Research Article

Automatic TV Broadcast Structuring

Gaël Manson and Sid-Ahmed Berrani

Orange Labs, France Telecom, Cesson Sévigné 35510, France

Correspondence should be addressed to Gaël Manson, gael.manson@gmail.com

Received 2 October 2009; Accepted 11 January 2010

Academic Editor: Jungong Han

Copyright © 2010 G. Manson and S.-A. Berrani. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

TV broadcast structuring is needed to precisely extract long useful programs. These can be either archived as part of our audio-visual heritage or used to build added-value novel TV services like TVoD or Catch-up-TV. First, the problem of digital TV content structuring is positioned. Related work and existing solutions are deeply and carefully analyzed. This paper presents then DealTV, our fully automatic system. It is based on studying repeated sequences in the TV stream in order to segment it. Segments are then classified using an inductive logic programming-based technique that makes use of the temporal relationships between segments. Metadata are finally used to label and extract programs using simple overlapping-based criteria. Each processing step of DealTV has been separately evaluated in order to carefully analyze its impact on the final results. The system has been proven on a real TV stream to be very effective.

1. Introduction

Broadcasted digital TV contents have incredibly increased over the last few decades. The resulting huge and continuously growing content has given rise to many novel services around TV and video platforms like TV-on-Demand (TVoD), interactive TV, Catch-up-TV, Network Personal Video Records (NPVRs), and so forth. This content is also part of our audio-visual heritage and must be properly archived. Archiving digital TV content is generally achieved by national public institutions like INA in France, Beeld en Geluid in Netherlands, ORF in Austria or BBC archives in UK. Therefore, the digital TV content has to be analyzed and indexed in order to be used within services and to be easily retrieved from archives.

Basically, analyzing and indexing a digital TV content consists in finding key instants in the content. These correspond to events of interest (e.g., goals in soccer footage or key scenes in a movie) users may like to directly find through either search engines (in the case of querying an archive) or services built on top of the content. These key instants could also be features and positions that allow structuring the content. Here, the objective is twofold: (1) to properly prepare the content before archiving it in

order to easily answer user queries later on; (2) to repurpose the content in another more convenient format to final users.

In the case of TV structuring, the main key instants are the start and end times of each program in TV broadcasts. These times allow automatically recovering the structure of the TV stream. They are at the root of novel added-value services or any archiving service. They allow extracting programs and making them available through a catalog, without any constraint on time. They can therefore be viewed in a nonlinear manner through the aforementioned services. They also allow identifying, isolating and properly archiving *useful* programs that might interest users later on.

This paper focuses on this latter use-case of analyzing and indexing digital TV content coming from TV broadcasts. This is generally referred to as TV broadcast structuring or TV broadcast macro-segmentation. The main contribution of the paper is a novel and fully automatic system for TV broadcast structuring. This system aims at precisely extracting useful TV programs.

The rest of the paper is organized as follows. Section 2 explains why TV broadcast structuring is required for real-world applications, finely analyzes related works and classifies existing approaches into three categories. Section 3 presents DealTV, our fully automatic system that analyzes

the video signal, segments it, classifies the segments, and extracts and annotates programs. Section 4 describes the application we address: “TV program extraction for TVoD services”. This application is evaluated and validated on real TV broadcasts collected from a French channel over two weeks. In these experiments, each processing step of our system is separately evaluated. An evaluation of the full system is also provided and compared to a manually created ground-truth.

2. TV Broadcast Structuring and Related Works

In short, the objective of TV broadcast structuring is to recover the original structure of the TV stream. In TV streams, useful long TV programs (like movies, news, series, etc.), short programs (like weather forecast, very short games, etc.) and interprograms (commercials, trailers, sponsorships, etc.) are concatenated and broadcasted without any precise and reliable flags that identify their boundaries. Hence, TV broadcast structuring consists in automatically and accurately determining the boundaries (i.e., the start and end) of each broadcasted program and inter-program as depicted in Figure 1. In addition to precise and automatic boundary detection, TV broadcast structuring gathers different parts of the same program and labels them when metadata are available.

Structuring a TV broadcast can be seen, by many, as a problem that TV channels could solve. Theoretically, this is true. TV channels aggregate the audio-visual content and broadcast it. Hence, they should be able to provide appropriate and reliable metadata on what they broadcast, namely the title, the type, the start and end times and any additional data on each broadcasted program or inter-program. In practice, most TV channels are technically unable to provide such data. Their broadcasting chains are too complex and do not have the appropriate tools to save and, more importantly, forward these metadata. The few other channels that may provide accurate metadata do not necessarily accept to provide them. On the other hand, archiving and building novel services might be done by third parties without any collaboration with TV channels. This is the case of national public institutions in charge of archiving audio-visual heritage. It is also the case of NPVR services for which provided services may even be considered by channels as competing services.

Existing techniques for structuring a TV broadcast are classified into 3 categories. They are described in the following three sections.

2.1. Manual Approaches. As most video analysis and indexing problems, TV broadcast structuring can be manually performed by skilled-workers. In this case, the TV broadcast can either be structured online or offline. Online, workers have to continuously watch the TV broadcast. Each time an event of interest is encountered, it is tagged. Offline, workers linearly browse the saved TV stream and annotate it. Both cases require adequate software applications that allow workers to efficiently structure the stream.

These approaches are currently the most widely used, in particular for structuring and indexing audio-visual heritage before archiving it. It is however prohibitively expensive and unable to handle the currently huge amount of available content. For instance, manually and offline structuring a TV stream of 28 days has taken more than 30 working days using a very powerful and customized software. We have performed this manual annotation in order to create the ground-truth which is required for the evaluation of automatic TV structuring approaches (cf. Section 4). It also suffers the imprecision and errors that workers can make. Indeed, contrary to what one could assume, manual approaches are not always the most reliable. Structuring a TV broadcast is a laborious repetitive task that requires permanent concentration.

2.2. Metadata-Based Approaches. There are mainly two types of metadata that are provided by TV channels and that are used to describe TV broadcasts: (1) metadata that are associated and broadcasted with the TV stream, and (2) metadata that can be retrieved from specialized websites which gather electronic program guides. Both metadata provide information on programs only. Interprograms are not mentioned.

Metadata associated with the stream depend on standards and broadcasting modes (analog/digital). In the case of analog TV that is in the process of becoming defunct, metadata were available within teletext (or Closed Caption in US). Teletext encloses a large amount of data such as news, weather forecasts, and so forth, and also *static* information on the program schedule. European standard teletext could also include Program Delivery Control (PDC) [1]. PDC is a system that properly controls equipped video recorders by using hidden codes in the teletext service. An equivalent service named VPS (Video Programming System) exists in some EU countries (e.g., Czech Republic). These codes allow the user to precisely control the record start and end times of a specific program.

In digital TV, broadcasted metadata are called Event Information Tables (EIT) and are of two types:

- (1) EIT schedule: stores the TV program over a number of days
- (2) EIT present and follow: contains the details (start and end times, title, and possibly a summary) of the program currently being broadcasted, as well as the following one.

If EIT “present and follow” is generally available, the EIT “schedule” is rarely provided.

Apart from PDC, both for digital and analog TV, broadcasted metadata are static, that is, they are not updated or modified in order to take into account any delay or change that may occur in the broadcast with respect to the initial program schedule.

Unfortunately, PDC cannot be used for TV structuring for many reasons. The main problem is that PDC is very rarely provided, as it allows users to skip commercials. Currently, these represent the main income for TV channels.

FIGURE 1: TV broadcast structuring. Detection of program and interprogram boundaries.

Another limitation of PDC is that it is well defined and standardized for analog TV, but in digital TV, standards are not built in and most broadcasters are not transmitting the appropriate data.

Metadata available on the web are typically program schedules. These are called Electronic Program Guides (EPG). Many companies (like *emapmedia*) also provide a service in which they gather EPGs from a large number of channels. These EPGs are then made available on a unique server that can be directly queried through the web service.

In order to assess the reliability and the precision of these metadata, a study has been conducted in which the EIT and the EPG metadata have been compared to a manually created ground-truth. This study, presented in [2], shows that metadata are generally imprecise, do not cover all the broadcasted programs and do not take into account late modifications of the schedule. For instance, it shows that over a 24 hour broadcast, more than 40% of the programs start more than 5 minutes earlier or later than that expected in the metadata. Another study [3] has been performed on program guides from 5 channels over 3 years. It has also shown that more than 75% of broadcasted programs and interprograms are not mentioned in the program guides. Based on these results, metadata cannot be *directly* used to structure TV broadcasts.

On the other hand, apart from traditional techniques for metadata aggregation and fusion [4, 5] which could be used to enrich them and increase their accuracy, only very few studies (among which [6]) have proposed novel ways to make use of these metadata.

In [6], Poli proposes a statistical predictive approach that allows correcting an EPG using a model learned from a ground-truth created on a one year TV broadcast. This approach is based on a simple observation: channels have to follow roughly the same schedule in order to increase their audience. The main drawback to this approach is the required ground-truth data for training. This ground-truth is very difficult and prohibitively expensive to be collected. Moreover, it has to be separately collected for each channel as the program schedule differs from one channel to the other. Poli's study was feasible because it was conducted at INA, the French National Audiovisual Institute in charge of indexing and archiving French channels (<http://www.ina.fr>). On the other hand, the model does not take into account program schedule of special events that may occur without

any regularity from one year to the next (e.g. political events, sports competitions, etc.).

2.3. Content-Based Approaches. Content-based approaches rely on the study of the basic audio and video signals in order to recover the high-level structure of the stream. Basic techniques one can think of are shot and scene (or story) segmentation. Low level features like motion and color are extracted from each frame and are analyzed in order to segment the stream into shots. Shots are then gathered into scenes with respect to their similarities and temporal order [7–9]. Scenes are subjective and not very well defined but, for instance for a movie, they generally try to meet the main chapters of the movie like those prepared for the DVD. In the context of TV structuring, precisely extracting programs would require clustering all the scenes of the same program and separating them from those of the other programs. However, TV programs, and thus their scenes, are very heterogeneous and do not generally share any common features or structure. It is therefore very hard to make use of scene detection and clustering to perform TV stream structuring.

Another content-based approach would focus on detecting boundaries between programs and interprograms in the TV stream. This has been investigated by Wang et al. [10] and a multimodal boundary classification system has been proposed. The system uses visual, audio and textual features within an SVM classifier in order to find transitions between programs among all the possible transitions in the stream. However, this solution cannot be used to structure any TV stream. It heavily relies on some assumptions on the structure of programs (e.g., presence of specific images at the beginning and at the end of each program and interprogram) and on a complex training procedure.

Finally, a novel content-based approach makes use of the fact that some sequences are broadcasted several times in the stream and are then repeated sequences. These are commercials, trailers, credits of TV series, sponsorships... If the occurrences of these repeated sequences can be automatically detected and identified in the stream, then the stream can be segmented. The resulting segments can be then classified and analyzed toward performing the structuring. This content-based approach is the most promising. Techniques following this principle are classified into two categories and are described in the two following subsections.

2.3.1. Reference Database-Based Techniques. The basic idea of reference database-based techniques is to manually label repeated sequences and store them in a reference database. Here, labeled and stored repeated sequences should include most of the interprograms, opening and closing credits of recurring programs (like TV series) and possibly any other program shot. These sequences are identified later on in the TV stream using a content-based matching technique. TV structuring is therefore reduced to content-based real-time sequence identification in an audio-visual stream. The start (resp., end) time of recurring programs that have a stable opening (resp., closing) credit is detected when the credit is identified. Other parts of the stream are segmented by detecting interprograms. Each gap of a significant duration between two consecutive segments of interprograms is considered as a program segment and is labeled using metadata (like EIT) when these are available. A program segment can also be labeled, if one of its shots has been matched with a shot that has previously labeled and stored in the reference database.

Methods following this principle can be built on top of audio or video fingerprinting [11, 12] techniques. These can be used to detect in the TV stream, referenced sequences stored in the database. Perceptual hashing can also be used [13–15].

Naturel et al. [16] propose a complete system for TV structuring based on this principle. In addition to a hashing-based identification of stored and labeled shots, a dynamic time warping procedure has been proposed in order to *match* extracted program segments with metadata provided in the EPG. The set of interprograms is also updated using a commercial detection method based on the same features as in [17] (i.e., monochrome frames, silence, etc.).

These approaches mainly have two drawbacks, both related to the reference database. First, this database has to be created manually for each TV channel. It must also contain a sufficient amount of interprograms, credits and labeled shots in order to achieve a good and precise TV structuring. Second, the database has to be periodically updated as new interprograms, new series (and hence new credits) are continuously introduced.

2.3.2. Techniques Based on Automatic Detection of Repeated Sequences. Following the same principle as reference database-based approaches, other techniques rely on segmenting the stream by automatically detecting interprogram segments, deducing program segments and then labeling them using metadata. Unlike reference database-based approaches, these techniques make use of the repetition property of interprograms in order to directly and automatically detect them using a non-supervised solution.

Inspired by video retrieval techniques, Gauch and Shivadas [18, 19] propose a video shot-based solution. Shots are described and indexed using perceptual hashing. Repeated shots are then detected using a two step procedure. The first step is based on collisions in the hash table. The second one is based on the visual similarity between shots. Adjacent repeated shots are merged and classified (commercials or not). Covell et al. [20] propose an approach

following the same principle as Gauch et al., but technically different. Repeated objects are detected using audio features and a hashing-based method. Detections are then checked using visual features. As for Herley [21], interprograms are detected as *repeating objects* using a correlation study of audio features. At time t , the current *object* (an audio segment of predefined length) is compared to a past stored buffer of fixed size in order to detect any possible correlation.

Even if fully automatic, these techniques are not sufficient to perform TV structuring. They all require a post-processing step in which automatically detected repeated sequences need to be mined before being used in the structuring process. Indeed, repeated sequences also include sequences that are broadcasted several times but that are not interprograms. Examples of such sequences are news reports, flashback sequences in movies and series, and so forth.

On the other hand, these solutions are technically limited. Solutions proposed in [18–20] focus on the detection of commercials. They suffer the drawbacks of content-based matching techniques using hash-tables, which are mainly related to the difficulty of choosing a suitable hash function with respect to the target similarity. They are also *brute force* in the sense that all descriptors of the whole audio-visual stream have to be inserted in the hash table and also saved, which could raise efficiency problems when dealing with a large amount of audio-visual data or with a continuous TV broadcast in a real-world system. The method by Herley [21] requires some crucial parameters related to the size of descriptors, the search window and the length of the search buffer. These restrict the depth of the search and limit the detection to a *pre-defined fixed* size range of repeating objects.

Additionally to all these approaches, it is worth pointing out in this section that the TV structuring research field also includes works on commercial detection and program genre classification. Commercials have attracted a lot of attention because of their importance in the business model of TV broadcasting. They are still the main income of TV channels. Existing works on commercial detection rely generally on intrinsic features of commercials (e.g., motion, audio, action) and on detecting separations between commercials (monochrome frames and audio cuts) [17]. They also rely on detecting the logo channel and on studying the shot duration [22]. Many program genre detection techniques have also been proposed [23–25]. They generally classify programs into categories like news, commercials, cartoon, sport, TV series, weather, and so forth. They assume that programs have already been properly segmented and extracted.

3. DealTV: A fully Automatic System

In this section, we describe our novel fully automatic system for TV Broadcast structuring. It is based on the same principle as techniques based on the detection of repeated sequences. It addresses, however, the limitations of existing approaches and focuses on extracting long useful programs. We recall that useful programs are long TV programs, International Journal of Digital Multimedia Broadcasting 5 namely movies, news, TV series, TV shows, etc. These are the most important content of TV streams.

Our system uses two methods from our previous works for repeated sequence detection [26] and for program segment classification [27]. These methods are improved, adapted and put together with other techniques in order to efficiently and effectively structure continuous TV broadcasts.

When launching the system for the first time, it needs to accumulate a sufficient amount of stream. The analysis of the stream starts when there are enough repeated sequences in the accumulated stream. As will be shown in the experiments (Section 4.1), 96 hours is sufficient. When the analysis is started, it structures the previously accumulated stream and delivers results. The system then starts accumulating the stream gain. However, this time, the structuring process can be launched anytime on demand or periodically. In this case, if the duration of the newly accumulated stream is not sufficient, it is merged with the previous period of the stream. This processing scheme is depicted in Figure 2.

Each time the stream structuring is launched on a portion of the stream (periodically or on demand), the following processing steps are performed (as depicted in Figure 3):

- (1) stream segmentation using detected repeated sequences,
- (2) resulting segments classification in order to detect useful program segments,
- (3) useful programs extraction and labeling.

First, repeated sequences detection uses a micro-clustering technique that does not make any assumptions on the length or frequency of the repeated sequences. Moreover, this detection can be performed whatever the length of the TV stream is. Our system also covers all of the steps from the first step of repeated sequence detection to the final step of program extraction. Detected repeated sequences are used to segment the stream and the resulting segments are then classified in order to isolate segments that belong to useful programs from the rest of the segments (inter-program and short program segments). This classification step is based on inductive logical programming. Program segments are finally labeled using a matching procedure with respect to metadata.

For the sake of simplicity, in the following we consider that we have accumulated a portion of the stream and we describe each step of the structuring process given that portion.

3.1. Stream Segmentation Using Detected Repeated Sequences. Stream segmentation is based on detecting repeated sequences in the stream, that is, sequences that are broadcasted several times. These include (but are not limited to) interprograms, whole programs and parts of programs that are broadcasted several times. Our repeated sequence detection relies on extracting and clustering visual features.

3.1.1. Stream Description. Our repeated sequence detection technique uses a two level visual description scheme. A first level at which an exhaustive description is performed.

A basic visual descriptor (BVD) is extracted from each frame of the video stream. It is used to match almost identical frames and only needs to be invariant to small variations due to compression, for instance. The second level focuses on carefully chosen keyframes of the video stream. The descriptor associated with these keyframes is called key visual descriptor (KVD). It is more sophisticated and has to be more robust. KVDs are used during the clustering step to cluster similar shots and to create the set of repeated sequences. However, at this stage, the boundaries of detected repeated sequences cannot be determined. A KVD is associated with a frame of the repeated sequence but does not provide any information on the sequence boundary. The BVD is thus used to precisely determine these boundaries by matching corresponding frames in all occurrences of the repeated sequence.

Both BVDs and KVDs are DCT-based descriptors. To compute a BVD, the frame is divided into 4 blocks and each block is sub-sampled to a matrix of 8×8 . A DCT is then applied on each block and one DC coefficient and the 15 first AC coefficients (according to the zig-zag order) are computed. Each coefficient is then binarized and a 64-bit descriptor is created. BVDs are compared using the Hamming distance.

As for KVDs, they are computed from keyframes. These are chosen after a shot segmentation of the video stream following the method described in [26]. To make the KVD robust to spatial variations, like subtitles incursion or logo insertion/removal, the keyframe is divided into 3×2 blocks. Six independent descriptors are computed on the six blocks and then concatenated into a single descriptor. To compute block descriptors, each block is first subsampled to a 8×8 matrix. A DCT is computed on this matrix and the first five coefficients (according to the zig-zag order) are selected to build the descriptor. The KVD is hence a 30-dimensional vector. The similarity between KVDs is measured using the L_2 distance.

3.1.2. Clustering Step. To gather similar keyframes that will be used to detect repeated sequences, a clustering technique is used. However, unlike most applications using clustering, we are interested in finding a large number of very small clusters within a huge amount of uniformly distributed and isolated vectors. The number of KVDs per cluster is determined by the number of times a sequence is repeated. If a sequence is repeated three times, and it is described by five KVDs (i.e., five keyframes have been selected from the sequence), then we should ideally discover five clusters with three KVDs inside each one. The number of KVDs per cluster ranges thus from two to few hundreds. The number of clusters corresponds to the number of KVDs in the repeated sequences. As for the rate of outliers, it corresponds to the rate of KVDs that do not belong to any repeated sequence, which could be very high. On the other hand, the clustering algorithm should also be able to process KVDs on the fly as they are computed from the video stream. Based on these criteria, we propose to use a micro-clustering technique similar to BIRCH [28].

FIGURE 2: Processing scheme of the TV broadcast by DealTV. (a) Periodical structuring. (b) On demand structuring.

FIGURE 3: Structuring steps of DealTV.

It is an iterative procedure that builds spherical clusters whose radii are controlled and must be below a threshold Ft . At the beginning, Ft is chosen to a very low value. During the first iteration, KVDs are inserted: a KVD is associated with a cluster of previously inserted KVDs only if the radius of the resulting bounding hyper-sphere of the cluster is less than Ft_1 (the value of Ft in the first iteration). If no existing cluster can *absorb* the KVD, it is put in a new cluster as a singleton (a cluster that contains only one KVD and of which the radius is equal to 0). To characterize clusters and facilitate their use, a clustering feature (CF) vector is associated to each cluster. A CF vector is a triple composed of the number and the sum of the KVDs belonging to the cluster, and the radius of its bounding hyper-sphere. When merging two clusters, the CF vector of the resulting cluster can be easily computed using only the CF vectors of the two clusters. That way, KVDs are read only once. In the following iterations, only CF vectors are manipulated.

After each iteration, the singletons are isolated as outliers. If the number of the remaining clusters falls below a fixed number then the clustering process is finished. Otherwise, the Ft is increased and a new iteration is performed.

As explained previously, the maximum number of clusters is related to the number of repeated sequences and the number of associated keyframes. It can, therefore, be experimentally determined by a study over a sample of TV broadcast.

On the other hand, in order to efficiently cope with the general periodical working scheme of DealTV, the clustering procedure should be incremental. This is important to reduce the cost of this crucial and costly processing step. In our system, this is taken into account at 2 levels. First, the CF vectors and the way the clustering technique works allows processing KVDs as they are computed. Second, when a new portion of the stream is accumulated and processed (P9 in Figure 4), the equivalent oldest portion in the clustering buffer is removed (P2 in Figure 4). The clustering process works hence as a continuously updated sliding window. This is allowed by the powerful way CF vectors are computed and manipulated. This way, when the structuring analysis is launched, the first (and the most costly) iteration of the clustering is already performed. Figure 4 depicts how the incremental clustering works.

3.1.3. Repeated Sequences Detection. Once the clusters are generated, they are analyzed in order to create repeated sequences. First, clusters that contain KVDs extracted from the same shot or from neighboring shots are removed. For example, this happens during a debate when shots are alternatively centered around the antagonists.

To make use of the rest of clusters, an inter-cluster similarity is defined. This similarity measures the chance for two clusters to generate a repeated sequence. To explain how it is computed, let's consider two clusters C_i and

FIGURE 4: Working scheme of incremental clustering.

FIGURE 5: Two interlaced clusters with n KVDs each.

C_j , containing n KVDs each. The set $\{I_1, \dots, I_n\}$ (resp., $\{J_1, \dots, J_n\}$) is the set of keyframes whose KVDs belong to C_i (resp. C_j). The first condition for two clusters to generate a repeated sequence concerns the temporal order of keyframes $\{I_k\}_k$ and $\{J_k\}_k$. They have to be alternating as depicted in Figure 5. In this case, we say that C_i and C_j are interlaced.

The temporal distance between each couple of keyframes from the two clusters is the second condition. Temporal distances between each couple (I_k, J_k) must be nearly the same if the $\{I_k\}_k$ and the $\{J_k\}_k$ are keyframes of a repeated sequence. The chance for C_i and C_j to generate a repeated sequence is thus related to the constancy of the temporal distance ($Tdist$) between their keyframes. The inter-cluster similarity must be one if all the temporal distances are identical and decreases when they differ. The following formula is used to compute the inter-cluster similarity between clusters C_i and C_j :

$$\text{Sim}_{ij} = \begin{cases} 0, & \text{if } C_i \text{ and } C_j \text{ are not interlaced,} \\ e^{-\sigma}, & \text{otherwise,} \end{cases} \quad (1)$$

where σ is the standard deviation of $Tdist_1, \dots, Tdist_n$ (cf. Figure 5).

The inter-cluster similarity is computed between each couple of clusters. The results are stored in a matrix B , where element b_{ij} is the similarity between clusters C_i and C_j (i.e., $b_{ij} = \text{Sim}_{ij}$). To process clusters, we define a basic relation between clusters as follows: if $b_{ij} > \delta_{\text{sim}}$ then clusters C_i and C_j are related. This relation is aimed at gathering clusters that have a high inter-cluster similarity and, hence, that are likely to generate a repeated sequence.

The rest of the process is summarized in the following steps.

FIGURE 6: Generation of a three times repeated sequence from two clusters.

- (1) Select clusters that have at least one relation with another cluster.
- (2) Select the set of most populated clusters, that is, clusters having the highest number of KVDs.
- (3) Perform transitive closure within the selected set. Again, the objective is to partition the set into subsets into which each cluster is related to one or more other clusters from the same subset.
- (4) Generate a repeated sequence from each subset as depicted in Figure 6 and remove the used clusters from B .
- (5) Continue with the process from step (1) until no cluster is selected.

In step (2), the process starts with the most populated clusters in order to first retrieve the most frequent repeated sequences. Indeed, as depicted in Figure 5, for a chosen subset of clusters, the number of occurrences of a repeated sequence is equal to the number of KVDs per cluster (we recall that all the clusters within the subset are interlaced and have the same number of KVDs each).

In step (4), the generation of repeated sequences consists in defining the boundaries of each occurrence of the repeated sequence. First, the boundaries are defined by the most left and the most right keyframes. These boundaries are then extended using BVDs computed for all the frames of the stream. The occurrences are extended to the left (resp., the right) if BVDs of all the left (resp., right) neighboring frames for all the occurrences are similar. To make the extension procedure more robust, we propose to simultaneously compare a set of neighboring frames, that is, the extension procedure compares the m left (resp., right) frames of all the occurrences every time. If the average dissimilarity is less than a threshold, then the occurrences are extended to the left (resp. right) by m frames.

The repeated sequence detection procedure is also able to retrieve trailers as repeated sequences and can sometimes match them with the corresponding program. This is not described here. The interested reader can refer to [26] for a complete and detailed description of the procedure.

3.1.4. Stream Segmentation. Finally, the stream is segmented as depicted in Figure 7. First, each occurrence of a repeated sequence is considered as a segment. Then, each *gap* between two consecutive segments is also considered as a segment.

FIGURE 7: Stream segmentation using detected repeated sequences. Eleven segments generated.

FIGURE 8: Illustration of adjacent segments to the occurrences of a repeated sequence.

3.2. Segment Classification Using ILP. Once the stream is segmented, the problem is to automatically detect segments that are part of long useful programs. It is a classification task with two classes: (1) the class of long program segments and (2) the class of the other segments that include segments of interprograms and segments of short programs.

To perform this task, local features of each segment (like the duration) can be used. However, the linear nature of the stream and the way long programs, short programs and interprograms are sequenced within the stream provide powerful features that greatly help distinguishing these segments. In the following, these features are referred to as neighborhood and relational features.

In order to take into account both kind of features (local and relational), our classification module uses Inductive Logical Programming (ILP) following the method described in [27]. ILP allows us to train offline a classifier that implicitly models the relational features and to easily take into account prior knowledge. Moreover, the resulting classification rules are easily understandable.

The features and the ILP classification module are described in the following two subsections.

3.2.1. Segment Features. Three kinds of features are used to characterize a segment: local, contextual and relational features.

(1) *Local Features.* These features describe a segment with features that do not depend on others segments. We use only the duration of the segment and the number of times it repeats in the stream (0 if the segment does not repeat).

(2) *Contextual Features.* These features take into account the context of the segment. We define the following features.

(i) If the segment is an occurrence of a repeated sequence, we compute the mean of the number of repetitions of the following segments adjacent to each occurrence. This is illustrated in Figure 8. This feature applies to occurrences $\{A_1, A_2\}$ and is equal to $(|B| + |C|)/2$, where $|X|$ denotes the number of occurrences of the segment X . In the same manner, we compute the mean of the number of repetitions of the *previous* segments that are adjacent to each occurrence (segments D and E in Figure 8). We propose this feature in order to help discriminating opening/closing credits from the set of repeated sequences. Indeed, when this feature is null, that means that the segment always lies before/after a long segment that does not repeat and that is most likely a long program. This feature may also help to classify segments that always lie between other very frequent repeated segments.

(ii) As contextual features, we also consider the local and contextual features of an adjacent segment: segments in the neighborhood of the considered segment.

(3) *Relational Features.* These features are the class of the neighboring segments. Therefore, they apply only when at least one of the neighboring segments has already been classified. They allow to take into account the class of neighboring segments.

3.2.2. ILP Classification. ILP can directly manage complex logical relationships between segments and returns explicit rules in the form of first order logic. Prior knowledge can also be easily taken into account. They just have to be encoded as first order logical rules and added to the background knowledge. However, as ILP does not handle numerical data, all the local and relational features have to be transformed to symbolic attributes. Categories for symbolic attributes are defined using numerical intervals based on prior knowledge.

An ILP system builds a *logical program* from the background knowledge and a set of training examples represented as a set of logical facts. This logical program is composed of a set of first order logical rules that cover all the positive and none of the negative training examples. ILP infers rules from examples by using computational logic as the representation mechanism for hypotheses and examples. An example of logical rule that can be learned is “If a segment A does not repeat and A is long then A is a long program segment” or “If a segment A repeats often and A is followed by a segment B that is not a long program segment then A is not a long program segment”.

The neighboring relationships are hence represented in the training set by a set of facts that gives the following segment for each segment of the stream. They are also represented by a recursive rule that transitively defines this relation, which allows to define a “distance” between segments.

In our implementation, we have used *Aleph* (*ex-P-Pradol*), a descendant ILP system which performs training from general to specific hypotheses [29, 30].

The logical rules computed by ILP define requirements for a segment A to belong to the class of “long programs” or the class of “others” (IP or short programs). They can be sorted into four categories following how they model segment features.

- (1) *Simple not recursive rules (SNR-rules)* rely only on the local and contextual features of A.
- (2) *Simple recursive rules (SR-rules)* rely in addition to (1) on the fact that some neighboring segments belong to the class of A.
- (3) *Relational not recursive rules (RNR-rules)* rely in addition to (1) on the fact that some neighboring segments belong to a class distinct of the class of A.
- (4) *Relational recursive rules (RR-rules)* rely in addition to (3) on the fact that some neighboring segments belong to the class of A.

In order to compute logical rules that define “long programs” or “others”, we first encode a part of the TV stream as a database of logical facts. This is the training set. The ILP system infers then a set of logical rules. Some of these rules are generic and very relevant. However, some other inferred rules are very specific to special cases of the training set and may confuse the classifier. Thus, in order to select the relevant rules, we propose to use an additional validation phase. The learned rules are applied on the validation set and depending on their precision, a confidence level is associated with each of them. The higher the precision, the higher the confidence level. Details on the number of considered confidence levels are given in Section 4.4.

The training phase provides hence a set of rules (SNR, SR, RNR, RR) ordered by their levels of confidence (the highest level of confidence = 0). The classification step takes into account the confidence levels and the types of rules. Prior knowledge rules are considered as the most reliable. They are applied first and at the beginning of each iteration. The classification phase consists in the following procedure:

- (1) apply prior knowledge rules,
- (2) select a subset E_i (initially $i = 0$) of the rules with the level of confidence i ,
- (3) select and apply SNR-rules for the class “long programs” from E_i ,
- (4) select and apply recursively SR-rules for “long programs” from E_i ,
- (5) do (3) and (4) for the class “others”.
- (6) select and apply RNR-rules for “long programs” from E_i ,
- (7) select and apply recursively RR-rules for “long programs” from E_i ,
- (8) do (6) and (7) for “others”.
- (9) select the next level of confidence: $i = i + 1$ and continue with step (1).

3.3. Program Extraction and Labeling. The ILP classification step detects and isolates segments that are parts of long programs from the whole step of segments. When no metadata is available, no further processing steps can be achieved. Segments can only be presented to a user to be manually annotated. However, at least the EPG is generally available for most channels. In this case, despite their imprecision, metadata are very helpful to fuse, extract and label long programs. Metadata only provide approximate start and end times of the broadcasted programs. These starts and ends are used to build the metadata segments that are analyzed and compared to the extracted segments in order to perform program extraction and labeling.

An interesting approach for TV segment labeling is presented in [16]. The authors have considered labeling as a problem of sequence alignment between the detected program segments and the metadata segments. A Dynamic Time Warping (DTW) algorithm has been used to fuse the segments. It uses the edit distance between the set of program segments and the set of metadata segments. The edit distance is a well-known method for aligning two sequences X and Y . It evaluates minimum weight for transforming X into Y by a set of weighted edit operations. The used operations are here defined as substitution, insertion and deletion. In order to drastically improve the results, the authors use a landmarked DTW to force local alignment. This forced alignment is based on previously manually labeled segments that are recognized.

In general, the labeling step is not a straightforward alignment issue. Indeed, TV programs may be cut into several parts that are separated by interprograms (commercials in particular) and this is not mentioned in the metadata. The basic substitution, insertion and deletion operations are not sufficient to deal with this. They are not able to fuse different parts of the same program.

In our system, the labeling procedure is depicted in Figure 9.

- (1) Segments classified as long programs (P) are selected.
- (2) Consecutive long program segments are gathered into a single long program segment.
- (3) Resulting long program segments are labeled using metadata segments following a temporal overlapping criteria.
- (4) Consecutive long program segments labeled with the same label are fused into a single TV program.

The labeling procedure is based on studying the temporal overlapping between the detected program segments and the metadata segments. For each detected long program segment, metadata segments that have a non empty temporal intersection with the long program segment are selected and for each one, the temporal overlapping rate is computed. The metadata segment with the highest overlap is selected if its overlapping rate is significantly greater than the one of the second most overlapping segment. If metadata segments having the highest overlaps present overlapping rates that are very close, the segment whose duration is the closest to the segment to label is selected.

FIGURE 9: Program extraction and labeling.

This labeling procedure is a local approach that does not heavily rely on the metadata. It is also able to extract the start and the end of each program while keeping information on the location of interprograms separating the different parts of the programs. This is very important to remove/replace commercials in services like TVoD.

4. Experiments

In this section, we evaluate and validate the DealTV system for automatic TV structuring. For this purpose, we have performed a set of experiments using real TV broadcasts. First, we present the real TV broadcast-based dataset that we used. Next, we evaluate each step that makes up DealTV namely, the repeated sequence detection, the TV stream segmentation, the ILP-based classification for program segment detection and finally, the TV program extraction.

The main and more important experiment is the last one. It allows us to validate DealTV for TV program extraction, our ultimate goal. The other experiments are presented in order to evaluate each processing step of DealTV and to understand its impact on the overall system performance.

All the algorithms have been developed in C++. Experiments have been performed on a PC under Windows XP. Its CPU is a 2 GHz Intel Xeon, with 3 GB of main memory.

4.1. Dataset and Ground-Truth. The dataset we have used is a real TV broadcast collected from a French channel over two weeks. It is called *TVData* in the following sections. We have selected one week for training. It is called *TVTrain*. The other week has been used for testing. It is called *TVTest*.

The dataset being collected from a French TV channel that is regulated by the European Union legislation, the duration and the frequency of commercial breaks are limited. We have measured about 15 hours and 22 minutes of interprograms over the 7 days of *TVTrain*. Table 1 presents the different categories of segment that compose *TVTrain*. The last category “other interprograms” gathers all the other

TABLE 1: Composition of *TVTrain*.

Long programs	87.63%
Commercials	5.02%
Trailers	3.16%
Short programs	2.99%
Sponsorships	0.54%
Jingle	0.33%
Logos	0.12%
Other interprograms	0.21%

TABLE 2: Day intervals.

Night	from 1 am to 8 am
Morning	from 8 am to 12 am
Midday	from 12 am to 2 pm
Afternoon	from 2 pm to 6 pm
Early evening	from 6 pm to 8 pm
Evening	from 8 pm to 1 am

small categories of interprograms and includes short channel games, single music clips, and so forth.

In order to present detailed results, we have chosen to partition the day into six intervals. These are defined in Table 2. They follow the structure of a TV guide.

In our evaluation, we were unfortunately unable to compare our results to related works. To the best of our knowledge, there is no evaluation campaign for TV structuring and there is no available international corpus that can be used for this purpose. The TREC Video Retrieval Evaluation (TRECVID) only provides a corpus of already segmented TV programs. It does not contain any continuously recorded TV broadcast over several days.

In order to evaluate our solution, we have manually segmented and labeled TV Data. This has provided a ground-truth that has been used to compute evaluation metrics such as precision and recall. This ground-truth has been also used

FIGURE 10: Proportion of repeated segments by categories in *TVTrain* with respect to the accumulated hours of TV stream.

to study the repetition rate of interprograms in *TVTrain*. It has also provided useful information on the structure of the stream. In particular, a set of 374 groups of repeated segments has been discovered with a total number of 2782 occurrences (repeated segments) in *TVTrain*. The most frequent segment is a sponsorship that has been broadcasted 34 times.

Within *TVTrain*, we have also focused on the three most important inter-program categories (i.e., commercials, sponsorships and trailers) and on short programs. We have then computed the proportion of commercials, sponsorships, trailers and short programs that repeat with respect to the accumulated TV stream. This is shown in Figure 10.

This figure shows two main points. First, more than 90% of commercials, sponsorships and trailers are broadcasted at least twice within 4 days. Moreover, this rate does not increase anymore. The remaining 10% of these interprograms do not repeat. They can thus only be detected using the neighborhood and the ILP classification. Nevertheless, this result validates our main idea behind our solution regarding the repetition property of interprograms.

The second point is that only about 30% of short programs are repeated. This implies that detecting short program segments heavily relies on the ILP-classification step. This also lets us suppose that detecting short programs is the key for an accurate TV program extraction.

As explained previously, our system performs a periodical analysis of the TV stream. For each period, the system computes a clustering of the accumulated TV stream in order to detect repeated sequences and to perform segmentation. The results of this first analysis on the content of *TVTrain*, allow us to choose the size of the required accumulated TV stream. With a background of 7 days, we ensure that most of interprograms are within the set of repeated segments. The period is therefore fixed to 7 days.

It is important to note that *TVTest* has been recorded one week after the end of *TVTrain* as depicted in Figure 11. This is required because the clustering step uses an accumulated

FIGURE 11: Positions of *TVTest* and *TVTrain* within the TV stream.

stream of 7 days for each processed day of *TVTest*. The week *TVTmp* that is between *TVTrain* and *TVTest* has thus also been recorded. This way, the stream used for training is completely separated from the stream used to testing.

4.2. Repeated Sequence Detection. The analysis of the *TVTrain* dataset shows that most of interprograms are broadcasted several times. In this section, we present experiments that evaluate the ability of our system to automatically detect repeated sequences in the stream.

This evaluation has been conducted separately on *TVTrain* and on *TVTest* in order to show also the stability of the repeated sequence detection. Precision and recall have been used as evaluation metrics.

4.2.1. Repeated Sequence Detection on *TVTrain*. Repeated sequence detection has been applied on *TVTrain*. A set of 775 repeated sequences has been discovered with a total number of 3718 occurrences. This is higher than the number of manually gathered repeated segments (i.e., 2782). The most frequent detected repeated sequence is also a sponsorship that is repeated 30 times. It is not the same sponsorship that has been manually annotated and that repeats 34 times.

This does not mean that our repeated sequence detection technique does not perform well. Due to the position of the keyframes, many repeated sequences have been divided into several repeated subsequences. Therefore, in order to evaluate the automatically detected repeated sequences, we have computed the recall R_{seqret} on a per shot basis. The number of shots that belong both to detected repeated sequences and to manually gathered repeated segments have been calculated. The recall R_{seqret} is then the proportion of this number with respect to the total number of shots of the manually gathered repeated segments. Table 3 shows the obtained results. We have also focused on the three main inter-program categories and on short programs. The results show that repeated segments are very well detected. We can notice the sponsorships are less detected than commercials. This can be explained as follows. In order to be detected, repeated segments must contain two keyframes that have to be gathered into similar clusters. However, sponsorships are often made up of only one shot with only one keyframe.

As for short programs, missed repeated sequences are due to only one short TV game. In this TV game, only a few portions of the segments are changing. There are too many versions of this short TV game that share too many features, which confuses our system.

We have also measured the precision on a per shot basis of the detected repeated sequences with respect to all the manually segmented interprograms and short programs. We call this precision P_{seqret} . The computed precision is

TABLE 3: Recall R_{seqret} of repeated segments per category in *TVTrain* on a per shot basis.

Commercials	98.55%
Trailers	97.85%
Sponsorships	92.54%
Short programs	92.38%

FIGURE 12: Variation of the number of detected repeated sequences.

64.54%. In other words, about 35% of the detected repeated sequences are repeated segments that are parts of long programs.

To evaluate the precision with respect to the repeated sequences, we have selected the 50 most repeated sequences and we have randomly chosen 50 other sequences. Then, we manually evaluate the results. The observed precision is equal to one. This means that all detected repeated sequences have been effectively repeated sequences.

These experiments allow us to conclude that our repeated sequence detection system is very reliable. However, an efficient classification step is required (as expected) in order to filter out the 35% of repeated sequences that are parts of long programs.

4.2.2. Repeated Sequence Retrieval over *TVTest*. Repeated sequence detection has been applied on each day of *TVTest*. We have assumed that at the end of each day, an on-demand analysis is launched. Each day is then processed with a sliding accumulated TV stream of 7 days (the six previous days + the analyzed day). We recall that for this reason, the TV stream of the week before *TVTest* has also been recorded. Table 4 summarizes the set of repeated sequences that has been detected with their total number of occurrences. Figure 12 shows how the number of detected repeated sequences varies on the days of *TVTest*. Both the figure and the table show that the system is stable. We can then suppose that we have a recall and a precision similar to those computed on *TVTrain*.

We have also studied the processing time of the repeated sequence detection step. Indeed, our system has to accumulate the TV stream and to periodically or on-demand

FIGURE 13: Processing time of repeated sequence detection on each day of *TVTest*. For each day, a sliding accumulated TV stream of 7 days has been taken into account.

analyze the accumulated TV stream. The analysis have thus to be processed before the next analysis is launched. Figure 13 shows the variation of the processing time for each day of *TVTest*. We recall that each day is processed within a sliding accumulated TV stream of 7 days. However, the clustering is not performed from scratch each time. It is updated as explained in Section 3.1. Figure 13 shows that the processing time is quite the same and is always less than 1 hour and 40 minutes. This suggests that the analysis can be launched every two hours which is sufficient in a real world service.

4.3. TV Stream Segmentation. In order to evaluate the segmentation that is performed based on the detected repeated sequences, we have focused on TV programs' boundaries. Indeed, this segmentation provides potential TV program boundaries. Its *quality* depends on the ability to provide boundaries that match those of TV programs given in the ground-truth.

To evaluate the alignment between the computed segmentation and the ground-truth program boundaries, we define specific precision and recall metrics that are tolerant to small imprecisions. The boundaries do not requires an accuracy at the frame level. In practice, a program that is extracted with the sponsorship sequence or without its opening/closing credit is still considered as correctly extracted. Hence, the tolerated imprecision corresponds to the average duration of a sponsorship or a credit; it is equal to 30 seconds.

For each ground-truth boundary, its nearest detected boundary is then found and the temporal distance is computed. The evaluation is achieved using the 3 following metrics.

- (i) The *precision* P_{bound} is the number of detected boundaries whose distances to their nearest ground-truth boundaries are less than 30 seconds divided by the total number of detected boundaries.

TABLE 4: Detected repeated sequences on each day of *TVTest*. For each day, a sliding accumulated TV stream of 7 days has been taken into account.

Days	Repeated Sequences	Number of occurrences	Most frequent sequence
1	849	4026	Commercial (34 times)
2	831	3887	Commercial (32 times)
3	823	3847	Sponsorship (35 times)
4	804	3851	Sponsorship (33 times)
5	815	3936	Sponsorship (35 times)
6	814	3940	Trailer (38 times)
7	813	3894	Trailer (41 times)

TABLE 5: Evaluation of the TV stream segmentation on *TVTest*.

Day interval	Precision P_{bound}	Recall R_{bound}	Imprecision I_{bound}
Night	0.37	0.94	1.6 s
Morning	0.35	0.95	2.9 s
Midday	0.47	1	0.8 s
Afternoon	0.46	1	1.5 s
Early evening	0.37	0.98	2.3 s
Evening	0.29	0.99	2.3 s

- (ii) The recall R_{bound} is the number of ground-truth program boundaries whose distances to their nearest detected boundaries are less than 30 seconds divided by the total number of ground-truth program boundaries.
- (iii) The imprecision I_{bound} is the mean of the absolute temporal distance between the ground truth program boundaries and their nearest detected boundaries.

These metrics have been evaluated on each day of *TVTest*. The obtained results are presented in Table 5. They are averaged and separately presented per day interval.

The obtained results show that the system has performed a good TV stream segmentation. In particular, for the midday and the afternoon intervals, all the ground-truth boundaries are correctly retrieved. Moreover, the average distance between a ground-truth boundary and its detected boundary match is always less than 3 seconds. This is very important for the TV program extraction. However, our system has failed to detect on average about 5% of boundaries from the night and the morning intervals. This is due to the fact that less interprograms and more specifically commercials are broadcasted during these intervals. In particular during the night, many long programs are sequenced without any separating inter-program. These results suggest also that TV program extraction is likely to be more accurate from midday to the evening.

4.4. ILP-Based Classification. In this experiment, the performance of segment classification using the ILP-based technique is studied. We recall that the problem here is to automatically detect segments that are part of long useful programs. It is a classification task with two classes: the class of long program segments and the class of the other segments that include segments of interprograms and short programs.

As we are interested at the end of the process on extracting long useful programs, in this study we focus on detecting long program segments.

To train our ILP classifier, we divided *TVTrain* into two. The first part contains 5 days and has been used for learning the logical rules. The second part of the 2 remaining days have been used for validation. As explained earlier, the validation aims at evaluating the effectiveness of the inferred rules. We have defined 4 levels of confidence and we have used only the three highest levels during the classification phase. Rules with the lowest level of confidence have been discarded. We have also defined numeric intervals for the symbolic attributes required by ILP. For example, we have partitioned the duration domain into the following intervals:]0, 2.5 s[, [2.5 s, 7.5 s[, [7.5 s, 12.5 s[, [12.5 s, 17.5 s[, [17.5 s, 22.5 s[, [22.5 s, 27.5 s[, [27.5 s, 32.5 s[, [32.5 s, 37.5 s[, [37.5 s, 42.5 s[, [42.5 s, 75 s[, [75 s, 1 m 45 s[, [1 m 45 s, 2 m 45 s[, [2 m 45 s, 7 m 30 s[, [7 m 30 s, ∞ [. We have chosen these intervals because they are centered around multiples of 5 that characterize the interprograms.

This training phase has created a set of 333 rules: 70 rules associated with the highest level of confidence, 96 rules with the second level of confidence, 16 rules with the third level of confidence and 151 rules with the lowest level of confidence. We have added one prior knowledge rule that states that “*if a segment A lasts more than 5 minutes, then A is a long program segment*”.

Precision and recall measures have been used to evaluate the results. They have been computed on a duration basis. The precision P_{class} is here the total duration of segments classified as long program segments that are effectively parts of long programs in the ground-truth, divided by the total duration of segments classified as long program segments. The recall R_{class} is in the same way the total duration of segments classified as long program segments that are effectively parts of long programs, divided by the total duration of long programs in the ground-truth.

The results contain also the total number of computed segments for each processed day, the total number of segments classified as long program segments and the total number of long program segments in the ground-truth.

Table 6 summarizes the obtained results. Both the precision and the recall are expressed in percentages; they are both very high.

In order to put into perspective the obtained results, we have calculated the score that a naive solution could

TABLE 6: Evaluation of the ILP-based technique in detecting long program segments on *TVTest*.

Days	1	2	3	4	5	6	7
Precision P_{class}	98.99	98.86	97.57	98.80	99.20	97.62	99.14
Recall R_{class}	97.69	96.58	95.43	94.60	95.57	97.57	98.22
No. of computed segments	674	681	727	712	739	602	500
No. of detected long program segments	136	121	143	134	154	131	99
No. of ground-truth program segments	40	30	42	39	36	42	35

obtained. The naive solution consists of applying a simple classification rule that classifies all the segments as long program segments. We have measured an average precision P_{class} of 88.09% with a recall R_{class} obviously equals to 100%. This naive solution classifies as long programs on average, each day, about 2 hours and 53 minutes of interprograms or short programs. However, our system reaches a precision of 98.59%. It is wrongly classified as long program segments only about 17 minutes each day.

In order to understand how these 17 minutes impact on the accuracy of extracting long programs, it is important to precisely study where they are located. This is presented in the next and last experiment.

We can also notice from Table 6 that the number of detected program segments is very high with respect to the number of actual program segments in the ground-truth. This is due to an over segmentation that is easily dealt with using a merge procedure. This procedure properly fuses consecutive program segments that belong to the same program during the labeling step.

4.5. TV Program Extraction. Finally, we present the experiment that evaluates the final step of our system, that is program extraction. We have conducted this experiment through a TVoD application. TVoD is a novel service that fulfills the needs of the users to make use of the huge and continuously growing audio-visual content without any constraint on time. It makes watching previously broadcasted TV programs possible at anytime and anywhere. In order to make TV programs available, they must firstly be automatically extracted and stored in a catalog. We evaluate here this automatic TV program extraction.

Despite the very good performance of previous processing steps, the labeling of detected long program segments and the extraction of long programs have to cope with many issues that have been revealed by the evaluation experiments of the previous steps. Not all inter-program segments and short program segments are isolated by the previous processing steps. Some of them could have been misclassified and then considered as long program segments. Moreover, long programs could have been over segmented.

TV program extraction also depends on the metadata information provided by TV channels. These metadata are required to be able to give names to the automatically extracted long program segments. To handle the most complete and accurate metadata, we have chosen to merge the EIT with the EPG. Based on the results in [2], EIT are more reliable than EPG. Therefore, we have completed

the EIT with the EPG when the EIT informations were not available.

This experiment evaluates the quality of the final extracted programs. It evaluates the effectiveness of our labeling and program extraction techniques and also its ability to deal with the limitations of the previous steps and the imprecision of metadata.

In order to evaluate the quality of extracted long programs, we have first counted the number of programs in the ground-truth, the number of programs mentioned in the metadata and the number of extracted programs by DealTV. We have then counted the number of programs from metadata that are effectively in the ground-truth and the number of extracted programs that are also effectively in the ground-truth. It is worth mentioning that the number of extracted programs is always less or equal to the number of programs in the metadata. This is due to the fact that metadata is used to label automatically extracted programs and that for the evaluation, we keep only labeled programs. The obtained results on *TVTest* are summarized in Table 7. They are presented per day interval.

The accuracy of extracted programs has also been evaluated. The start (resp. end) of each extracted program has been compared to the actual start (resp. end) given by the ground-truth. The temporal difference is summed up and averaged for all the extracted programs. It is referred to as the imprecision. This imprecision has also been computed for metadata programs. The obtained results on *TVTest* are presented in Table 8.

From Tables 7 and 8, we can notice that DealTV greatly outperforms the metadata and provides very good results. In particular, the obtained results for the intervals of the midday and the evening are very accurate.

In order to further analyze these results with respect to a TVoD service, we have selected five categories that are the most relevant for the users. These categories are: movies, in live show, series, sport and news. We have then classified extracted TV programs into these five categories and we have separately evaluated the imprecision for each category. The obtained results are presented in Table 9.

Table 9 shows that movies are very accurately extracted. This is very interesting as movies are the most relevant content for real world services, TVoD in particular. Another very important content is TV news. Results show that their starts are very accurately detected. However, their ends are sometimes missed. This is mainly due to the weather that is broadcasted right after the end of the news and that is wrongly classified as a long program segment. The results

TABLE 7: *TVTest*. The number of long programs in the Ground-Truth—programs mentioned in the metadata—extracted programs by DealTV.

Day intervals	No. of ground-truth	Metadata		DealTV	
	GT programs	No. of programs	No. of programs $\in$ GT	# programs	No. of programs $\in$ GT
Night	45	46	37	33	30
Morning	29	24	24	24	24
Midday	15	20	15	16	15
Afternoon	17	14	12	12	11
Early evening	14	13	12	13	12
Evening	27	26	24	24	24

TABLE 8: Accuracy of extracted programs by DealTV, comparison with metadata.

Day intervals	DealTV		Metadata	
	Start mean imprecision	End mean imprecision	Start mean imprecision	End mean imprecision
Night	8 min 10 s	9 min 27 s	9 min 57 s	10 min 19 s
Morning	5 min 16 s	3 min 32 s	8 min 25 s	9 min 46 s
Midday	16 s	1 min 37 s	2 min 49 s	7 min 52 s
Afternoon	1 min 32 s	3 min 15 s	3 min 23 s	9 min 58 s
Early evening	4 min 41	7 min 2 s	8 min 3 s	11 min 9 s
Evening	13 s	1 min 21 s	2 min 45 s	6 min 52 s

TABLE 9: Accuracy per category of extracted programs by DealTV from *TVTest*.

Category	Extracted	Start imprecision	End imprecision
Movies	7	1 s	5 s
In live show	1	10 s	1 s
Series	14	27 s	1 min 50 s
Sport	12	42 s	1 min 48 s
News	14	2 s	2 min 56 s

shown in Table 9 do not match the results in Table 8. The high imprecision values in Table 8 are caused by others categories.

In the previous experiment presented in Section 4.4, we have measured that about 17 minutes of TV stream are incorrectly classified each day. A careful analysis of this duration has shown that it is mainly made up of few short programs. These short programs are classified as long programs and they are hence handled as parts of programs. This reduces the overall imprecision of our automatically extracted programs. Based on the results presented in Table 8, it is likely that these mis-classified segments are located outside the midday and the evening intervals.

To sum up, the obtained results show that overall our system is able to perform an accurate and fully automatic TV structuring that greatly outperforms a metadata-based structuring.

5. Conclusion

In this paper, we have studied one aspect of the problem of audio-visual content analysis and indexing. We have focused

on TV structuring that is needed to automatically and precisely extract long useful programs. These can be either archived as part of our heritage or used to build added-value novel TV services like TVoD and Catch up TV.

We have first positioned the problem and then carefully and deeply presented related works and existing solutions.

We then presented DealTV, our fully automatic system. It is based on studying repeated sequences in the TV stream in order to segment it. Segments are then classified using an ILP-based technique that makes use of the temporal relationships between segments. Finally, metadata are used to label and extract programs using simple overlapping-based criteria.

Each processing step of DealTV has been separately evaluated in order to carefully analyze its impact on the final results. The system has been proven on real TV streams to be very effective.

Future work will focus on further improving the performance of the system. In particular, short program segments must be better filtered out during the ILP-classification step. It is the main source of imprecision of extracted programs. We will also study the ability of the system to structure thematic TV streams that are collected from specialized channels, like sport channels.

References

- [1] ETSI EN 300 231, “Television systems; specification of the domestic video programme delivery control system,” European Standard (Telecommunications series), European Telecommunications Standards Institute, 2003.
- [2] S. A. Berrani, P. Lechat, and G. Manson, “TV broadcast macro-segmentation: metadata-based vs. content-based approaches,” in *Proceedings of the ACM International Conference on Image*

- and Video Retrieval, pp. 325–332, Amsterdam, The Netherlands, July 2007.
- [3] J. P. Poli and J. Carrive, "Television stream structuring with program guides," in *Proceedings of the IEEE International Symposium on Multimedia*, pp. 329–334, San Diego, Calif, USA, December 2006.
  - [4] I. Foster and R. L. Grossman, "Data integration in a bandwidth-rich world," *Communications of the ACM*, vol. 46, no. 11, pp. 51–57, 2003.
  - [5] A. Motro and P. Anokhin, "Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources," *Information Fusion*, vol. 7, no. 2, pp. 176–196, 2006.
  - [6] J. P. Poli, "Predicting program guides for video structuring," in *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, pp. 407–411, Hong-Kong, November 2005.
  - [7] Z. Rasheed and M. Shah, "Detection and representation of scene in videos," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1097–1105, 2005.
  - [8] J. R. Kender and B. L. Yeo, "Video scene segmentation via continuous video coherence," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 367–373, Santa Barbara, Calif, USA, January 1998.
  - [9] A. Hanjalic, R. L. Lagendijk, and J. Biemond, "Automated high-level movie segmentation for advanced video-retrieval systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 580–588, 1999.
  - [10] J. Wang, L. Duan, Q. Liu, H. Lu, and J. S. Jin, "A multimodal scheme for program segmentation and representation in broadcast video streams," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 393–408, 2008.
  - [11] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proceedings of the 3rd International Symposium on Music Information Retrieval*, Paris, France, 2002.
  - [12] A. Joly, C. Frelicot, and O. Buisson, "Content-based video copy detection in large databases: a local fingerprints statistical similarity search approach," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. 505–508, Genova, Italy, September 2005.
  - [13] J. Oostveen, T. Kalker, and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems*, pp. 117–128, Hsin Chu, Taiwan, 2002.
  - [14] J. Barr, B. Bradley, and B. Hannigan, "Using digital watermarks with image signatures to mitigate the threat of the copy attack," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 3, pp. 69–72, Hong Kong, 2003.
  - [15] B. Coskun and B. Sankur, "Robust video hash extraction," in *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference*, pp. 292–295, Vienna, Austria, 2004.
  - [16] X. Naturel, G. Gravier, and P. Gros, "Fast structuring of large television streams using program guides," in *Proceedings of the 4th International Workshop on Adaptive Multimedia Retrieval*, pp. 223–232, Geneva, Switzerland, March 2006.
  - [17] R. Lienhart, C. Kuhmunch, and W. Effelsberg, "On the detection and recognition of television commercials," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 509–516, Ottawa, Canada, November 1997.
  - [18] J. Gauch and A. Shivadas, "Identification of new commercials using repeated video sequence detection," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, pp. 1252–1255, Genova, Italy, 2005.
  - [19] J. M. Gauch and A. Shivadas, "Finding and identifying unknown commercials using repeated video sequence detection," *Computer Vision and Image Understanding*, vol. 103, no. 1, pp. 80–88, 2006.
  - [20] M. Covell, S. Baluja, and M. Fink, "Advertisement detection and replacement using acoustic and visual repetition," in *Proceedings of the 8th IEEE International Workshop on Multimedia Signal Processing*, pp. 461–466, Victoria, Canada, 2006.
  - [21] C. Herley, "ARGOS: automatically extracting repeating objects from multimedia streams," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 115–129, 2006.
  - [22] A. Albiol, M. J. Fulla, A. Albiol, and L. Torres, "Detection of TV commercials," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 3, pp. 541–544, Montreal, Canada, 2004.
  - [23] S. Fischer, R. Lienhart, and W. Effelsberg, "Automatic recognition of film genres," in *Proceedings of the 3rd ACM International Conference on Multimedia*, pp. 295–304, San Francisco, Calif, USA, 1995.
  - [24] A. G. Hauptmann and M. J. Witbrock, "Story segmentation and detection of commercials in broadcast news video," in *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries*, pp. 168–179, Santa Barbara, Calif, USA, April 1998.
  - [25] B. T. Truong, S. Venkatesh, and C. Dorai, "Automatic genre identification for content-based video categorization," in *Proceedings of the 15th International Conference on Pattern Recognition*, pp. 4230–4233, Barcelona, Spain, 2000.
  - [26] S. A. Berrani, G. Manson, and P. Lechat, "A non-supervised approach for repeated sequence detection in TV broadcast streams," *Signal Processing: Image Communication*, vol. 23, no. 7, pp. 525–537, 2008.
  - [27] G. Manson and S. A. Berrani, "An inductive logic programming-based approach for TV stream segment classification," in *Proceedings of the IEEE International Symposium on Multimedia*, pp. 130–135, Berkeley, Calif, USA, 2008.
  - [28] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 103–114, Montreal, Canada, 1996.
  - [29] S. Muggleton, "Inverse entailment and prolog," *New Generation Computing*, vol. 13, no. 3–4, pp. 245–286, 1995.
  - [30] A. Srinivasan, "Aleph: a learning engine for proposing hypotheses," 2007, <http://web2.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.pl>.