.g., due to heavy compression or multiple re-encodings of the video.

Following this, we calculate the video crop window that abides to the target aspect ratio, specifically the dimensions of the final crop window. To minimize the loss of visual content, we select the maximum dimension of the crop window as the minimum dimension of the

[4] https://ai.googleblog.com/2020/02/autoflip-open-source-framework-for.html

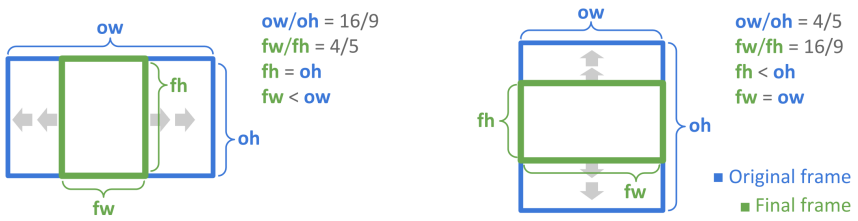

Figure 8: Calculation of the final crop window size that respects the target aspect ratio when transforming a video from a) 16:9 to a 4:5, b) 4:5 to a 16:9 aspect ratio. Blue color denotes the original frame, while green color denotes the final (transformed) frame. The grey arrows depict the dimension along which the final frame can move within the original frame.

original frame size. Employing this practice, we retain as much as possible of the original video content and also simplify the algorithm's computations since we only have to calculate the movement of the crop window in one dimension. For example, when transferring a 16:9 video to a 4:5 target aspect ratio, the final crop window height will be equal to the original video's height and the crop window will be able to move only in the x-axis (Fig. 8.a). Similarly, when transferring a 4:5 video to a 16:9 target aspect ratio then the final width will be equal to the original video's width the crop window will be able to move only in the y-axis (Fig. 8.b).

To find the center of the viewer's attention in a frame, we compute a salience map of the frame. A salience map is a gray-scale image of the same size to the original frame that shows how much each pixel stands out from its neighbors - a measure of visual attention of humans towards important information. The higher the value of a pixel the higher the salience. For each of the input frames we compute the salience map by employing the UNISAL [19] state-of-the-art visual saliency method.

We continue to eliminate regions of small salience by zeroing the pixel values of the salience map that are below a pre-specified threshold. Note that even after the thresholding procedure, the salience can be concentrated in a small region of the whole frame or be in the form of multiple blobs. Aiming to select the main part of the viewer's focus, we cluster the coordinates of the non-zeroed pixels' locations in the salience map. To do so, we selected the HDBSCAN [68] clustering algorithm due to its numerous merits for the specific type of application:

- HDBSCAN does not require pre-specifying the number of clusters - a particular trait for our case since it is impossible to know how many (if any) salient blobs result from the salience map inference stage.
- The clusters extracted by HDBSCAN can take any irregular shape of uneven size - unlike other common clustering algorithms, e.g., K-Means, where clusters are more or less spherical.
- HDBSCAN performs DBSCAN [47] over varying epsilon values, a crucial parameter of the later algorithm. The results are then integrated to find a clustering that gives the best stability over epsilon allowing the finding of clusters of varying densities. We have no way of knowing the density of clusters we want to detect beforehand therefore tuning the epsilon parameter in DBSCAN would be a tedious task.
- HDBSCAN forms clusters at dense regions of data points. Ignoring the low-density areas, it can identify data points (i.e., pixel positions in our case) as outliers. Excluding these

outliers from the rest of the smart-cropping procedure can further aid the discard of scattered or spurious salient pixels.

After applying the clustering procedure, we select the cluster with the highest weight - expressed as the sum of pixel values - and continue to zero the pixel values of the rest of the clusters (as well as data points marked as outliers), producing a filtered salience map.

We go on to find the center of mass of the filtered salience map by performing K-Means with $k = 1$ on the set triplets of x, y position coordinates and pixel intensity values. Incorporating the pixel intensity values in this process we make sure that brighter (i.e. more salient) pixels of the salience map will have highest weight in the final outcome. We consider the position of K-Means resulting cluster centroid as a single point center of the viewer's attention. In an effort to speed-up the K-Means process we initialize the algorithm using the $[x_c, y_c, v_c]$ triplet as the initial center of cluster, where x_c, y_c are the x, y coordinates of the location of the salience map's pixel with the maximum intensity value and v_c is this maximum pixel intensity value. We also significantly limit the number of iterations that the algorithm can perform.

Having a single point location where the viewer's attention is centered, we move on to fit a temporary crop window to this center, to be able to compute a "coverage" score as the ratio of the sum of all salience map pixels values to the values of pixels inside the temporary crop window, as a means to quantify how much salient visual information is covered by the temporary crop window. It is easily understood that the higher the value of the "coverage" score across all frames of the video, the better the quality of the final product of the smart-cropping procedure.

After the above described procedure has been conducted for all frames, we end up with a time series of the locations of the center of the main attention as a result of its movement throughout the video and a series of "coverage" scores for each frame. If the mean "coverage" scores of all frames is above a pre-specified t_c threshold we proceed to process the time series of the locations of the center. We consider each of the x and y coordinates independently and apply a Butterworth fourth-degree low-pass filter at 3Hz (considering as sampling rate the frame rate of the original video) in order to reject sudden movements. Consequently, we employ the LOESS (locally weighted smoothing) method, a popular tool used in regression analysis, to fit a smooth curve to the data points of x and y coordinates time series. Finally, we infer the final crop window for each frame based on x and x coordinates and the calculated final crop window size. The temporary crop window calculated earlier is discarded since this was not computed on the smoothed times series (the sole purpose of that temporary crop window was to be able to calculate the "coverage" score). In the case where the mean "coverage" scores of all frames is below or equal to the t_c threshold, we pad the video frames with zeros (i.e., we add black borders) so as to reach the target aspect ratio. All various parameters of low-pass filters and LOESS smoothing procedure were decided upon preliminary experiments and visual inspection of the results. The t_c threshold was set after specific experiments that are reported in Section 2.2.4.

In an effort to further speed-up the whole process, and based on the fact that temporally-closeby frames of videos most often present high visual similarity, we decided to skip the three next consecutive frames for each frame we process. We interpolate the sparse time-series of the attention's center location, prior to the low-pass filtering process, in order to fill-in the missing values resulting from this frame skipping process. Once again, the number of frames we skipped was decided upon preliminary experiments and visual inspection of the results, noticing negligible difference in the quality of the smart-cropping results. Results of all performance

optimization efforts are reported in Section 2.2.4.

The described smart-cropping method in this section was deployed as a module in the VS service and is employed after the generation of a video summary, if the user explicitly sets a target aspect ratio in the call to VS. API details on the setting of this parameter are detailed in Section 2.4.2.

2.2.3. Implementation Details and Use

Smart-cropping was implemented using Python. All deep learning modules were employed using the PyTorch[5] open-source neural-network library version 1.5. All experiments were conducted on Intel i5 or i7 PCs with 32 of RAM, running Windows 10, equipped with Nvidia GeForce GPUs.

2.2.4. Results

As discussed in Section 2.2.1, to our knowledge there is no established benchmark dataset for the task of video smart-cropping. To be able to test our approaches and the effect of adjusting various parameters during the development of our algorithm, we formed a development/validation dataset by selecting 50 videos from the DHF1k dataset [92] as well as from YouTube. Our selection criteria were devised with the goal to find a balanced set of both straight-forward and challenging videos for the task at hand. We proceeded to generate two smart-cropped versions for each video, one with target aspect ratio of 1:3 and another one with target aspect ratio of 3:1. We selected these target aspect ratios (despite not being used in real life applications) in order to test our algorithm under extreme circumstances. All smart-cropped versions of videos were manually evaluated with a "quality" score ranging from 1 to 5, with 1 being an unacceptable cropped version of the video - mainly due to loss of significant visual content of the original video - and 5 being a perfect cropped result of the original video.

In Figure 9 we plot the "coverage" score that our algorithm calculates against the manually assigned "quality" score to the smart-cropped versions for all of the 50 videos in our manually curated development/validation dataset. We observe that these two measures are related in the sense that when our algorithm cannot fit a sufficient portion of the salient regions in the smart-cropped video version (i.e. there is a low "coverage" score), then the quality of the result will also be lacking, i.e. there will be also a low "quality" score. It is clear that by setting $t_c = 0.76$ all smart-cropped versions with "quality" equal to 1 and 2 will be discarded, i.e., our method will instead choose to produce a padded version for each of these videos in order to reach the target aspect ratio. At the same time, most of the rest of the cropped versions will have "quality" score above 3. The detection of a good quality smart-crop result and the automatic decision to produce an alternative padded version of the original video, was our original intent as discussed in Section 2.2.1.

Aiming to avoid introducing an extra stage in the video summarization process that presents heavy time-complexity, in Section 2.2.2 we discuss our performance optimization efforts of the whole smart-cropping process. In Table 7 we list all stages of our method and include the ratio of the process time to the original video duration, before (second row of the table) and after (third row of the table) applying any speed-up scheme. Of course, these timings depend on the available hardware, especially the *salience map inference* stage which utilizes a GPU. We notice the significant speed-up that the optimization efforts offer. The final achieved

[5] `https://pytorch.org/`

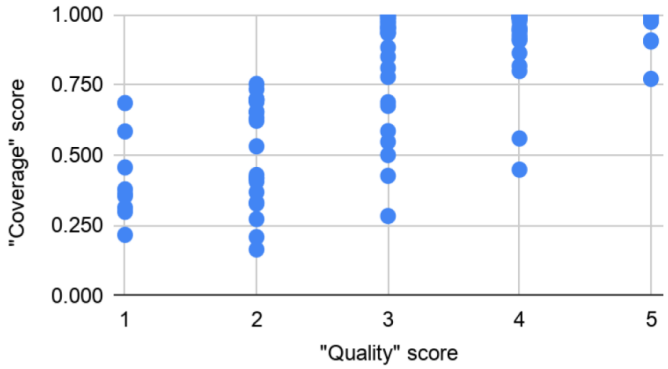

Figure 9: Scatter plot of "coverage" score against "quality" score for the 50 videos of our development/validation dataset.

processing time is 55% of the video duration, which was deemed as an acceptable time budget for the smart-cropping task.

Stage	**Processing time (% of video duration)**	
	V1	**V2**
Border detection	0.118%	0.120%
Crop window size calculation	<0.001%	<0.001%
Saliency map inference	111.782%	33.315%
Saliency map thresholding	0.862%	0.19%9
HDBSCAN	42.903%	11.709%
Center of mass calculation	13.137%	4.243%
"coverage" score calculation	3.581%	1.461%
Interpolation	∼	0.009%
Low-pass filtering	0.162%	0.158%
LOESS smoothing	1.001%	1.006%
Final crop window calculation	0.009%	0.011%
Frames transforming	1.782%	1.918%
Total	229%	55%

Table 7: Average timings for each stage of our smart-cropping method.

We observe in Table 7 that the second most time demanding stage is that of HDBSCAN. One could argue that skipping this stage would result to a further speed-up. However, looking at the examples of Figs. 10, 11 and 12 the importance of this stage becomes clear, since there are multiple salient blobs and the designed filtering-through-clustering procedure manages to select the blob of the main focus. In all of these examples, the first image depicts the original frame. The second depicts the inferred salient map, the third image depicts the thresholded salient map, while the fourth image depicts the filtered salient map as a result of applying the clustering procedure. Finally, the fifth (right-most) image depicts the filtered salient map overlayed on top of the original frame. For an indicative example see Fig 11, where the UNISAL model identified as salient areas the face of the person and a region of hands holding a camera (in the lower left side of the original frame), resulting in two salient blobs (in the post-processed center image of the figure). However, our cluster-and-filter procedure successfully discarded the region containing the camera. Note that if the center of mass was calculated on the post-processed image (instead of the clustering-filtered image, as our method dictates) the center of attention would be calculated to be somewhere in the middle of these two blobs and thus

none of the objects of interest would be included in the final smart-cropped video.

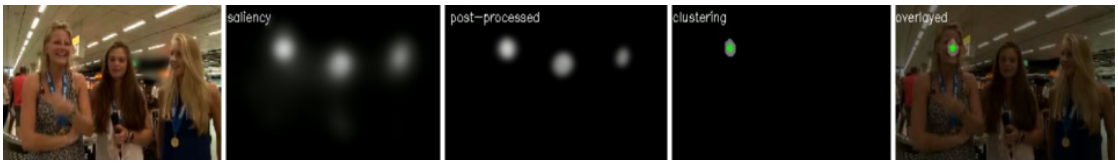

Figure 10: Example frame from processing the 700.AVI video of the DHF1k dataset. Notice how the salient map contains a blob for each of the interviewed persons. After employing the HDBSCAN clustering the filtered salient map contains only the face with the highest salience, the one of the talking person in our case.

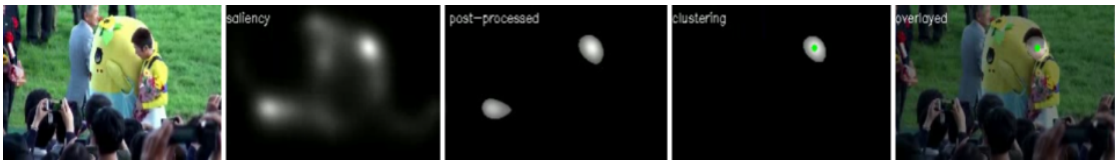

Figure 11: Example frame from processing the 722.AVI video of the DHF1k dataset. Note how the clustering procedure filters the salient region in the lower left area of the frame.

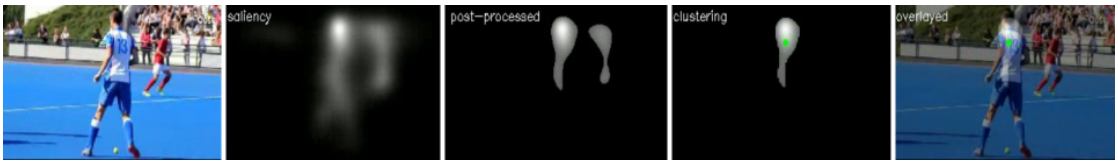

Figure 12: Example frame from processing the 915.AVI video of the DHF1k dataset. Note how the clustering procedure filters the second person in the background (in the red uniform).

2.3. Text to Video Matching

2.3.1. Updated Problem Statement and State of the Art

Given a set of unlabeled video shots and an unseen textual query, the goal of text to video matching, also known as ad-hoc video search (AVS), is to retrieve the most related video shots, ranked from the most relevant to the least relevant shot for the query. In the context of ReTV, text to video matching aids Content Recommendation as discussed in Section 3.3.

Early solutions to the AVS problem were based on large pools of visual concept detectors, and NLP techniques for query decomposition in order to identify concepts in the textual queries. In [67], a set of NLP rules and a variety of pre-trained deep neural networks for video annotation were used in order to associate visual concepts with the provided textual queries. In [51], a large amount of concept, scene and object detectors were used along with an inverted index structure for query-video association.

Recent SoA approaches rely on deep neural networks for directly comparing textual queries and the visual content in a common space [31]. Also, inspired by problems similar to AVS, e.g., cross-modal retrieval or visual question-answering, solutions that have been proposed for these problems were modified and adapted to AVS. In [22], an improved multi-modal embeddings system was proposed, together with a loss function that utilizes the hard negative samples

of the dataset; this approach was adapted to the AVS problem in [8]. In [55], an improved version of the image-to-text matching method of [16] was proposed for the AVS task. More specifically, [55] used the method of [16] together with the triplet loss function of [22] and an improved sentence encoding strategy. In [72], a weakly-supervised method was proposed to learn a joint visual-text embedding space using an attention mechanism to highlight temporal locations in a video that are relevant to a textual description. This mechanism was also used for extracting text-depended visual features. Recently, the dual encoding network proposed in [18] encodes videos and queries into a dense representation using multi-level encodings for both text and videos and the improved loss function of [22]. In [25], the problem of video retrieval was addressed by training three different networks using different training datasets, and combining them by using an additional neural network.

2.3.2. Text to Video Matching Approach

To address the text to video matching problem, we developed an improved dual encoding method designed for Ad-hoc Video Search. Inspired by the dual encoding network presented in [18], we create a network that encodes video-caption pairs into a common feature subspace. In contrast to [18], our network utilizes attention mechanisms for more efficient textual and visual representation, and exploits the benefits of richer textual and visual embeddings.

Let $\mathbf{V}$ be a media item (e.g., an entire video or a video shot) and $\mathbf{S}$ the corresponding caption of $\mathbf{V}$. Our network translates both $\mathbf{V}$ and $\mathbf{S}$ into a new common feature space $\Phi(\cdot)$, resulting in two new representations $\Phi(\mathbf{V})$ and $\Phi(\mathbf{S})$ that are directly comparable. For this, two similar modules, consisting of multiple levels of encoding, are utilized, for the visual and textual content respectively. Moreover, two new attention components are integrated into the baseline network. The overall network architecture is illustrated in Fig. 13.

For every video three different encodings are created, $\phi(V)^1$, $\phi(V)^2$, $\phi(V)^3$. We consider a video or a video shot as a sequence of n keyframes $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$, where each keyframe vector v_i is the output of a pre-selected hidden layer of a pretrained deep network, e.g., the pool5 layer of Resnet [33] or Resnext [100]. The first encoding is the global representation of every video and is obtained by mean pooling the individual keyframe representations, as follows: $\phi(V)^1 = \frac{1}{n}\sum_{i=1}^{n} v_i$.

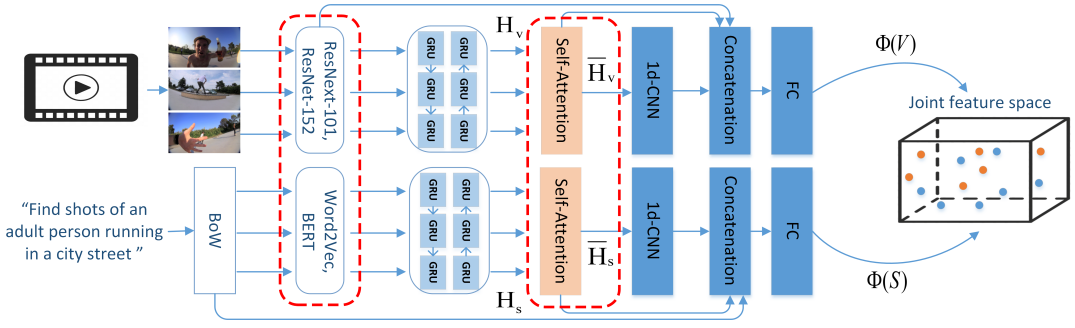

Figure 13: The overall dual encoding network incorporating the self-attention mechanism in both visual and textual modules. A pair of video+text is fed into the network in order to be represented into a joint feature space. The dotted red rectangles indicate the contributions of this work beyond [18]: the self-attention mechanisms, and the multiple video/text encodings.

Next, the keyframe representation vectors $\{v_1, v_2, \ldots, v_n\}$ are fed in a sequence of bi-directional Gated Recurrent Units [11] (bi-GRUs). The hidden state in time t of a forward $\overrightarrow{GRU}$ is

defined as $\overrightarrow{h_t} = \overrightarrow{GRU}(v_t, \overrightarrow{h_{t-1}})$, and in the backward $\overleftarrow{GRU}$ as $\overleftarrow{h_t} = \overleftarrow{GRU}(v_t, \overleftarrow{h_{t+1}})$. All GRU's hidden states are represented as a feature matrix $\mathbf{H_v} = [h_1, h_2, \ldots, h_n]$, where $h_t = concat(\overrightarrow{h_t}, \overleftarrow{h_t})$. To obtain the second level of encoding, mean pooling of the h_t values is performed as follows: $\phi(V)^2 = \frac{1}{n}\sum_{t=1}^{n} h_t$.

Subsequently, a 1-d CNN is built and fed with the feature matrix $\mathbf{H_v}$. A convolutional layer $Conv1d_{k,r}$ is used, with r filters of size k. After applying ReLU activation and max pooling to the layer's output, the $c_k = maxpool(ReLU(Conv1d_{k,r}(\mathbf{H_v})))$ vector is produced. Multiple representations of the video are created, using different $k = 2,3,4,5$ values. The third-level video representation is the concatenation of the produced c_k vectors: $\phi(V)^3 = [c_2, c_3, c_4, c_5]$.

Finally, the concatenation of the previously generated features is used as the global and multi-level feature representation of a video:

$$\phi(V) = BN(W_v concatt(\phi(V)^1, \phi(V)^2, \phi(V)^3) + b_v)$$

where W_v and b_v are trainable parameters and BN a batch normalization layer.

Similar to the visual content encoding network, a multilevel encoding $\phi(S)^1$, $\phi(S)^2$, $\phi(S)^3$ is generated for the textual content. Given a sentence S containing m words, the $\phi(S)^1$ representation is created by averaging individual one-hot-vectors $\{w_1, w_2, \ldots, w_m\}$. Next, as the second level of textual encoding, a deep network-based representation for every word is used as input for the bi-directional GRU module, and similarly to $\phi(V)^2$, $\phi(S)^2 = \frac{1}{m}\sum_{t=1}^{m} h_t$.

Next, the feature matrix $\mathbf{H_s}$ of the textual bi-GRUs is forwarded into a 1-d convolutional layer with filter sizes $k = 2,3,4$ and $\phi(S)^3$ is calculated similarly to $\phi(V)^3$ above. The final textual representation is:

$$\phi(S) = BN(W_s concatt(\phi(S)^1, \phi(S)^2, \phi(S)^3) + b_s)$$

Following [22], [71] and [18], the improved marginal ranking loss is used to train the entire network.

Self-Attention Mechanisms

The 1-d CNN layer that is fed with $\mathbf{H_s}$ or $\mathbf{H_v}$ in the original network of [18] treats each item of the words or frames sequence equally. Our target is to exploit the most meaningful information from the textual and visual sequences, particularly the words with the highest semantic importance and the keyframes which are more representative for a video shot. For this, we introduce a self-attention mechanism [7][60] in each modality, in order to find the relevant importance of each word in the input sentence, and to find important temporal locations in a video-shot. An overview of this self-attention mechanism is illustrated in Fig. 14.

In the textual encoding part of the network, given the output of the bi-GRU $\mathbf{H_s}$, the attention model outputs a vector a:

$$a = softmax(w_{s2} tanh(W_{s1} \mathbf{H_s}^T))$$

where W_{s1} is a trainable weight matrix of size $d \times 2u$, where d is a hyper-parameter, $2u$ is the size of a single bi-GRU unit and w_{s2} a parameter vector of size d. The w_{s2} vector is extended

in a $z \times d$ matrix W_{s2} for multi-head attention, by modeling z semantic aspects of the text, as in [60], resulting in a weight matrix $\mathbf{A_s}$:

$$\mathbf{A_s} = softmax(W_{s2} tanh(W_{s1} \mathbf{H_s}^T))$$

The $softmax()$ is used for weight normalization, so that all the weights sum up to 1. Then, the attention matrix $\mathbf{A_s}$ is multiplied with the initial $\mathbf{H_s}$, resulting in matrix:

$$\overline{\mathbf{H}}_\mathbf{s} = \mathbf{A_s} \mathbf{H_s}$$

$\overline{\mathbf{H}}_\mathbf{s}$ is forwarded into the 1-d convolutional layer instead of the feature matrix $\mathbf{H_s}$. This text-based self-attention mechanism is denoted as *Att* in the sequel.

A similar self-attention mechanism, denoted as *Atv*, is integrated in the visual encoding module. In this case $\mathbf{H_v}$ is used for calculating the attention weighted matrix $\mathbf{A_v}$, resulting in

$$\overline{\mathbf{H}}_\mathbf{v} = \mathbf{A_v} \mathbf{H_v}$$

Dealing with such a demanding task, where typically the SoA methods achieve accuracy of about $10-22\%$ on different evaluation datasets, it is vital to exploit the advantages of different signal encodings. Regarding the video module, two SoA deep neural network architectures are used for frame feature extraction: the ResNext-101 [100] and ResNet-152 [33] models. Concerning the text module, the performance of the Word2Vec [70] model, as well as the bidirectional transformer-based language model BERT [15], are examined. We also examine early fusion (as shown in Fig. 13) i.e., concatenation of encoding vectors, versus late fusion (i.e., merging of ranked lists, each obtained using a different text-visual encoding pair), for jointly exploiting the multiple encodings.

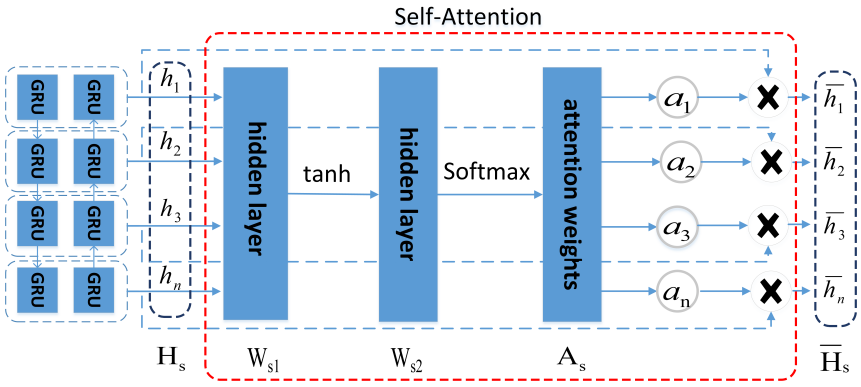

Figure 14: An illustration of the employed self-attention mechanism.

Complying with the agreed ReTV architecture, all text to video embeddings (both from videos and text) are extracted in the Video Analysis service of WP1 using three alternative end-points (see Section 6.2 of D1.3). The permanent storage and comparison of video and text embeddings comparison is conducted by GENISTAT, as discussed in Section 2.4.2.

2.3.3. Implementation Details and Use

All implementations regarding text to video matching were completed using the PyTorth[6] open-source neural-network library version 1.3. Also the deep learning framework MXNet v1.4.1 was

[6] https://pytorch.org/

Table 8: Results (MXinfAP) of the proposed networks and their combinations, compared with the baseline [18]. The best results for each dataset are indicated with bold, while those that are worse than the baseline are given in parenthesis. All reported training/inference times are in hours, for a single setup (should be multiplied by 6 for the *Combination of 6 setups*) and for processing the whole training/test dataset. These numbers are not to be confused with query execution time; this is approximately 30 sec. for all but the late fusion methods, and 4 times higher for the latter.

		I. Combination of 6 setups			II. Best of 6 setups			Avg. training time	Inference time
		AVS16	AVS17	AVS18	AVS16	AVS17	AVS18		
(a)	W2V + ResNext-101 [18]	0.142	0.2189	0.1187	$0.1457^{\dagger,1}$	$0.212^{*,3}$	$0.1165^{*,2}$	6.66	1.72
(b)	(a) + *Att*	0.1544	0.2264	0.1233	$0.1477^{\dagger,2}$	$0.2298^{\dagger,1}$	$0.1183^{\dagger,2}$	7.53	1.81
(c)	(a) + *Atv*	0.1497	0.2274	0.1231	$0.1464^{\dagger,2}$	$0.2165^{\dagger,1}$	$0.1237^{\dagger,1}$	7.52	1.80
(d)	BERT + ResNext-101	0.1532	0.2248	0.1194	$0.1576^{*,2}$	$0.2288^{\dagger,1}$	$(0.1126)^{\dagger,2}$	6.7	1.70
(e)	W2V + ResNet152	0.1464	(0.2033)	(0.0986)	$0.1507^{*,1}$	$(0.2062)^{*,1}$	$(0.1043)^{\dagger,2}$	6.64	1.71
(f)	BERT + ResNet152	0.1501	(0.2141)	(0.103)	$0.1464^{\dagger,2}$	$(0.208)^{*,2}$	$(0.099)^{\dagger,2}$	6.68	1.71
(g)	Early fusion of W2V + BERT + ResNext-101 + ResNet-152	0.1614	0.2312	0.122	$0.1544^{\dagger,2}$	$0.2327^{\dagger,1}$	$0.12^{*,1}$	9.1	1.73
(h)	(g) + *Att*	0.1635	**0.2427**	**0.1266**	$0.1594^{\dagger,2}$	$0.2444^{\dagger,2}$	$0.1261^{*,2}$	9.21	1.84
(i)	(g) + *Atv*	0.1637	0.2352	0.1265	$0.1583^{\dagger,2}$	$0.2307^{\dagger,2}$	$0.1265^{*,3}$	9.2	1.84
(j)	Late fusion of (a), (d), (e), (f)	0.1658	0.2414	0.1206	0.1683	0.2499	0.1272	26.7	6.84
(k)	Late fusion of (b), (d)+*Att*, (e)+*Att*,(f)+*Att*	**0.1663**	0.2413	0.1240	0.1658	0.2469	0.1283	27.6	7.01
(l)	Late fusion of (c), (d)+*Atv*, (e)+*Atv*, (f)+*Atv*	0.1655	0.2415	0.1245	**0.1693**	**0.2576**	**0.1288**	27.4	7.00

† Adam optimizer * RMSprop optimizer ^{1} learning rate: 1×10^{-4} ^{2} learning rate: 5×10^{-5} ^{3} learning rate: 1×10^{-5}

used for video feature extraction. The experiments were conducted on an Intel i7 3770K PC with 32GB of RAM, running Ubuntu 18, equipped with an Nvidia GeForce GPU (GTX 2080 Ti) as well as additional PCs with similar specifications.

2.3.4. Results

We trained our network using the combination of two large-scale video datasets: MSR-VTT [102] and TGIF [58]. We evaluate its performance on the official evaluation dataset of the TRECVID AVS task for the years 2016, 2017, and 2018, i.e., the IACC.3 test collection consisting of 4,593 videos and altogether 335,944 shots. As evaluation measure we use mean extended inferred average precision (MXinfAP), which is an approximation of the mean average precision suitable for the partial ground-truth that accompanies the TRECVID dataset. As initial frame representations, generated by a ResNext-101 (trained on the ImageNet-13k dataset) and a ResNet-152 (trained on the ImageNet-11k dataset), we use the publicly-available features released by [55]. Also, two different word embeddings are utilized: i) the Word2Vec model [70] trained on the English tags of 30K Flickr images, provided by [17]; and, ii) the pre-trained language representation BERT [15], trained on Wikipedia content.

For comparison reasons, we used the publicly available code of [18] to re-train the network with the same configuration and features we use in our methods. This method is indicated as *W2V+ResNext-101* in Table 8 and is used as a baseline for our experiments. Overall, three general network architectures are trained, i) the baseline network, ii) the network with the text-based attention mechanism, and iii) the network with the visual-based attention. Each network is trained using one or both available word embeddings (i.e., Word2Vec [a.k.a. W2V

for short] and BERT) and one or both visual representations (i.e., ResNext-101 and ResNet-152). The *Combination of 6 setups* column presents the results after late fusion of 6 different experimental setups for the same network, using two optimizers, i.e., Adam and RMSprop, and 3 learning rates ($1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}$), similarly to [56][82], while the *Best of 6 setups* column presents the results of the best-performing among these setups.

The results reported in Table 8(a)-(f) show that both attention mechanisms improve the performance of the baseline method. Furthermore, using better word embeddings (BERT) consistently improves the performance in comparison to W2V.

In the (g), (h) and (i) configurations of Table 8, the results of the early fusion of the text and visual embeddings are presented. The results indicate that the combination of different visual (ResNext-101, ResNet-152) and textual (W2V, BERT) features leads to improved performance. Moreover, the integration of the aforementioned attention mechanisms further improves performance.

Subsequently, in configurations (j), (k) and (l) of Table 8 the performance of late-fusion combinations of the previously-examined networks is presented. In configuration (j), the late fusion of the baseline network trained with different textual and visual features is presented. When considering the combination of 6 setups, this approach usually outperforms by a small margin the corresponding early fusion model (g). The late fusion of models with text- or visual-based attention performs similarly to, or a bit better when combining the best of 6 setups (columns II) compared to the corresponding early fusion approaches, however at the expense of considerably higher training and inference times.

Method	AVS16	AVS17	AVS18
VSE++ [22]	0.123	0.154	0.074
Video2vec [31]	0.087	0.150	-
W2VV++ [55]	0.151	0.220	0.121
Dual encoding [18]	0.159	0.208	-
(h) from Table 8	**0.1594**	**0.2444**	0.1261
(i) from Table 8	0.1583	0.2307	**0.1265**

Table 9: Comparison with published SoA results (MXinfAP).

In Table 9 the recommended single-setup early fusion configurations of the proposed method (shaded rows (h) and (i) of Table 8) are compared with the literature SoA works (included the top-performer of the TRECVID 2018 competition [55]), based on the results reported in the corresponding papers for the same evaluation datasets. We can see that the proposed method's configurations (h), (i), outperform the SoA published results on all three datasets.

Compared to early results reported in D3.2 our method achieved 230% relative improvement in terms of MXinfAP in the AVS16 dataset, increasing MXinfAP from 0.0482 to 0.1594.

2.4. Updated Component, Workflow and API

2.4.1. Video Summarization Component Updated Functionalities and Outputs

The CERTH Video Summarization (VS) component was deployed as a REST service before M21 and is described in detail in Section 4.7 of D3.2. We updated the component to utilize the new models for video summarization, discussed in Section 2.1.2. The workflow and the format of results remain exactly the same as presented in Section 4.7 of D3.2. The video features required for use by the VS service are extracted when a video is ingested by the Video

Analysis (VA) service of WP1. A prerequisite to generate a video summary using the learning-based summarization method described in this deliverable is to set the "lbvs" field to 1 in the JSON-structured body of the HTTP call to VA, as described in Section 6.2 of D1.3. Otherwise the resulting summary is generated using the older non-learning based video summarization method described in Section 4.3 of D3.2, since the necessary frame importance scores of the new method will not be available.

In order to optimize the performance of the process of the VS component and specifically the stage of the video summary's rendering we implemented a change that proved to provide a speed-up. In the "VA service modifications for ingesting large video files" paragraph in Section 4.1.1 of D1.3, we described a memory scheme in which large arrays are written on disk but cached in RAM in order to retain the fast processing of VA while overcoming the physical limit on the duration of videos that can be ingested. This ensures that in all cases we have readily available the frames of the original video, either in RAM or memory mapped in disk. However, these frames are resized and processed based on the requirements of the Video Analysis component's requirements and thus cannot be used for the rendering of the final video summary. Therefore, the need to re-read the original video file to select the appropriate segments (as dictated by our video summarization algorithm) becomes clear. We employed the multi-threaded read-and-decode workflow discussed in "Performance Optimization" paragraph in Section 4.1.1 of D1.3 to read faster the original video. Our experiments showed an average 10% speed-up in the video summary rendering process because of this modification.

Regarding text to video matching, the extraction of text to video embeddings is again conducted by the VA service of WP1, according to the final architecture that optimizes the use of the ReTV services. The available endpoints for the extraction of embeddings from either video shots or text, are described in Section 6 of D1.3. These embeddings are saved in the respective VA session and are available for download for 48 hours - the time that a session results folder remains in CERTH's repository before it is deleted. The permanent storage and comparison of embeddings is done by GENISTAT in the project binary repository. See Section 3.3.1 for how we built a search API on top of those embeddings.

In Section 6.1 of D1.3 we discuss the implemented endpoints for text to video embeddings extraction from video files (retv.iti.gr:8090/t2v) and text strings (retv.iti.gr:8090/t2vs). The extraction of text to video embeddings from videos can take place in an offline fashion - a large volume of videos can be pre-processed to extract and have readily available the embeddings. However, the extraction of the corresponding embedding from the text depends upon the user's search actions in the user interface of the ReTV tools, and therefore this procedure should happen as close to real-time as possible. We remind here that the calls to the main video analysis, video summarization and text to video embeddings extraction endpoints are all inserted in the same queue. A delay of a t2vs call being inserted in the queue after a va call concerning a large video file would be unacceptable. To alleviate this, CERTH made an identical copy of the t2vs to a different server. The API, and the formatting of the results remains the same, the only change is the different IP of the underlying server in the call. These t2vs calls are inserted to separate queue and are served immediately.

2.4.2. Component Updated API and Usage Instructions

The API of the Video Summarization component remains very similar to its previous version presented in Section 4.7 of D3.2, with a few necessary modifications. An update made to the VS API is the replacement of the "utilize_story_percent" field, which can be included in the JSON-structured body of the HTTP POST start call to the VS service (described in Section 4.7

of D3.2) with the "utilize_range" field, which can be utilized in the same way. This parameter now accepts a list of two floats that are in the range $[0,1]$. These represent percentages of the start- and end- parts of the portion of the video to not take into consideration when generating the summary, thus allowing the user of VS to avoid "spoilers" - a segment of the video which reveals important plot elements. For example, since most of the time a "spoiler" might reside in the ending or near-ending shots of a video, setting "utilize_story_percent"$=[0.0, 0.75]$ will make sure that no shot from the last quarter of the original video will be used in the summary. Similarly, a user that wants to exclude the first half of the video from the generated summary, can set "utilize_story_percent"$=[0.5, 1.0]$. The default value of "utilize_range" parameter is $[0.0, 1.0]$, i.e., the whole video content is utilized to generate the video summary.

An additional update made to the VS API is the introduction of the "output_ratio" field, which can be included in the JSON-structured body of the HTTP POST start call to the VS service. This parameter can be used to set a target aspect ratio for the generated video summary, which can be different than that of the original video. It accepts a string which consists of two numbers separated by the ":" character or "-". Valid values of this parameter include "16:9", "5:4", "4:5", "1:1", etc. The default value of "output_ratio" parameter is "-", i.e., the video summary dimensions will remain equal to these of the original video.

2.4.3. Component Testing and Software Quality Assessment

The entire VS REST service and the updated video summarization method that is implemented in the service have undergone extensive testing, both by their developers and by other ReTV partners that use the service for generating video summaries. The generated summaries have been assessed by users in ReTV, both within the consortium and as part of the user testing performed in relation to the use cases of the project, and the results are very encouraging. A preliminary evaluation of video summaries with the stakeholders was presented in D5.2 and more extensive results will be presented in D5.3. Concerning unit testing, similarly to the VA service of WP1, 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 earlier in this section, and also cover edge cases (such as very short or very long videos submitted for summarization). The proper exchange of information with the WP1 VA service, for the types of calls of the VS service that require the previous correct execution of the VA one, was also tested. In terms of stress testing, the VS REST service implements a queuing mechanism similar to that of the VA service; this mechanism was also tested with a few hundred summarization requests that were submitted and were successfully queued and eventually processed. The final VS service, using the learning-based summarization algorithm, is mature for use in ReTV.

3. Content Recommendation and Scheduling

Content recommendation and scheduling lies at the core of our use cases. In D3.2, we demonstrated how we included the recommendation and scheduling engine in the Content sWitch use case. Since then, we have decided to focus our efforts on the 4u2 Messenger, as we believe it has a more substantial potential for addressing the needs of our stakeholders and it needs to clear a lower barrier to market entry: unlike the Content sWitch, the 4u2 Messenger does not need to be tightly integrated with an existing video streaming platform to offer value. The final version of the Content sWitch was presented in D6.2. However, as we described in [74], both the Content sWitch and the Messenger used the same core recommendation and scheduling engine. The extended approach to recommendation and scheduling presented in this deliverable is to be seen as the evolution of this common component.

Section 3.1 explains what data can be used to recommend content to users in the 4u2 Messenger. Section 3.2 presents our approach to recommendation and scheduling and shows how it is integrated in the the 4u2 Messenger. Section 3.3 explains how we improved the recommendation and scheduling capabilities of the Content Wizard by integrating different ReTV services. Finally, Section 3.4 shows our approach to recommendation and summarisation in the Storypact Editor, which will help to finetune the wording of Content Wizard postings.

3.1. Measuring User Engagement in the 4u2 Messenger

In the 4u2 use case we develop a novel channel for distribution of personalized video content. We will explain the setup of the 4u2 Messenger in detail in D6.3 "Second validation of Personalization Prototype" due in M39. The current 4u2 Messenger is our second implementation, after having built first version for the Telegram messaging application. Practical use showed that the chosen chatbot-framework did not meet our requirements and that Facebook was a better platform than Telegram for testing the application with real users of the use case partners. Specifically, NISV was interested in the 4u2 Messenger integration into its existing Facebook pages as a means to deliver personalised video content to its followers. However, Facebook's glacial approval process for public messengers made us incur a delay in the development, deployment and testing of the application. We further had to build features that Facebook wouldn't unlock for us: For example, our 4u2 Messenger comes with support to add test-users that are separate from normal users.

The 4u2 Messenger has clearly defined interactions with interested users. To start the interaction with the Messenger application, users subscribe through the Facebook Messenger. Our approach and system is generic, and will support other platforms if needed. The target group are visitors of the NISV museum who would like to engage with archival content after their visit to the museum. Human editor can interact with the 4u2 Messenger through an admin interface that displays past and future messages sent to subscribed users. This allows them to monitor the messages that the recommendation engine sends. Editors can browse the connected video archive and craft new messages by hand if they so desire (see Figure 15). Those hand-crafted messages can also be scheduled to be sent at a specific time (see Figure 16). After delivering a message to a user, the messaging platform sends back events that describe how the user reacts to the message. Those events are then used as signals by the recommendation engine to optimize for user retention.

Measuring user engagement when we control the video streaming platform delivering the content is relatively simple: the server logs show which user requested which video. Looking at which video chunks the streaming server delivered, we can even infer which parts of the video

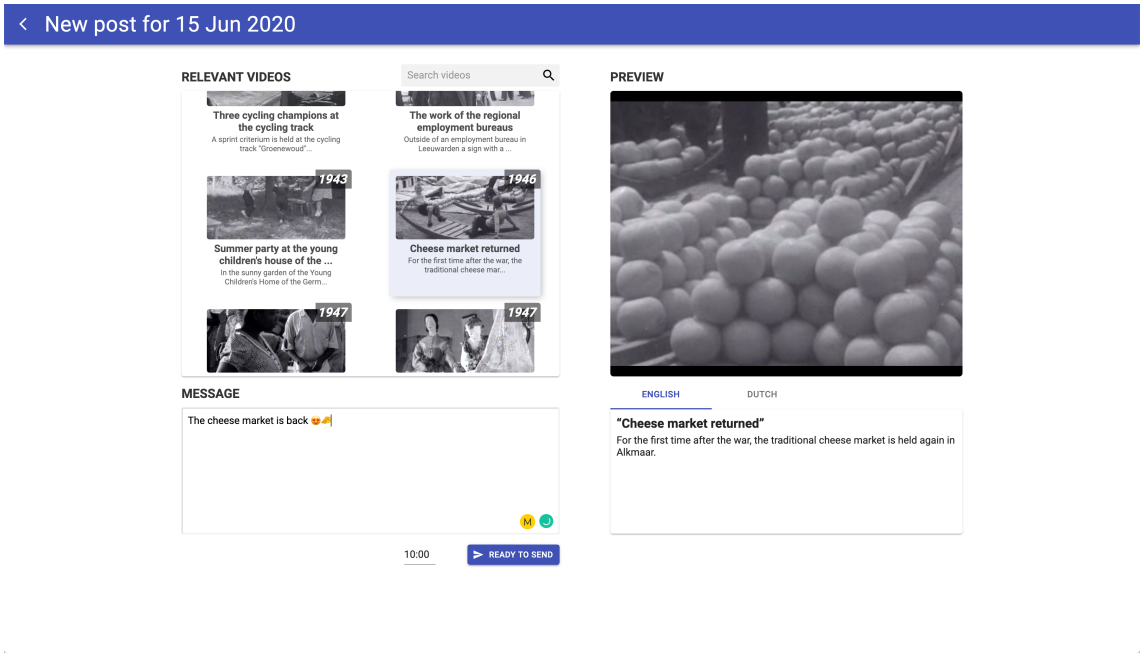

Figure 15: Interface editors use to create new messages based on archive videos.

the user saw. As with all audience measurements, we need to assume that the user watched the screen when the video played. This assumption can be wrong for multiple reasons, though: a viewer might get up to answer the door and leave the video playing. Alternatively, they might have the browser window displaying the video covered by another application. In practice, however, this assumption is good enough for most purposes.

Measuring user engagement in a messenger has proven to be more difficult because we do not control the streaming platform that delivers the video content. When sending a video to a user through a chat platform like Facebook Messenger or Telegram, we lose control over the video as the platform takes over the hosting and streaming of the content. The platforms do this by design, and we have not found a way to circumvent this. Sending a link pointing to a video on our streaming servers is possible, but it will not display as an in-app video which makes for poor user experience. See figure 17 for a video playing directly in the chat conversation. Sending a video message to a user on Facebook is a two-step process:

1. Upload the video to Facebook through an API. The API returns a `FacebookID` which is a unique identifier for the uploaded content.
2. Send the message to the user and specify the video using the `FacebookID`. The Facebook Messenger platform will then display the video inside the app.

We assume that Facebook and other platforms chose to host the videos themselves to control and validate the content before showing it to users. It also allows them to measure more engagement data themselves: if users watched videos hosted on external streaming platforms, they would not be able to measure their engagement. Since we depend on chat platforms like Telegram and Facebook Messenger to reach users, we had to find a way to measure user engagement without using the server logs from the streaming servers.

We want the 4u2 Messenger to work across all messaging platforms. We therefore decided to only use measurement signals that we believe all platforms will provide:

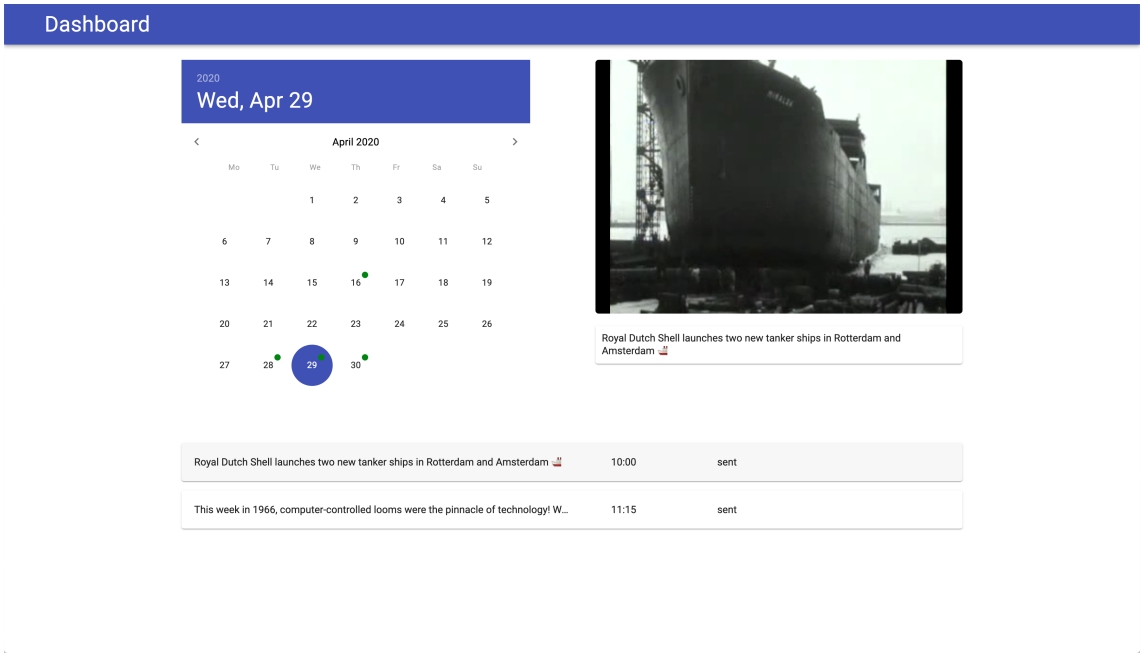

Figure 16: Overview of sent messages.

- If a user has seen a message. It only signifies that they opened the chat with our Messenger but it does not mean that they watched a corresponding video or even read the message. This we can't measure.
- A user reaction to a message. Some chatbot platforms limit users to choose from a set of predefined emojis to react to a message (see figure 18 for an example).
- A user sends a message. We get notified by any message a user sends to our Messenger because otherwise, we could not respond.

3.2. Optimized Retention in the 4u2 Messenger

In the D3.2 deliverable, we presented a recommendation approach based on explicit content attributes, in particular:

- TV broadcast genres
- Sports events attributes (e.g., tennis match players, football team nationality, tournament event stage)

We built a user profile model based on these attributes (such as the most engaging genres, the match between user nationality and event location). Based on such data, we proposed a forecasting-based model to predict the optimal scheduling time and a recommendation model based on Field-aware Factorization Machines model (FFM [39]). We have refined the approach to sending content in the 4u2 Messenger and we see five different phases that transition smoothly into each other:

1. Manual crafting of messages: Editors can manually search for video content, craft messages and schedule them. This is meant to be a fallback from automatically sent messages. However, the ability to schedule messages for the week ahead already creates large time-savings for editorial teams.

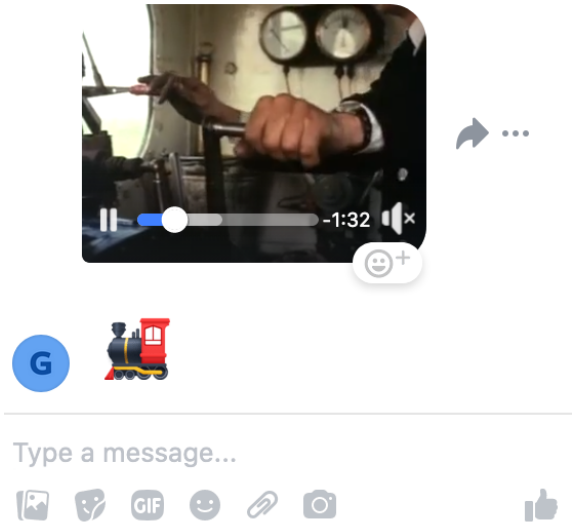

Figure 17: Videos play directly in the chat application. It makes for a seamless user-experience, but reduces the amount of data we can use for the recommendation engine.

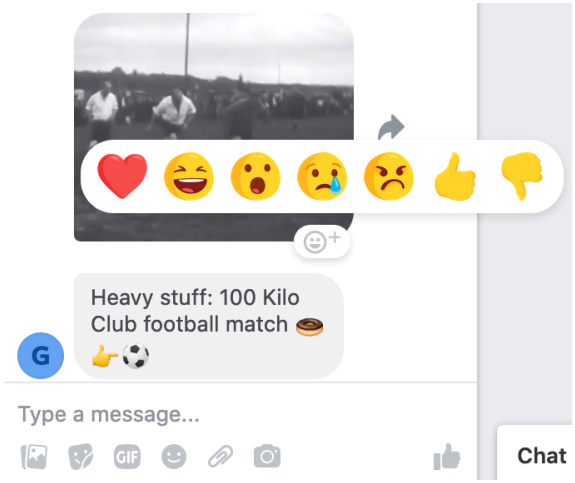

Figure 18: Facebook messenger lets users choose predefined reactions to a message.

2. Content-based recommendations: To solve the cold-start problem inherent in recommendation tasks, we start to recommend videos to users with a content-based algorithm.
3. Aggregating data to optimize the publication time: As we gather data on user-behaviour, patterns will emerge when messages should be sent to optimize retention. See 3.2.3.
4. Aggregating data to understand the meaning of the emoji reactions: As noted in 3.1, different emoji reactions might mean different things depending on the context. As we gather user-data, we will be able to learn more about what those reactions mean (see 3.2.2).
5. Collaborative filtering: With enough user-data our system can find similar users based on their behaviour. Together with the content-based recommendations this forms a hybrid, best-of-both-worlds approach.

We step from one phase to the next as we gather more user-data. The first phase, manual

crafting, is described at the start of Section 3.1. See 3.2.1 for the details of the second phase and the content-based algorithm and how it builds up on the Text to Video matching described in section 2.3. The third and fourth phases are described in sections 3.2.2 and 3.2.3 and can be seen as iterative improvements that discover heuristics to guide the recommendation. Finally, the last and most data-intensive phase, collaborative filtering, will help us to improve all previous phases and allow us to learn the ideal publication time and what the emojis in response to videos mean in different contexts. We present our approach in section 3.2.4.

Contrary to the approach presented in D3.2, we want to avoid explicit aggregation of such content vectors to create user profile attributes. Aggregation to a user profile happens implicitly via a recommendation model that considers (a) interactions between users and content vectors and their reactions to the presented content and (b) the similarities between content vectors themselves.

Measuring user engagement needs to serve a definite purpose. We chose to optimize the recommendation engine for retention, i.e., to help build long-lasting relationships between the content owners and their audience. Retention tells us how many active users a day, a week, or a month ago are still active today. For each user, we store all of the events described in the previous section. We interpret any reaction as an interaction and, therefore, as having retained the user. We focus on retention and not actual user reactions, as they are highly correlated. Receiving a series of thumbs-down emojis as a response to a video does not have the same meaning as receiving a series of thumbs-up emojis. However, we believe that users who send a series of thumbs-down emojis will stop opening our Messenger conversations altogether.

3.2.1. Overview of Content-based Recommendations

Below we present our extended approach to content-based recommendation, which uses the Text to Video embeddings described in Section 2.3. Instead of explicit attributes, we use a set of vectors in 2048 dimensional space to describe each video shot. Contrary to the collaborative filtering approach presented later in section 3.2.4, we do not assume that we have enough usage data from user-content interactions to estimate the collaborative filtering model (users who like video X typically also like video Y). Instead, we focus on the aspect of video content similarity.

We used the Open Images[7] collection from NISV, which contains both videos and short descriptions. We set up a video embeddings database by running 1837 videos through the Text to Video endpoint of CERTH (cf. Section 2.3). We stored a 2048 dimensional signature vector on disk for each of the 51712 video shots (individual scenes).

Content-based User Profile

The user profile consists of the list of videos a user saw and timestamps of the viewing event. Each video is represented by a list of embeddings, one embedding per video shot. We process only M first scenes of a video. Currently, M is 10, which corresponds to more or less 20-30 seconds of the video content. There are several reasons for that decision:

- If none of the first few scenes of the video is relevant for the user, she will most likely skip it (especially in the context of viewing on a mobile device)
- Longer videos are not preferred over shorter ones during recommendations

[7] See `https://openbeelden.nl/about.en`

- Recommendations are faster to calculate

We explain the usage of timestamps in the following section.

Finding Relevant Video Candidates

Finding videos relevant for a given user (and their associated profile U, i.e., the list of embeddings and associated timestamps) is based on the similarity between user profile U and a list of embeddings a candidate video shots. We construct a k-NN (K Nearest Neighbors) search space of embeddings based on cosine similarity distance. To ensure k-NN searches' scalability, we use the Faiss library[8].

In the first step, for each of the video shots (first M ones) in each of the videos that users have seen ($S \in U$), we run a k-NN search and collect K candidate video shots (currently K is set to 100). We group search results in U_pos and U_neg lists, representing positive and negative user reactions. Next, we calculate the relevance score for each candidate video shot D according to the equation 9. The intuitive interpretation is that the more similar candidate D is to the video shots that users liked, and the more dissimilar from shots that the user disliked, the higher the relevance is.

$$relevance(V_S) = \sum_{S \in U_{pos}} \sum_{i=1..M} w(S_i) \cdot cosine(S_i, V_S) - \sum_{S \in U_{neg}} \sum_{i=1..M} w(S_i) \cdot cosine(S_i, V_S) \tag{9}$$

Finally, we aggregate relevance scores from video shots to videos by summing individual scores, and obtain the video ranking by sorting videos in a descending order by the score:

$$relevance(V) = \sum_{V_S \in V} relevance(V_S) \tag{10}$$

Diversified Recommendations

Recommendations based only on relevance scores would keep the user in the very narrow bubble of video topics. To achieve diversified results and avoid a significant increase in computation time, we first created a fuzzy K-Means clustering model for each of the 51712 video shots (individual scenes). Some of the NISV collection videos discuss multiple topics so that individual scenes may belong to different clusters. For each video shot and each cluster, we obtained a distance value. The lower this distance is, the more typical a given shot is for a given cluster (i.e., topic).

We estimated a model with a relatively high number of small clusters to get clusters that are thematically more homogeneous. Precisely speaking, we set the number of clusters K to 1024. Additionally, we set constraints on the minimal and maximal number of video shots in each cluster (5 and 100, respectively).

Based on such a clustering model, we defined a conditional diversity score. Conditional diversity compares the similarity of the distribution of cluster memberships of all video shots that user has seen already (R) with cluster memberships of the new candidate video D (consisting of multiple video shots). Function C in equation 11 maps a list of video shots into a multi-set, where each element is a cluster identifier, and quantity of the element is equal to the sum of

[8] See `https://github.com/facebookresearch/faiss`

membership levels from all input video shots. For instance, if $R = [v1, v2]$, video shot $v1$ fuzzy cluster memberships are $\{c1 : 0.2, c2 : 0.4, c3 : 0.7\}$, video $v2$ fuzzy cluster memberships are $\{c2 : 0.3, c3 : 0.1, c4 : 0.3\}$, then $C(R) = \{c1 : 0.2, c2 : 0.7, c3 : 0.8, c4 : 0.3\}$

Now, considering the new video candidate D not seen yet by the user, we compare its cluster memberships to $C(R)$ by $\cap$ operator, i.e., multi-set intersection. The smaller the intersection is, the higher the serendipity effect of the new video is.

The last operation is the aggregation of all resulting memberships just by a simple summation $\sum$. Again, the smaller the sum value is, the higher is the serendipity effect and the higher value of the diversity score in equation 11. The diversity score ranges between 0 (no diversity) and 1 (highest diversity). Also, with more videos that user has already seen (i.e., the bigger the list R is), the average diversity score diminishes.

$$diversity(D) = \left[1 + \sum [C(R) \cap C(D)]\right]^{-1} \tag{11}$$

The final step is to achieve a good balance between relevance and diversity scores: a trade-off between exploration mode (more serendipity at the potential cost of lower relevance) and exploitation mode (the user stays in her information bubble with high relevance but no new information). We rank all the candidate videos by a product of relevance and diversity, with the potential scaling effect of diversity:

$$rank(D) = relevance(D) \cdot F(diversity(D))^P \tag{12}$$

The constant P is set to 1 by default, but it can be used to control the trade-off between diversity and relevance factors.

In the next section, we present the impact of selecting various F functions on the resulting ranking (in particular, $F = exp$ and $F = sqrt$).

Multiple Recommendations

If we present a user with just the highest-ranked video, the approach to diversification presented in the previous section is sufficient. However, if we present more than one video to a user, we also need diversification *within* the list of results. Otherwise, we can end up with a list of videos covering the same topic.

The solution to this issue is to re-calculate the diversity score iteratively, video by video. In other words, we calculate relevance scores for all candidate videos once (as discussed earlier), but diversity scores multiple times. In the first iteration, we calculate diversity against $R0$, the user's list of videos already seen. We take the candidate $D1$ with the highest $rank(D1)$. Then we calculate $R1 = R0 \cup D1$ and diversity for the second recommended video is calculated against $R1$, instead of $R0$. We repeat this until the list of candidates is full.

Exemplary Results

Let us consider a fictional user who positively reacted to videos on sport (*Football match The Netherlands - Belgium*), Amsterdam airport (*Schiphol Airport exists 30 years*) and traffic (*25 years of driving licenses in the Netherlands*). The output of the model is presented in Tables 10, 11, 12 and 13.

Table 10: Ranking based on relevance score only

rank	relevance	diversity	title
1	13.536	0.095	Opening of the Feyenoord stadium
2	11.933	0.150	Schiphol Airport prepares itself for the jet engine era
3	10.879	0.152	Schiphol Airport exists 40 years
4	10.074	0.150	The Boeing Stratocruiser at Schiphol Airport for the first time
5	8.627	0.167	Schiphol Airport expands

Table 11: Ranking based on relevance and diversity scores ($F = exp$ scaling)

rank	relevance	diversity	title
1	13.536	0.095	Opening of the Feyenoord stadium
2	11.933	0.150	Schiphol Airport prepares itself for the jet engine era
3	6.590	0.715	25-year jubilee of the 'Hendrick de Keyser' association
4	8.031	0.461	American pilot thanks helpers in The Hague
5	10.879	0.152	Schiphol Airport exists 40 years

There is a clearly visible trade-off between the relevance-only based ranking (Table 10) and the rest of results, where some relevance is sacrificed at the expense of more topically diversified results. Finally, in the last Table 13, we can see that more topics are covered in the top-5 results (e.g. the traffic topic that user was interested in), and that the results are also more diversified between themselves (note the high diversity scores), and not only with respect to the current user profile. It allows to make a user familiar with a broader set of topics, and by collecting user reactions to these topics, to converge faster to a more accurate user profile.

Cold-start Problem

The 4u2 Messenger, as deployed at NISV, sends messages following the approach of "This week in history": Videos might be 50 years old, but they always reference the current week of the year. This meaningfully reduces the number of videos we can send a given user when starting out. In order to let the algorithm capture quickly what a user enjoys, we base our solution to the cold-start problem on a clustering model, described in Section 3.2.1. The model clusters all the videos into multiple (1024), small clusters that represents various topics.

Having a new user that we know nothing about yet, we collect all of the videos from the current week of the year (spanning over multiple decades, approximately 15 to 20 videos for each week) and the topical clusters to which they belong. The user is presented with 2-3 videos (cf. section 3.2.1) that come from a different clusters. Clusters are selected randomly, where the selection probability is proportional to the relevance of a given topical cluster for the current week of the year. Cluster relevance is measured as the total similarity of all the

Table 12: Ranking based on relevance and diversity scores ($F = sqrt$ scaling)

rank	relevance	diversity	title
1	11.933	0.150	Schiphol Airport prepares itself for the jet engine era
2	8.031	0.461	American pilot thanks helpers in The Hague
3	6.590	0.715	25-year jubilee of the 'Hendrick de Keyser' association
4	13.536	0.095	Opening of the Feyenoord stadium
5	10.879	0.152	Schiphol Airport exists 40 years

Table 13: Ranking based on relevance and *dynamic* diversity scores

rank	relevance	diversity	title
1	8.876	1.000	Plaatwellerij Velsen
2	6.590	0.715	25-year jubilee of the 'Hendrick de Keyser' association
3	8.031	0.423	American pilot thanks helpers in The Hague
4	9.741	0.362	Polygoon Dutch News
5	4.931	0.566	Demonstration by the Rotterdam motorcycle police

allowed videos belonging to a given cluster (i.e., the sum of similarity scores). Such a topical diversification allows for a faster convergence to an accurate profile of a user interests.

Demo Frontend

In order to facilitate evaluation of the recommender algorithm (including its hyper-parameters, such as the Diversity factor that controls the trade-off between relevance [information bubble] and serendipity [topical freshness]), we developed a simple interactive frontend that allows to simulate user behavior in the context of 4u2 Messenger.

The process flow for the user that tests the recommendation algorithm in the frontend is the following:

1. The editor defines an initial user profile - a list of videos that a simulated user found interesting (cf. Figure 19).
2. The editor calls the recommendation algorithm and gets back a ranked list of K (currently $K = 5$) videos, sorted by a total score (product of relevance score and diversity score, cf. equation 12). All three scores (total, relevance, diversity) are visible to the editor. The result is a list of videos as depicted in Figure 20.
3. The editor evaluates videos in the ranking. Each video is evaluated with respect to the three dimensions (the three scores mentioned above).
4. The editor evaluations are added to the user profile, and a new call to the recommendation algorithm can be executed. This simulates dynamics of the user behavior.
5. The editor saves the full history of updates to a file. Collected data can be analyzed (correspondence between scores produced by the algorithm and the evaluation of the human editor) in order to tune up the algorithm and its hyperparameters.

A crucial hyperparameter that can be tuned is the Diversity factor. The editor can modify the Diversity value at the top of the screen (cf. Figure 20). The default value is set to 1, which is a compromise between focusing on the topics that user liked so far ("information bubble") and some serendipity effect (exploration of some new topics, but also the diversity among the top-5 recommended videos). If the diversity is set to 0, only the relevance score (i.e., similarity to the current user profile) will be taken into account. If the value is increased over 1, the freshness effect should prevail over relevance. There is an intuitive trade-off between these two optimization criteria (exploration of the video collection while keeping the relevance for the user interests) that needs to be addressed in the set of the videos proposed to the end-user.

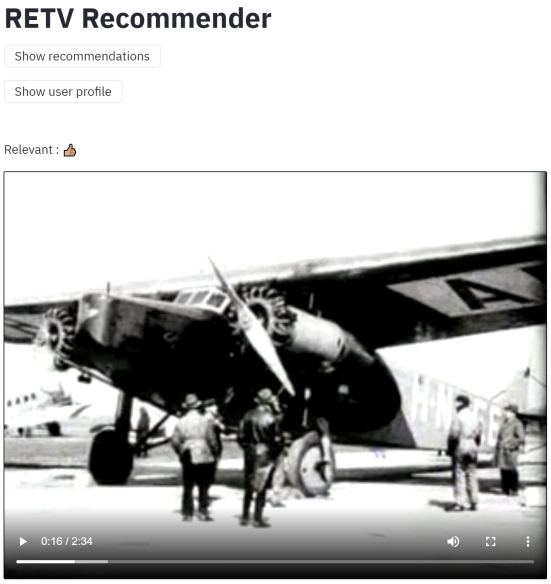

Figure 19: Recommender evaluation frontend: User profile

Initial Evaluation

The evaluation of the recommender algorithm was carried out by four professional users at NISV. They were selected based on their expertise in content curation, editorial activities and audience engagement on digital channels, and their knowledge of the Open Images collection used for the evaluation. Each person created one simulated user profile using different videos. They were asked to the score the initial five recommended videos with the diversity score set at 1.0 and then repeat this exercise with an adjusted diversity score. This was done based on their preference to see either more serendipitous results (higher than 1.0) or results closer matching their simulated profile (lower than 1.0). During the evaluation sessions, testers were also asked to provide qualitative feedback on the video recommendations they were receiving.

Quantitative analysis showed a high correspondence between metrics produced by the recommendation algorithm and professional user perception of relevance, diversity and the overall interestingness of presented results.

Four sessions produced 35 evaluations of the recommended videos. Out of these 35 videos, 32 were labeled as interesting. 23 videos were marked as surprising, and also 23 as relevant to the initial user profile. Also the agreement between numerical values produced by the algorithm and the qualitative labels is positive. The average diversity score among the videos labeled as surprising was 0.4, while for the videos labeled as not surprising it was 0.32. Similarly, for the videos marked as interesting, the average score was 25.9, while for three videos marked as not interesting, the scores values were: 15, 24.6 and 25.3 respectively (the average: 21.6).

Qualitative feedback from the professional users confirmed the high relevance of topical diversity in the recommendation engine. All testers commented that this approach would allow them to achieve their goals of promoting lesser-known content, introducing their audiences to a wide range of historical topics and gaining more value from the long tail of the collection. On a couple of instances, the engine recommended videos on topics that the testers were not even aware that they existed in the collections, or would not have thought to search for. One of the testers mentioned that the surprise element that came with getting diverse and more unexpected video recommendations would be their preferred strategy for keeping audiences

Figure 20: Recommender evaluation frontend: recommendation results

engaged in the long-term and regularly coming back to consume more content.

3.2.2. Understanding the Meaning of User Reactions

Assuming that we can collect a big enough and representative dataset on user behavior, we should learn each reaction's meaning in a data-driven way. A data-driven approach and more precisely - matrix factorization that embeds user reactions [emojis] and user attributes (activity and retention in particular] - is of great importance for understanding user reactions' actual meaning. Even more than the same reaction can mean a different thing for different users and in the context of different videos. For example, a sad emoji on a video depicting archive footage from WWII might be a positive sign, as it shows engagement with the content that is sad. The same thumbs-down emoji on a report on a caravan driving contest from the 1970s might signal dissatisfaction with the content.

For smaller organisations who are still growing their audiences and whom ReTV wants to provide solutions for, it might not be possible to collect enough data on user behaviour. We therefore decided to apply a more straightforward approach temporarily. The idea is to directly link emojis to user retention statistics (such as the number of videos that the user has seen, or even better - watched and reacted to). Knowing that the average activity among users

that used a given emoji (weighted by the number of emoji uses), we can weigh the reactions on a continuous, numerical scale between +1 (most positive impact on user retention) and -1 (most negative impact on user retention). Finally, the recommendation algorithm will use such a mixture of positive and negative weights to recommend videos that are as close as possible to the reactions that increase user retention and, simultaneously, as far as possible to the reactions that decrease retention.

3.2.3. Optimizing the Publication Time

Similar to the previous section's approach, we approach the automatic selection of the message publication time. It is a step before the full-fledged data-driven solution (discussed in section 3.2.4) which is based on collaborative filtering. It considers implicit patterns of interactions between user attributes, content attributes (in particular, t2v embeddings), and temporal attributes (time of the day, day of the week, but also, e.g., yearly seasonalities).

As again, organisations with small audiences might not be able to generate enough behavioural data we use a more straightforward solution until we can apply collaborative filtering. We collect the timestamps of the "open message" events from all users and aggregate them by user segments. Segments are obtained by clustering users according to their content-based interest profiles. To allow for faster convergence to the optimal time optimization by collecting more diversified data on user behavior, we add a randomization factor to selecting the publication time. More precisely, a list of candidates for publication time is created. Each candidate timestamp is assigned with a numerical value equivalent to the fraction of users from a given segment that opened a message at this time. We randomly select one candidate as a publication time with the probability proportional to this fraction.

3.2.4. Extended Approach based on Collaborative Filtering

Our approach to recommending relevant and fresh content to the end-user presented above focuses on content similarity. This is the approach that we will use at the start of the longitudinal user-testing of T6.3. However, we hope to collect enough usage data on user-content interactions to estimate a collaborative filtering model (users who like video X typically also like video Y). The most significant advantage of such an approach in the 4u2 Messenger recommendations is that we can learn how to weigh various categorical reactions instead of trying to weigh them arbitrarily. Encoding nuanced interactions has the advantage that we do not need to teach our model about different reactions.

In the collaborative filtering approach, we encode the content and the user behavior in the same matrix. This combined matrix allows our model to pick-up on complex interactions of features and reactions. By focusing on retention, our model can pick up nuanced interactions.

State of the art approaches presented in the scientific literature on recommender systems typically use deep neural networks, such as Google's "Deep Neural Networks for YouTube Recommendations" [14], implemented in the YouTube video service. Google's paper exploits a neural network architecture based on multiple embeddings, including averaged embeddings of watched videos and the history of user search tokens, user location embedding (for localized recommendations), and explicit user attributes (such as age or gender), cf. Figure 21. The optimized metric is the expected viewing duration, and the recommendation is driven mostly by implicit feedback (i.e., time spent on various kinds of content/channels).

In the context of ReTV recommendations, a neural network-based approach has two major drawbacks:

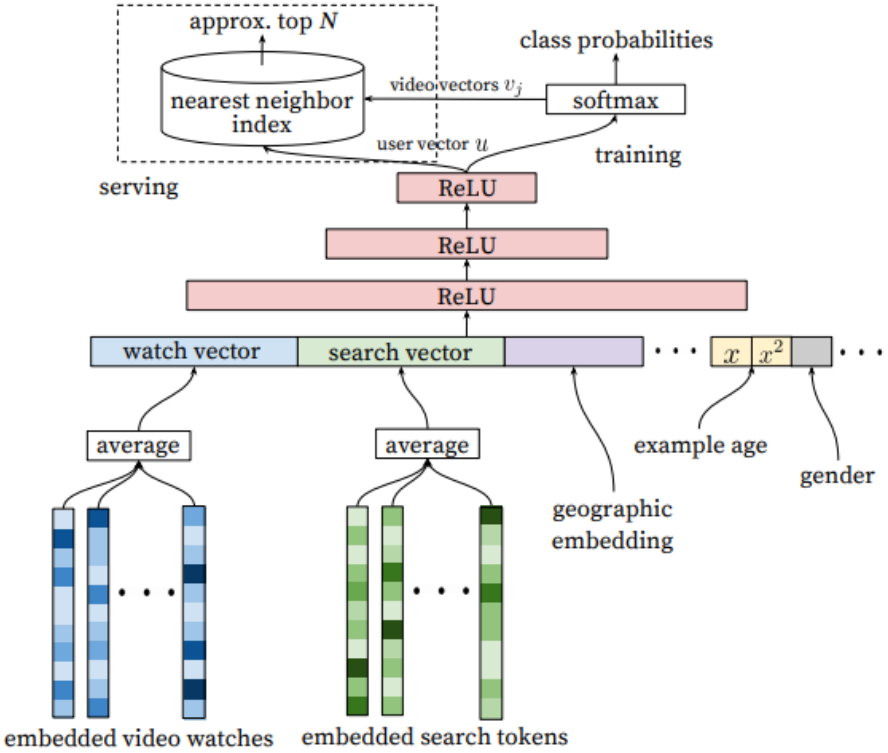

Figure 21: Youtube DNN recommender architecture (source: [14])

- DNN requires high volumes of data (in particular, a higher number of users) that we will not have shortly
- it requires an arbitrary decision of how to aggregate vectors of the content that user interacted with into a single "user profile" vector (like e.g., average pooling in Google's approach)

Instead, we will let the model implicitly create user profiles based on content features (on a single video scene granularity level) and the collaborative filtering model.

Due to the limitations of the chat platforms' data, our recommendations are based on explicit user reactions to videos. Implicit features like the duration watched are not available.

Paper [99] presents an interesting idea that we chose to use in the context of recommendation in ReTV. Multiple user-content interactions (browsing behavior, click actions, purchases, searches) are represented as graphs and embedded by so-called multi-modal fusion into a single vector. Contrary to Google's arbitrary average pooling, this kind of aggregation is model-based and data-driven. The most significant advantage of such an approach in the 4u2 Messenger recommendations is that we can learn how to weigh various categorical reactions instead of weighing them arbitrarily.

Paper [20] develops this idea even further, to multiple types of interactions as well as multi-modal input (including explicit user attributes and embeddings such as Text to Video embeddings described in Section 2.3). They claim the solution is robust against the cold start problem that we will need to be very cautious about in messenger recommendations where user feedback is very sparse.

Finally, thanks to content embedding performed on a very granular level of a single video scene, we can represent original and adapted (e.g., summarized) content in the same embedding space. Thus, we can use the same models for personalization and scheduling of original and adapted content. Moreover, we can create a reinforcement loop between the adaptation/summarization process and optimization of other success metrics (such as viewing probability).

As briefly discussed above, we base our approach on collaborative filtering, namely extension of standard matrix factorization:

- Standard matrix factorization operates on a matrix representing each user by a single column, each content piece (e.g., video or video scene) is represented in a single row. Each cell is a numerical value that measures interaction (can be either binary - watched/not-watched, or continuous - e.g., proportional to the number or duration of interaction).
- The matrix is sparse; most elements are zeros since each user only interacts with a fraction of all available content. The decomposition creates its dense approximation, where we use non-zero elements to rank items.
- In our approach, the item row is extended by its embedding. We add 2048 additional columns for each dimension of the embedding), cf. Figure 22 .
- Optionally, we can also extend each user column with user-related attributes or even user embeddings.
- We expect such an extended factorization to be more accurate since it implicitly factorizes similarity between items.

	User 1	User 2	.....	User M	Dim 1	Dim 2	.....	Dim 2047	Dim 2048	Length	Snappiness	Genre X	.....
Content 1	-1	+1	.....	-1	.027	.512	.....	.311	.127	365	0.1	1	.....
Content 2		-1	.....		.131	.101	.....	.001	.054	123	0.9	0	.....
...	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
Content N-1	+1		.....	-1	.001	.129	.....	.211	.385	1000	0.2	0	.....
Content N		-1	.....		.432	.625	.....	.019	.451	540	0.5	1	.....

Figure 22: Extended matrix factorization with Text to Video embeddings

In addition to explicit matrix factorization, we will test collaborative filtering based on manifold projections, as described in [20].

3.3. Recommendation and Scheduling in the Content Wizard

In the context of recommendations in the Content Wizard, we focused on making it easier for editors to create relevant social media posts. The main contribution of WP3 is the Text to Video search, which is a newly developed API and search database built on top of the Text to Video mapping (see Section 2.3). To narrow down the set of recommended videos, we integrated it with the Event and Anniversaries API.

In Sections 3.3.1 and 3.3.2 we present the Text to Video search, its integration into the Content Wizard and we evaluate its quality. In Section 3.3.3 we show our approach to optimizing the publication time in the Content Wizard.

3.3.1. Text to Video search

The Content Wizard and the 4u2 Messenger both work on archives of video content. The API endpoints provided by CERTH do not allow for long-term storage of features, and the process that extracts the embeddings is time-consuming. Therefore, we need to precompute the embeddings and make them available in a database that allows for fast querying.

When a user clicks on an event in the planning view of the Content Wizard, we show them related videos from their video archive. We use the title of an event and pass it into the Text to Video search. The Text to Video search turns the event's title into a signature vector (cf. Section 2.3).

We compare this signature vector to all of the signature vectors of videos that are available in the archive and return the best matches. For example, if the event was "Jurassic park opens in cinemas," we turn this string of text into a signature vector. Since the text and the videos are in the same vector space, we can then find videos close to this text. In the case of the NISV archive, we then find videos that deal with cinema in general. The user of the Content Wizard can then select the video, summarize it if desired, and then post it on social media with an accompanying message.

To show videos for an event, we integrated the search into a workflow that extracts embeddings from videos and stores them for later retrieval and comparison.

The API is an extension of the one described in D3.2 which only provided support for summaries. It has the following endpoints:

Endpoint	Parameters	Description	Response
/v2/video/	via JSON	Register a video for feature extraction (va and t2v)	{JSON with TOKEN}
/v2/video/		Gets status of all video	[{JSON representation of video object}]
/v2/video/TOKEN	Get status for one video	{JSON representation of video object}	
/v2/video/search/	q="Search string"	Search for a video	List of TOKENS with optional metadata (title etc.) ordered by relevance
/v2/summary/	via JSON	Returns the summary for the given parameter	Summary script
/v2/summary/ download/TOKEN		Download features from VA service	HTTP status 204 if successful
/v2/video/search/ download/TOKEN		Download embeddings from Text to Video service	HTTP status 204 if successful

As described in 2.1.2, the summarization can be configured. We extended the `/v2/summary/` endpoint to allow passing along parameters to configure the summarization. As figure 23

shows, we surface those configuration parameters to the user in the Content Wizard. We will describe its relevance for the use case in more detail in D6.3.

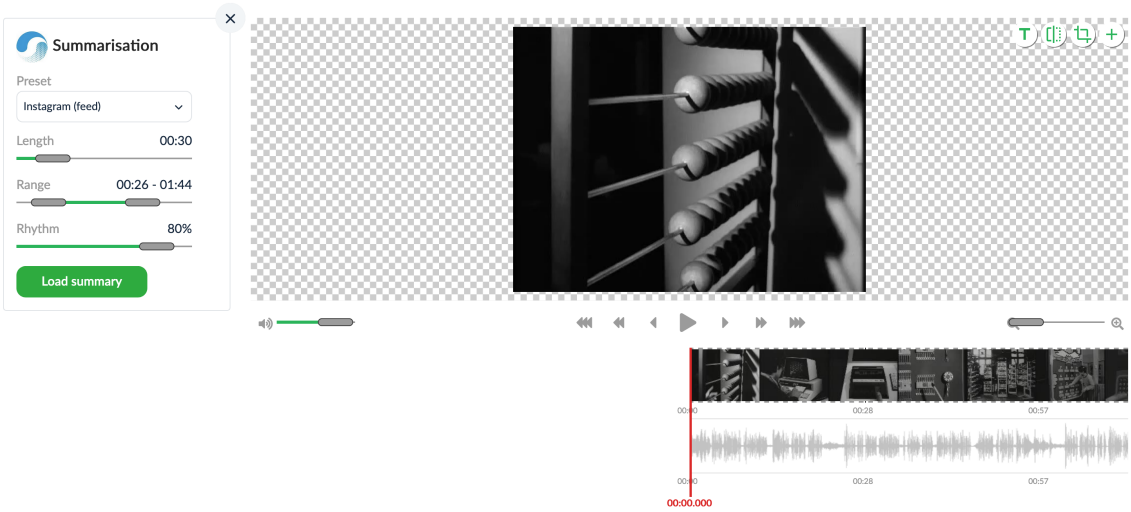

Figure 23: Summary configuration in the Content Wizard

The features from the Text to Video endpoint are stored in Elasticsearch[9]. We chose Elasticsearch over a relational database like Postgres as it has dedicated support for storing vectors[10].

We use the cosine similarity metric between the vectors find the video that matches best. Finding the best match is somewhat arbitrary, as we have a signature vector for each sub-shot of a video, and we, therefore, get a matching score for each of those sub-shots. How we aggregate the scores of those different shots can impact the search results. Let us assume we have two videos, each containing ten sub-shots. The first video contains a scene that discusses a dinosaur movie and nine other scenes with no relation to the cinema or dinosaurs. The second video is a sequence of 10 scenes shot in a dinosaur museum. One could argue that the first video scene is the most relevant, but the second video is more relevant on average. We have decided to aggregate the distance metrics for each scene in such a way as to rank the first video higher: Social media managers want short content that is highly relevant.

However, we envision other cases where the average relevant might be more critical. Therefore, our search implementation is flexible in ranking logic and can even support multiple strategies that can be interchanged on the fly to allow for A/B testing.

The Text to Video search API can be used with any query. However, in the context of the Content Wizard, it is directly integrated with the Events and Anniversaries API (See section 2 of D2.3 for a detailed description). We learned that social media managers focus on posts around anniversaries and other current events. To find such events, they often use websites like On This Day[11]. For example, they might learn that on the 11th of June 1993, "Jurassic Park" opened in cinemas. They would need to manually comb their archive for content relevant to dinosaurs or movies and create a similar post.

This process of finding events and matching videos is laborious. Human editors can now see

[9] See https://www.elastic.co/elasticsearch/
[10] See https://www.elastic.co/guide/en/elasticsearch/reference/current/dense-vector.html#
[11] See https://www.onthisday.com/

what events or anniversaries are relevant today and for weeks in advance. This integration allows them to do all of their planning in the same tool without manually searching external websites.

We load the events directly out of the WLT Metadata Repository through simple API calls. The call below is an example of how we can load events for June 2020:

```
curl -X POST -H "Authorization: Bearer $TOKEN"
-H "Content-Type: application/json"
--data '{"from_date": "2020-06-01", "to_date": "2020-06-30"}'
https://api.weblyzard.com/1.0/skb/events
```

3.3.2. Analysis of Search Integration

We analyzed the quality of the Text to Video service and tested our search integration. We used the Open Images[12] collection from NISV, which contains both videos and short descriptions. We set up a video embeddings database by running 1837 videos through the Text to Video endpoint of CERTH. For each video shot, we stored a signature vector on disk.

Figure 24: Example of a video from the Open Images collection

The videos each have a title and a short, human-created description. The title for the example video in 24 is "Car soccer," and the description is "Soccer match in Overloon with altered cars, a ball with a diameter of one meter, flagpoles as goal posts and a referee on a motorbike."

We then created embeddings for each title and each description as well. Descriptions that contained multiple sentences had one signature per sentence. We compared it to all video embeddings using the cosine similarity metric for each of those text embeddings.

Since each video has multiple embeddings, one per shot, we had to aggregate the results. As described in the previous section, we chose to aggregate by maximum, making the best matching shot represent the whole video.

For each title and description, we searched the ranking for the video they described. If the car soccer video had a rank of 0 in the description similarity ranking, it would mean that the video signature and the description signature matched best. If it had a rank of 3, it would mean that

[12] See https://openbeelden.nl/about.en

three other videos matched the description better. If it had a rank of 1836, it would mean that it was the worst overall match since we compared 1837 videos against each other.

Intuitively, we expect all videos to have a decent rank since the descriptions and the titles were created by humans and meant to represent the video's content. Since we ran the experiment for both the titles and the descriptions of the videos, we have two sets of results. We will first present the results for using the description as a search term and then the title.

Table 15 describes the ranks for each of the videos compared to the embeddings of their descriptions.

count	1837
mean	145.73
std	258.19
min	0
5%	0
10%	2
20%	7
30%	15
40%	28
50%	48
60%	75.6
70%	122
80%	203
90%	396
95%	633
max	1802

Table 15: Distribution of description-based similarity ranks. 5% of videos have rank 0. 20% have rank 7 or lower. Lower is better.

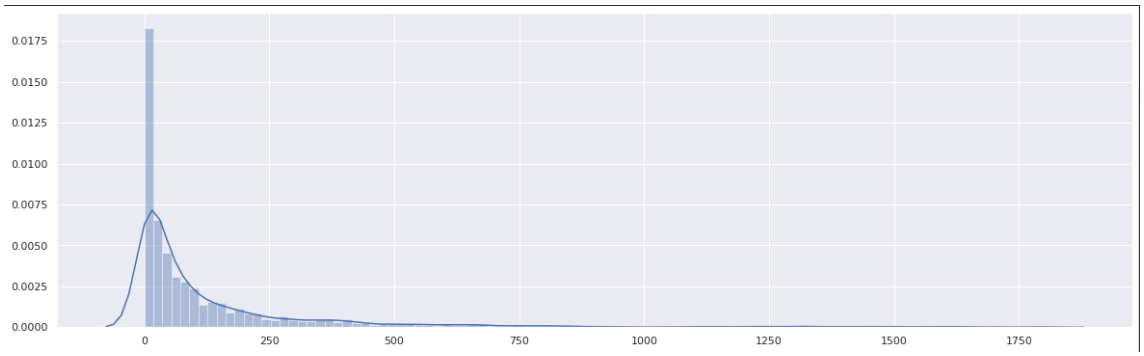

Figure 25: Rank distribution for description-based similarity

See figure 25 for a histogram of the description ranks. Individual bars in the plot represent discrete bins of ranks, and blue line shows the continuous (Gaussian) distribution estimate based on discrete observed histogram.

Table 16 describes the ranks for each of the videos when compared to the embeddings of their titles.

The results are satisfactory for both title- and description-based querying. However, longer queries seem to be more descriptive and result in ranks closer to the top of the ranking (mean rank: 145.73 for descriptions vs. 319.48 for titles, median rank: 48 for descriptions vs. 116 for titles).

count	1837
mean	319.48
std	436.57
min	0
5%	1
10%	3
20%	12
30%	30
40%	57
50%	116
60%	197.6
70%	336
80%	589.6
90%	1053.4
95%	1315.4
max	1834

Table 16: Distribution of title-based similarity ranks. 5% of videos have rank 0 or 1. 20% have rank 12 or lower. Lower is better.

As can be seen in the plot, the rank distribution is highly skewed towards ranks closer to 0, and it rarely happens that the ranks are in the long tail region (the lowest rank being 1837, i.e., the total number of NISV videos). Over 30% of all videos were ranked at positions 0 to 15, i.e., relatively close to the top.

This experiment informed our decision to use Elasticsearch as a database because naive querying on disk was prohibitively slow. Even with 32-thread parallelization, naive querying (1837 queries each looping through 1837 videos) took 82 minutes for the descriptions. For titles (which are shorter and usually have just one signature), querying took 11 minutes.

It is worth investigating what features of video descriptions or titles, or what visual features of videos themselves cause some videos to be ranked near the end of the ranking. However, the real test of the quality of the search will be the feedback from users of the Content Wizard and 4u2 Messenger which we'll gather as part of T5.3 and T6.3.

3.3.3. Optimal Publication Time for Social-Media

In the Content Wizard, we do not have the same freedom in addressing individual users as in the 4u2 Messenger. We cannot decide on an individual level when a user receives our post: the social media platforms have algorithms that decide which user gets to see what content at what time. We see this as an additional layer of misdirection between the user and our content.

We believe that we can have a significant impact on user engagement by optimizing the publication time in the 4u2 Messenger and have therefore focused our efforts on building a recommendation model that can recommend optimal publication time (see Section 3.2).

We will add support for broadly optimizing the publication time in the Content Wizard as part of the integration work done in T5.3. We plan to use predictive analytics from WP2 for this purpose (see Section 4 in D2.3). Since we can predict audience sizes for certain terms, we will publish our content shortly before the predicted peak of the topic.

3.4. Recommendation and Summarization in the Storypact Editor

Content creators do not only need tools to edit and optimize video content for a particular publishing vector, but also to author and refine text optimised for their target audiences. Ideally, the underlying analysis encompasses both the text that is being edited as well as recent trends in the public debate - i.e., other documents on similar topics that have been published in the past few weeks. In many cases, the goal is to increase the reach of an online presence and maximize the impact of social media postings and other dissemination activities. For this purpose, the Storypact Editor bundles the text summarization and recommendation technologies outlined in the following section into a Web-based tool to assist content creators. It includes the visual widgets of WP4 to convey context information in a simplified, user-friendly setting that is less complex than the fully synchronized dashboard experience of the Topic Compass.

Overview of the Functionality

Offering data-driven text optimisation and summarization features, the editor helps content creators maximise the impact of their content assets. The system analyzes the text while users write, suggesting modifications to increase the achievable impact and align the text with an organization's communication goals. Suggestions are based on an analysis of the public debate, recommending related keywords and identifying opinion leaders who shape this debate.

The M34 prototype processes text snippets of variable lengths, allowing for an evaluation in different usage contexts. Future versions will also customize its recommendations to specific output formats such as press releases, Web pages and social media postings.

Language Selection and Focus Keywords

To fine-tune the recommendations of the Storypact editor, it is important to set the correct language in the upper right corner (see screenshot of Figure 27). In addition, there is an optional text input field in the header to specify one or more keywords such as innovation or sustainability (multiple keywords need to be separated by a comma). Such focus keywords help to tailor the editor's recommendations to a specific topic or domain, ignoring trends in the public debate that are irrelevant to a given target group.

Editor Modes

Storypact supports a wide range of communication goals and workflows. The text editor offers four modes with recommendations tailored to the specific aspect of a story that the users wants to analyze and strengthen:

- **Suggest New Terms** provides a source of inspiration for hashtags or keywords to include that do not yet appear in the provided text. Users can filter these terms by sentiment, recency, source or the popularity as a Google search term (the optional Whitespace Analysis optimizes the suggestions by giving preference to terms that do not yet appear on many other Web sites). Users can display the recommendations either as a tag cloud or as a phrase list to identify two- or three-word combinations.
- **Explore Related Content** analyzes news media coverage in different countries. Users can select between a list of top-ranked documents, a geographic map showing the current hot spots of the coverage, and a list of opinion leaders (frequently referenced persons of interest or influential news media outlets that have recently published about similar topics).

- **Analyze Current Text** evaluates the current wording and uses green and red colors to highlight terms to keep and those that should rather be rephrased or replaced - if underlined in blue, Storypact has identified related expressions expected to perform better (see Figure 26). Users can focus the analysis on specific word forms (nouns, verbs or adjectives) and update the display with the refresh button in the upper right corner. The drop-down below offers a "Quality of Text" option that includes a spell checker to spot typos, and a highlighting module to indicate frequently used words since accidental repetitions can negatively impact the quality of a text (although a certain level of redundancy is often intended).

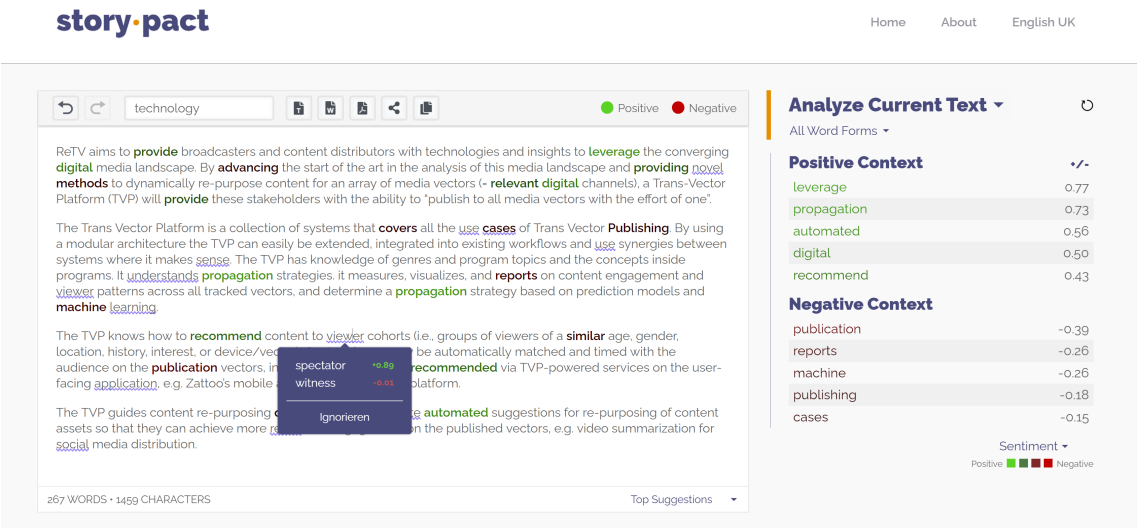

Figure 26: Screenshot of the text analysis feature of the Storypact Editor

- **Generate Summary.** Shortens the text to a desired number of sentences, facilitating its reuse across channels with specific length requirements. The two icons on the right hide the orange "..." separators that indicate removed sentences, and copy the shortened text to the clipboard (see Figure 27).

Text Summarization Options

The M34 text summarization component of Storypact returns condensed representations of the input text, based on a computation of term significance values using co-occurrence analysis. The mean value of term significance values in a sentence is combined with additional additional metadata to change and optimize the ranking order of the sentences contained in the shortened document. The component provides three ranking options to adapt the condensed representation, tailored to the users' specific goals and tasks:

- *Content.* The focus is on key aspects of the story by selecting the most characteristic sentences - computed via keyness metrics that rely on co-occurrence analysis and the distribution of sentences between a target corpus (= the current text) and a reference corpus (= the entire set of documents published within the past 30 days in the selected content source, for example UK News Media).
- *Google Search* favors sentences that contain popular search terms, using the average search volume over the past <n> weeks to distinguish words likely to increase organic search traffic from words that are less popular as a search term. Historic values are

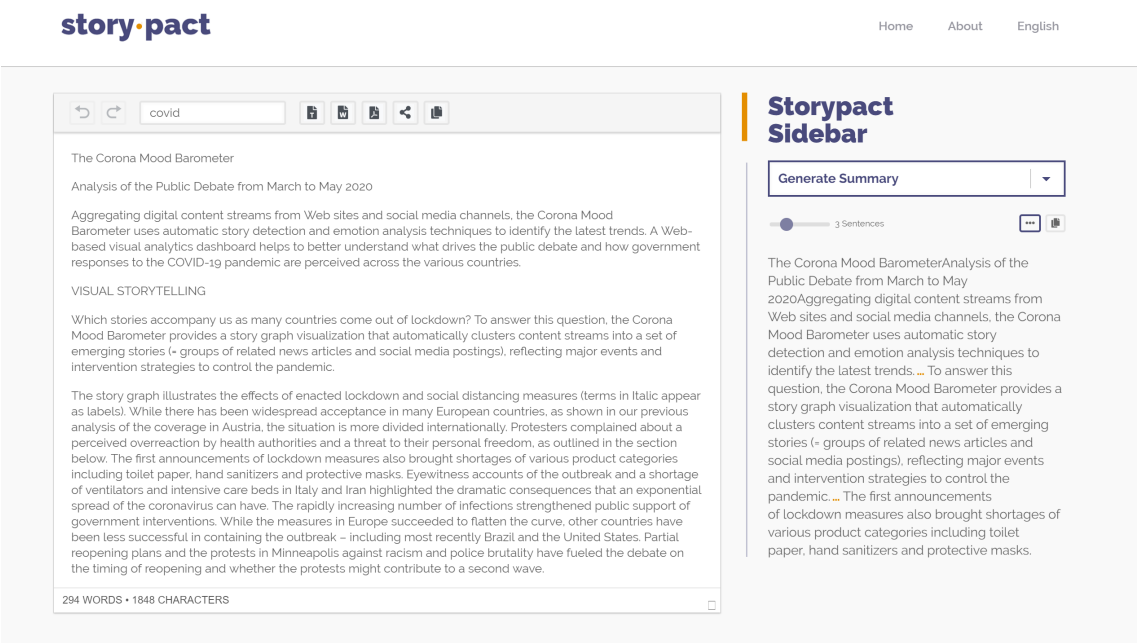

Figure 27: Screenshot of the text summarization feature of the Storypact Editor

stored in a term cache, for increased scalability and improved recall in the case of less popular terms. The term cache is an evolving repository of term-specific metadata that helps to overcome the latency of third-party API calls.

- *Sentiment* ranks those sentences higher that are likely to convey a positive attitude, avoiding sentences that contain expressions with a negative connotation in the public debate. It is based on the same lexical sentiment analysis algorithms that are also used to highlight the sentiment of keywords with green and red colors in the "Analyse Current Text" mode of *Storypact*, as well as the various visualisations of the *Topic Compass*.

We plan to further extend the summarization feature shown in Figure 27 and the corresponding REST API calls to provide the option to adapt the ranking of sentences, taking into account their vector-space similarity to a keyword triple. This will unlock several new features, particularly in conjunction with the scheduling of posts outlined in the *Content Wizard* section above. The ranking process will boost sentences that contain terms co-occurring with one of more of those keywords, thereby increasing the expected impact of a message to be sent out at a given future point in time. If a date range is supplied rather than a specific date, both the exact date and sequence of sentences will be part of the recommendation - using the maximum expected impact as the optimization criterion.

Iterative Development and Evaluation

The goal of Storypact is to merge the traditionally disparate processes of information seeking and authoring when working on press releases, Web pages or social media postings that accompany published videos. The evaluation of the editor followed an evolutionary and user-centered development approach, with rapid feedback cycles focusing on usability, the accuracy of the provided recommendations, and the added value in real-world use cases. We conducted usability inspections, asking project partners and third-party experts to assess the interface design against recognized usability principles, collecting their feedback by means of an online

form. Since the WP4 editor integration and corresponding evaluation tasks are still ongoing, a summary of the results will be provided as part of *D4.3: Trans-Vector Platform, Final and Optimized Version* in M36 of the ReTV Project.

4. Conclusion and Outlook

In this deliverable, we have shown improved approaches to content adaptation and summarization, as well as recommendations and scheduling. We have seen the pace of integration accelerate, both within WP3 but also for services from other work packages. We will continue improving the relevant results from WP3 as part of the integration work of WP4, WP5 and WP6.

In the context of video adaptation and repurposing, we developed a new architecture for unsupervised video summarization, that combines Actor-Critic with GANs, showcasing its competitive performance among literature methods and its superiority over the summarization methods presented in D3.2. This method was combined with the previous non-learning based video summarization method and was deployed to a service that generates summaries that fully comply with the requirements and editorial rules that are relevant for the ReTV use cases. We will continue optimizing the performance and accuracy of the deployed method and we will test its performance on videos of different genres than those that are currently used, until M36, while further robustifying the implementation of the integrated method in the context of ongoing work in WP4. We also introduced a new module that can transform the final video summaries to a different target aspect ratio than the original video. The developed method achieves this by either intelligently cropping the original video frames at the center of the viewer's predicted attention, or by padding the video frames. The decision for which is the best approach to use is automatically taken by our algorithm.

In the context of Text to Video matching we developed a technique that achieves state-of-the-art results in standard benchmark datasets, deployed this method as a service and use the text to video embeddings for the Video search in the Content Wizard. We will continue our work to improve the quality of embeddings and the quality of the software implementation of this technique, again as part of the ongoing work in WP4 until M36. We have improved the manual scheduling in the Content Wizard by integrating with the Event and Anniversaries API of WP2 and linked it to the Text to Video search.

We extended our approach to recommending videos in the 4u2 Messenger by using the CERTH API's video-embeddings. After positive evaluation results from expert users, we will now test and iteratively improve the recommendation engine in the context of WP5 and WP6, with a focus on usability and user satisfaction.

As part of the system integration activities in WP4, work will also continue on the Storypact editor introduced in Section 3.4. It is planned to (i) extend the stand-alone Web version and make it embeddable into other applications, (ii) support additional languages including Dutch, Spanish and French, and (iii) integrate the developed summarization function with the affective metadata extraction work of WP2, allowing users to select different types of summarization. In addition to the "keyness" sorting of sentences already offered in the current prototype, this will add the capability to sort by sentiment or match with a specific set of desired keywords.

References

[1] G. Abdollahian, Z. Pizlo, and E. J. Delp. A study on the effect of camera motion on human visual attention. In *2008 15th IEEE International Conference on Image Processing*, pages 693–696. IEEE, 2008.

[2] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras. Performance over random: A robust evaluation protocol for video summarization methods. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1056–1064, New York, NY, USA, 2020. Association for Computing Machinery.

[3] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras. Unsupervised video summarization via attention-driven adversarial learning. In Y. M. Ro, W.-H. Cheng, J. Kim, W.-T. Chu, P. Cui, J.-W. Choi, M.-C. Hu, and W. De Neve, editors, *Proc. of the MultiMedia Modeling 2020*, pages 492–504, Cham, 2020. Springer International Publishing.

[4] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras. Unsupervised video summarization via attention-driven adversarial learning. In *International Conference on Multimedia Modeling*, pages 492–504. Springer, 2020.

[5] E. Apostolidis, A. I. Metsai, E. Adamantidou, V. Mezaris, and I. Patras. A stepwise, label-based approach for improving the adversarial training in unsupervised video summarization. In *Proc. of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*, AI4TV '19, pages 17–25, New York, NY, USA, 2019. ACM.

[6] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 papers*, pages 10–es. Association for Computing Machinery, 2007.

[7] M. Barrett, J. Bingel, N. Hollenstein, M. Rei, and A. Søgaard. Sequence classification with human attention. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 302–312, 2018.

[8] M. Bastan, X. Shi, J. Gu, Z. Heng, C. Zhuo, D. Sng, and A. Kot. Ntu rose lab at trecvid 2018: Ad-hoc video search and video to text. In *TRECVID 2018 Workshop. Gaithersburg, MD, USA*. NIST, USA, 2018.

[9] Y. Chen, L. Tao, X. Wang, and T. Yamasaki. Weakly supervised video summarization by hierarchical reinforcement learning. In *Proceedings of the ACM Multimedia Asia*, MMAsia '19, New York, NY, USA, 2019. Association for Computing Machinery.

[10] Y. Chen, L. Tao, X. Wang, and T. Yamasaki. Weakly supervised video summarization by hierarchical reinforcement learning. In *Proc. of the ACM Multimedia Asia*, MMAsia '19, New York, NY, USA, 2019. Association for Computing Machinery.

[11] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[12] W.-T. Chu and Y.-H. Liu. Spatiotemporal modeling and label distribution learning for video summarization. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2019.

[13] W.-T. Chu and Y.-H. Liu. Spatiotemporal modeling and label distribution learning for video summarization. In *Proc. of the 2019 IEEE 21st Int. Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2019.

[14] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *RecSys*, pages 191–198. ACM, 2016.

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[16] J. Dong, X. Li, and C. G. M. Snoek. Word2visualvec: Image and video to sentence matching by visual feature prediction. *arXiv preprint arXiv:1604.06838*, 2016.

[17] J. Dong, X. Li, and C. G. M. Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Transactions on Multimedia (TMM)*, 20(12):3377–3388, Dec. 2018.

[18] J. Dong, X. Li, C. Xu, S. Ji, Y. He, G. Yang, and X. Wang. Dual encoding for zero-example video retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9346–9355, 2019.

[19] R. Droste, J. Jiao, and J. A. Noble. Unified Image and Video Saliency Modeling. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, 2020.

[20] J. Dąbrowski, B. Rychalska, M. Daniluk, D. Basaj, P. Babel, and A. Michałowski. An efficient manifold density estimator for all recommendation systems, 2020.

[21] M. Elfeki and A. Borji. Video summarization via actionness ranking. In *IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, HI, USA, January 7-11, 2019*, pages 754–763, Jan. 2019.

[22] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[23] J. Fajtl, H. S. Sokeh, V. Argyriou, D. Monekosso, and P. Remagnino. Summarizing videos with attention. In G. Carneiro and S. You, editors, *Asian Conference on Computer Vision (ACCV) 2018 Workshops*, pages 39–54, Cham, 2019. Springer International Publishing.

[24] L. Feng, Z. Li, Z. Kuang, and W. Zhang. Extractive video summarizer with memory augmented neural networks. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 976–983, New York, NY, USA, 2018. ACM.

[25] D. Francis, P. Anh Nguyen, B. Huet, and C.-W. Ngo. Fusion of multimodal embeddings for ad-hoc video search. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1868–1872, Oct. 2019.

[26] N. Gonuguntla, B. Mandal, N. Puhan, et al. Enhanced deep video summarization network. BMVC, 2019.

[27] A. Goyal, N. R. Ke, A. Lamb, R. D. Hjelm, C. J. Pal, J. Pineau, and Y. Bengio. Actual: Actor-critic under adversarial learning. *ArXiv*, abs/1711.04755, 2017.

[28] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Discontinuous seam-carving for video retargeting. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 569–576. IEEE, 2010.

[29] Y. Guo, F. Liu, J. Shi, Z.-H. Zhou, and M. Gleicher. Image retargeting using mesh parametrization. *IEEE Transactions on Multimedia*, 11(5):856–867, 2009.

[30] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV) 2014*, pages 505–520, Cham, 2014. Springer International Publishing.

[31] A. Habibian, T. Mensink, and C. G. Snoek. Video2vec embeddings recognize events when examples are scarce. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(10):2089–2103, Oct. 2017.

[32] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2007.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] X. He, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan. Unsupervised video summarization with attentive conditional generative adversarial networks. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, pages 2296–2304, New York, NY, USA, 2019. ACM.

[35] X. He, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan. Unsupervised video summarization with attentive conditional generative adversarial networks. In *Proc. of the 27th ACM Int. Conf. on Multimedia*, MM '19, page 2296–2304, New York, NY, USA, 2019. Association for Computing Machinery.

[36] C. Huang and H. Wang. A novel key-frames selection framework for comprehensive video summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):577–589, 2020.

[37] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[38] Z. Ji, K. Xiong, Y. Pang, and X. Li. Video summarization with attention-based encoder-decoder networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2019.

[39] Y. Juan, D. Lefortier, and O. Chapelle. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 680–688, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[40] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *2009 IEEE 12th international conference on computer vision*, pages 2106–2113. IEEE, 2009.

[41] Y. Jung, D. Cho, D. Kim, S. Woo, and I. S. Kweon. Discriminative feature learning for unsupervised video summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8537–8544, 2019.

[42] Y. Jung, D. Cho, D. Kim, S. Woo, and I.-S. Kweon. Discriminative feature learning for unsupervised video summarization. In *Proc. of the 2019 AAAI Conf. on Artificial Intelligence*, 2019.

[43] D. Kaufman, G. Levi, T. Hassner, and L. Wolf. Temporal tessellation: A unified approach for video analysis. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 94–104, Oct. 2017.

[44] J. Kiess, R. Garcia, S. Kopf, and W. Effelsberg. Improved image retargeting by distinguishing between faces in focus and out of focus. In *2012 IEEE International Conference on Multimedia and Expo Workshops*, pages 145–150. IEEE, 2012.

[45] J. Kiess, D. Gritzner, B. Guthier, S. Kopf, and W. Effelsberg. Gpu video retargeting with parallelized seamcrop. In *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 139–147, 2014.

[46] J. Kiess, B. Guthier, S. Kopf, and W. Effelsberg. Seamcrop: Changing the size and aspect ratio of videos. In *Proceedings of the 4th Workshop on Mobile Video*, pages 13–18, 2012.

[47] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. Density based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[48] S. S. Kruthiventi, K. Ayush, and R. V. Babu. Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26(9):4446–4456, 2017.

[49] S. Lal, S. Duggal, and I. Sreedevi. Online video summarization: Predicting future to better summarize present. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 471–480. IEEE, 2019.

[50] S. Lal, S. Duggal, and I. Sreedevi. Online video summarization: Predicting future to better summarize present. In *Proc. of the 2019 IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pages 471–480, 2019.

[51] D.-D. Le, S. Phan, V.-T. Nguyen, B. Renoust, T. A. Nguyen, V.-N. Hoang, T. D. Ngo, M.-T. Tran, Y. Watanabe, M. Klinkigt, et al. NII-HITACHI-UIT at TRECVID 2016. In *TRECVID 2016 Workshop. Gaithersburg, MD, USA*, 2016.

[52] O. Le Meur, P. Le Callet, D. Barba, and D. Thoreau. A coherent computational approach to model bottom-up visual attention. *IEEE transactions on pattern analysis and machine intelligence*, 28(5):802–817, 2006.

[53] L. Lebron Casas and E. Koblents. Video summarization with lstm and deep attention models. In I. Kompatsiaris, B. Huet, V. Mezaris, C. Gurrin, W.-H. Cheng, and S. Vrochidis, editors, *MultiMedia Modeling*, pages 67–79, Cham, 2019. Springer International Publishing.

[54] J. Lei, Q. Luan, X. Song, X. Liu, D. Tao, and M. Song. Action parsing-driven video summarization based on reinforcement learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(7):2126–2137, 2019.

[55] X. Li, C. Xu, G. Yang, Z. Chen, and J. Dong. W2vv++: Fully deep learning for ad-hoc video search. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1786–1794. ACM, 10 2019.

[56] X. Li, J. Ye, C. Xu, S. Yun, L. Zhang, X. Wang, R. Qian, and J. Dong. Renmin university of china and zhejiang gongshang university at trecvid 2019: Learn to search and describe videos. In *TRECVID 2019 Workshop. Gaithersburg, MD, USA*, 2019.

[57] X. Li, B. Zhao, and X. Lu. A general framework for edited video and raw video summarization. *IEEE Transactions on Image Processing*, 26(8):3652–3664, Aug. 2017.

[58] Y. Li, Y. Song, L. Cao, J. Tetreault, L. Goldberg, A. Jaimes, and J. Luo. Tgif: A new dataset and benchmark on animated gif description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, June 2016.

[59] S.-S. Lin, C.-H. Lin, I.-C. Yeh, S.-H. Chang, C.-K. Yeh, and T.-Y. Lee. Content-aware video retargeting using object-preserving warping. *IEEE transactions on visualization and computer graphics*, 19(10):1677–1686, 2013.

[60] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[61] F. Liu and M. Gleicher. Automatic image retargeting with fisheye-view warping. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 153–162, 2005.

[62] F. Liu and M. Gleicher. Video retargeting: automating pan and scan. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 241–250, 2006.

[63] H. Liu, X. Xie, W.-Y. Ma, and H.-J. Zhang. Automatic browsing of large pictures on mobile devices. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 148–155, 2003.

[64] Y.-T. Liu, Y.-J. Li, F.-E. Yang, S.-F. Chen, and Y.-C. F. Wang. Learning hierarchical self-attention for video summarization. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3377–3381. IEEE, 2019.

[65] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial lstm networks. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2982–2991, 2017.

[66] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proc. of the 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2982–2991, 2017.

[67] F. Markatopoulou, D. Galanopoulos, V. Mezaris, and I. Patras. Query and keyframe representations for ad-hoc video search. In *Proceedings of the 2017 ACM International Conference on Multimedia Retrieval*, ICMR '17, pages 407–411. ACM, 2017.

[68] L. McInnes and J. Healy. Accelerated hierarchical density based clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 33–42. IEEE, 2017.

[69] J. Meng, S. Wang, H. Wang, J. Yuan, and Y.-P. Tan. Video summarization via multiview representative selection. *IEEE Transactions on Image Processing*, 27(5):2134–2145, May 2018.

[70] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, Workshop Track Proceedings*, ICLR '13, 2013.

[71] N. C. Mithun, J. Li, F. Metze, and A. K. Roy-Chowdhury. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, pages 19–27. ACM, 2018.

[72] N. C. Mithun, S. Paul, and A. K. Roy-Chowdhury. Weakly supervised video moment retrieval from text queries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11592–11601, 2019.

[73] J. Pan, E. Sayrol, X. Giro-i Nieto, K. McGuinness, and N. E. O'Connor. Shallow and deep convolutional networks for saliency prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 598–606, 2016.

[74] B. Philipp, K. Ciesielski, and L. Nixon. Automatically adapting and publishing tv content for increased effectiveness and efficiency. In *Proceedings of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*, pages 51–52, 2019.

[75] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision (ECCV) 2014*, pages 540–555, Cham, 2014. Springer International Publishing.

[76] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *2009 IEEE 12th International Conference on Computer Vision*, pages 151–158. IEEE, 2009.

[77] M. Rochan and Y. Wang. Video summarization by learning from unpaired data. In *Proc. of the 2019 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7894–7903, 2019.

[78] M. Rochan, L. Ye, and Y. Wang. Video summarization using fully convolutional sequence networks. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *European Conference on Computer Vision (ECCV) 2018*, pages 358–374, Cham, 2018. Springer International Publishing.

[79] M. Rubinstein, A. Shamir, and S. Avidan. Improved seam carving for video retargeting. *ACM transactions on graphics (TOG)*, 27(3):1–9, 2008.

[80] M. Rubinstein, A. Shamir, and S. Avidan. Multi-operator media retargeting. *ACM Transactions on graphics (TOG)*, 28(3):1–11, 2009.

[81] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen. Gaze-based interaction for semi-automatic photo cropping. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 771–780, 2006.

[82] C. G. Snoek, X. Li, C. Xu, and C. D. Koelma. University of amsterdam and renmin university at trecvid 2017: Searching video, detecting events and describing video. In *TRECVID 2017 Workshop. Gaithersburg, MD, USA*, 2017.

[83] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsum: Summarizing web videos using titles. In *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5179–5187, June 2015.

[84] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic thumbnail cropping and its effectiveness. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 95–104, 2003.

[85] Y. Sun and R. Fisher. Object-based visual attention for computer vision. *Artificial intelligence*, 146(1):77–123, 2003.

[86] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[87] E. Vig, M. Dorr, and D. Cox. Large-scale optimization of hierarchical features for saliency prediction in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2798–2805, 2014.

[88] J. Wang, M. J. Reinders, R. L. Lagendijk, J. Lindenberg, and M. S. Kankanhalli. Video content representation on tiny devices. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 3, pages 1711–1714. IEEE, 2004.

[89] J. Wang, W. Wang, Z. Wang, L. Wang, D. Feng, and T. Tan. Stacked memory network for video summarization. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, pages 836–844, New York, NY, USA, 2019. ACM.

[90] J. Wang, W. Wang, Z. Wang, L. Wang, D. Feng, and T. Tan. Stacked memory network for video summarization. In *Proc. of the 27th ACM Int. Conf. on Multimedia*, MM '19, page 836–844, New York, NY, USA, 2019. Association for Computing Machinery.

[91] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 20–36, Cham, 2016. Springer International Publishing.

[92] W. Wang, J. Shen, J. Xie, M. Cheng, H. Ling, and A. Borji. Revisiting video saliency prediction in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[93] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel. Motion-aware temporal coherence for video resizing. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–10. Association for Computing Machinery, 2009.

[94] Y.-S. Wang, H.-C. Lin, O. Sorkine, and T.-Y. Lee. Motion-based video retargeting with optimized crop-and-warp. In *ACM SIGGRAPH 2010 papers*, pages 1–9. Association for Computing Machinery, 2010.

[95] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y. Lee. Optimized scale-and-stretch for image resizing. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–8. Association for Computing Machinery, 2008.

[96] H. Wei, B. Ni, Y. Yan, H. Yu, X. Yang, and C. Yao. Video summarization via semantic attended networks. In *2018 AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[97] H. Wei, B. Ni, Y. Yan, H. Yu, X. Yang, and C. Yao. Video summarization via semantic attended networks. In *Proc. of the 2018 AAAI Conf. on Artificial Intelligence*, pages 216–223, 2018.

[98] L. Wolf, M. Guttmann, and D. Cohen-Or. Non-homogeneous content-driven video-retargeting. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–6. IEEE, 2007.

[99] A. Wroblewska, J. Dabrowski, M. Pastuszak, A. Michalowski, M. Daniluk, B. Rychalska, M. Wieczorek, and S. Sysko-Romanczuk. Multi-modal embedding fusion-based recommender, 2020.

[100] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[101] Xinhui Song, Ke Chen, Jie Lei, Li Sun, Zhiyuan Wang, Lei Xie, and Mingli Song. Category driven deep recurrent neural network for video summarization. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, July 2016.

[102] J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

[103] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2852–2858. AAAI Press, 2017.

[104] L. Yuan, F. E. H. Tay, P. Li, L. Zhou, and J. Feng. Cycle-sum: Cycle-consistent adversarial lstm networks for unsupervised video summarization. In *Proc. of the 2019 AAAI Conf. on Artificial Intelligence*, 2019.

[105] Y. Yuan, H. Li, and Q. Wang. Spatiotemporal modeling for video summarization using convolutional recurrent neural network. *IEEE Access*, 7:64676–64685, 2019.

[106] Y. Yuan, T. Mei, P. Cui, and W. Zhu. Video summarization by learning deep side semantic embedding. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):226–237, Jan. 2019.

[107] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Summary transfer: Exemplar-based subset selection for video summarization. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1059–1067, 2016.

[108] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV) 2016*, pages 766–782, Cham, 2016. Springer International Publishing.

[109] Y. Zhang, M. Kampffmeyer, X. Zhao, and M. Tan. Dtr-gan: Dilated temporal relational adversarial network for video summarization. In *Proc. of the ACM Turing Celebration Conf. - China*, ACM TURC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[110] Y. Zhang, X. Liang, D. Zhang, M. Tan, and E. P. Xing. Unsupervised object-level video summarization with online motion auto-encoder. *Pattern Recognition Letters*, 2018.

[111] B. Zhao, X. Li, and X. Lu. Hierarchical recurrent neural network for video summarization. In *Proc. of the 25th ACM Int. Conf. on Multimedia*, MM '17, page 863–871, New York, NY, USA, 2017. Association for Computing Machinery.

[112] B. Zhao, X. Li, and X. Lu. Hierarchical recurrent neural network for video summarization. In *Proceedings of the 2017 ACM on Multimedia Conference*, MM '17, pages 863–871, New York, NY, USA, 2017. ACM.

[113] B. Zhao, X. Li, and X. Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *2018 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7405–7414, 2018.

[114] B. Zhao, X. Li, and X. Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7405–7414, 2018.

[115] B. Zhao, X. Li, and X. Lu. Property-constrained dual learning for video summarization. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[116] B. Zhao, X. Li, and X. Lu. Property-constrained dual learning for video summarization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2019.

[117] K. Zhou and Y. Qiao. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *2018 AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[118] K. Zhou, T. Xiang, and A. Cavallaro. Video summarisation by classification with deep reinforcement learning. In *2018 British Machine Vision Conference (BMVC)*, 2018.