Content-based inference of structural grammar for recurrent TV programs from a collection of episodes

Bingqing Qu

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Bingqing QU

Préparée à IRISA UMR 6074
Institut de recherche en informatique et systèmes aléatoires
Composante universitaire: IFSIC

Inférence de la grammaire structurelle d'une émission TV récurrente à partir du contenu

Thèse soutenue à INA
le 03 décembre 2015

devant le jury composé de :

Frédéric BÉCHET
Professeur + Aix Marseille Université / *rapporteur*

Sid-Ahmed BERRANI
Chercheur Senior + Orange Labs / *rapporteur*

Bernard MERIALDO
Professeur + Eurecom / *examinateur*

Bertrand COUASNON
Maître de conférences + INSA Rennes, IRISA Rennes/ *examinateur*

Hervé BREDIN
Chargé de recherche + CNRS, LIMSI / *examinateur*

Pascale SÉBILLOT
Professeur + INSA Rennes, IRISA & Inria Rennes / *examinateur*

Guillaume Gravier
Directeur de recherche + CNRS, IRISA & Inria Rennes / *directeur de thèse*

Jean Carrive
Chercheur Senior + INA Research / *co-directeur de thèse*

Abstract

TV program structuring raised as a major theme in the last decade for the task of high quality indexing of multimedia content. Studying program structuring focuses on finding key instants of events of interest, understanding the underlying structures of programs. In earlier literature, some studies of TV program structuring addressed the problem of event detection or scene segmentation, which is not practically usable for indexing tasks because of the lack of semantic understanding of the program structure. Alternately, some approaches structure programs with semantic interpretations, however making use of massive prior knowledge of program structure. Therefore, recent studies of program structuring attempt to structure programs with minimal prior knowledge but more semantic interpretations of programs.

In this work, we follow the last path, addressing the problem of TV program structuring from the point of view of grammatical inference, i.e., discovering a common structural model shared by a collection of episodes of a recurrent program. Recurrent program refers to programs with multiple episodes periodically broadcasted, e.g., on a daily or weekly basis. Recurrent programs exhibit two properties, namely, the repetitiveness and temporal stability of structural elements, i.e., across episodes the same structural elements are involved in almost the same order and with similar duration. Using grammatical inference on recurrent programs makes program structuring possible to rely on only minimal domain knowledge. In particular, we assume no prior knowledge on the structural elements that might be present in a recurrent program and very limited knowledge on the program type, e.g., to name structural elements, apart from the recurrence. With this assumption, we propose an unsupervised framework operating in two stages.

The first stage aims at determining the structural elements that are relevant to the structure of a program. We address this issue by making use of the property of element repetitiveness in recurrent programs, leveraging temporal density analysis to filter out irrelevant events and determine valid elements. By adopting the property of repetitiveness, the determination of structural element is achieved in an unsupervised manner, just minimal domain knowledge being involved. Therefore, the proposed approach is not specific to a certain type of programs but addresses a large category of recurrent programs. Having discovered structural elements, the second stage is to infer a grammar of the program, the key idea of which is to find an optimal alignment of the structural elements between episodes to bring to light regularities and infer a common model shared by the episodes. We explore two inference techniques based either on multiple sequence alignment or on uniform resampling. A grammar model of the structure can be derived from the grammars and used to predict the structure of new episodes.

Evaluations are performed on a selection of four different types of recurrent programs. Focusing on structural element determination, we analyze the effect on the number of determined structural elements, fixing the threshold applied on the density function as well as the size of collection of episodes. For structural grammar inference, we discuss the quality of the grammars obtained and show that they accurately reflect the structure of the program. We also demonstrate that the models obtained by grammatical inference can accurately predict the structure of unseen episodes, conducting a quantitative and comparative evaluation of the two inference methods by segmenting the new episodes into their structural components.

Finally, we summarize our work, discuss a number of open issues in structure discovery, and point out three new research directions to address in future work: Small object mining is addressed for discovering more complete program structures; Regular expression is proposed to improve the ability of extracting more generalized model for the episodes sharing the same structure model but with evident difference; Content-based segmentation aims at ameliorating the effectiveness of structuring new episodes using inferred grammar models in practical use.

Résumé

La structuration des programmes télévisés est une des grandes thématiques des dix dernières années pour la tâche d'indexation de haute qualité. La structuration des programmes focalise sur la recherche des instantes clés des événements d'intérêt et la compréhension des structures des programmes. Dans la littérature, certaines études travaillent sur le problème de la détection d'événements ou de la segmentation des scènes, qui ne sont pratiquement pas utilisable pour des tâches d'indexation en raison du manque de la compréhension sémantique sur la structure du programme. Alternativement, certaines approches structurent des programmes avec des interprétations sémantiques, mais demandant énorme connaissances préalables sur la structure du programme. Par conséquent, des études récentes abordent le problème sur la structuration des programmes avec minimum de connaissances préalables mais plus de interprétations sémantiques sur la structure du programme.

Dans cette thèse, suivant la dernier direction, on aborde le problème de structuration des programmes télévises de manière non supervisée à partir du point de vue de l'inférence grammaticale, focalisant sur la découverte de la structure des programmes récurrents à partir une collection homogène. Programme récurrent se réfère à des programmes avec plusieurs épisodes diffusés périodiquement, par exemple, quotidienne ou hebdomadaire. Les épisodes des programmes récurrents se présentent deux propriétés, la répétitivité et la stabilité temporelle des éléments structuraux, c'est-à-dire, à travers des épisodes les mêmes éléments structuraux se trouvent dans presque le même ordre et avec une durée similaire. L'inférence grammaticale sur des programmes récurrents permet de utiliser minimum des connaissances de domaine a priori pour atteindre la découverte de la structure des programmes. En particulier, on suppose qu'il n'y a aucune connaissance préalable sur les éléments structuraux qui pourraient être présents dans un programme et une connaissance très limitée sur le type de programme. Avec cette hypothèse, on propose un cadre non supervisé se déroulant en deux phases.

Premièrement, on vise à découvrir les éléments structuraux qui sont pertinents à la structure du programme. Afin d'atteindre cet objectif, des détecteurs multimodaux sont implémentés et appliqués sur une collection d'épisodes d'un programme pour détecter des événements généraux qui sont potentiellement liés à la structure du programme. En suite, un filtrage par la densité temporelle des événements généraux est utilisé afin de déterminer les éléments structuraux de programme. En adoptant la propriété de répétition, la détermination des éléments structuraux est réalisée d'une manière non supervisée, ou seulement minimum des connaissances du domaine sont impliquées. Par conséquent, l'approche proposée n'est pas spécifique à un certain type de programmes, mais répond à une large catégorie de programmes récurrents. Deuxièmement, ayant les éléments structuraux pour les

épisodes dans la collection, on s'intéresse à l'inférence grammaticale de la structure des programmes, qui vise à trouver un alignement optimal sur des éléments structuraux entre les épisodes afin de apporter à régularités légers et en déduire un modèle commun partagé par les épisodes. Deux méthodes différentes pour inférer les grammaires structurelles, technique d'alignement des séquences multiples et technique d'alignement par ré-échantillonnage uniforme, sont adoptées afin de découvrir une grammaire pour ces épisodes. Enfin, un modèle structural issu de la grammaire est généré pour segmenter des nouveaux épisodes de même programme.

L'expérimentation est appliquée sur quatre programmes de différents types pour évaluer la performance des grammaires. Focalisant sur la détermination des éléments structuraux, on analyse l'effet sur le nombre d'éléments déterminés détectés, afin de déterminer le seuil appliqué sur la fonction de densité ainsi que le nombre des épisodes dans la collection. Pour inférence de la grammaire, on discute de la qualité des grammaires obtenues et montrons qu'ils reflètent fidèlement la structure du programme. On démontre également que les modèles obtenus par inférence grammaticale peuvent prédire la structure des nouveaux épisodes, effectuant une évaluation quantitative et comparative entre deux méthodes par segmenter les nouveaux épisodes.

Enfin, on conclut nos travaux, discute des questions ouvertes sur la découverte de la structure, et indique trois nouvelles directions de recherche pour aborder dans les travaux futurs : L'exploitation des petites objets est adressée pour découvrir des structures de programmes plus complets ; Expression rationnelle est proposé d'améliorer la capacité d'extraction de modèle plus généralisé pour les épisodes partageant le même modèle de structure, mais avec la différence évidente ; La segmentation basée sur le contenu vise à améliorer l'efficacité de la structuration de nouveaux épisodes en utilisant des modèles de grammaire dans des utilisations pratiques.

Acknowledgements

First of all, I would like to express gratitude to my supervisors, Guillaume Gravier and Jean Carrive, for their continuous support. They have been nurturing and advising me throughout my study and their support and guidance have been fundamental to shape my research and focus my efforts. I also want to thank all jury members, especially two reviewers, Béchet Frédéric and Berrani Sid-Ahmed.

I am grateful to those with whom I have co-authored papers over the last three years : Félicien Vallet for his guidance during the progress of my research. In addition, I would like to thank my colleagues with whom I have collaborated in various occasions, particularly Pierre Letessier for many interesting discussions and great insights.

I would also like to show my gratitude to all my other colleagues and friends for their kindness and help during my stay at Institut National de l'audiovisuel, particularly Feriel Abboud, Ludivine kuznik, Rakia Jaziri, Akila Ghersedine, Valentin Leveau, Olivier Buisson, Jean-Hugues Chenot, Valerie Gauffreteau, Elisabeth Chapalain. As well as my colleagues in IRISA, François Coste, Vincent Claveau, Ewa Kijak, Jonathan Delhumean, thank you for their patient guidance. It was a great experience working with these smart people.

Finally, I do not even know how to thank my beloved Dingqi. She has been through this Ph.D. with me, day after day, reading drafts of my papers and listening to unpolished versions of my talks. I hope we will remember these years forever, together. In addition, I thank my parents and my brother for their unwavering belief in me and support of my endeavors. I know that I would not accomplish this journey without their infinite love and support.

My heartfelt thanks to you all.

Bingqing @ Paris, France

December, 2015

Table of contents

Chapter 1

Introduction

Content

1.1 Background

The last decade has seen a rapid increase in multimedia content, and large scale audiovisual archives are available for users and content providers. Consequently, data organization tools to efficiently manipulate and manage multimedia archives are needed, for which many audiovisual archive institutions were built. The French National Institute of Audiovisual (INA) [1] is one of them to gather, store and share professional multimedia content.

INA has more than twelve million hours of radios and television programs stored, which are prepared for later use, for professionals or individuals. The volume of audiovisual content managed by INA raises many problems. Beyond the technological difficulties related to capture of audiovisual streams and storage of content, such a large collection is useless in practice if its content is not described and indexed so as to know what it contains and to provide easy access to a particular portion of its content. Therefore, indexing of such large-scale archives is indispensable for browsing, sharing and Internet re-diffusion.

1. http://www.institut-national-audiovisuel.fr/

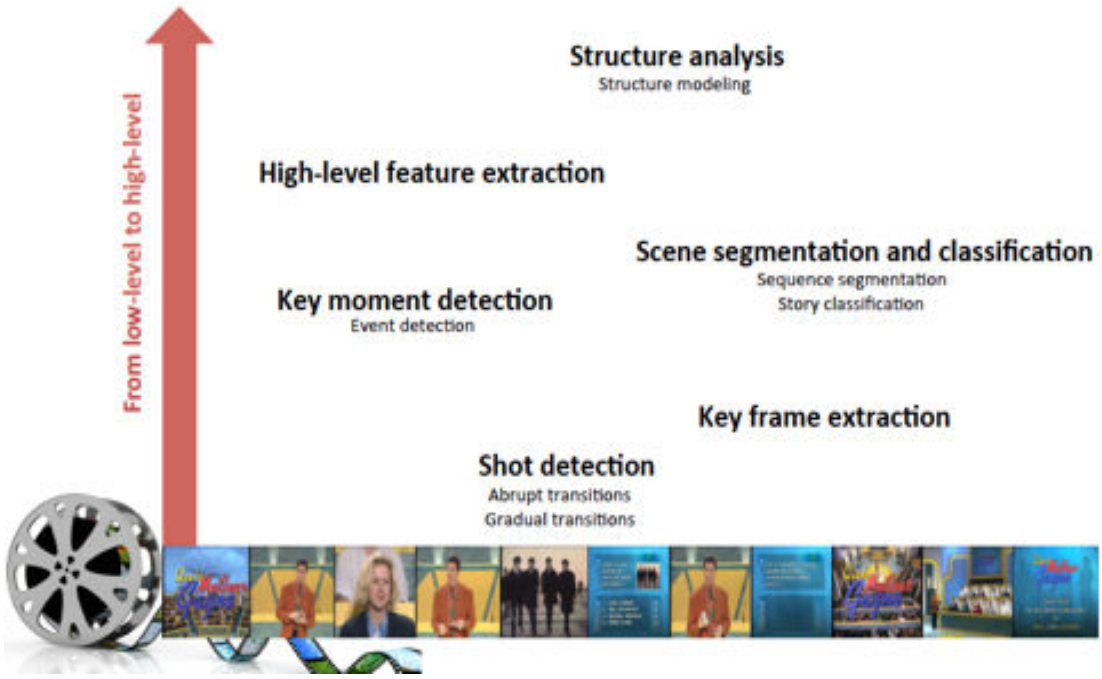


Figure 1.1: Terminologies of automatic video structuring

Currently, audiovisual content indexing, in practice, mainly depends on manual operations by librarians, which is time consuming and costly in human resources. So automatic indexing raised as one of the major themes in the last decade. It can be defined as the use of software methods to establish an index for a set of documents and thus facilitate access to audiovisual content. This area has boomed directly correlated to the increased volume of data to be processed, as well as insufficient human resources. As indicated in [12] [23] and [65], the challenge is to supplement the human labor, safest and also the most expensive, by automatic algorithms for the simplest actions.

Automatic structuring of audiovisual documents is a particular aspect of indexing. Literally, one can define the term structuring as the ability to extract the editorial organization of the documents, while automatic structuring refers to audiovisual document structuring by means of software methods. However, the concept of structuring is very vague and there does not really exist a more precise consensual definition. Structuring could refer to any attempt of content discovery. Taking TV programs, an essential part of audiovisual archives, as an example, TV program structuring includes disparate tasks, like shot detection, scene segmentation, automatic structure analysis, etc. In Figure 1.1, some common terminologies of TV programs structuring are illustrated. These tasks target structuring

videos from low-level to high-level analysis according to their level of semantic interpretation. Semantics is the study of meaning, which requires prior knowledge to interpret digital representations of a computing machine (low-level) into contextual concepts (high-level). From shot detection to structure analysis, program structuring is accomplished from low-level to high-level analysis according to the use of prior domain knowledge.

In earlier literature, automatic program structuring mainly adopted low-level analysis. Although programs are segmented with very limited prior knowledge, e.g., shot detection [49, 54], the segmentation is not practically usable for indexing tasks because of the lack of semantic understanding of the program structure. Alternately, shifting to high-level analysis, some work makes use of massive prior knowledge of program structure, i.e., expert-craft models or tedious annotations, to detect events of interest or to segment programs. For example, some work focused on certain unique type of programs, structuring programs into their constitutive components, e.g., tennis match [21,42], by exploiting prior knowledge on the structure of such a program. Some work tried to find the structure of a video stream, segmenting the program stream and inter-programs [36, 52] into coherent types of video clips. Other work learns recurrences from an electronic program guide, as at INA [56]. The structures of programs are better exploited, however supervisions and annotations are required, which need a lot of manual ancillaries.

Therefore, recent work attempts to structure programs with less prior knowledge but more structural information. Several studies have taken this path, focusing on unsupervised methods to discover the program structure with very limited prior knowledge on programs. For example, at INA, Vallet [70] exploited the structure of talk shows centered on speakers' interventions. Another work, conducted by Abduraman [1], was based on the discovery of separators, i.e., short video sequences that are inserted between different parts of a program. These work achieved to structure programs without supervisions and annotations. However they are limited to a certain type of program or a particular constitutive elements of the program. In other words, the proposed approaches are not generic enough to process various types of programs with different constitutive elements.

Taking the same path of structuring TV programs with minimal prior knowledge, the work presented in this thesis, accomplished at INA, attempts to enhance previous work's advantage, i.e., adopting minimal prior knowledge, and compensate for their deficiencies, i.e., covering more program types.

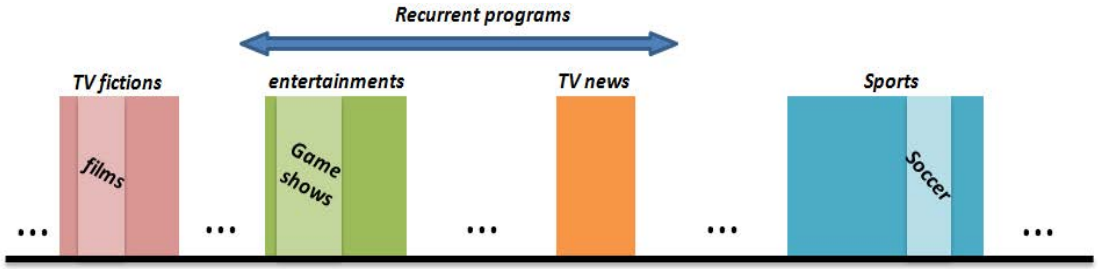


Figure 1.2: Positioning of recurrent programs

1.2 Thesis objectives and contributions

This thesis focuses on TV programs and addresses the problem of automatic TV program structuring. We propose to structure programs in the range of high-level analysis but adopt minimal prior knowledge, aiming at discovering the overall structure of programs.

Practically, it is difficult to find a general method that covers all the types of existing TV programs, hence we focused on a broad scope of programs, getting rid of the limitation to a certain type of programs. As illustrated in Figure 1.2, we work on **recurrent programs**, which designate programs with multiple episodes periodically broadcasted (e.g., daily, weekly). Recurrent programs are numerous on most TV channels, with news programs, magazines and entertainments as typical examples. Such programs are usually well structured while being edited, and the content of each part is usually detailedly verified before being broadcasted. Taking TV news as an example: News usually start with a brief outline of the reports, followed by an alternation of anchorperson's announcement of the upcoming topic and of the corresponding news report; Most news programs end with interview segments, sports highlights or program teasers. Similarly, a particular game might start with the presentation of the competitors, followed by the different phases of the game, ending with the prize awarded to the winner. We designate the constitutive elements of the program—headline, report, interview, trailer in the news example—as **structural elements**, referring to a video segment with a particular syntactic meaning. Interestingly, most episodes of a recurrent program follow the same editorial structure: Across episodes, the same structural elements are involved in almost the same order and with similar duration. In summary, recurrent TV programs have a stable structure with well defined structural elements repeating across episodes, which can be seen as two properties, **element repetitiveness** and **temporal stability**.

With the main objective, i.e., TV program structuring with minimal prior knowledge,

the properties of recurrent programs hints us that we could determine structural elements in an unsupervised manner considering their repetitiveness, and infer a model for the program representing its overall structure since the temporal structures of its episodes are stable. Having this idea, two challenges are being faced.

The first one lies on unsupervised structural elements determination. We assume that we have no prior knowledge on the structural elements that might be present in a recurrent program and very limited knowledge on the program type, e.g., to name structural elements. In practice, different types of programs lead to a diversity of programs structures, hence various types of structural elements may be present. With this assumption, identifying such structural elements relevant to the structure of a program is not a trivial task. Therefore we propose to determine the structural elements by mean of their repetitiveness. The same structural elements across episodes are usually identical, e.g., the opening jingles, or inherit the same audio/visual properties, e.g., commercials sequences. By leveraging their properties of repetitiveness, we are able to discover the structural elements repeating across episodes in an unsupervised manner. The proposed approach for determining structural elements is not specific to a certain type of programs but addresses a large category of recurrent programs.

Having the determined structural elements, the second challenge is to derive a structural model representing the overall structure of the program. Although, temporal stability of recurrent program structures results in similar structural elements with approximate temporal positions, slight differences still exist among different episodes. Discovering a common model shared across episodes is also a tough task. To reach this objective, we propose to make use of grammatical inference techniques, the key idea of which is to find an optimal alignment of the structural elements between episodes to bring to light regularities and infer a grammar for the program. Beyond this, across episodes, the structure of a program may vary according to editorial purposes, which lead to multiple structures for a recurrent program. Taking the news example, episodes where invited people are present usually end with an interview. There is also a number of episodes that end with movie teasers, e.g., when a new film is released. Thus identifying multiple structures and grammars are required to avoid structure ambiguities. So the grammar inference task is bound up with identifying multiple structures for a given recurrent program.

Practically, program structuring based on grammar inference has specific meaning for TV program indexing tasks in industry. Recurrent programs usually have a lot of episodes, which leads to a very heavy workload. As mentioned, at INA, the most common case of indexing such recurrent programs is manual structuring, which is costly in both time and

Figure 1.3: Applicable purpose in industry

human resources. Targeting this, our studies contribute a way of facilitating the manual structuring tasks to the industrial life. As illustrated in 1.3, by adopting a small quantity of episodes, without manual annotations, we infer a grammar of the program, indicating its temporal structure and building a grammar model with time information of its structural elements. Such a grammar model of programs can be utilized to segment existing episodes or process upcoming episodes. Particularly, our studies provide a structural reference (i.e., grammar models) for librarians to have an overall understanding of program structures, thus helping them to rapidly find the boundaries of structural elements according to the grammar model. It allows in practice facilitating the manual indexing tasks and improving the efficiency of archiving tasks. This measure attempts to make the manual indexing moving a step forward towards the automatic indexing in industrial life.

1.3 Chapter outlines

In this dissertation, we discover the structure of recurrent programs based on grammatical inference. Since automatic program structuring has been widely studied in recent years, in Chapter 2, we first present the existing work in this area. In Chapter 3, we present the overview of the proposed approach, before which the recurrent programs are described, as well as data sets used in the work. The rest of the dissertation presents our main contributions, which are organized in two parts.

The first part consists of two chapters, describing the discovery of structural elements. It begins in Chapter 4 with unsupervised structural element determination by a broad scope audiovisual detectors and temporal distribution analysis. In Chapter 5, after obtaining the structural elements, we present the results with experimental analysis.

The second part having three chapters, introduces the two grammar inference techniques, i.e., multiple sequence alignment and uniform resampling, which are explained respectively in Chapter 6 and Chapter 7. In Chapter 8, we report the experimental evaluations by comparing the pros and cons of the two proposed techniques.

At last, in Chapter 9, we conclude this thesis with a summary of the contributions and a review of some short and long term research perspectives.

Chapter 2

Automatic TV Program Structuring: A Literature Review

Content

Due to the significant growth in the content of broadcasted TV programs, manual operations for program managements and manipulations are no longer sufficient. Consequently, automatic program indexing is required for complementing the manual services. In particular, program structuring, which this thesis focuses on, is a crucial step for high-quality indexing, and amount of work has been targeting this subject. Based on whether prior knowledge of the program structure is adopted or not, previous work in the abundant literature on TV program structuring can be classified in two categories: prior-knowledge-based methods and prior-knowledge-free methods. Prior-knowledge-based methods exploit the prior knowledge of programs in order to get the structural elements presented in the program or to construct a structure model of the program. Alternatively, some notable studies

address the problem of program structuring without prior knowledge of the structure, attempting to shift from supervised to unsupervised techniques for program structuring.

2.1 Prior-knowledge-based methods

Prior-knowledge-based methods extensively work on the prior knowledge of program structures in order to get massive information used for structuring programs, focusing on either typical structural elements or overall structures of programs.

2.1.1 Approaches for structural element detection

Numerous studies focus on the detection of typical structural elements using prior domain knowledge. By adopting the specificity of anchorperson shots while editing the news, some studies focus on detecting anchorpersons in newscasts. [34] builds a sequence-specific template, using the template and a dissimilarity measure to distinguish all anchorperson shots from the others. [77] proposes a frame model combining a set of region models of distinct compositions for the anchorperson frame, including reporters' names and the logo of the broadcast channel. Recognizing an anchorperson shot involves testing every frame over the set of region models. [32] also uses a region match scheme to cluster anchorperson shots. Object detection and tracking are adopted to define the region of interest by considering the boundary of the object. Face detection and skin color classification are used to find the possible regions where anchorpersons might be suited. These regions are then compared to templates stored in the application database. The prior knowledge of the spatial configuration of the objects is used to limit the regions of interest. [24] detects anchorpersons based on face similarity, where the interview segments, in the form that interviewer (anchorperson) and interviewee recursively appear, are identified using a technique called interview clustering. A very recent work [19] targets on fast detection of anchorperson shots in TV sports news, where a temporal aggregation method applied for sports news videos detects and aggregates two kinds of shots, i.e., sequences of long shots and studio shots, in order to significantly reduce the number of frames analyzed in content-based indexing of TV sports news. The anchorperson shot detection is then realized using traditional methods for example face detection and video categorization.

Sports programs have also been studied since decades, due to the fact that sports videos are characterized by a defined structure, combining the game rules as well as knowledge of visual environment. In [5] and [18], goal events are identified for soccer videos. [5] presents

a supervised learning scheme, i.e., support vector machines, for detecting goals during a soccer match, using images acquired by a single camera placed externally to the field. Using combined visual/audio features analysis and decision tree model, [18] is also able to determine soccer goal events in soccer videos. [33] tracks in real time specific moments in baseball game, as the ball during pitches, by mean of extensive prior knowledge about the game and camera setup rules. Some recent studies work on diving and jump games. [45] utilizes hidden Markov models to recognize the player actions, where the highlights in diving videos are detected by motion segmentation and a hierarchical agglomerative clustering. Then the action is recognized based on body shape segmentation and shape transitions modeled by continuous hidden Markov models. Using the trajectory of the action, [78] aims at detecting and representing player's action in panoramic background. The global motion is first estimated to obtain dominant motion by calculating the homographies between video frames with RANSAC [29] technique. After building the panoramic background by the inter-frame relationship, object segmentation is implemented by subtraction between the panoramic background and each warped frame in order to represent the player's action in each frame on the panoramic background.

2.1.2 Approaches for overall structure recovery

Multiple studies have attempted to recover the overall structure of programs. All these studies target the entire structure of a specific type of program. In the news domain, [9] proposes a news browsing system with automatic video annotations based on visual features and textual strings. A statistical analysis is performed to classify news shots as anchorperson shots, as well as report shots, the content of which is further described through the use of both visual and textual information. [26] presents an automatic learning system for news video indexing, where hidden Markov models are used and trained by presenting manually indexed video sequences. While [27] uses a neural network-based face detector and machine-learning-based techniques to classify the shots of news video into six predefined categories, e.g., *anchor*, *interview*, *forcast*, etc. [25] proposes a method for the automatic segmentation of TV news videos into stories, which requires a very small amount of manual annotation. A multiple-descriptor based segmentation approach is used to select multimodal features and give insights about story boundaries. The story boundaries are then predicted using machine learning techniques. Zlitni *et al.* make use of the contextual and operational characteristics as prior knowledge to automatically structure TV news in [79]. The prior knowledge is modeled as video grammar for identifying a news show in TV stream. Then the news programs are segmented into different topics, like outlines, sports, etc.

Many studies use extensive prior knowledge of sports and editing rules to model the structure. Soccer and tennis are spotlights in sports domain, as they attract a large audience and have important commercial applications. [73] adopts low-level features to detect general states of soccer, i.e., *play* and *break*, and models the stochastic structures of each state of the game with a set of hidden Markov models. Fine grain states, i.e., *goal*, *referee*, *penalty*, are targeted in [76] combining features extracted from the video and additional information from the semantic gamelog for live soccer video. This additional information facilitates the system to achieve accurate and very fast boundary detection of the states. [28] tries to use more cinematic features, such as dominant color region detection and shot boundary detection, and employ limited object based features, such as goal detection, referee detection and penalty-box detection, to detect certain events, like *goal events*, *red-yellow cards* and *penalty*, in soccer. Hidden Markov models are used in [41] and [42] for tennis video structuring, relying on prior information about tennis video content and production rules. [41] proposes a tennis scene classification and segmentation method that automatically labels each segment of a tennis video with one of the following four types: *first missed serve*, *rally*, *replay*, and *break*. HMMs are used to merge audio and visual information, and to represent the hierarchical structure of a tennis match, where domain knowledge is implicitly taken into account through the learning process associated with HMMs. Similarly, [42] also proposes scene classification and segmentation of a tennis match using HMMs, aiming at exploiting multimodal information and temporal relations between shots in order to identify the global structure. Multimodal fusion however is performed at two levels: feature level and decision level. At the feature level, low-level audio and visual features are combined into a single audiovisual feature vector before the classification. At the decision level, classifying separately each modality before integrating the classification results is considered.

These approaches focus on high-level analysis of programs, providing abundant semantic interpretations of the structures of programs. However, most of these approaches are highly supervised. Prior knowledge of the program structure is required to construct models either for typical structural elements or for entire program structures. Even though, for some cases, there is no prior models, massive prior knowledge are utilized to define and detect the typical events. Highly relying on prior knowledge to accomplish the structuring tasks, these approaches can usually target one typical structural elements or one type of programs.

2.2 Prior-knowledge-free methods

Having the same goal, prior-knowledge-free methods try to recover the structures with minimal, even without, prior knowledge of programs, attempting to shift from supervised to unsupervised techniques for program structuring.

2.2.1 Approaches for structural element detection

Same as the prior-knowledge-based category, some approaches focus on detecting typical structural elements. However, abandoning the prior knowledge about program structures, they consider inherent properties of certain structural elements across diverse types of programs.

[37] adopts a frequent pattern approach that exploits video's self-similarity for anchorperson detection, relying on a matrix of image feature similarity between different time points in the video. The method is very general and may be used to anchorperson detection or other related problems. [30] also targets anchorperson detection in news. It utilizes a fuzzy C-means algorithm to detect the shot boundaries and partition the video frames into video shots. Then adopts a graph-theoretical cluster analysis algorithm to classify the video shots into anchorperson shots and news footage shots, so as to detect all the anchorperson shots. A very recent work [16] presents a robust framework using a hybrid I-vector representation to extract speaker identity features from the audio data, and a deep neural network system to perform anchorperson detection based on audio streams of video content. This framework is able to detect the anchorpersons for different types of programs, such as talk shows. Ji *et al.* recently propose a novel algorithm for anchorperson detection in news video sequences in [40]. The anchorperson detection is conducted from the key frames of detected shots by using a clustering-based method based on a statistical distance of Pearson's correlation coefficient. [11] also focuses on anchorperson detection in news videos. The algorithm exploits audio, frame and face information to identify major cast in the content. These three components are firstly processed independently during the cluster analysis and then jointly in a compositional mining phase. A differentiation of the role played by the people in the video has been implemented, exploiting the temporal characteristics of the detected anchorpersons. In [57], customized temporal video segmentation and repeated video clips are adopted for real-time video processing, having the purpose of typical content analysis, such as commercial detection.

2.2.2 Approaches for overall structure recovery

Recent work also addresses the entire structure of programs in an unsupervised manner. [75] uses a dissimilarity index and a proximity matrix based on color and luminance information, to match video shots and cluster them into different scenes. [72] aims to recover the temporal and structural characteristics of TV programs with visual, auditory, and textual information. The general idea is to group successive shots into meaningful clusters along the temporal dimension so that all the shots in a scene are relevant to a common event. [61] proposes a shot similarity graph method. The program structuring is achieved by splitting a weighted graph into subgraphs, verifying the weights between two shots computed based on their color, motion similarity and the temporal distance. In [7], recurrent segments exhibiting audiovisual consistency are firstly discovered. The underlying events throughout the video are detected using discriminative models and filtered according to their relevance to the video structure. Event mining is applied to unsupervised video structure analysis, using simple heuristics on occurrence patterns of the events discovered to select those relevant to the video structure. In [22], sequences of audio and shot clusters are automatically identified using unsupervised audio diarization and shot segmentation techniques. Audio-visual clusters are then formed by ranking the co-occurrences between these two segmentation and selecting those that significantly go beyond chance.

2.2.3 Approaches based on recurrence detection

Beyond traditional approaches, i.e., approaches for typical structural element detection or for overall structure recovery, a recent approach based on recurrence detection is highly studied, which is proved to be an effective way to avoid to use massive prior knowledge for program structuring but provide more semantic understanding of program structures.

For example, event repetitiveness and temporal distribution analysis are leveraged by considering visual/audio recurrence in [2, 3] to detect separators, where a separator refers to certain short sequences which may appear before or after the events of interest to signal their start or end. [31] develops a repetition-based approach to commercials detection. The system utilizes customized video segmentation techniques to automatically partition the video into semantically sensible shots and scenes. Then a repeated sequence detection is adopted, followed by a feature-based classification technique in order to classify the video sequences as commercials or non-commercials. Yang *et al.* propose a method to model and analyze video syntactical structure based on short video repeat detection in [74]. Two detectors in a cascade structure are employed to repeated clip mining, where very short

video repeats (<1 s) and long ones can both be detected by a single process. And a method of video structure discovery and syntactical segmentation is proposed based on short video repeats detection. In [8] repeated audiovisual sequences are detected and used for TV broadcast segmentation. It relies on a micro-clustering technique to group similar audio/visual descriptors, allowing the identifications of inter-program structures.

Repetitions are also used in audio structure analysis. In [48], repeating pattern discovery and structure analysis of acoustic music data is proposed. Using a self-similarity matrix of the music, the algorithm detects repetitions that have similar melody in music and songs for music summarization, indexing and retrieval. Music summarization is also performed in [47] where *key phrases* are detected after parameterizing the song into features, and a top-down clustering is adopted based on unsupervised learning of hidden Markov models. [15] performs segmentation and summarization of music, achieved by tonality analysis and recurrent structure analysis. [14] propose a software MODIS: a generic approach to mine repeating audio sequences, with tolerance to motif variability. MODIS is basically based on the unsupervised algorithm [51], where repetitions are detected by extending matches of motif fragments, and a template matching technique is used to detect acoustically close segments, based on dynamic time warping and self-similarity matrix.

Making use of general solutions as well as the properties of recurrence, prior knowledge of program structures is no longer indispensable to detect structural elements or program structures. Program structuring tasks can be achieved without supervision nor training data, on rare occasion with very limited prior domain knowledge. Besides, these approaches are no longer limited to certain types of programs, most of them can be applied on various programs.

2.3 Conclusion

As stated in the introduction, TV program structuring is very important for a better understanding of the video content, and provides an efficient way for video indexing. All the methods presented in the previous sections try to meet these objectives.

On the one hand, for prior-knowledge-based approaches, although giving promising results, they highly rely on prior knowledge about program structures. Requiring prior knowledge, such as editing rules, annotations or template models, leads to a lack of generality of processed videos, hence they can be applied to only very specific types of programs. Most of them are supervised. Even though some do not make use of supervision nor train-

ing process, called unsupervised methods, massive domain knowledge is needed. On the other hand, prior-knowledge-free approaches try to structure videos without prior knowledge about program structures, shifting from supervised to unsupervised techniques. Since limited prior domain knowledge is involved, the approaches are able to process various types of programs.

Considering the main objective of the thesis, i.e., structuring recurrent programs with minimal prior knowledge, making use of recurrence seems to be a promising approach. As the recurrent programs refer to a large varieties of programs, it is not trivial to cover different types of programs by means of prior-knowledge-based structuring methods. Since we focus on the common structure shared by a collection of episodes from a recurrent program, the invariants across episode is the key elements to be detected, which hints us to choose the approaches based on recurrence detection. We thus use it as a starting point for the thesis, trying to discover the structure of recurrent programs with as much structural information as possible but adopting very limited domain knowledge.

Chapter 3

Overview of the proposed approach

Content

Since we aim at discovering program structures with minimal prior knowledge, the use of recurrence seems to be a promising way to achieve the goal, especially when we try to cover not only one type of TV programs. Thus we propose to work on recurrent programs as stated previously. Before stepping to the the proposed approach, it is important to study systematically recurrent programs to better understand them, identifying their key properties, and the importance in TV industry. Furthermore, we make a simple description on the four different recurrent TV programs which will be involved in the thesis. At last, an overview of the proposed framework will be introduced to provide a general understanding of the work presented in the thesis.

Table 3.1: percentage of recurrent programs during one week of broadcast [1]

Channel	Percentage on program	Percentage on duration
TF1	35.38%	31.49%
France2	73.75%	72.62%
M6	33.33%	34.57%

3.1 Recurrent TV programs

A recurrent program refers to a program having multiple episodes that are periodically broadcasted (e.g., daily, weekly). For instance, a TV news show is usually broadcasted daily and a game show may broadcast an episode per week. Recurrent programs are numerous on most TV channels, in addition to news and games, magazines and entertainments are also typical examples. Here we give a general idea of the importance of recurrent programs on TV channels to show the necessity of working on it.

We make a statistical analysis on three different French TV channels, i.e., *TF1*, *France2* and *M6*, according to studies of TV streams conducted by [1, 55]. TV channels usually broadcast various types of programs. Figure 3.1 shows an extract from the program schedule of *France2* collected by Poli from INA [55]. The legend below the schedule indicates the types of different programs with the corresponding colors. As shown in the figure, although TV fictions, i.e., films, TV films and TV series, take a large proportion of all programs, there is also a fair amount of recurrent programs, including news, magazines, games, news magazines and entertainments.

A study on recurrent programs is made by [1], the percentage of recurrent programs during a week of broadcast being presented in Table 3.1. The table does not include weather forecasts, as well as some other short service programs like lotteries, although they are also recurrently broadcasted. The interest of structuring lies on long-duration programs, i.e., programs longer than 10 minutes, thus the service programs and short magazines will be out of our scope. As shown, recurrent programs have a high proportion both on percentage of programs and on the time occupied in TV timetables. The significant broadcast volume fully explains the necessity of working on structuring recurrent programs, which will benefit TV program indexing and archiving.

As stated previously, recurrent programs exhibit two structural properties, i.e., *repetitiveness* and *temporal stability*. Concretely, most episodes of a recurrent program follow the same editorial structure: Across episodes, the same structural elements are involved in almost the same order and with similar duration. Figure 3.2 illustrates a typical recur-

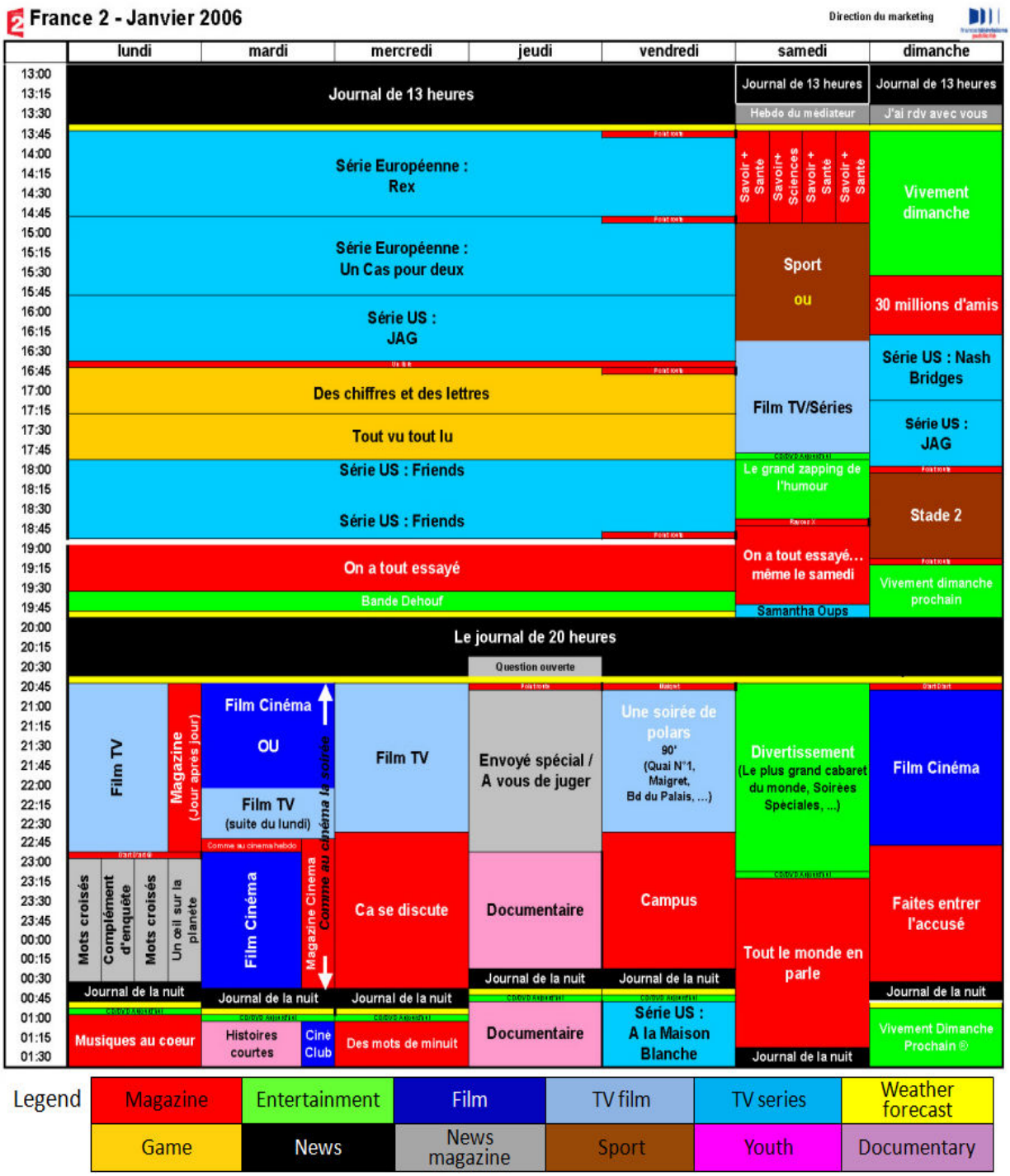


Figure 3.1: Extract from the program schedule of France2 from January to June 2006 [55]

Figure 3.2: The structure of TV news

rent program—News—with its typical structural elements. TV news usually start with a brief outline of the reports (*outline*), followed by an alternation of anchorperson's announcement of the upcoming topics and the corresponding news reports; Most news programs end with interview segments (*interview*), sports highlights (*sports*) or program trailer (*trailer*). Anchorperson's announcements and news reports are the typical structural elements. Particularly, news reports can be further decomposed into finer-grained structural elements, e.g., sports, interviews or forecasts.

However, according to different editorial purposes, across episodes, the organization of a program may partly vary. Considering the TV news example above: The days when there are invited people, the show usually ends with the interview, while the days when a new film is released, the show usually ends with the trailer. As a result, for the same program, *multiple structures* may exist [60].

In summary, the characteristics of recurrent programs could be concluded as three properties:

- **Temporal stability** The episodes from the same program usually follow the same temporal structure with similar structural elements.
- **Repetitiveness** Across episodes, the same structural elements are involved in almost the same order and with similar duration.
- **Multiple structures** For one program, multiple structures could be found according to the editorial issues.

The properties, i.e., temporal stability and repetitiveness, make the task of unsupervised structuring feasible, and provide the possibility of covering a large scope types of programs. While the property, i.e., multiple structures, brings new challenges when structuring recurrent programs, especially when discovering a common model shared by the episodes.

3.2 Vocabulary employed for recurrent TV programs

In order to avoid any confusion and to facilitate the reading of the thesis, we present the lexicon employed in the thesis.

We firstly highlight the terms for the description of recurrent programs:

- **Program** A recurrent TV program.
- **Episode** A simple occurrence of the program.

As illustrated in Figure 3.3, *20h News* is a program whose type is deemed as TV news, while a piece of *20h News* dated 19 March 2007 is an episode of the program.

Considering the terms for program structure, the key notions used in the thesis are defined as follows:

- **Structural element** A video segment with a particular syntactic meaning. It includes the constitutive elements of a program, such as outline, anchorperson, report in the example of news.
- **Structural grammar** A graphical representation of the optimal alignment of structural element between episodes. Since a recurrent program has stable structure across episodes, it is possible to discover the common structure shared across episodes and generate the corresponding grammar.
- **Grammar model** An abstracted model derived from the structural grammar, including structural elements, their temporal organizations, relative duration as well as presence probabilities.

For example, reports and anchorperson's announcements are structural elements for a news program, on top of which a structural grammar can be formed by aligning the common elements shared by the episodes, as well as a grammar model with occurrence duration and probabilities can be built.

Regarding structural elements, we want to highlight two terms, separators and chapters, which are frequently seen in games or other entertainment programs:

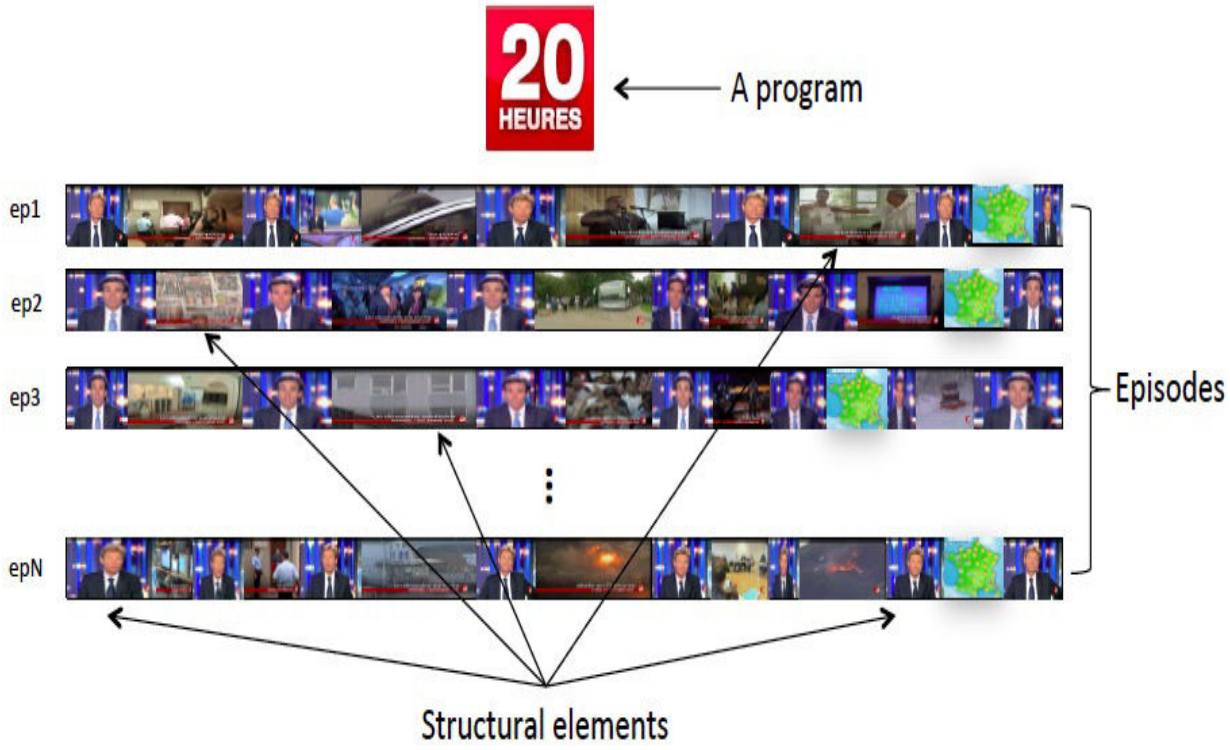


Figure 3.3: Terms for recurrent programs

- **Separator** Short audio/visual sequences that appear before or after the events of interest to signal their start or end. Separators are usually used to separate a program into different chapters, signifying different stages of a program, especially for entertainment programs.
- **Chapter** The set of segments between two separators.

An example of a game show is illustrated in Figure 3.4, where the program is divided into four chapters by four separators.

3.3 Presentation of the datasets employed in the thesis

In this thesis, four recurrent programs are considered, namely a TV news show, a game show, a talk show and a magazine. We describe each of them by giving their basic information and presenting their general structure. Then we present a manual annotation tool for getting ground truth program structures.

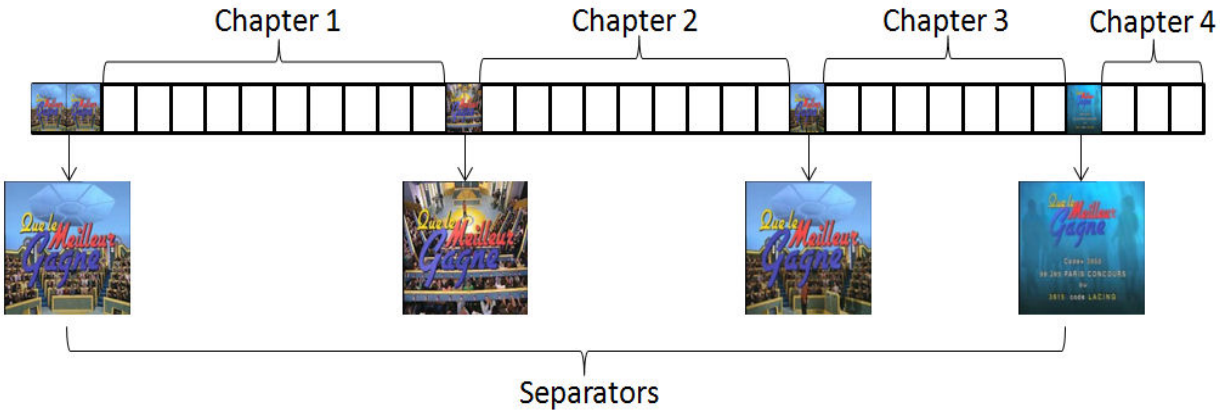


Figure 3.4: Example of separators and chapters in an episode of a game show

Table 3.2: Description of the datasets employed in the thesis

Dataset	Program	Type	Date	Volume	Average duration
NEWS	*20h news*	TV news	2007	24 episodes	37.9 m
TALK	*Le grand journal*	Talk show	2014	24 episodes	71.3 m
GAME	*Que le meilleur gagne*	Game	1991 - 1992	24 episodes	31.9 m
MAGZ	*Telematin*	Magazine	1989	24 episodes	61.9 m

3.3.1 General presentation of the datasets

The general information of each program is listed in Table 4.1. In order to be representative, we adopt four programs of different types. Episodes of the four programs are taken from certain years over twenty-five years (from 1989 to 2014). We detail each program hereunder.

- **NEWS** *20h News* follows a very standard pattern for a daily news show. Starting with an outline, it is composed of anchorperson's announcements and news reports. As illustrated in 3.2, news reports usually consist of daily events, sports, weather, etc. Episodes in the dadaset, lasting about half an hour, are randomly taken from the year 2007.
- **GAME** *Que le meilleur gagne* is a game show hosted by a conductor. The episodes of GAME were taken over two years (1991 and 1992). It has four parts divided by separators. During the first part, 200 candidates must answer multiple choice questions. A candidate will be eliminated once he/she does not respond properly.

Figure 3.5: Extracts of GAME (*Que le meilleur gagne*)

After this round, only ten candidates are kept. In the second part, the ten candidates are eliminated on the same principle to select the two bests. In the third part, the remaining two finalists meet to win the final position. In the last part, the finalist must give a correct order for a question of sorting to win a big prize. In summary, the program is well structured, mainly containing interview scenes and question/answer scenes with frames having full texts. Figure 3.5 shows the frame extracts of GAME.

- **TALK** *Le grand journal* is a talk show, whose episodes are taken from the first months of 2014. The talk show is hosted by a conductor and contains three main parts between which separators and commercials are inserted. For the first two parts, the talks mainly lie on the discussions of politics between the conductor and the invited people. In the third part, new invited people are welcomed, and the subjects depend on the guests, e.g., movies when actors are invited while musics when singers are invited. During each part, some special columns may be added, such as weather reports, musical performances, etc. Comparing to GAME, the structure of TALK varies more because of its longer duration. That can also be explained by the fact that its episodes are taken from a more recent year. Figure 3.6 shows its frame extracts.
- **MAGZ** The magazine *Telematin* was firstly broadcasted in 1985, and today is still aired. It is a morning program proposing news and topics about culture and daily life. The dataset was taken with episodes selected from the year 1989. The program lasts about 150 minutes on average, but we just captured the first hour of each episode for MAGZ, as the structure of remaining parts is somehow a repeat of the first hour. The program is conducted by a main conductor thought out the program, and several sub-conductors for different topics. During one hour's broadcast, various topics are introduced, including flash news, weather reports, sports, musics, fashions, etc, each

Figure 3.6: Extracts of TALK(*Le grand journal*)

Figure 3.7: Extracts of MAGZ(*Telematin*)

of which is an individual part separated by separators. The length of these parts lasts from less than 5 minutes to about 20 minutes. Figure 3.7 shows its frame extracts.

The types of these four programs are the most seen ones for recurrent programs on television, which are representative enough to show the feasibility of the structuring approach presented in the thesis for various types of recurrent programs.

3.3.2 Ground truth annotation

Evidently, for later use of evaluation of the proposed approach in the thesis, the ground truth of program structures is needed. The annotations is done by a data management tool, named Feria2 [13], which is developed in INA and not yet open to public. Figure 3.8 gives a screen capture Feria2.

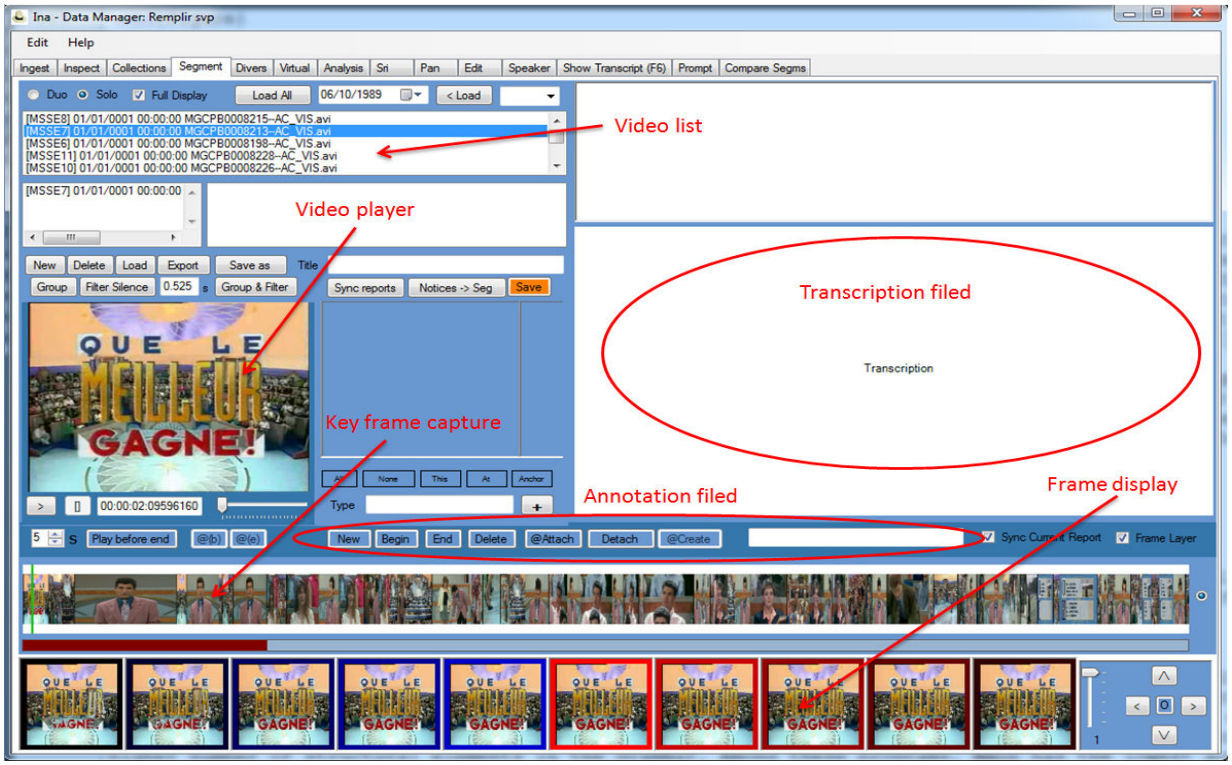


Figure 3.8: Screen capture of the annotation tool Feria2

Feria2 is an experimental framework, designed for facilitating the development of research prototypes, experimentation for documentation and visualization of audiovisual content. For aspect of visualization and annotation of video contents, Feria2 mainly consists of a video player, the capture of key frames, a field of transcriptions, a field for displaying every frame of the video and a bar of annotation tools.

We manually segmented and annotated the structural elements of each episode with their corresponding types and start/end times. The program structures annotated by Feria2 can achieve frame accuracy.

3.4 General presentation of the proposed framework

This thesis casts the task of recurrent program structuring as a grammatical inference task, contributing to demonstrate that by leveraging grammatical inference techniques, the program structure can be discovered with only minimal prior knowledge and a model can

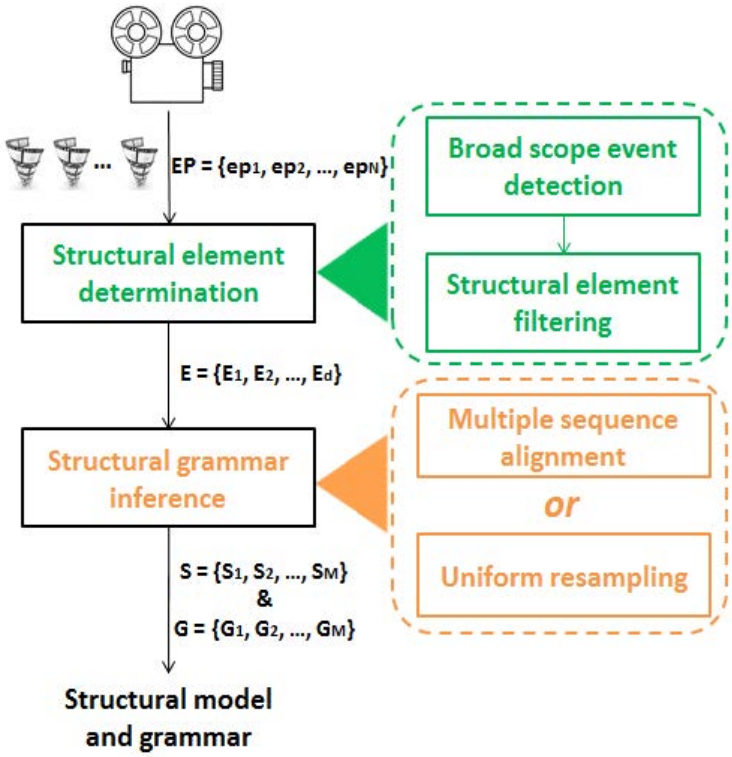


Figure 3.9: Framework for grammatical inference of recurrent programs

be designed to accurately segment new episodes.

To achieve that, we proposed a two-step framework illustrated in Figure 3.9. The objective that we are targeting is to ultimately create a model of a recurrent TV program given a collection of episodes EP = $\{\text{ep}_1, \text{ep}_2, .., \text{ep}_N\}$. The collection, for a certain program, is randomly selected from its existing episodes, and usually a small volume of episodes is used.

The first step, in green, aims at discovering elements in the video that are relevant to the structure of the program, e.g., jingles, outlines, anchorshots. In the absence of prior knowledge on the structural elements that might be present in a recurrent program, the determination of structural elements is achieved by running a number of broad scope audiovisual event detectors to generate a large number of potential general events. As structural elements of recurrent programs are supposed to repeat across episodes, the temporal distribution of the general events is analyzed and filtered to keep only the repeated ones, which are deemed to be relevant to the program structure. Elements relevant to the structure, called structural elements, are finally identified using minimal domain knowledge

from the events deemed as repeating across episodes. The outcome of the first stage is thus a collection of d structural elements $E = \{E_1, E_2, ..., E_d\}$ with their corresponding occurrences in the N episodes.

Having the collection of episodes labeled with the structural elements in E, the second step consists in grammar inference. Although, temporal stability of recurrent TV program structure results in similar structural elements with approximate temporal positions, slight differences still exist among different episodes. Discovering a common model shared across episodes is targeted, where two different approaches are considered to align structural elements across episodes: Standard multiple alignment techniques can be used; Alternately, uniform resampling can be performed after normalizing the length of each episode. From the alignments, a grammar is generated and transformed into a grammar model. In practice, the structure of a program may vary across episodes according to editorial purposes, thus identifying multiple structures and grammars is required. The outcome of the grammar inference stage is thus a set of structures, $S = \{S_1, S_2, ..., S_M\}$, from which a set of grammars, $G = \{G_1, G_2, ..., G_M\}$, is derived.

3.5 Conclusion

This chapter gives an overview of the thesis. We systematically described the recurrent programs for further understanding them and identifying their key properties, and several frequently used terms in the program structuring lexicon were highlighted. Furthermore, we had a simple description on the four different recurrent TV programs which will be used in the thesis. At last, a general presentation of the proposed framework was introduced to provide a general understanding of the work that will be presented in the last parts of the thesis.

Two main stages are involved in the proposed framework, i.e., structural element determination and grammatical inference. The first stage aims at discovering the elements relevant to the program structure with only minimal domain knowledge. Two grammar inference techniques, i.e., multiple sequence alignment and uniform resampling, are adopted in the second stage to infer a structural grammar for the program. Hence in the last parts of the thesis, we will detail each stage with experimental evaluations to show the effectiveness of the proposed framework.

Part I

Structural Element Discovery

Chapter 4

Determination of Structural Elements

Content

This thesis casts the task of recurrent program structuring as a grammatical inference task, as stated in Chapter 1, including two stages: structural element determination and structural grammar inference. With the main assumption that there is no prior knowledge about the program structure, i.e., about the structural elements that may be present, determining structural elements that are relevant to the structure of the program must be performed in an unsupervised manner. Having all discovered structural elements across episodes, the second stage consists in the grammar inference to infer an overall structural model for the program.

In this chapter, we focus on the first stage, determining the structural elements that are relevant to the structure of the program across episodes. To skirt the unsupervised issue, we search for elements that repeat across episodes with relative temporal stability,

giving priority to the basic and common elements. Practically, we apply a large amount of audiovisual feature detectors on the collection of episodes from a program to detect general purpose events. By analyzing their temporal distribution, events exhibiting the property of repetitiveness are filtered. The repeated events are supposed to be relevant to the structure, hence being identified as structural elements with the corresponding types based on prior domain knowledge of TV programs.

4.1 Broad scope event detection

In order to determine structural elements generic enough for various types of programs in an unsupervised manner, the most straightforward solution is to search for all possible recurrent elements throughout the episodes and select the ones which are relevant to the structure of the program. Ideally, a large number of event detectors should be used, which are generic enough to apply to a large number of programs. However, this is not very practical because of implementation and run time issues. A number of key event detectors must therefore be selected based on a trade-off between the type of programs to process, the complexity of run time and, to a lesser extent, the implementation complexity. Considering a trade-off between these three issues, nine key detectors are adopted to detect general purpose events that may potentially be relevant to the program structure.

4.1.1 Visual detector

Among the nine key detectors, seven are visual detectors, which focus on basic visual features.

4.1.1.1 Shot transition detection: Hard cut detector

Shots are basic units for program structures which are required for structuring purposes. Hard cut is the most common shot transition for TV programs, which segments a program into different sequences of frames running for an uninterrupted moment. In addition, hard cut detection also serves for other event detection, such as person clustering or shot reverse shot detection. Hence it is important to detect hard cuts for structuring programs into shots. Hard cut is a sudden transition from one shot to another, i.e. one frame belongs to the first shot, the next frame belongs to the second shot. Figure 4.1 shows an example of a hard cut.

Figure 4.1: Shot transition: Hard cut

Figure 4.2: Shot transition: Dissolve

For seeking hard cuts, shot boundaries are detected using the implementation of J. Mathe et al. [1]. The technique is based on color histogram comparison between two successive frames. The result produced is a simple XML file containing the start time (frame) and end time (frame) of each detected shot. Furthermore, each shot can be labeled as short shot (e.g., less than 30 frames), medium shot (e.g., between 30 and 60 frames) or long shot (e.g., more than 60 frames) considering the length of shots.

4.1.1.2 Shot transition detection: Dissolve detector

Gradual transitions are also frequently seen in TV programs, which usually signal the start and end of a scene. We focused on dissolve transitions, the most common gradual transition. A dissolve overlaps two shots for the duration of the transition from the first shot to the second shot. An example of a dissolve is illustrated in Figure 4.2.

Dissolve is detected using an extension of [49]. As explained by Figure 4.2, for a dissolve transition, there always is a frame being approximately the overlap of two frames from the first and the second shot respectively. In order to detect dissolve transitions, we leverage a dissolve window going through all the frames and verifying their chromatic difference.

Figure 4.3 illustrates procedure of the dissolve transition detection. Given a set of

1. http://johmathe.name/shotdetect.html

frames, denoted f_0, f_1, ..., f_n, a window with length N ($N << n$) runs through the frames. We suppose that there exists a frame f_k with its intensity $I(x, y, f_k)$ equal to the average intensity of the start and end frames of the dissolve window:

$$I(x, y, f_k) = \frac{I(x, y, F_1) + I(x, y, F_N)}{2} \ , \tag{4.1}$$

where F_1 and F_N are start and end frames of the dissolve window.

In order to measure whether there exists a dissolve transition, a factor, $F_{dissolve}$, is defined as:

$$F_{dissolve} = min\left(\frac{\sum_{x,y}(\mid \frac{I(x, y, F_1) + I(x, y, F_N)}{2} - I(x, y, f_k) \mid)}{\sum_{x,y}(\mid I(x, y, F_1) - I(x, y, F_N) \mid)}\right)(k = 2, 3, ..., N - 1) \ , \tag{4.2}$$

In each displacement of the window, a $F_{dissolve}$ is calculated. Hence, there will be a sequence of $F_{dissolve}$ after the window goes through all the frames. Considering the definition of the factor $F_{dissolve}$, a small value of $F_{dissolve}$ implies that it is more possible for a dissolve window being a dissolve transition, or conversely.

Hence, we propose to use the threshold T_f to select the dissolves, which is automatically determined by the Gaussian distribution:

$$T_f = \mu - 3 * \sigma \ , \tag{4.3}$$

where μ represents the expectation of the set of $F_{dissolve}$, and σ is its standard deviation. As for a Gaussian distribution, about 99% of the values lie within the tolerance interval $[\mu - 3 * \sigma, \mu + 3 * \sigma]$, where the values of $F_{dissolve}$ represent the case that there dose not exist a dissolve transition. Therefore, the dissolve can be selected when the value of $F_{dissolve}$ is smaller than the threshold T_f. The positions of dissolves are deemed as the minimal peaks under the threshold.

Figure 4.4 gives two examples of dissolve detection: The blue one represents a sequence of frames without dissolve; the red one represents a sequence of frames with two dissolves. After applying the dissolve window running through the video, the time (frame) position of dissolve transitions can be detected. A quantitative evaluation of dissolve transition detection can be found in our publication [59], where dissolve transition detection is evaluated in terms of recall and precision with a good performance for various recurrent programs.

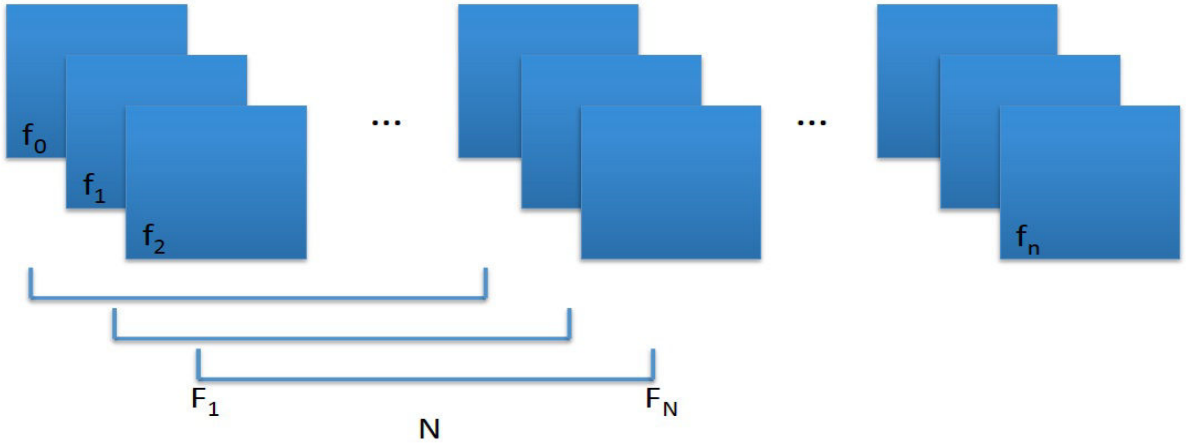


Figure 4.3: Illustration of dissolve detection

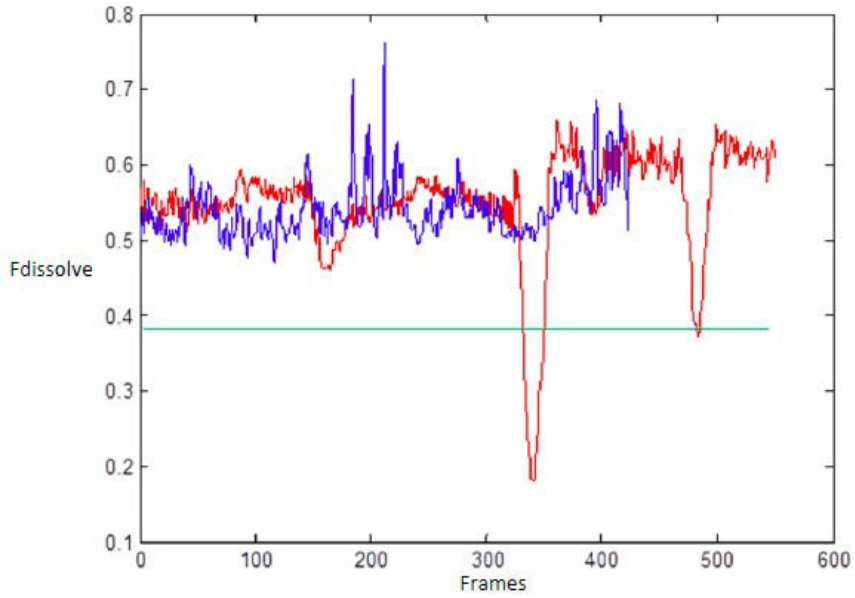


Figure 4.4: Two example of $F_{dissolve}$ for dissolve detection

4.1.1.3 Monochrome image detector

Monochrome images—mostly white or black—are usually added to the TV streams at edition time to separate the different parts of a program. They are mostly seen before or after separators, or between two commercials segments. Detecting monochrome images is therefore an obvious choice for structure discovery in TV programs, in particular due to the simplicity of its implementation.

In this work, monochrome images are detected by simply verifying the histogram variance of the images. For each frame, after converting to a gray image, the standard deviation of the its histogram is calculated. If its standard deviation is less than a mall value, e.g., 0.01, the frame is deemed as a monochrome image. Applying the detector to the frames of the video, we can label the time instant of all monochrome frames

4.1.1.4 Text region detector

Text region detection aims at detecting and localizing text(-like) regions in image. Text is often related to some scenes with particular meanings in programs, especially the scenes with full text on screen, such as a data summarization in a TV news or a question scene in a game show.

Text region detection in [17] is adopted for our use. The implementation of the proposed algorithm can be found on line[2]. It focuses on detecting regions containing text in unstructured scenes in an image, employing edge-enhanced Maximally Stable Extremal Regions (MSER) detectors to separate non-text regions from the text regions. Using connected component analysis and stroke width filtering, the bounding boxes enclosing text regions are then determined. The proposed text detection algorithm starts with a large number of text region candidates and progressively removes those less likely to contain text. By adopting the text region detector, all the frames containing texts are output with the bounding boxes of the text regions.

4.1.1.5 Motion activity detector

As said in the previous section, the scenes with full text on screen may show particular meaning for program structures. However, there also exist scenes with full text, but not actually contributing to the program structures, for example, a street scene with plenty of text panels. Practically, full text scenes on screen appearing with a still background (e.g., a data summarization in a TV news or a question scene in a game show) would make more sense to the program structures, which hints us to estimate motion activities of the frames to enhance the full text scenes.

We adopt a block matching algorithm [6] to estimate the motion activity in current frame with respect to the previous frame of a video. The idea behind block matching is

2. http://ch.mathworks.com/help/vision/examples/automatically-detect-and-recognize-text-in-natural-images.html

to divide the current frame into a matrix of 'macro blocks' that are then compared with corresponding block and its adjacent neighbors in the previous frame to create a vector that stipulates the movement of a macro block from one location to another in the previous frame. This movement calculated for all the macro blocks comprising a frame, constitutes the motion estimated in the current frame. The search area for a good macro block match, the matching of one macro block with another, is based on the output of a cost function, e.g., Mean Squared Error (MSE) or Mean Absolute Difference (MAD). The macro block that results in the least cost is the one that matches the closest to current block. Several search algorithms could be adopted to calculate the cost function at each possible location in the search window, e.g., exhaustive search in our case, finding the best possible match.

After running through the motion activity, each frame from the video can be estimated with a corresponding motion activity value, further labeled as low, medium or high motion activity. Motion activity is often associated with text region localization to detect still text scene, i.e., frames with large text regions and low motion activity.

4.1.1.6 Person clustering

Persons are essential features for almost any types of TV program. Especially, in many programs, a few number of key persons appear and are strong structure cues, such as the anchorperson in news shows or the host in game shows. TV show conductors usually appear as the most dominant face in a program, i.e., the one which appears most. Dominant person is usually implemented using person clustering based on faces and clothing. Taking news as an example, the anchorperson's clothes are usually carefully chosen so as to be easily distinguishable from guests (and one from another in case of multiple anchors) and obviously do not change within an episode. Hence person clustering is an obvious choice for discovering person cues for program structures.

Person clustering is implemented using Viola and Jones face detection [71] and dress bounding box determination [38], as illustrated in Figure 4.5. By applying face detection, the shots with only one face are firstly selected. For each key frame with a face, clothing histograms are then used in a K-means clustering to obtain person clusters, where the number of person clusters is automatically determined by verifying the quality of clusters [62]. Generally, from two clusters up to an upper limit on the number of clusters, it verifies a validation measure based on the intra-cluster and inter-cluster distance measure. The optimal value of the number of clusters is determined when the clustering procedure gives a minimum value for the validity measure.

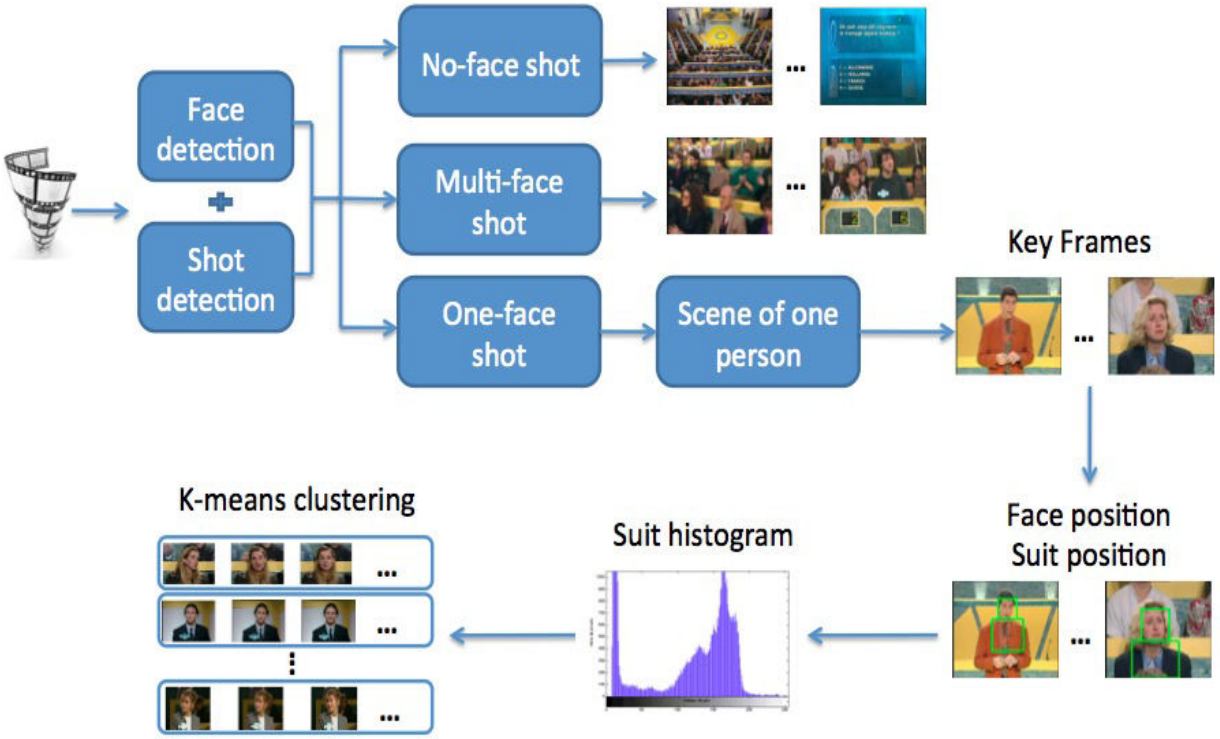


Figure 4.5: Illustration of Person clustering

The person clustering is applied to individual episode, classifying the persons from one episodes into different clusters. Figure 4.6 gives an example of several clusters from an episodes of GAME. The evaluation of the proposed person clustering can be found in the paper [59] by means of cluster purity, where the clustering results were found reliable enough for structure inference.

4.1.1.7 Shot reverse shot detector

Shot reverse shot is a classical video technique depicting shots alternating between two characters facing one another, usually engaged in some face to face interaction. In the case of TV programs, we assume that a segment of shot reverse shot represents a dialog and that such interaction between two characters are relevant to the structure.

Based on the results of person clustering, we detect shot reverse shot segments by a straightforward analysis of the cluster interlacing. We focus on just the shots containing one person and eliminate other shots. For the one-person shots, we detect the segments, where

Figure 4.6: Example of person clusters from GAME

two person alternate one after the other. The segment is considered as a shot reverse shot segment if two persons appearing alternatively and successively, without being interrupted by other shots (e.g., the shots containing no person or the shots containing the third person).

4.1.2 Audio detector

Besides, two audio detectors, i.e., speech/music/silence detector and audio recurrence detector, aim at detecting generic audio features for program structuring.

4.1.2.1 Speech/music/silence detector

Speech/music/silence detection evidently allows identifying different types of audio sequences. Music might refer to songs, jingles, etc. Evident silences might appear in commercials segments, (before or after) separators, etc. Also, outlines for a program, i.e., a brief summary of the main points of a show, are mostly found when speech and music appear at the same time.

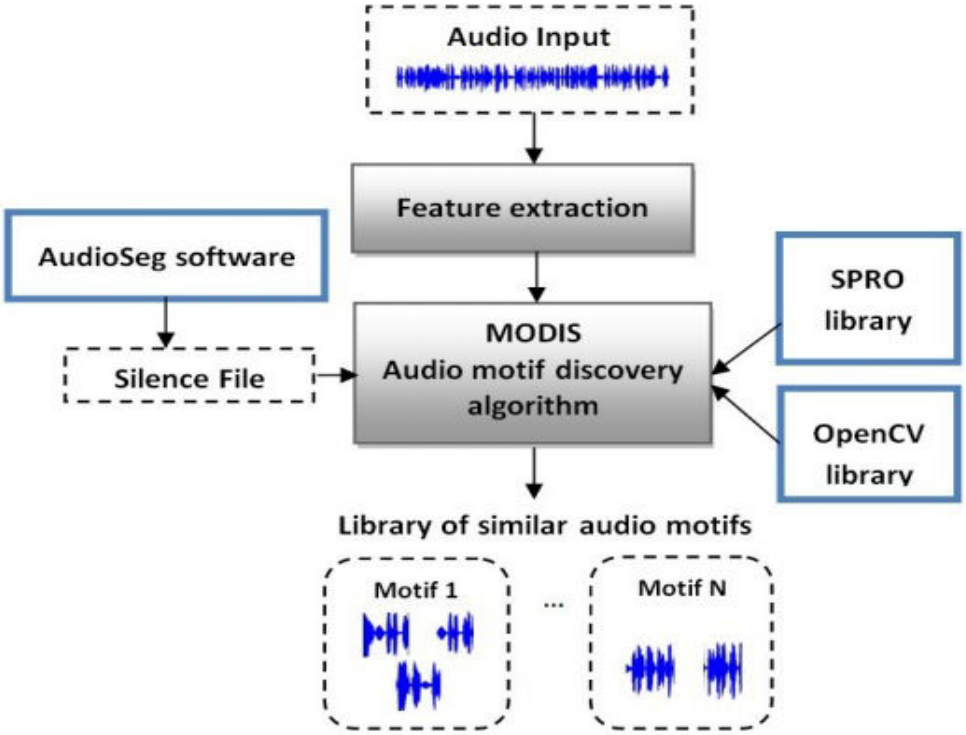


Figure 4.7: A general overview of software MODIS [14]

We adopt a speech/music/silence discrimination algorithm proposed in [53]. The audio sequence is at first segmented by detecting the change of audio types. A change is detected based on the distribution of the root mean square values, which differ between speech, music and silence. Based on the central frequency measured by the zero-crossing rate, classification is then performed for each segment extracted by the segmentation stage, discriminating the segments into one of the four categories: silence, music, speech, or music overlapping speech. We mention that the basic characteristics, i.e., the signal amplitude and the zero-crossings, are computed over 20 ms intervals, hence the output audio sequences labeled with corresponding audio types are specified with an accuracy of 20 ms.

4.1.2.2 Audio recurrence detector

Audio recurrence detection, based on the motif discovery MODIS software [14], aims at detecting identical audio sequences repeated in the audio stream, which allows detecting separators as they repeat within or among different episodes of a program.

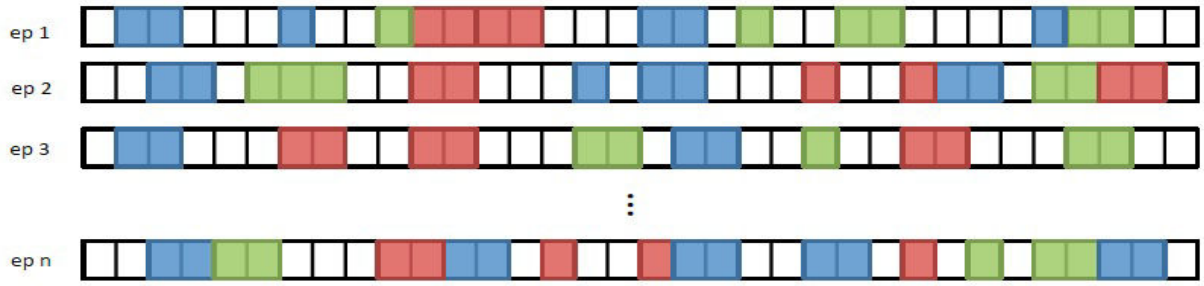


Figure 4.8: An illustrative example of detected general events by audiovisual feature detectors

A general overview of MODIS is illustrated in Figure 4.7. MODIS requires an audio features file input, e.g., Mel frequency cepstral coefficient. The detailed algorithm for mining repeating audio sequences is introduced in [51] and [50], where the search of local similarities between the seed block and a buffer is performed using variations of dynamic time wrapping procedure. The results of the audio recurrence discovery is a library of motifs. Each motif is a set of identical/quasi-identical occurrences, each occurrence being represented by a starting and an ending point relative to its location in the input audio stream.

By applying these event detectors, a large number of general events are detected. We keep the occurrences of each general event, with their temporal positions across episodes, and consider them as the ones which are potentially related to the program structures. Figure 4.8 shows an illustrative example of the general events detected by audiovisual feature detectors, different colors representing different types of general events. For instance, the red segments could be monochrome images, the blue ones could represent the positions of dissolve transitions. These general events need to be further analyzed to find the ones which are actually relevant to the program structures.

4.2 Event filtering

A considerable amount of general events is detected in the previous step. Obviously, not all are relevant to the structure. For instance, a short sequence of black frames could indicate a separator inserted between two successive parts of the program. It could however also be found in a night scene. The key idea that we exploit for the determination of the *structural elements* is firstly to find among the detected *general events* those that

exhibit the property of repetitiveness, i.e., *repeated events*, and to analyze the temporal distribution of the repeated events to select their *repeated occurrences*. Considering the example above, the event of monochrome images (i.e., a general event) is firstly analyzed to verify if it exhibits repetitiveness across episodes. If yes, it is deemed as a repeated event, and its occurrences are further filtered to select the repeated occurrences (i.e., the ones who contribute to the separators) and eliminate the sporadic ones (i.e., the ones who contribute to the night scene). In a second step, by means of minimal domain knowledge, the selected occurrences of the repeated events are, individually or jointly, identified as the structural elements which are named with their corresponding types, e.g., as separator or dialog. Repetitiveness is measured by means of temporal density filtering. Before applying density filtering, a complementary strategy, i.e., role recognition, is performed to detect the dominant person of each episodes, e.g., anchorperson or conductor, as it is an important clue of program structures.

4.2.1 Role recognition

Role recognition is adopted to further characterize the outcome of person clustering and identify the most important person of each episode, such as the conductor or the anchor, which is clearly a strong cue with respect to the program structure. Since no prior knowledge of the programs' content, we assume that there always is a dominant person for each episode. In the case of a program without dominant person, the person who is mistaken for the dominant person will be eliminated in the step of event filtering, as the role recognition here serves for finding repeated elements related to anchor persons. A dominant person is assumed to ideally have the following properties: he/she is the one that appears most; he/she is filmed the most frequently; he/she participates the most frequently in dialog; his/her range of appearance and longest time of appearance should not be the lowest.

Based on these insights, we use five measures [39] to characterize each person cluster with the idea of finding the person that covers at best a significant amount of time in an episode:

- Total duration of appearance;
- Total number of distinct appearances, i.e., number of non consecutive segments;
- Duration of the longest segment in which the person appears;
- Time range between the first and last occurrence;

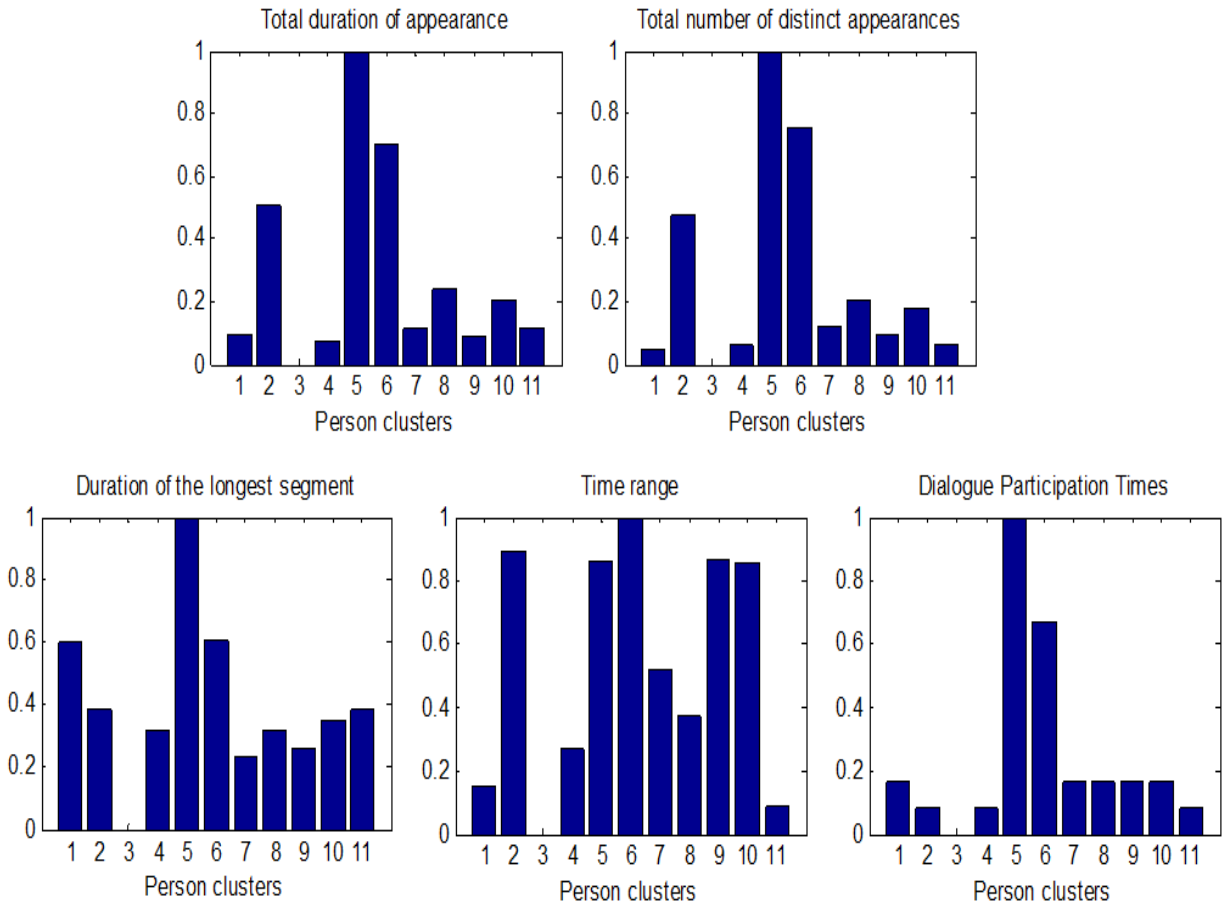


Figure 4.9: Time related features for person clusters of an episode from GAME

- Number of times in which the person is engaged in a dialog, i.e., the person is found in a shot reverse shot segment.

Given such features, a dominant person of each episode is determined by verifying these defined features. To account for varying episodes and program lengths, all five measures are scaled to $[0, 1]$. Decision on the dominant person is made based on the sum of the five normalized measures, the cluster having the maximal sum being identified as the dominant person. Figure 4.9 shows these features for person clusters of an episode from the program GAME. There are eleven clusters in total, each cluster representing a person. The defined features of each cluster are summed up and compared to find the maximum, hence to determine the most dominant person. In the case of the example, it is evident that the 5^{th} cluster, having the most significant time related features, corresponds to the dominant person for the episode.

Once the dominant person of each episode is identified, the shots containing the dominant person are considered as a general event and will be further analyzed using temporal density filtering, as all other general events.

4.2.2 Temporal density filtering

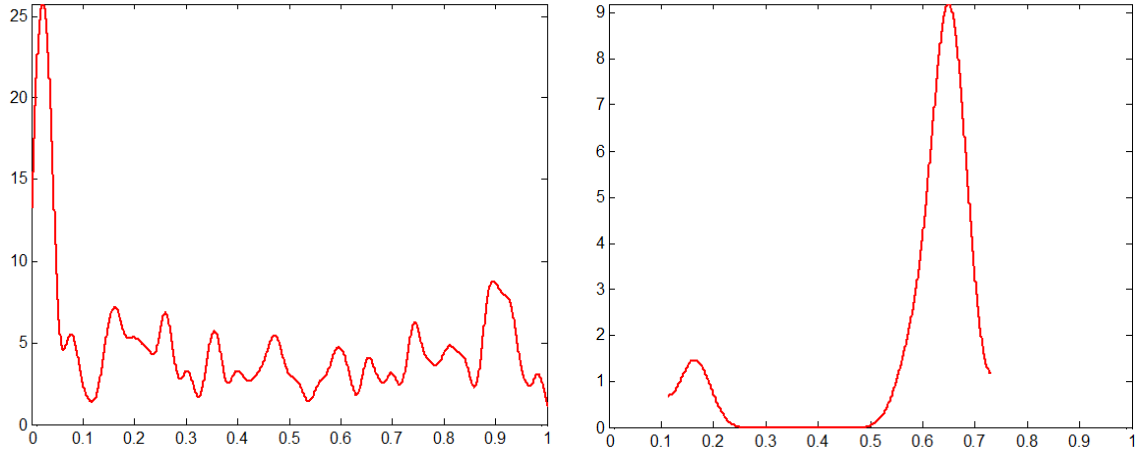
The key property that we are using to determine the structural elements is their repetitiveness, i.e., they appear at roughly the same time instant in all episodes. Therefore, density filtering is adopted to analyze the temporal distribution across episodes of each general event detected by audiovisual detectors. The idea is to first find the events that repeat across episodes, then to select the segments of the events with high density of its temporal distributions. These segments are likely to be the occurrences that are significantly repeated and thus deemed relevant to the program structure. Isolated occurrences are probably the ones appearing quasi-randomly without any temporal stability.

4.2.2.1 Repeated event filtering

A prior filtering step is firstly adopted to remove the general events that do not exhibit the property of repetitiveness. In particular, a general event is deemed as not repeating across episodes, if one of the two case occurs: First, the occurrences of a general event only come from a small minority of episodes, i.e., less than one third of the total number of episodes in the collection. For example, a sequence of monochrome images just found in one single episode is very unlikely to be relevant to the program structure. It is probably just a night scene. Second, a general event whose segment of occurrences with highest concentration of the temporal distribution (which will be detailed in next sections) does not reach a given value, i.e., one third of the total number of episodes in the collection. For instance, the music sequence with its occurrences appearing in all episodes but not found in similar temporal positions across episodes, is also not considered as a repeated event. It may be just different scenes with musical accompaniment. We thus eliminate the general events exhibiting no repetitiveness, then analyze the temporal density of the repeated events to select their repeated occurrences.

4.2.2.2 Repeated occurrence filtering

For a collection of episodes, considering different episodes with various lengths, we normalize the length of each episode. For each type of repeated event, we project onto



(a) Density function of *music&speech* for NEWS (b) Density function of *repeated audio sequence* for TALK

Figure 4.10: Examples of density function for NEWS

the same temporal axis the occurrences across episodes of the event, and measure for its empirical density across episodes at every time instant. In other words, we estimate the probability that a general event appears at a given time instant based on all the episodes. To avoid artifacts and account for slight time variations, a kernel-based density estimator is used in practice. Formally, for an event, we compute at each time instant the density as

$$f(t;h) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{t-t_i}{h}\right) \quad , \tag{4.4}$$

where n is the number of instances of the event considered, t_i is the time position of the i^{th} occurrence calculated as the mean of the start and end times of the i^{th} occurrence. K is a zero mean unit variance Gaussian kernel function whose optimal bandwidth h is automatically chosen as in [10]. In plain words, $f(t;h)$ measures how frequently an event occurs at time t across episodes.

Figure 4.10 shows two examples of the estimated density function. Obviously, in Figure 4.10(a) the segments of *music&speech* have high concentrations at the beginning of the program, corresponding to the max peak in the density function. Figure 4.10(b) shows the temporal distributions of *repeated audio sequences* for TALK, where evident areas of repetitiveness can be found. Having the temporal density of the repeated events, we aim at selecting the areas with high concentrations of the events to determine their repeated occurrences.

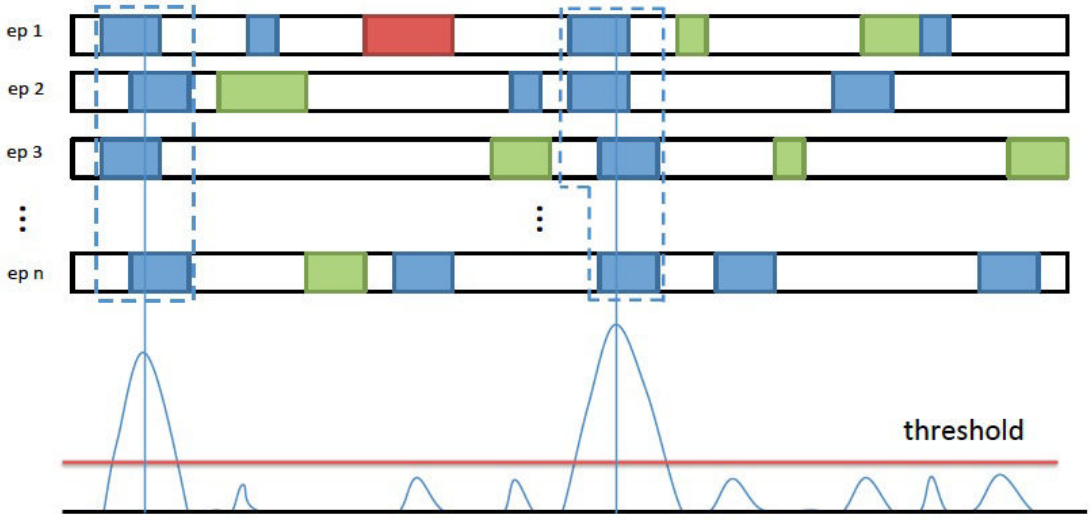


Figure 4.11: Example of repeated event filtering

Selecting repeated occurrences of a repeated event is finally performed by thresholding $f(t;h)$, considering only frequent occurrences as structurally relevant and ignoring sporadic ones. In order to adapt this threshold to each particular case of program, the threshold is empirically determined as a fraction of the mean of all the peak values in the density function, defined as:

$$threshold = \frac{\frac{\sum_{i=1}^{m} p_i}{m}}{n} \tag{4.5}$$

where p_i is the local peak values of the density function and m is number of the peaks. The fraction n is a parameter that can be experimentally determined. All the peaks that are under this threshold are rejected, and consequently the corresponding occurrences of the general events too. The choice of the threshold, i.e., the choice of the fraction n, will be discussed in Chapter 5.

Figure 4.11 illustrates temporal density filtering on a collection of episodes, depicting the occurrences of three detected general events (colored rectangles). The red and green events are the ones exhibiting no repetitiveness across episodes and need to be removed. The red event is found just in one episode and the occurrences of the green event do not appear in similar temporal positions across episodes. The blue event is a repeated event with its occurrences found in similar temporal positions across episodes. A density function is estimated considering all the occurrences projected on the same axis. The occurrences enclosed in dashed boxes are selected by thresholding the density function $f(t;h)$, while the occurrences out of the dashed boxes are discard. The selected occurrences are deems

as repeated ones with the type of monochrome image.

4.3 Structural element identification

From the selected repeated events as well as their repeated occurrences, we identify the structural elements of a program using a set of rules based on minimal domain knowledge referring to common practice in TV programs. As opposed to repeated events that have no particular meaning per se, structural elements are syntactic units that compose the different parts of the program, similar to words in a grammatical inference task. For instance, a structural element corresponding to a sequence of white images (the event) is a separator (the structural element), while a long duration shot containing the dominant person at the beginning of the program (events) is deemed as an anchor's opening (structural element). There are cases that we need more than one repeated events to identify a structural element. Commercials is a typical such structural element, identified by jointly considering three repeated events, i.e., silences, monochrome images and shots with short duration. With these basic rules of identifying structural elements in TV program domain, we can identify the structural elements for the programs based on the repeated events.

For the four datasets, different types of structural elements are presented in Table 4.1, where the corresponding event detectors of each structural element are listed. Specifically, if an event detector contributes to identifying a specific structural element, it is marked as "yes" or with the corresponding result. Some structural elements can be simply determined by a sole event, such as dialogs, music/songs or separators for NEWS, TALK and MAGZ. While some structural elements are discovered by considering more than one event. Person cluster results with long shot duration are used to find person's monologues, where anchor person's shot can be found by dominant person prediction. Besides, commercials are characterized by the presence of (any combination of) black frame, short shot duration and audio silence. In NEWS the outlines pronouncing the main topics in each episode always has short shot duration and anchor person's speech with back ground music. With this basic domain knowledge, all structural elements are identified and named with their types.

Figure 4.12 illustrates some examples of elements determined based on visual detectors. For instance, Separators for NEWS are detected by sole white images (monochrome images), while separators for GAME are determined by both a dissolve transition and black images (monochrome images). With centralized texts and low motion activities, a full text screen scene can also be identified for GAME.

Table 4.1: Structural elements determination using a broad scope of detectors

Structural element	Symbol	SD	DT	MI	CT	MA	DP	SRS	SMS	AR
Separator NEWS	S			yes						
Separator GAME	S		yes	yes						
Separator TALK	S									yes
Separator MAGZ	S									yes
Dialog	D							yes		
Anchor's monologue	A	long					yes			
Music/song	M								music	
Commercials	C	short		yes					silence	
Outline NEWS	T	short							music&speech	
Full screen text	E				yes	low				

SD: Shot duration
DT: Dissolve transition
MI: Monochrome image
CT: Centralized text
MA: Motion activity
DP: Dominant person
SRS: Shot-reverse-shot
SMS: Speech/music/silence
AR: Audio recurrence

(a) Separator with monochrome images for NEWS

(b) Separator with dissolve and monochrome images for GAME

(c) Full screen scene with centralized text and low motion activity for GAME

Figure 4.12: Examples of elements determined based on visual detectors

We emphasize that the identification of structural elements is the only step in the whole process where domain knowledge is required, however being limited to standard edition rules that apply to a wide variety of programs.

4.4 Conclusion

This chapter addresses the problem of determining structural elements for various types of recurrent programs in an unsupervised manner. We have shown that, by leveraging the property of repetitiveness, structural element can be discovered using a broad scope of event detectors and density filtering techniques, with only minimal domain knowledge. Comparing with previous work, which either focuses on finding a certain kind of structural element or utilizes massive of prior knowledge of a certain program to detect its structural elements, we are able to break the limitations of these previous work. Without prior knowledge of the program structure, the proposed approach takes advantage of repetitiveness of recurrent programs, making it possible to covering various of structural elements for numerous categories of programs.

The proposed determination approach is a big step of advance to explore common structural elements for various types of programs in the absence of program structures. In order to show its effectiveness, we will report in Chapter 5 the experimental evaluations of the determination of structural elements on recurrent programs of different types.

Chapter 5

Experimental Evaluation for Structural Element Determination

Content

Evidently, the performance of structural element detection is strongly influenced by the repeated events, determined by density filtering. As presented in the last chapter, the threshold applied to the density function is obviously a factor who has direct influence on the determination of repeated events. Another importance factor is the number of episodes utilized, which has influence on the produced density function, hence indirectly affecting the determination of repeated events. Therefore we hereunder evaluate the performance of repeated events, considering two different factors respectively, i.e., the threshold applied on the density function and the number of episodes utilized for density filtering.

5.1 Analysis of the threshold applied on the density function

In the last chapter, we have presented the use of a temporal filter needed to select the repeated events, where a threshold is applied on the density function. To recall, the threshold is defined as $threshold = \frac{\frac{\sum_{i=1}^{m} p_i}{m}}{n}$, i.e., a fraction of the mean of all the peak values in the density function.

In order to adapt this threshold to each particular case of program, the choice of the parameter n in the threshold is crucial to filter out the minor peaks, i.e., areas with isolated occurrences of general events, and keep the major peaks, i.e., areas with the repeated ones. As shown in the illustrative example in Figure 5.1, for the density function of a general event, when increasing n, the threshold decreases from $threshold_1$ to $threshold_3$, and the number of repeated segments selected by the temporal filter augments, the same as the selected occurrences of the general event. When the threshold keeps decreasing, from $threshold_3$ to $threshold_4$, the number of repeated segments gets increasing, and the selected occurrences of the event keep augmenting. In plain words, a very high threshold, corresponding to small values of n, filters out not only the sporadic occurrences of the event but also the repeated ones. Conversely, a very small threshold, corresponding to big values of n, keeps not only the repeated occurrences but also the sporadic ones. Hence an appropriate threshold needs to be chosen for keeping the repeated occurrences and eliminating the sporadic ones.

To precisely detect which part of the peaks should be considered, we conduct experiments by varying the parameter n. Three evaluation metrics, i.e., precision, recall and F measure, are adopted to examine the influence that the parameter produces on the occurrences of repeated events. The three metrics in our cases are defined as follows:

$$P = \frac{N_p}{N_{re}} \ , \tag{5.1}$$

$$R = \frac{N_p}{N_{se}} \ , \tag{5.2}$$

$$F\ measure = \frac{2 * P * R}{P + R} \ , \tag{5.3}$$

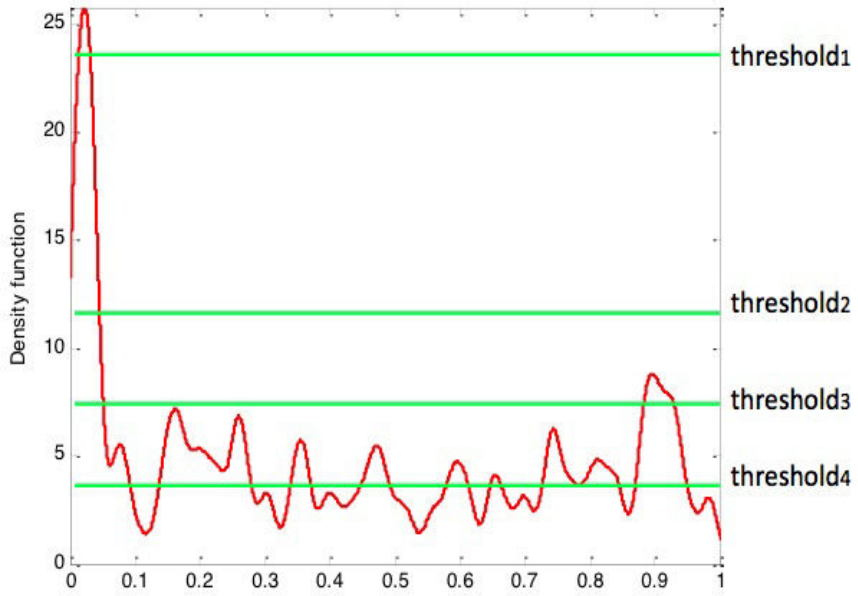
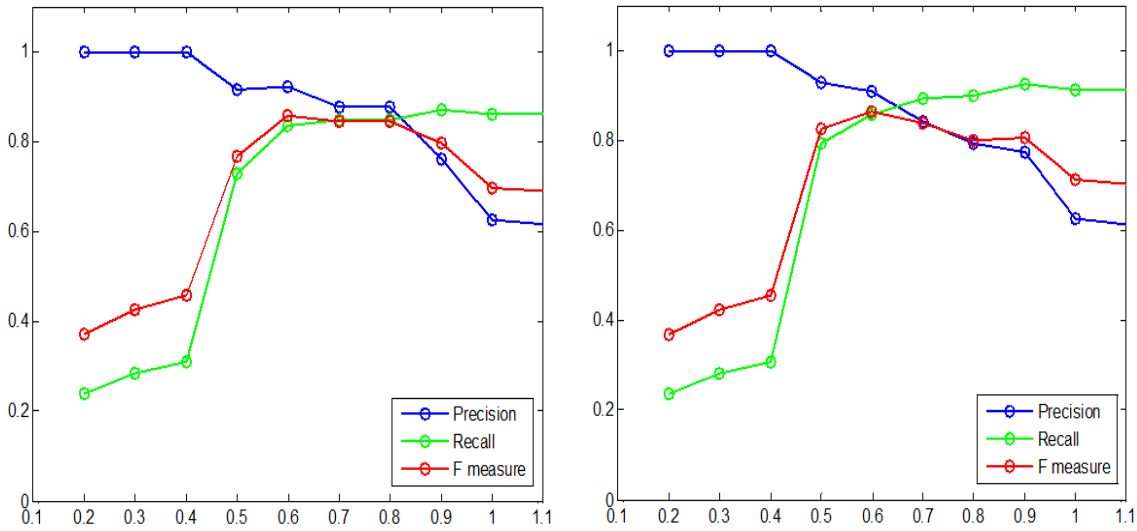


Figure 5.1: An example to illustrate the threshold applied on the density function

N_{re} refers to the total number of selected occurrences after adopting the given threshold; N_p refers to, among all the selected occurrences, the ones which contribute to the identified structural elements; N_{se} refers to, for the manually labeled structural elements in the collection of episodes, the number of occurrences of the repeated events which contribute to the structural elements .

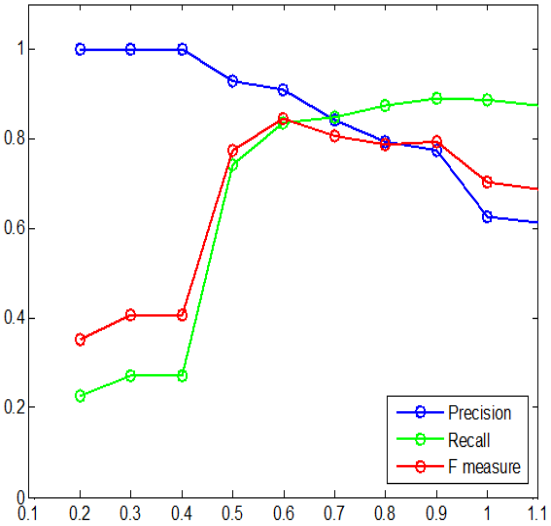
As mentioned previously, the number of episodes also produces influences on the estimated density function, hence we fix several numbers of episodes used and observe the effect of the parameter n. We conduct the experiments on the four programs, i.e., NEWS, GAME, TALK and MAGZ, respectively. In order to choose the best value of the parameter n for various types of programs, for each given number of episodes, we average the results of the four data sets (Individual results for the four datasets can be found in Appendix) and obtain the averages of precision, recall and F measure, as illustrated in Figure 5.2.

In the figures, on the vertical axis, the obtained values for the precision, recall and F measure are illustrated; on the horizontal axis, the different values corresponding to the parameter n are given. As shown, the increase of the parameter n (decrease of the threshold) results in an increase of recall and a decrease of precision, as more sporadic occurrences are involved. Consequently, F measure exhibits a decrease after an initial increase. When F measure goes to its maximum, it implies a best trade-off between precision and recall. Based on the tables for the cases of different numbers of episodes, when the parameter



(a) ep = 6

(b) ep = 9



(c) ep = 12

Figure 5.2: Precision, recall and F measure for an average of NEWS, GAME, TALK and MAGZ (ep: number of episodes involved for the evaluation)

n lies between 0.5 and 0.8, the F measure shows its highest values. Among them, when $n = 0.6$, the average of F measures stabilizes to a maximum value. Consequently, from now on, we will use $n = 0.6$ for the threshold for the temporal filters.

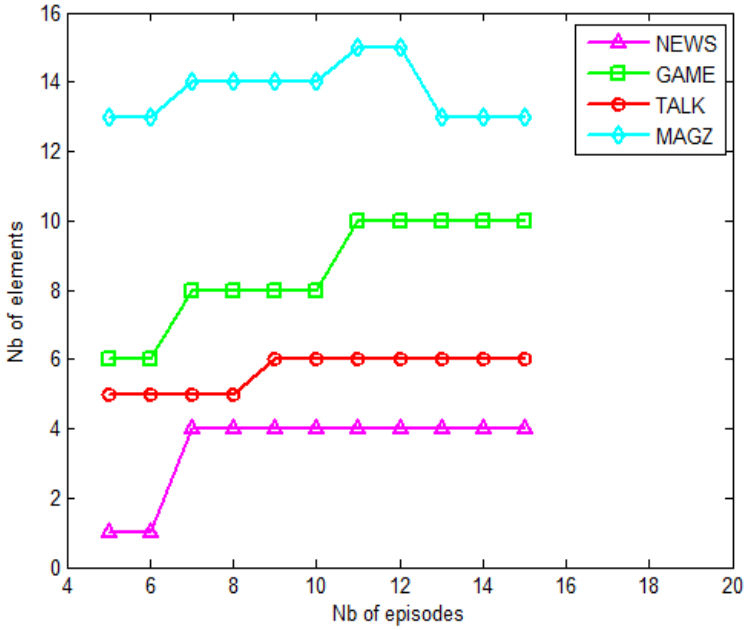


Figure 5.3: Number of events determined as a function of the number of episodes for the four types of programs

5.2 Analysis of the number of episodes used for determining structural elements

After fixing the threshold of the temporal filters, we have to choose a promising number of episodes involved for the structural element determination. In particular, the number of episodes involved strongly influences the produced density function, affecting indirectly the number of repeated events. Hence it is crucial to find an appropriate number of episodes for determining the structural elements, to further infer structural grammars.

In order to choose the number of episodes used for determining structural elements, we fix the threshold and analyze the effect that the number of episodes may produce on the number of structural elements that is determined. Figure 5.3 reports the number of repeated events determined when varying the number of episodes involved in the density filtering step. On the vertical axis, the number of obtained repeated events is illustrated. On the horizontal axis, the different values corresponding to the number of episodes utilized for the structural element determination task.

We conduct the experiments on the same programs, i.e., NEWS, GAME, TALK and MAGZ. Evidently, a very small amount of episodes is not sufficient to conduct density filtering with high confidence. On the contrary, repeated events may be drowned in a large number of episodes. For all four data sets, with a small quantity of episodes, i.e., less than nine episodes, the numbers of events increases along with the number of episodes. For NEWS, GAME and TALK, the number of repeated events tend to be stable as the amount of episodes increases above 9. For MAGZ, however, the number of events discovered drops down after 12 episodes. This can be explained by the fact that some events, such as the ones corresponding to separators, are very short audio or visual sequences, which are easily drowned in a large number of episodes. Based on these observations, 11 or 12 episodes are the promising number of episodes and considered as a collection used for structural element determination task.

5.3 Conclusion

This chapter reports the experiments on structural element determination. As repeated events are selected based on the temporal distribution of general events, we study the threshold that is applied on the density functions by analyzing a series of possible values of the parameter which consists of the threshold, and choose the most appropriate one based on quantitative examinations. Furthermore, a promising number of episodes for the collection utilized for element discovery is also determined by observing the number of repeated events.

Having fixing the threshold for density filers and the number of episodes in the collection used for structural element determination, we can discover a promising number of structural elements relevant to program structures. Moving to the stage of grammar inference, we infer the structural grammars and build the grammar models using the same collection for each recurrent programs.

Part II

Structural Grammar Inference

Chapter 6

Grammar Inference by Multiple Sequence Alignment

Content

Having a collection of episodes with their structural elements, identified after temporal density filtering of general events, the next stage is to infer a grammar of the corresponding program. The key idea is to find an optimal alignment of the structural elements between episodes to bring to light regularities, so as to discover a structural model representing the overall structure of the program. The alignment in the thesis is proposed to be done in one of two ways, using multiple sequence alignment or resampling uniformly the episodes to apply direct comparison. We will introduce them separately, multiple sequence alignment in this chapter and uniform resampling in Chapter 7. The two alignment techniques have their distinct features and properties. Consequentially, they yield grammars having their own features. We will experimentally compare the pros and cons of the two techniques in light of a segmentation use-case in Chapter 8.

In this chapter, multiple sequence alignment techniques are firstly introduced, followed

by a hierarchical architecture designed to enhance the grammatical inference, providing high quality structural grammars. At last, based on the grammar, a grammar model is built to be later utilized as a structural model for segmentation tasks.

6.1 Multiple sequence alignment

The temporal stability of recurrent TV program structure results in similar episodes in terms of the structure, however slight differences exist among different episodes. Multiple sequence alignment is a promising choice to discover a common pattern shared by the episodes, as the key of finding the common pattern shared by the episodes is to extract the commonality and eliminate variations of the episodes, which is exactly what multiple sequence alignment techniques focus on.

6.1.1 General presentation of multiple sequence alignment

Multiple sequence alignment techniques are mostly used for biological sequences, generally protein, DNA, or RNA. They aim at aligning a set of biological sequences and outputting the set of sequences, where homology can be inferred and the evolutionary relationships between the sequences can be obtained [1]. Specifically, across the output sequences, the biological symbols in a given position are homologous, superposable or play a common functional role.

There exist many multiple sequence alignment techniques. Considering our case, the input sequences is a small quantity of the episodes, i.e., about a dozen of episodes.. We here propose to use ClustalW [67], because that it is a general purpose multiple sequence alignment tool for three or more sequences. It does not specifically target a large amount of sequences with great length comparing with other multiple sequence techniques, which have high speed or additional functions but requiring more complexities and computations. Besides, ClustalW can perfectly align a set of sequences with different length, which is the common case for the episodes of determined structural elements in our case. Additionally, ClustalW is one of the highly recommended sequential tools for multiple sequence alignment in bioinformatics due to its high accuracy, effectiveness and free availability [43]. Before applying multiple sequence alignment to grammatical inference tasks, on which we are focusing, we generally explain how ClustalW works.

1. http://www.ebi.ac.uk/Tools/msa/

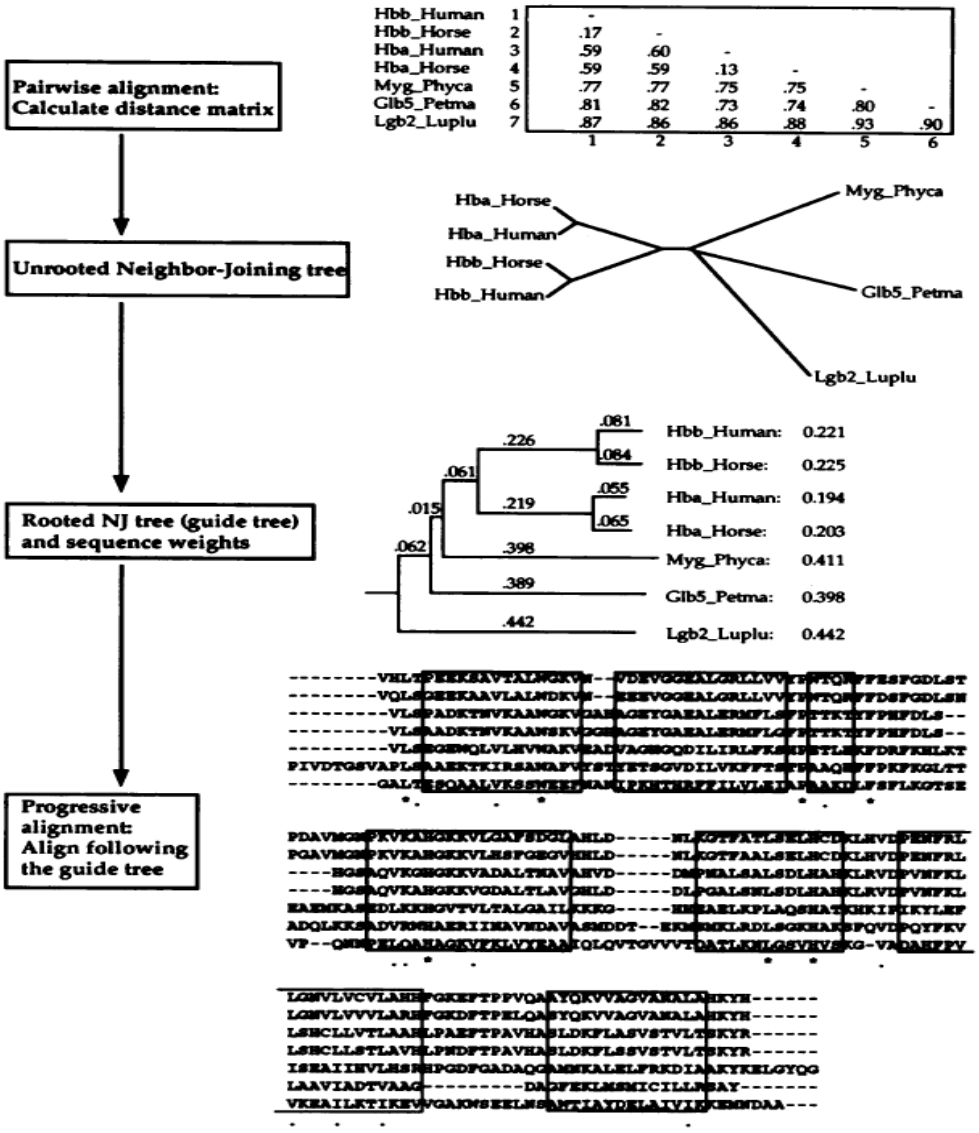


Figure 6.1: Illustration of the basic progressive alignment procedure [67]

The algorithm uses a progressive alignment technique, which builds up multiple alignments progressively by a series of pairwise alignments. It consists of three main stages: pairwise alignment, guide tree and progressive alignment. We now detail each stage by giving an example of seven globin sequences in Figure 6.1.

For multiple sequences, all pairs of sequences are aligned separately in order to calculate a distance matrix giving the divergence of each pair of sequences. Using the full dynamic programming method, scores of pairwise sequences are firstly calculated as the number of

identities in the best alignment divided by the number of residues. Then the distances of pairwise sequences are obtained by subtracting the scores from one. As in Figure 6.1, a 7x7 distance matrix between the 7 globin sequences is calculated, from which the trees used to guide the final multiple alignment process are generated.

Based on the relationship of pairwise alignments, an unrooted tree is firstly built using Neighbor-Joining method [63]. The branch length of the unrooted tree is proportional to the estimated divergence along each branch, and the root is placed by the 'mid-point' method [66] at the position where the means of the branch lengths on either side of the root are equal. Following the unrooted tree, a rooted tree can be derived with a weight for each branch. The weight of each sequence is dependent on the distance from the root of the unrooted tree, where sequences having a common branch with other sequences share the weight derived from the shared branch. Concretely, in the example in Figure 6.1, the sequence 'Hbb_humain' gets a weight consisting of the length of the branch leading to it, which is the length of the branch not shared with any other sequences (0.081) plus half the length of the branch shared with the 'Hbb_Horse' (0.226/2) plus one quarter the length of the branch shared by the first four globins (0.061/4) plus on fifth the length of the branch shared by the first five globins (0.015/5) plus one sixth the length of the branch shared by all the six globins (0.062/6). This sum is to a total of 0.221, which is the weight for the sequence 'Hbb_humain'.

Following the branching order of the rooted tree, the basic procedure of the last stage is to use a series of pairwise alignments to align larger and larger groups of sequences, where the most closely related sequences are firstly aligned, gradually progressing to the most distant ones. The order of progressive alignment is given in Figure 6.2. Each step consists of aligning two sequences. The alignment starts with the pair of sequences at the tips, progressing to the sequences to the node connecting those tips. Hence a multiple sequence alignment is built for each internal node of the tree, where the alignment at a given internal node contains all of the sequences connecting to that node. At each alignment, gaps may be introduced to fill up a possible loss or gain of a symbol, while the old gaps introduced in previous alignments remain fixed. During the process of the progressive alignment, the weights of sequences are used for a scoring function i.e., down-weight near-duplicate sequences and up-weight the most divergence ones, correcting the bias of the input sequences that are rather distantly related. Pairwise alignment of very closely related sequences can be carried out very accurately, leading to the accurate multiple alignment. By the time that the most distantly related sequences are aligned, we can have a set of output sequences, which gives important information about the commonality and variability of the biological symbols at each position.

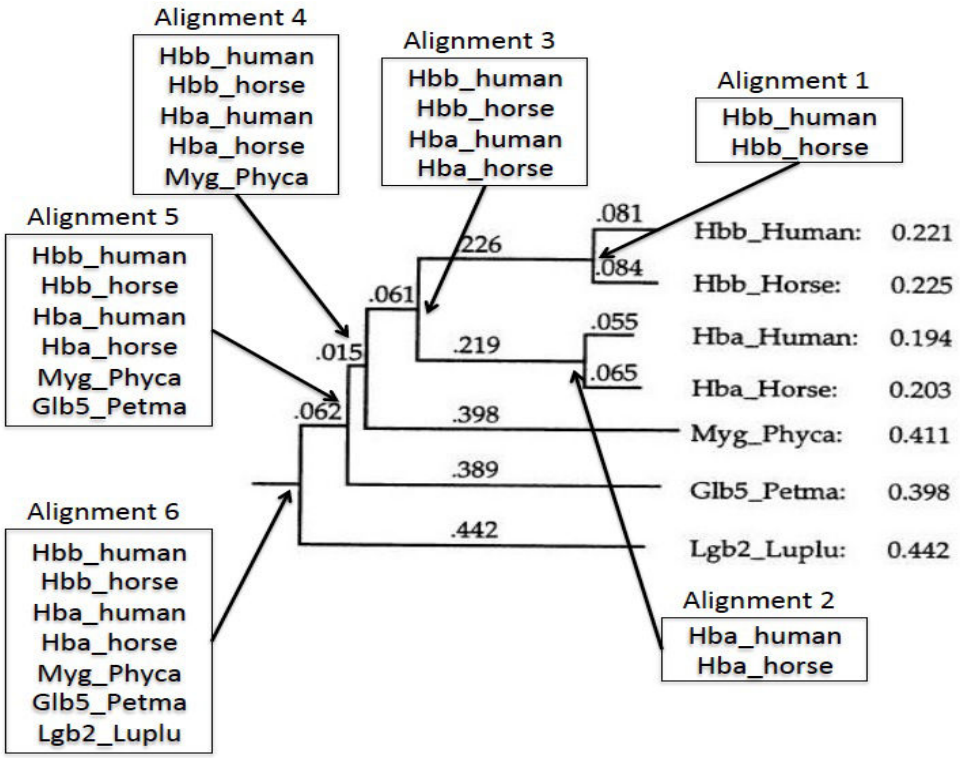


Figure 6.2: Progressive alignment order

6.1.2 Multiple sequence alignment for grammar inference

In our work, multiple sequence alignment is used to align the set of episodes labeled with structural elements, and to discover a structural grammar across these episodes. As described in previous chapters, a symbolic representation was adopted after identifying the structural elements, where each structural element is an alphabet symbol. As a result, each episode is represented as a sequence of alphabet letters representing the sequence of structural elements present. Looking from the projection of biological sequences, a symbolized structural element corresponds to a biological symbol, while an episode corresponds to a biological sequence. Alignment of the symbolic sequences is done in the way that alphabet symbols in a given position are homologous, superposable or play a common functional role, thus permitting to derive a model from the alignment.

With ClustalW, the whole collection of symbolic episodes is aligned, based on which a common pattern can be derived for the program structure. The process of multiple sequence alignment is illustrated in Figure 6.3, where three structural elements are identified beforehand. After the symbolization, the elements are represented by different symbols,

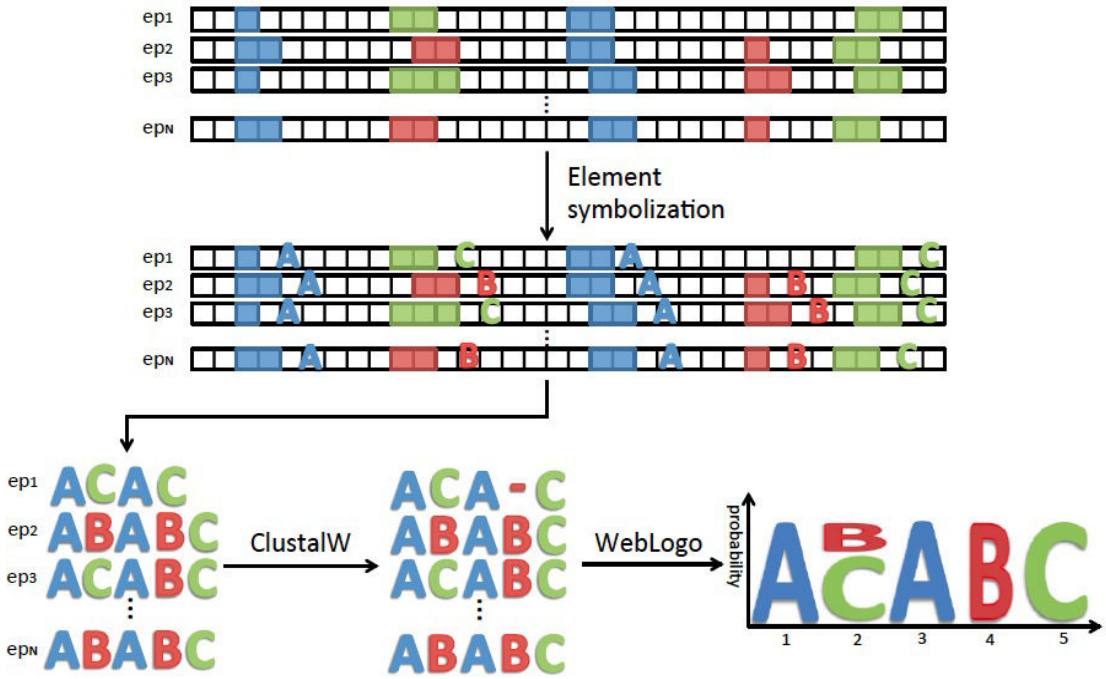


Figure 6.3: Illustration of multiple sequence alignment from a collection of episodes

i.e., A, B, C. The set of symbolic sequences is aligned by ClustalW, where a gap in the alignment indicates a possible loss or gain of a symbol. Having aligned sequences means that homology between sequences is inferred, based on which we are able to derive a common pattern shared by the sequences, i.e., a grammar for the collection of episodes. A grammar of program refers to an optimal alignment shared by the collection of episodes in a form of graphical representation. There exist many possible graphical representations for deriving a common structure from the aligned episodes. We here propose to use the WebLogo [20] representation for its concise visualization. WebLogo is originally designed for a graphical representation of biological multiple sequence alignment. Each logo consists of stacks of symbols, one stack for each position in the sequence. In our case, a stack of symbols is used to illustrate each position in the grammar: The overall height of symbols within the stack indicates the relative frequency of each symbol while the stack width is proportional to the fraction of valid symbols in that position.

In summary, by adopting the symbolic representation and the technique of multiple sequence alignment, we are able to find the regularities between the episodes labeled with determined structural elements. A structural grammar can be derived directly from the aligned sequences in the form of WebLogo.

6.2 Hierarchical architecture

The multiple sequence alignment technique can be effectively used for modeling recurrent programs with concise structures, i.e., the ones with short duration or limited numbers of structural elements. However it may fail in the case of programs with more complex structures, i.e., the ones with longer duration or more structural elements. The reason is that increasing the number of structural elements and the structure complexity also increases ambiguity in the process of multiple sequence alignment. For example, comparing with NEWS, MAGZ has a more complex structure, where various of structural elements are found in several chapters. Various structural elements and chapters make it more possible to lead to wrongly aligned elements between episodes. To reduce this ambiguity, we propose to apply multiple sequence alignment recursively in a hierarchical manner.

The idea of the hierarchical method is that multiple sequence alignment is applied to short segments of a program, replacing aligning long sequences of entire programs by aligning short sequences of certain segments of a program. To segment the program, we rely on a key structural element—separators, as most of recurrent programs make use of separators to define a clear structure with several main parts, i.e., chapters. Most common separators can be easily detected using basic event detectors, e.g., monochrome images and short repeated audiovisual sequence, and can be identified based on domain prior knowledge. Based on separator determination and multiple sequence alignment, a coarse-grained structure is first obtained considering only separators and chapters, whose structure is to be discovered. For a structure at a finer grain, multiple sequence alignment is applied independently to each chapter, considering the sequences of symbols that belong to the same chapter across episodes. Therefore, the structure of each chapter is inferred, and a fine-grained structure can be obtained. Moreover, we can still infer a structure for each detected structural element if its structure can be further decomposed.

The proposed hierarchical architecture is illustrated in Figure 6.4. A program is firstly structured only with separators represented by S_i and chapters represented by C_i. The structural grammar of each chapter is then inferred. For instance, the structure of chapter C_1 is composed of two structural elements A and B; The process can be iterated again where, in the example, the structural element C is further decomposed into M and N. The hierarchical grammar inference replaces aligning long sequences of entire programs by aligning short sequences of each chapter of the program, which obtains a more deterministic structural grammars. Besides, grammars at different granularity from coarse grain to fine grain can be obtained at the same time.

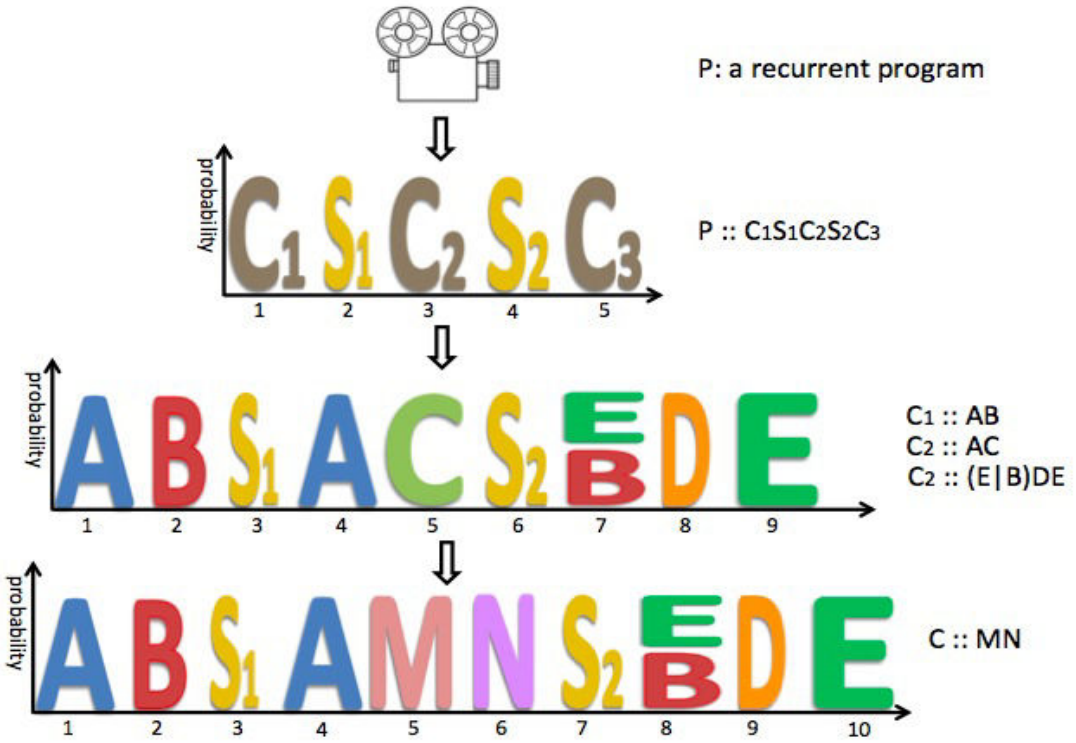


Figure 6.4: A Hierarchical architecture for multiple sequence alignment

6.3 Grammar model construction

The inferred grammar in the form of Weblogo just provides a straightforward understanding of the program structure, which is not usable in practice to segment new episodes. In order to derive a practical model for the program, we construct a grammar model based on the grammar, by combining time information of the structural elements. Specifically, a grammar model is designed as an abstracted representation of the corresponding grammar, including structural elements, their temporal organizations, their relative duration as well as presence probabilities.

In order to extract the relative duration of each structural element in the grammar, we take the average value of the position of each corresponding element across episodes. In particular, as in Figure 6.5, let t_{si} and t_{ei} be the start and the end instants for the i^{th} structural element in the grammar, and t^j_{si} and t^j_{si} be the occurrence of the element from the j^{th} episode. For the relative position of i^{th} structural element in the grammar, we should take into account of the positions of all corresponding occurrences across episodes.

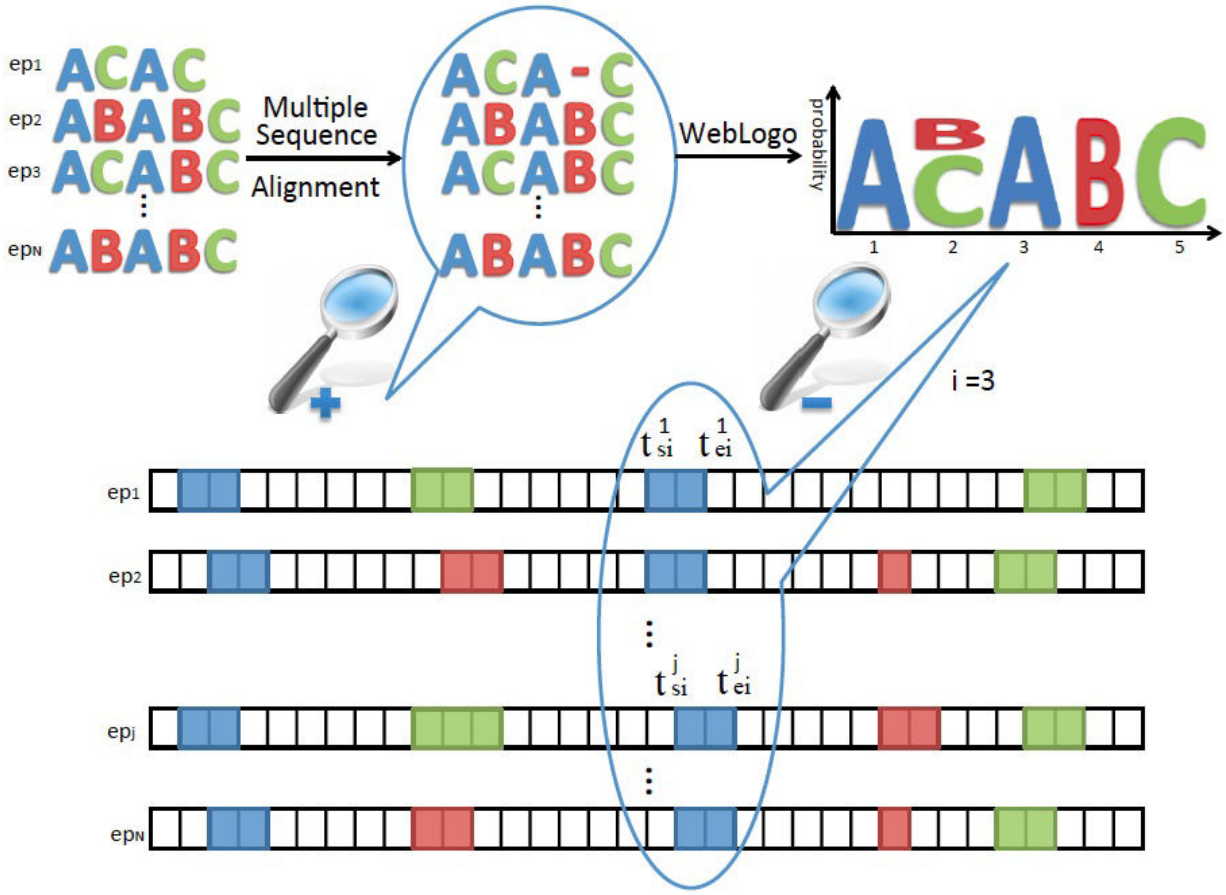


Figure 6.5: Relative duration of structural elements

Therefore, the start and end instants of the i^{th} structural element in the grammar model are computed as

$$t_{si} = \frac{\sum_{j=1}^{n_i} t_{si}^{j}}{n_i} \tag{6.1}$$

$$t_{ei} = \frac{\sum_{j=1}^{n_i} t_{ei}^{j}}{n_i} \tag{6.2}$$

where $n_i (n_i \leqslant N)$ refers to the number of occurrences from the episodes contributing to the i^{th} structural element in the grammar (the gaps are not considered). In the case of multiple elements present in one stack, we do not differentiate them, computing the position of the stack by taking consideration of all elements in that stack.

The relative presence probability of structural elements can be reflected in the grammar.

Here we give a quantitative definition of the presence probability of each structural element in the grammar:

$$P_i = \frac{n_i}{N} \tag{6.3}$$

where P_i refers to the presence probability of the element in the i^{th} stack, and n_i is the number of occurrences from the episodes contributing to the i^{th} structural element in the grammar. Different from the position of structural element, in the case of multiple elements present in one stack, the presence of probability of each element is computed independently, i.e., the presence probability of each element in the same stack is computed separately by counting the number of the occurrences contributing to the same element.

Having the positions and presence probabilities of structural elements in the grammar, the next step is to construct a grammar model based on the duration and presence probability of the structural elements. Evidently, as in Figure 6.5, adjacent elements in aligned sequences are not necessarily contiguous because of segments with no particular semantic interpretation appearing between two structural elements. In order to give prominence to these blank segments and make clear the positions of structural elements, we propose to in practice use a particular symbol N to represent such segments. In Figure 6.6, a grammar in the form of Weblogo is transformed into the grammar model: A rectangle filled with a symbol is used to illustrate each structural element for the program; The height of rectangles indicates the relative presence probability of each element while the width is proportional to its duration.

Once a grammar model is constructed, the model can be utilized to segment new episodes, finding the respective start and end times of structural elements in new episodes. This alignment process is purely based on time, the result is thus the estimated boundaries of the structural elements in the grammar, without disambiguation when multiple symbols can occur at the same time.

6.4 Conclusion

In this chapter, we have presented the technique of multiple sequence alignment, showing that multiple sequence alignment is indeed an efficient way to infer a structural grammar for a recurrent program from a collection of episodes. During the process of alignment, introducing gaps for lose of structural elements makes it possible to align sequences with different lengths, which is the case that we are facing. Based on the grammar derived from the aligned episodes, a grammar model is then constructed by combining time related

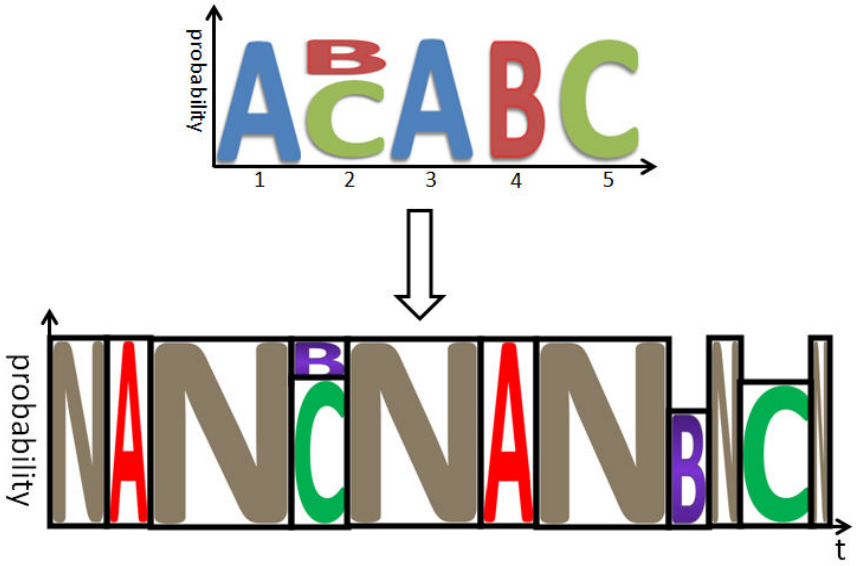


Figure 6.6: Example of grammar model inferred by multiple sequence alignment method

information about structural elements. Comparing with the grammar, the corresponding grammar model can not only provide understanding of the program structure, but also be utilized to, in practice, segment new episodes from the same program.

However, the proposed method of multiple sequence alignment can derive just one common model for the collection of episodes. Since practically there may exist multiple structures for one program, the grammar obtained, in this chapter, may be a superposed model of multiple structures. Discovering multiple structures with multiple sequence alignment is costly and difficult, in particular because episodes do not always have the same length, thus requiring numerous dynamic alignments between sequences. Hence in the next chapter another approach of grammar inference will be introduced, allowing turning the episodes of different length to the sequences of the same length, hence facilitating the task of multiple structure identification.

Chapter 7

Grammar Inference by Uniform Resampling

Content

Practically, some recurrent programs have not only one structure, but can rather have multiple structures, e.g., depending on the day of the week or on the phase of a game. For instance, there may exist two structures in news program. Sharing the same structural elements at the beginning, one ends with interview when people are invited, while the other ends with trailer when a new film is out. Multiple structures can mislead sequence alignment because that superposition of multiple structures can be considered as one during the process of multiple sequence alignment. Therefore multiple structures need to be detected before inferring the program grammar. A rather straightforward way to identify multiple structures is to run clustering techniques on the episodes. However, clustering with multiple sequence alignment is difficult, as symbolized episodes do not always have the same length, which needs costly computations of alignments.

To circumvent this high computational cost, we propose in this chapter a method based on uniform resampling, which represents episodes as sequences of symbols of the

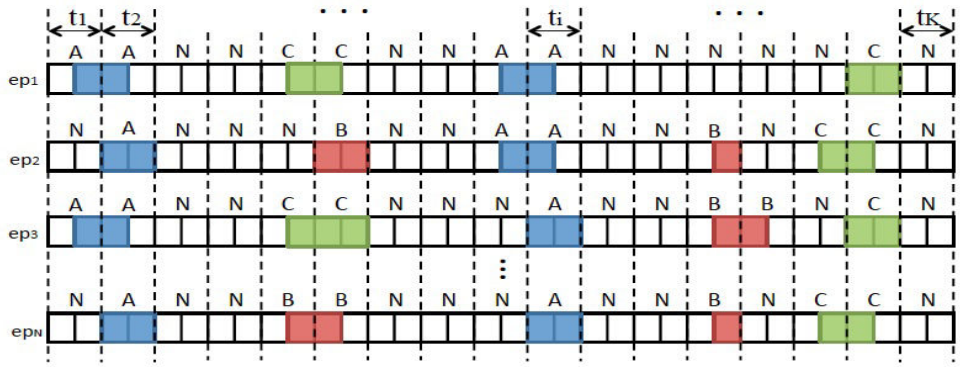


Figure 7.1: Uniform resampling

same length by uniform resampling after length normalization. Uniform resampling makes the clustering techniques much easier on the collection of episodes, as computing pairwise distance between elements is much simpler for input sequences having the same length. Particularly, after uniformly resampling the episodes, we then model the structure of the program with a categorical distribution, as well as multiple structure identification with a clustering technique.

7.1 Uniform resampling

The idea of uniform resampling is to segment each episode into a fixed number of time intervals, K, after normalizing the length of the episode. Each time interval is represented by the symbol of the corresponding structural element, if any, present in the interval. As a result, all episodes are discretized into a fixed length representation of K time intervals.

Figure 7.1 shows an illustrative example of a collection of episodes $EP = \{ep_1, ep_2, ..., ep_N\}$ with d discovered structural elements $E = \{E_1, E_2, ..., E_d\}$, i.e., a set of symbols (in the case of example, $E = \{A, B, C\}$). The colored rectangles represents the occurrences of structural elements, colors differentiating one element from another. Given such a collection of episodes, we firstly normalize the length of the episodes as the length K, namely, the temporal positions of structural elements are normalized into the rang $[0, K]$. Specifically, for a structural element, supposing that the temporal position of one of its occurrence in the jth episodes is $(t_s, t_e)^j$ for the start instance and the end instance. After the length

normalization, the temporal position of the occurrence is turned to:

$$(t_s, t_e)^j_{norm} = \frac{(t_s, t_e)^j * K}{L_j} \tag{7.1}$$

where L_j is the length of the jth episodes.

We uniformly segment the episodes into a fixed number of intervals, which are represented by a set of time stamps $T = \{t_1, t_2, ..., t_K\}$. We then mark each time interval t_i ($t_1 < t_i < t_K$) with the corresponding structural element E_k from the d detected structural elements. Arbitrarily, N denotes the absence of any structural element, specified as E_0. If more than one element are detected in an interval, the one that has the longest duration in the time interval is considered. Particularly, $e^j_i \in [0, d]$ denotes the element taken in episode ep_j at time interval t_i. In other words, $e^j_i = k$ means that E_k is present in ep_j during the time interval t_i.

For example, in a news program, two structural elements, $E = \{E_1, E_2\}$ are determined, where E_1 represents *separators* (S in symbol) and E_2 represents *outlines* (T in symbol), i.e., $E = \{S, T\}$. Therefore, $e^5_2 = 2$ means that at t_2 in the 5th episode, the present structural element is the *outline* (T in symbol), while $e^3_{10} = 0$ means that at t_{10} in the 3rd episode, no determined element is found (N in symbol).

Hence after uniform resampling, the collection of episodes can be represented by a set of symbolic sequences, where the sequences takes the same length.

7.2 Categorical distribution modeling

After turning a collection of episodes into a set of fixed-length sequences, the next step is to find a common structure shared by the sequences. To do this, we first assume that a program has a unique structure shared by all episodes. The case of multiple structures will be discussed in Section 7.4.

For each of the K time intervals, we may straightforwardly use a categorical distribution to model the probability of observing a given structural element. Let x_i be the categorical random variable representing the type of structural elements at interval t_i. We model the program structure as a categorical distribution matrix, $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_K]$, where each column $\mathbf{P}_i = [p(x_i = 0), p(x_i = 1), ..., p(x_i = d)]^T$ is an independent probability mass function over the d structural elements plus the empty one.

A set of number sequences representing a collection of episodes

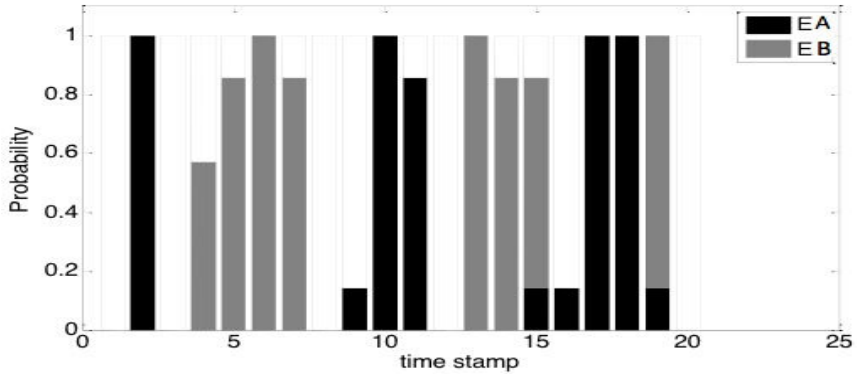
ep_1 NANBBB…NAABN
ep_2 NANNBB…NAABN
ep_3 NANNBB…AAABN
⋮
ep_N ANABBN…NAAAN

MLE

Probability distribution matrix for the collection

Structural element \ Time stamp	t_1	t_2	…	t_K
N	$p(x_1=N)$	$p(x_2=N)$	…	$p(x_K=N)$
A	$p(x_1=A)$	$p(x_2=A)$	…	$p(x_K=A)$
B	$p(x_1=B)$	$p(x_2=B)$	…	$p(x_K=B)$

(a) Generation of probability distribution matrix



(b) Illustration of probability distribution matrix

Figure 7.2: Example for unique structure modeling with two structural elements.

The structure of the collection $EP = \{ep_1, ep_2, ..., ep_N\}$ modeled by some distribution matrix $\mathbf{P}$ yields

$$P(EP \mid \mathbf{P}) = \prod_{ep_j \in EP} \prod_{i=1}^{K} p(x_i = e_i^j) \tag{7.2}$$

considering all episodes and time intervals are independent.

According to the maximum likelihood estimation (MLE) criterion, the best generative model can be defined as the optimal solution to the following log likelihood maximization problem:

$$\max_{\mathbf{P}} \left\{ L(\mathbf{P} \mid EP) = logP(EP \mid \mathbf{P}) = \sum_{ep_j \in EP} \sum_{i=1}^{K} log\ p(x_i = e_i^j) \right\} \tag{7.3}$$

Taking the well-known solution to 7.3, the probability mass functions hence are estimated as

$$p(x_i = k) = \frac{\sum_{\text{ep}_j \in \text{EP}} 1_{e_i^j = k}}{|\text{EP}|} \tag{7.4}$$

where $\sum 1_{e_i^j=k}$ counts the number of occurrences where $e_i^j = k$ among all episodes. In other words, $p(x_i = k)$ is the relative frequency of structural element E_k at t_i across all episodes from EP.

A simple example with two discovered structural elements is given in Figure 7.2(a), illustrating the collection of probability mass functions $\mathbf{P}$. Figure 7.2(b) provides a visual representation of the estimated probability matrix. In other words, it is a grammar based on the alignment of the collection of episodes in the form of time stamps, depicting the time ordered structural elements and their relative probability at each time stamp.

7.3 Grammar model construction

The grammar represented by time stamps with a probability distribution matrix has however very limited abstraction capabilities and is not concise enough owing to information redundancy, i.e., consecutive time stamps repeating with similar probability distribution. Hence before constructing a grammar model, we thus propose to separate the time stamps into coherent states, grouping consecutive time stamps with similar distributions, for having a usable grammar model.

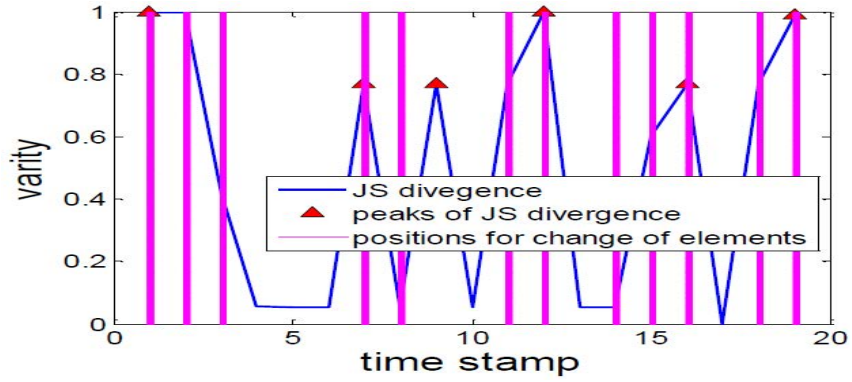
To achieve this goal, we firstly introduce the Jensen-Shannon (JS) divergence, which measures the similarity between two probability distributions. The JS divergence, based on the Kullback-Leibler divergence, between two probability distribution matrices $\mathbf{P}$ and $\mathbf{Q}$, is defined as

$$D_{JS}(\mathbf{P} \parallel \mathbf{Q}) = \frac{1}{2} D_{KL}(\mathbf{P} \parallel \mathbf{M}) + \frac{1}{2} D_{KL}(\mathbf{Q} \parallel \mathbf{M}) \tag{7.5}$$

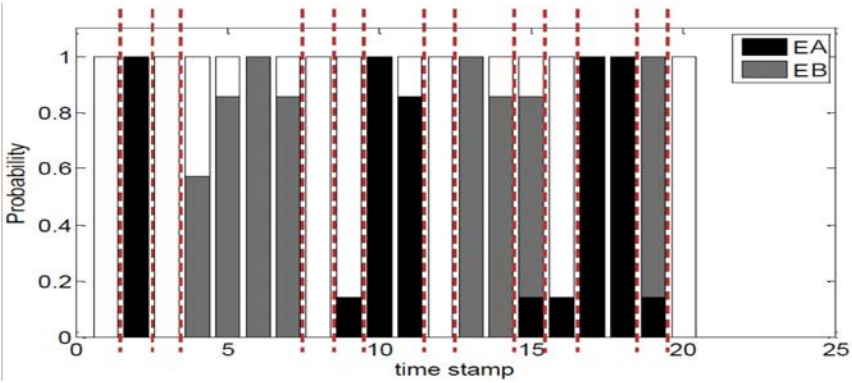
where $\mathbf{M} = \frac{1}{2}(\mathbf{P} + \mathbf{Q})$, and $D_{KL}()$ denotes the Kullback-Leibler divergence defined as

$$D_{KL}(\mathbf{P} \parallel \mathbf{Q}) = \sum_{i=1}^{K} \sum_{k=1}^{d} p(x_i = k) log \frac{p(x_i = k)}{q(x_i = k)} \tag{7.6}$$

A small $D_{JS}(\mathbf{P} \parallel \mathbf{Q})$ means that the two distribution matrices $\mathbf{P}$ and $\mathbf{Q}$ are similar, and



(a) State change positions



(b) Structure segmentation

Figure 7.3: Illustration of structure segmentation

conversely.

In order to segment the time stamps into coherent states, we verify between two successive time stamps the variations of two indicators: the composition of structural elements and the JS divergence. If one of the two indicators has a significant change, we consider that there is a rupture of state. More concretely, as illustrated in Figure 7.3(a), the positions where the JS divergence has a peak (its local maximum) or where the combination of structural elements changes are deemed to be state changes. This is illustrated in Figure 7.3(b), where state changes are marked based on the measures reported in Figure 7.3(a).

Based on the segmented grammar, we can visualize the states and build a grammar model. The same as multiple sequence alignment model, the grammar model for uniform resampling is finally obtained by introducing temporal organizations of structural elements, their relative duration as well as presence probabilities. Since each state may consist of

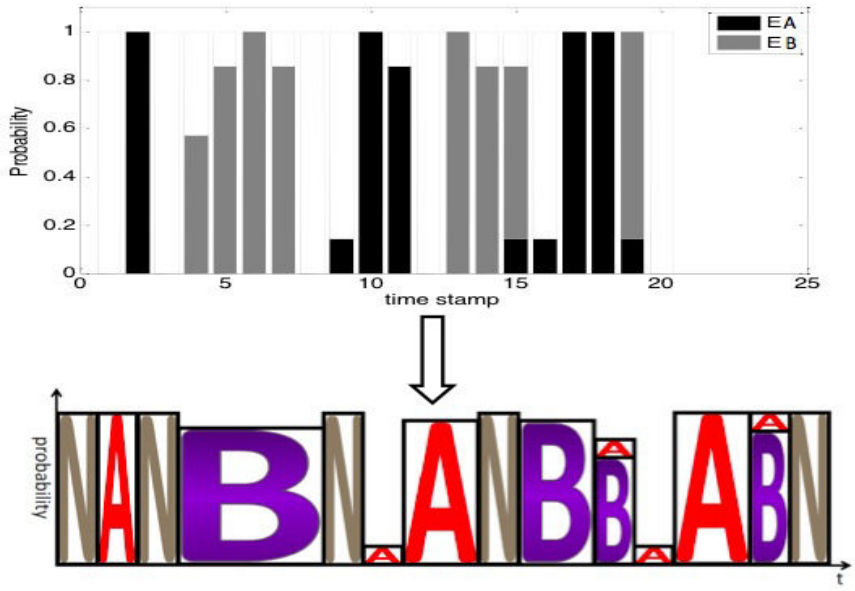


Figure 7.4: Example of grammar model inferred by uniform resampling method

several time stamps, For each state, the probability matrix is taken the average value of the probability matrix (the average value of each structural element is computed separately) of all time stamps. Using the same representation as for grammar models coming from multiple sequence alignment, the grammar model is constructed based on the temporal positions of the segmented states as well as their probability matrices. The grammar model corresponding to the grammar in Figure 7.3(b) is shown in Figure 7.4, where two structural elements are in symbols A and B, the segments corresponding to no determined structural element are denoted as N.

7.4 Multiple structure identification

The grammar presented above clearly assumes that the structure is unique across episodes, which is not always the case in practice. Recurrent programs may have different structures according to editorial rules. Given a collection of episodes from the same program, we aim at identifying multiple structures using a clustering technique, because the episodes that belong to the same structure should have similar probability distributions of structural elements.

Supposing that there are M underlying structures, each of them has its own structural

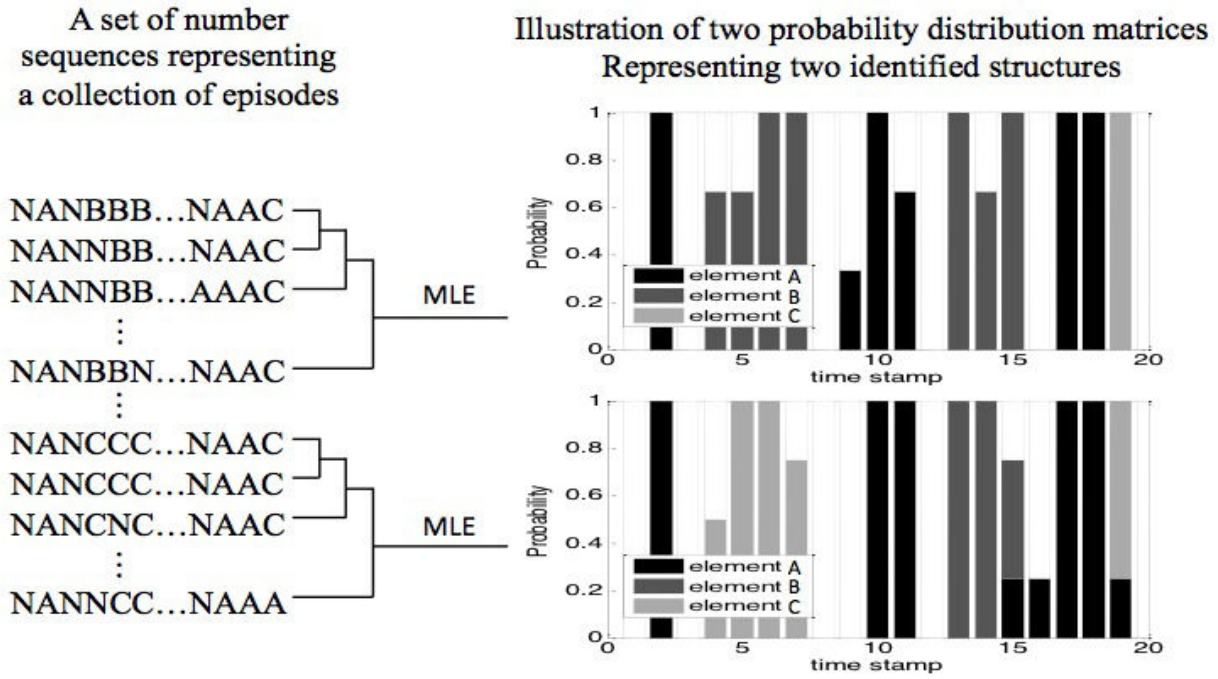


Figure 7.5: Illustration of multiple structure identification

elements and probability distribution matrix. The grammar is in the form of time stamps, represented by the set of categorical distribution matrices $\mathbf{P} = \{\mathbf{P}^1, \mathbf{P}^2, ..., \mathbf{P}^M\}$. The episodes $EP = \{ep_1, ep_2, ..., ep_N\}$ should be partitioned into M clusters, each cluster representing one structure. Figure 7.5 shows an example of a collection of episodes partitioned into two clusters, representing two different structures.

There are many clustering techniques to classify the episodes into M clusters. As no prior knowledge about the program structure is adopted, the number of underlying structures (i.e., M) is unknown in advance. Hence we propose to use a hierarchical agglomerative clustering technique to group episodes, while at the same time determining the optimal number of clusters.

The key to this type of bottom-up clustering is the distance between clusters on which agglomeration is based. Since a structure is represented by a probability distribution matrix, the distance between two structures is determined by their probability distribution matrices. We again adopt JS divergence, which was defined in the equation 7.5, because it measures the similarity between two probability distributions with a symmetric and bounded value, required for hierarchical clustering techniques.

We define the distance between two clusters C_1 and C_2 based on the JS divergence using their probability distribution matrices $\mathbf{P}^1$ and $\mathbf{P}^2$, i.e.,

$$\text{dist}(C_1, C_2) = D_{\text{JS}}(\mathbf{P}^1 \parallel \mathbf{P}^2) \tag{7.7}$$

At each iteration of the hierarchical clustering, the two clusters with minimal distance are merged, the new cluster inheriting the episodes owned by the original clusters C_1 and C_2. The probability distribution matrix corresponding to the newly merged cluster is obtained as

$$\mathbf{P}^{new} = \frac{\mid C_1 \mid}{\mid C_1 \mid + \mid C_2 \mid}\mathbf{P}^1 + \frac{\mid C_2 \mid}{\mid C_1 \mid + \mid C_2 \mid}\mathbf{P}^2 \tag{7.8}$$

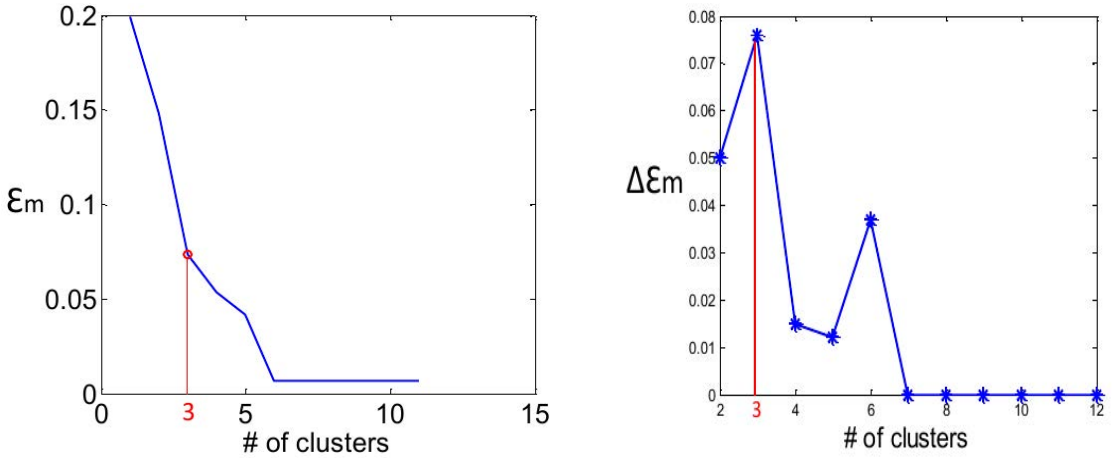
With such distance measure and cluster merge method, we now identify the multiple structures of the program. However without the number of underlying structures, M is unknown. In order to automatically determine the structure number for different types of programs, we propose to determine the optimal cluster number while exhibiting hierarchical agglomerative clustering by monitoring the quality of the clusters by means of an impurity factor. For a partition into m cluster, the impurity factor is given by

$$\varepsilon_m = \frac{1}{m}\sum_{i=1}^{m}\text{IM}(C_i) \tag{7.9}$$

where $\text{IM}(C_i)$ [46] is the impurity of each cluster, defined as

$$\text{IM}(C) = \frac{\sum_{\text{ep}_j \in \text{EP}}\sum_{i=1}^{K} 1_{e_i^j \neq 0} \cdot (1 - p(x_i = e_i^j))}{\sum_{\text{ep}_j \in \text{EP}}\sum_{i=1}^{K} 1_{e_i^j \neq 0}} \tag{7.10}$$

The impurity factor measures the error that, at a particular time stamp, all the episodes in the cluster are not concentrated in one single element. Note that here we exclude the element N which essentially means the element is unknown. The optimal number of clusters is obtained by monitoring ε_m, where a significant increase of the impurity indicates that the optimal number of clusters has been reached. Particularly, when the number of clusters turns from m to $m-1$, if the impurity factor ε_m increases significantly comparing with ε_{m-1}, it indicates that m might be the correct structure number. Because the significant increase of ε_m signifies that the newly merged cluster may contain two different structures.



(a) Impurity factor (b) Increase of impurity factor from m to $m-1$

Figure 7.6: Determination of cluster number by impurity factor

Therefore, the optimal number of clusters (and hence structures) is determined when the impurity factor has maximal increase while the cluster number turns from N, i.e., the total number of episodes in the collection, to 1.

The observation of a such change, i.e., $\triangle\varepsilon_m$, can be expressed as the difference value of ε_m and ε_{m-1}. Since a sudden change of ε_m will result in a peak in its difference value, we can find the optimal a number of underlying structures M as $M = arg\, max_m \triangle\varepsilon_m$. We illustrate a real example of news program in Figure 7.6, where the impurity factor suddenly increases at $m = 3$, thus indicating that there are three structures for the program. However, we observed cases where some clusters contain a single episode. This can be explained by the fact that the isolated structure may result from an episode badly processed in earlier stages or some special episode, e.g., on Christmas or special issues. We treat such episodes as an isolated structure for which no grammar will be made to make sure that the discovered structure is common enough to represent the program.

7.5 Conclusion

This chapter have proposed an approach of uniform resampling for the task of grammar inference, which allows identifying multiple structures for a recurrent program. By leveraging uniform resampling, episodes are turned to fixed-length sequences, which makes clustering techniques feasible with low computational cost to identify the existence of mul-

tiple structures, at the same time determining the optimal number of structures. Based on the grammar, a corresponding grammar model is constructed using the same design as for grammar models coming from multiple sequence alignment, for the purpose of practical use.

Hereto, we have proposed two approaches for grammar inference. The two alignment techniques, i.e., multiple sequence alignment and uniform resampling, exhibit different properties, leading to distinct features for the inferred grammars and models. We will experimentally compare the pros and cons of the two strategies in light of a segmentation use-case in the next chapter.

Chapter 8

Experimental Evaluation for Grammatical Inference

Content

Two different alignment techniques, i.e., multiple sequence alignment and uniform resampling, have been presented in the last two chapters. They have distinct features and properties. Generally speaking, on the one hand, multiple sequence alignment offers more flexibility than uniform resampling, however to the expense of a high computational cost. We use this flexibility to enable hierarchical structures, recursively applying alignments. Recursion also lowers the computational cost by simplifying the alignment problem at each step. On the other hand, uniform resampling provides a simple method to turn the episodes of different length into sequences of the same length, which enables to cluster episodes when multiple structures are present across these episodes, a strategy that is only achievable with multiple sequence alignment at a very high computational cost.

In this chapter, we propose a quantitative and comparative evaluation of the two meth-

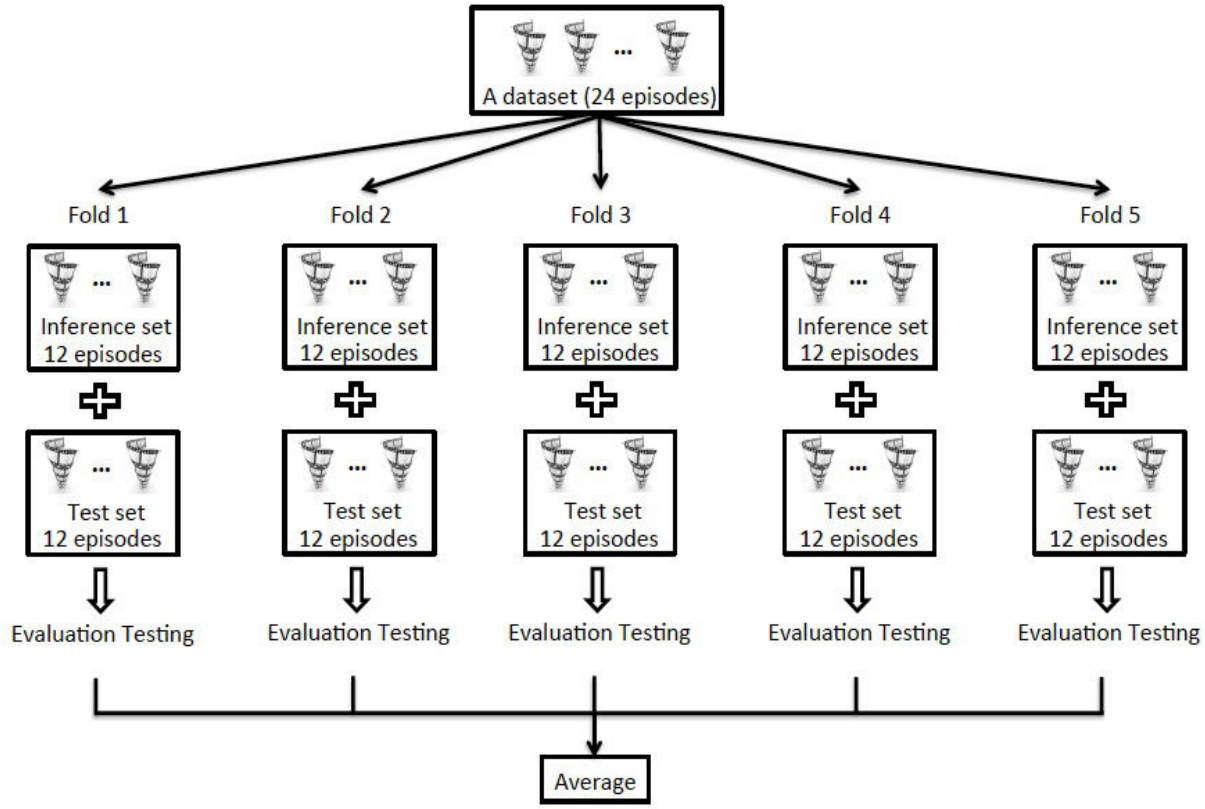


Figure 8.1: Cross-validation strategy

ods in light of a segmentation use-case. To do so, we propose to apply the inferred grammar model to new episodes, i.e., episodes not used for inference, so as to segment the new episodes into their structural components. We first discuss the grammar inference step, mostly from a qualitative standpoint before evaluating grammars quantitatively.

8.1 Experimental setting

We discussed in Chapter 5 about the number of episodes used to infer structural grammars. In our case, the optimal number is around 11 or 12 episodes based on the observations of the quantity of structural elements determined. We fix 12 as the size of the collection of episodes used to infer grammars, named as *inference set*. As we will perform segmentation tasks, using inferred grammars, on the new episodes, we therefore choose 12 additional episodes as a *test set* for quantitative evaluations.

The experiment is again applied to the four datasets, i.e., NEWS, GAME, TALK and MAGZ. For each program, the dataset comprises 24 episodes, hence divided into two sets: one set for inferring grammars (inference set), the other for the use-case application (test set). Due to the limited quantity of data and to avoid experimental biases, experiments are conducted using a cross-validation strategy as illustrated in Figure 8.1: For each fold, part of the episodes is randomly selected for the inference set from the 24 episodes, the remaining ones being used for segmentation in the test set. Quantitative results reported hereunder are averaged over 5 folds.

8.2 Qualitative analysis of grammars

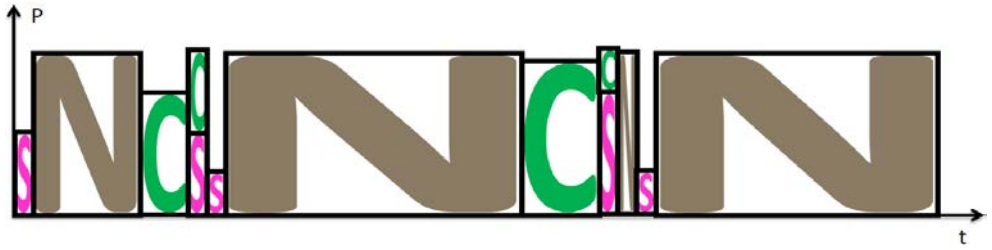
Having randomly chosen the episodes in the inference set, we now analyze the resulting grammar models as obtained with multiple sequence alignment (MSA) and with uniform resampling (UNR). Multiple sequence alignment may produce hierarchical grammars with different granularity (examples of grammars with different granularity can be found in [58]): We consider here the finest-grained grammars in order to compare with the ones obtained by uniform resampling. Examples of obtained grammar models are represented in Figures 8.2, 8.3, 8.4 and 8.5 for TALK, GAME, MAGZ and NEWS, respectively.

For TALK, three grammar models are illustrated in Figure 8.2, where three semantically interpretable structural elements are identified, i.e., separator (S), commercials (C) and musical performance (M). Figure 8.2(a) shows a grammar model obtained with multiple sequence alignment, while Figure 8.2(b) and 8.2(c) are the two models obtained with uniform resampling. All three grammars describe a program with three main chapters bounded by separators and commercials. The clustering stage in the uniform resampling strategy enables to identify two distinct grammar models, depending on whether the episode ends with a musical performance.

For the two grammars obtained with uniform resampling in Figures 8.2(b) and 8.2(c), the evident difference is the presence of musical performance segment, based on which the episodes are clustered in two different groups, hence resulting in two different grammar models. In other words, all episodes belonging to the grammar model in Figure 8.2(b) are supposed to have the musical performance segment. However, the presence probability of the musical performance segment is not one. This can be explained by the fact that the musical performance does not appear exactly at the same moment in different episodes. Hence the musical segments are not always found in the time intervals at the same position across episodes, even though all episodes have a musical segment. After transforming the

(a) multiple sequence alignment

(b) uniform resampling, live musical performance



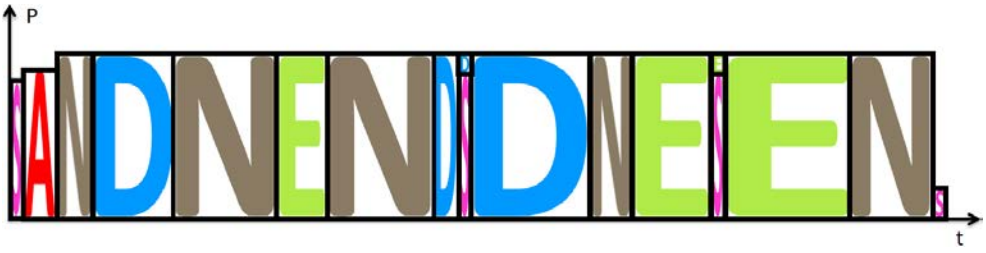
(c) uniform resampling, no live musical performance

Figure 8.2: Grammars for TALK with multiple sequence alignment (a) and with uniform resampling (b and c). Structural elements: separator (S), commercials (C), musical perforamnce (M) and undefined (N).

grammar in the form of time stamp into the grammar model, there is not a full value for the presence probability the musical performance segment.

Note that the musical performance is not totally lost in the grammar model obtained with multiple sequence alignment but appears with a frequency less than that of the other elements in the model. In fact, the MSA grammar model can be seen as a superposition of the two models identified with uniform resampling. However the contents of chapters, i.e., the structural elements denoted as N, need be further determined.

Results on TALK also show subtle differences between the MSA and UNR grammar models regarding the relative duration of the structural elements and their presence prob-



(a) multiple sequence alignment

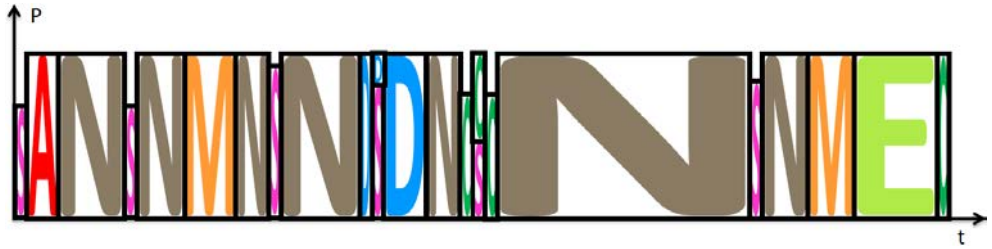
(b) uniform resampling

Figure 8.3: Grammars inferred for GAME with multiple sequence alignement (a) and with uniform resampling (b). The grammar's structural elements are: separator (S), anchor (A), dialog (D), full text (E) and undefined (N).

ability. The relative duration of the structural elements in the UNR grammars is mostly longer than the ones in the MSA grammar model, such as musical segments and the separators. On the contrary, the presence probability for some elements is higher in the MSA grammar model than the ones in the UNR models, such as for commercials and for separators.

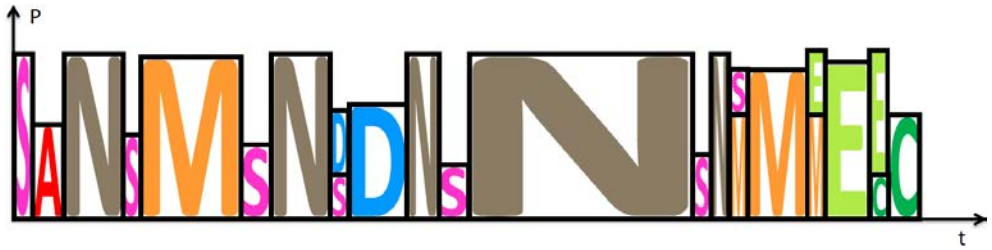
These two phenomena result from the different techniques used for grammatical inference. Multiple sequence alignment relies on a dynamic alignment of the symbolic sequences, which leads to a relative high presence probability for each element. On the contrary, uniform resampling does not allow warping between episodes (apart from the duration normalization), and computes the probability of each structural element in each time interval. In other words, one occurrence of the element could be found in more than one time intervals, which reduces its presence probability in each time interval and somehow expands the element duration in grammar models. Besides, the element duration in the MSA grammar model is an average of the aligned ones across episodes. So generally the elements in the UNR grammar model usually have longer duration and lower presence probability than the ones inferred by MSA.

Similarly, the same phenomena could also be noticed for the three other types of pro-



(a) multiple sequence alignment

(b) uniform resampling, grammar 1

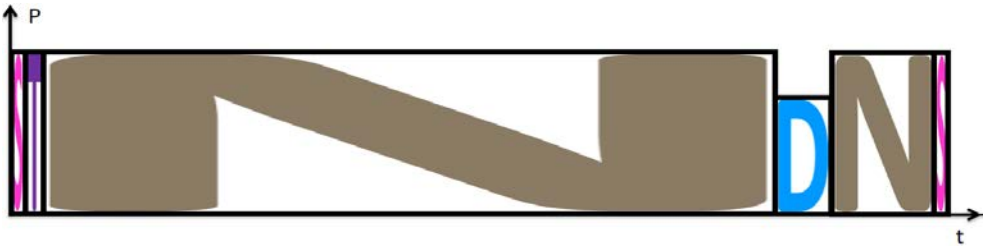


(c) uniform resampling, grammar 2

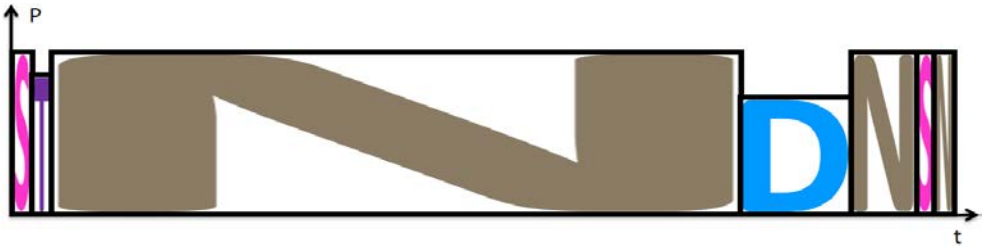
Figure 8.4: Grammars obtained for MAGZ with multiple sequence alignment (a) and with uniform resampling (b and c). Structural elements: separator (S), anchor (A), commercials (C), dialog (D), full text (E), musical performance (M) and undefined (N).

grams. Figure 8.3 shows the grammar models for GAME respectively inferred by MSA and UNR. GAME has just one structure, hence for both multiple sequence alignment and uniform resampling, there is just one grammar model inferred. Reading the two models in Figure 8.3(a) and 8.3(b), the program structure for GAME is: Starting with anchor person's opening, and its main content is interviews as well as full text scenes. Comparing the two grammar models, the elements in the UNR grammar model generally have longer duration and lower presence probability than the ones inferred by MSA, which can be evidently observed from all the full text segments, separators and the first segment of dialog.

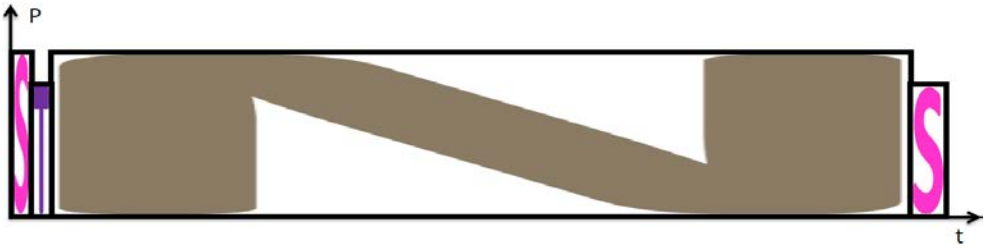
MAGZ shows a more complex structure comparing with others. The grammar mod-



(a) multiple sequence alignment



(b) Uniform resampling, interview



(c) Uniform resampling, no interview

Figure 8.5: Grammars inferred for NEWS with multiple sequence alignement (a), with uniform resampling including clustering (b and c). The structural elements are: separator (S), outline (T), dialog (D) and undefined (N).

els are in Figure 8.4, where anchorperson's opening, music, dialog, full screen text and commercials are determined as the elements structural of the grammar's vocabulary. Two different structures can be inferred with uniform resampling method, depending on whether the episodes start with a separator or not. Generally, MAGZ is divided into many chapters by separators, and content of each chapter varies a lot. The difference between two types of grammar models can be noticed, i.e., the elements in the UNR grammar model generally have longer duration and lower presence probability than the ones inferred by MSA. The anchor's opening at the beginning of the program, the dialog segments and the separators are the elements exhibiting the evident differences.

For NEWS, the three grammar models in Figure 8.5 correspond to the coarse-grain

structure of a classical news program: The program is introduced by a separator (S) and starts with the headlines (T); The following non interpreted element (N) correspond to the alternation of anchor's announcements and reports, possibly including an interview (D). The clustering stage in the uniform resampling strategy enables to identify two distinct grammars, depending on whether an interview is included at the end of the program or not. However, the inference techniques used did not allow identifying the structure of the program at a finer grain, e.g., to discover the alternation of anchor's announcements and reports, mostly because the number of reports varies across episodes. Lacking of the alternation of anchor's announcements and reports in NEWS shows a limitation of the proposed grammar inference approach, which makes us think of a further work direction, i.e., extracting a generalized model from the set of sequences sharing a common structure but with evident difference. In the case of NEWS, the alternation of anchor's announcements and reports is the common structure shared across episodes, while the various numbers of the reports is the evident difference. More concrete thinking on this issue will be introduced in perspectives.

Up to now, we have two types of grammar models for each program obtained with multiple sequence alignment and with uniform resampling. Even though there still exist segments that are not identified, the grammar models are deemed to well represent the structure of the programs considering the granularity that we can reach. In order to prove that, quantitative evaluations of grammar models will be given in the next section.

8.3 Use-case: segmentation of new episodes

Quantitatively evaluating the quality of the grammars inferred from a number of episode cannot be done in a direct manner. We thus rely on a use-case scenario to verify the effectiveness of the grammar models. As the thesis was collaborated with French National Institute of Audiovisual (INA), we propose preferentially to position ourselves in the context of indexing and archives. In a reminder of that up to now the indexing task at INA is quasi exclusivement performed manually, it would be of great interest for Ina to have softwares facilitating the work of librarians during indexing of TV programs. Hence, providing automatic time indications for events of interest during manual indexing is a possible use case. Following this insight, we propose to design a use-case to segment new episodes, finding the temporal positions of structural elements, using the inferred grammar models.

8.3.1 Use-case description

Considering the objective in industry of the thesis, we design the following scenario for use-case: Given a dataset, we want to infer a grammar model from episodes in the inference set, and segment the remaining episodes (the test set) according to the inferred grammar model. By segmenting, we mean predict the structural elements that are present in an episode and determine their respective start and end times, thus effectively providing a dense structure or, equivalently, a dense segmentation, for all episodes within the test set.

Figure 8.6 shows how to segment the new episodes using a grammar model. For a grammar model having d structural elements (including no identified element, i.e., segments N), the temporal position of each structural elements in the grammar model is

$$\mathbf{P}_{model} = \{P_1, P_2, ..., P_i, ..., P_d, P_{d+1}\} \tag{8.1}$$

where P_i is the boundary position between two successive elements. Evidently, P_1 equals to 0, and P_{d+1} is 1.

For a new episode from the test set with the length denoted as L, the boundaries of the predicted structural elements in the episode are computed as follows:

$$\mathbf{P}_{episode} = L * \mathbf{P}_{model} = \{L * P_1, L * P_2, ..., L * P_i, ..., L * P_d, L * P_{d+1}\} \tag{8.2}$$

With such a segmentation strategy, the structural elements can be predicted by giving their start and end instants. In the case of multiple elements at one position, the type of structural elements can be one of the elements in that position according to the grammar model. In other words, during evaluation process, any of the possible elements (the elements at that position in the grammar model) being found in the position is considered as the case of a correct prediction, while in practice during the indexing task, multiple possible elements should be considered in that position.

The segmentation task just relies on time information of structural element provided by grammar models, and no content-based interpretation of segmented episodes are involved. Because the purpose of the use-case strategy is to evaluate the quality of the inferred grammar models. Hence, the inferred grammar model is the only reference involved to predict the structural elements and find their boundaries in the new episodes. In the case of practical use, such as at INA, the content-based interpretation of new episodes should be considered, which will be introduced in further perspectives.

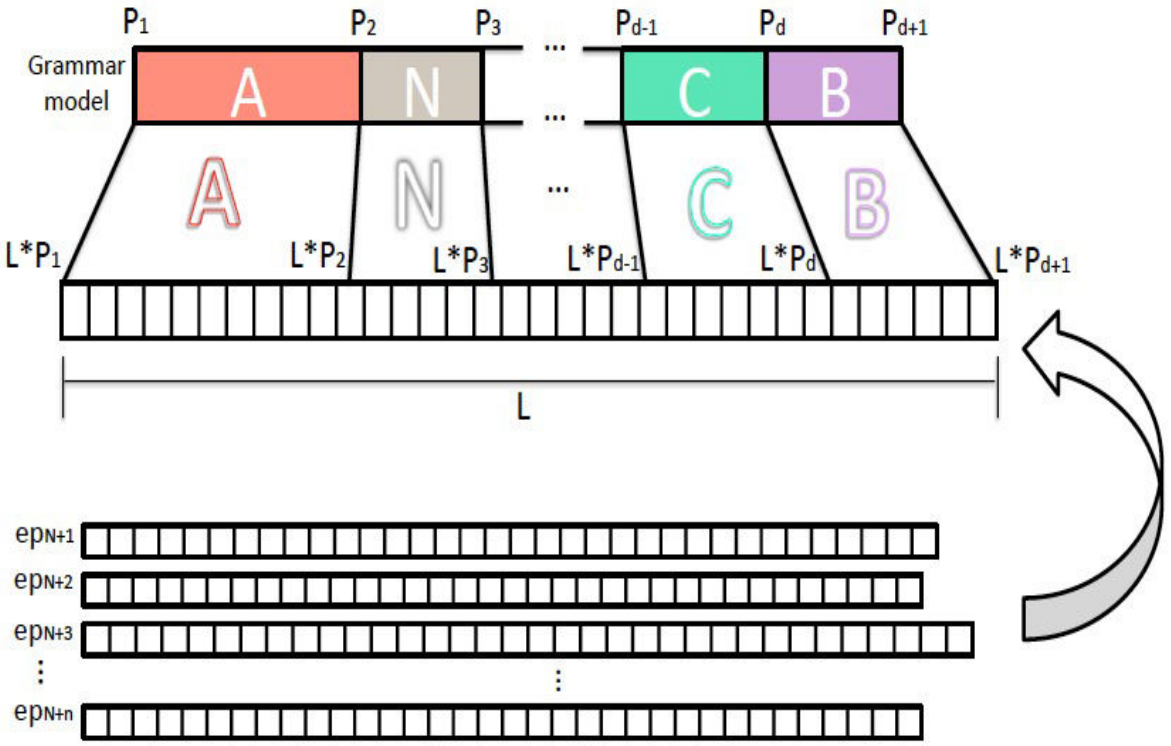


Figure 8.6: Segmentation strategy

8.3.2 Baseline model construction

For comparison purposes, we construct a baseline model, whose segmentation results are compared with the two introduced grammatical inference methods. The construction of the baseline model is illustrated in Figure 8.7. After turning a collection of episodes into a set of fixed-length sequences based on uniform resampling, for each time interval, the element (including symbol 'N') appearing most frequently is voted as the element in that position. The boundary of the element is computed by counting the number of the successive time intervals having the same structural elements, and the presence probability is always deemed as 1.

8.3.3 Experimental result comparison

In our experiments, we compare the segmentation results using standard performance measures such as precision (P), recall (R) and F measure (F). The metrics are computed on a time basis. An element is considered as being correctly predicted if it overlaps with

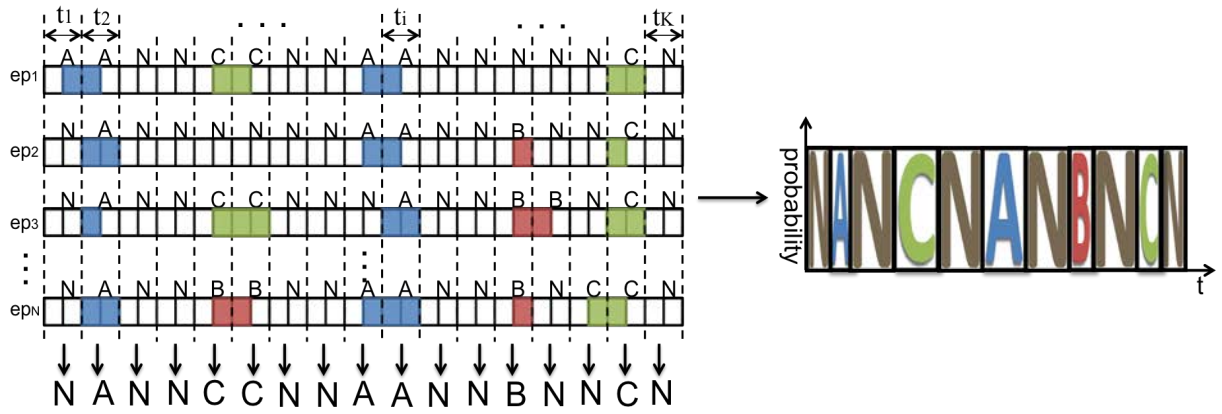


Figure 8.7: Baseline model construction

an element of the same type from the ground-truth, which is manually annotated. In the case of multiple elements at one position, we consider all the present elements. If one of them is match, it is deemed as correctly predicted. Recall is measured as the amount of time that the elements are correctly predicted, i.e., the time of overlapping, by the total duration of the elements in ground-truth. Precision is defined in a similar way by the total duration of the elements in the predicted structure. Recall and precision measures are averaged across episodes. In the case of multiple structures, the structural model having the best F measure is chosen as the final results.

Results for the four recurrent programs are reported in Table 8.1. NEWS has the best score in terms of precision. This result owes to the very stable structure of news programs and the fact that simplicity of the structure is found. Comparing with NEWS, the other programs exhibit lower performance, mostly because of the higher number of determined structural elements in the grammar. Especially, when some structural elements having very short duration, such as the elements in MAGZ, structural elements with short duration are easier to be missed than longer ones. The duration of structural elements in GAME is on average longer than the one for TALK and MAGZ, which result in a slight better performance for GAME. Furthermore, in the case of TALK and MAGZ, for all three inference methods, recall values are rather high whereas precision values are relatively low. This reveals that the overlapping segments temporally match better to the elements in annotations than to the predicted elements. Conversely, for the case of GAME and NEWS.

Comparing the three inference methods, one can see that generally the two proposed methods perform better that the baseline method in term of precision and recall. These

Table 8.1: Segmentation performances by multiple sequence alignment model (MSA), uniform resampling model (UNR) and baseline model

	MSA			UNR			Baseline		
Dataset	P	R	F	P	R	F	P	R	F
NEWS	0.69	0.54	0.61	0.82	0.55	0.66	0.55	0.37	0.44
TALK	0.53	0.54	0.53	0.48	0.86	0.61	0.46	0.64	0.53
GAME	0.69	0.50	0.58	0.56	0.53	0.54	0.53	0.49	0.51
MAGZ	0.42	0.64	0.50	0.33	0.73	0.45	0.34	0.58	0.42

results can be explained by the nature of grammatical inference techniques. First, both multiple sequence alignment and uniform resampling methods consider at a given position all the elements across episodes, where multiple sequence alignment relies on a dynamic alignment and uniform resampling adopts all elements at the each time interval. While the baseline model just counts the most frequent element at a given position, which means that the elements from some episodes are abandoned, thus providing incomplete boundary information for the model. Second, for the two proposed inference methods, multiple types of elements may be found at one position with their presence probabilities, which augment the chance of a structural element being correctly predicted. Comparing the results given by the two proposed inference methods, one can see that in case of simple structures, i.e., NEWS and TALK, uniform resampling is more precise. These results can be explained by the fact that the UNR method is capable of identifying multiple structures for a program, which highly raises the prediction precision, as the tested episodes are supposed to be more targeted by a certain grammar. However, in the case of more complex structures, i.e., GAME and MAGZ, multiple sequence alignment performs slightly better than uniform resampling. These results owes to the way of boundary determination of structural elements for the two inference methods. The duration of structural elements in the UNR grammar models is somehow extended by counting the length of time intervals, while the boundary of structural elements in the MSA grammar models is the average value of the aligned elements across episodes. The extended boundary in the UNR models leads to a lower prediction performance.

Considering the performance achieved, even though the time-based error rate is relative high, the prediction precision for both inference methods is promising for a segment task considering other research work (e,g, results in [2]), especially for the programs having simple structures, such as NEWS and GAME in our case. As for the time-based error rate, content-based interpretation (will be introduced in perspectives) will highly improve the

precision on element boundaries, which will enhance the feasibility of the proposed method in practice use.

8.4 Conclusion

Qualitatively comparison of the grammar models inferred by the two inference techniques shows the distinct features of each type of models. Multiple sequence alignment infers a unique grammar model for each program, combining all possible structures exhibited by the collection of episodes. Uniform resampling achieves to identify existence of multiple structures for one program, inferring individually the grammar model for each corresponding structure.

Although evident differences can be noticed between the grammars obtained by multiple sequence alignment and uniform resampling, they factually reflect the structure that one would expect. It can be demonstrated by the qualitative analysis, proving that the grammatical inference can be used to infer grammars by adopting minimal domain knowledge for recurrent TV programs. Furthermore, the quantitative evaluation conducted in the way of segmentation use-case shows that the idea of grammatical inference can be used for segmentation tasks.

The two inference methods also shows their limitations. The most evident one is that there still exist the segments that are not identified, including missed structural elements as well as some scattered segments which are not relevant to the overall structure of the program. Generally, almost half of a program's structure (measured in time) is undetermined. As mentioned, the lack of ability of extracting a generalized model from the set of sequences with evident difference is another limitation needed to be considered, like the alternation of anchor's announcement and reports in NEWS (the number of alternations varies across episodes). Besides, content-based interpretation is also important, for the practical segmentation task, which is repeatedly mentioned in this chapter. Targeting these limitations, in the next chapter, we will discuss several perspectives after concluding the thesis.

Chapter 9

Conclusion and Perspectives

Content

This dissertation focuses on the content-based TV program structuring, aiming at inferring a common structural model for a recurrent program shared by a collection of episodes. The work targets recurrent TV programs, involving various types of programs, but adopting very limited prior domain knowledge. To discover the program structure with maximal semantic understanding but minimal domain prior knowledge, we propose thus an unsupervised framework addressing the problem of program structuring based on grammatical inference. The proposed framework allows us to determine the structural elements composing the program, and construct a grammar model representing the overall structure of the program. Having such a structural grammar for a program allows facilitating the segmentation of the episodes from the same program, which is the practical purpose of the proposed work.

We summarize hereafter our contributions to the problem of recurrent TV program structuring and we discuss some perspectives.

9.1 General conclusions

We have proposed an unsupervised framework addressing the problem of structure discovery for recurrent TV programs. Leveraging grammatical inference techniques, we have shown that relevant structures can be discovered with only minimal domain knowledge, based on which a structural model can be designed to accurately segment new episodes. In order to achieve the task of grammatical inference, we have proposed an unsupervised framework consisting of two main stages: Structural element determination and grammatical inference.

Structural element determination aims at detecting all relevant constitutive element composing the program. Considering the assumption that no prior knowledge of the program structure is adopted, i.e., no prior knowledge on the structural elements which may be present and very limited knowledge on the program genre. The discovery of structural element is achieved by a variety of broad scope event detectors applied on a collection of episodes and temporal distribution filtering. The only usage of domain knowledge is to identify the structural elements, i.e., naming them from general event detectors exhibiting the property of repetitiveness. The minimal domain knowledge used to name the elements provides semantic interpretations of the program structure. Besides, the corresponding symbolic representation of structural elements provides the feasibility of further studying the episodes, i.e., the grammatical inference in our case. Symbolic representation of structural elements can also be used for other purposes, such as being an input to a number of symbolic data mining algorithms to extract knowledge from a collection of episodes.

Grammatical inference focuses on inferring a structural grammar shared by the collection of episodes, leveraging two alignment techniques, i.e., multiple sequence alignment and uniform resampling. Two proposed grammar inference methods are proved to be capable of inferring grammars for a collection of episodes, even though the resulting grammars show their distinct pros and cons. With symbolic representation of structural elements, grammatical inference allows discovering relevant structures of recurrent programs, from which a grammar model can be designed for segmentation tasks. Experimental evaluation on various types of programs demonstrates that the inferred grammar model can well reflect the program structures, and can be efficiently utilized to segment the new episodes of the same program.

The proposed framework is unsupervised, as during the process of structure discovery, no supervision is involved. As no prior knowledge of program structure is considered, another contribution herein comparing practical algorithms is that the proposed approach

can be applied on a large variety of programs, which is a big step of advance for the task of program structuring. The idea of grammatical inference is proved to be a promising method for understanding general structures of TV programs, further for segmentation tasks.

9.2 Discussion and perspectives

The thesis has demonstrated that grammatical inference is feasible for recurrent program structure discovery with minimal prior knowledge but providing maximal semantic interpretation of program structures. However, there are several open issues that we want to discuss. Targeting the problems, we propose new research directions where the proposed approach could be improved and exploited.

9.2.1 Small object mining

The general event detectors adopted in the thesis can be successfully used to discover the structural elements for various types of programs. However, there may exist better choice of event detectors, i.e., having less detectors but more structural elements that are discovered, or adding various types of detectors to enrich the structural elements. In particular, in order to discover generic structural elements, we just incorporated a large number of audiovisual detectors to detect global-scaled audiovisual events, i.e., the general events that are usually extracted from a whole video frame (e.g., monochrome image) or the most important part of a video frame (e.g., person clustering). However, according to inferred grammar represented in Chapter 8, almost half of a program's structure (measured in time) is undetermined. The content of some segments is not always evident to be explored even though they target on the same theme across episodes. For example, in the talk show, the flash question sections conducted by different people with various questions can hardly be detected.

In order to improve the completeness of a program structure, i.e., recovering the structure with all potential structural elements, we propose that the small visual objects present in a recurrent TV program serve as another important clue for further understanding program structure. Some small logos (usually less than 20% of the image) often appear at specific parts of a program. For example, in talk show programs, some small objects, such as the same screen, usually appear in flash question parts across episodes. Different from the global-scaled audiovisual events, such small visual objects are usually extracted from

partial image of a video frame.

Based on this insight, we propose to mine such small visual objects for augmenting the completeness of structure of recurrent programs. Concretely, using the proposed approach in the thesis, we firstly infer a raw structural grammar from a collection of episodes of a program, where the structural elements are determined using global-scaled event detectors. We advocate for the existing methods of determination of structural elements and structural grammar inference, because the global-scaled events are indeed the most important clue for understanding the program structures. Given the raw structural grammar, the second stage mines the presence of small objects and analyzes their temporal distribution for the video segments that are labeled as unknown structural elements. A possible approach for mining small objects is presented in [44]. The small objects are automatically discovered and classified into different groups, where the frames assigned to the same cluster are supposed to have the same small objects. By analyzing the temporal distribution of the mined small objects, for instance with density filtering proposed in Chapter 4, we can discover a number of new structural elements, where similar small objects are present across episodes, and a finer structure can be inferred.

Some preliminary results are obtained by running the small object mining technique on the datasets used in the thesis. Figure 9.1 illustrates five instances extracted from the programs, where each row represents the images from a cluster containing the same small objects. The corresponding datasets are indicated at the left of each row. Obviously, we can tell that the screen and the lighted circles are the same objects shared by the images in the first row. Practically, these frames correspond to a flash question scene at the end of different episodes. Similar observation can also be found in others examples: the frames in the second row, containing television logos and black regions, refer to a flash news scene; the third row with the same weather cliparts and the striped backdrops obviously corresponds to the weather forecast. The clusters in the last two rows seem less obvious that each row of them shares the same visual objects. However, very similar shapes can be found for each of them: in the forth row, the flowers and leaves are the clue for the frames belonging to a plant representation scene; while the yellow panes in the last row are deemed as the clue for a scene of game winner for the game show. These scenes can hardly be detected using the global-scaled audiovisual detectors. One may says that some cases above could also be discovered by global-scaled audiovisual detectors, such as the examples at the first and the third rows. However, detecting them usually needs massive prior knowledge on the program, even more requiring the detailed characteristics on the frames. By leveraging small object mining, no prior knowledge are required.

Figure 9.1: Instances for small object mining

In general, small object mining can easily discover and cluster the images sharing partial identical (quasi-identical) to deeply discover the structure of programs, serving as an important clue for further exploring program structures, increasing the variety of determined structural elements.

9.2.2 Regular expression

In this thesis, grammatical inference has been demonstrated as a feasible approach for structure discovery task. The inferred grammars can accurately reflect the overall structures of recurrent programs. However, the grammars considered in the thesis still have limited capacity for structure expression. This is evident in the case of NEWS: The alternation of anchor's announcement and news reports that constitute the bulk of a news program can not be expressed, because of the varying number of such alternations across episodes. Thus combining regular expressions [4,69] into the grammar model constructions should be a good choice.

Regular expression is to specify certain regular patterns by leveraging logical operators. In formalism, the operations can be combined to form arbitrarily complex expressions [64].

For example, an asterisk "$*$" is a typical regular expression used to indicate that there is more preceding element, like AB^* referring to AB, ABB, $ABBB$, etc. A boolean or can be expression by "|", referring to alternatives, such as $A|BC$ matching with AC or BC. Besides, parentheses can combine different operators into one construction, defining the scope and precedence of the operators.

Making use of such regular expressions can enhance the structure expression capacity of grammars, hence improving the structure granularity of certain programs. For instance, In NEWS, the alternation of anchor's announcement and news reports can be easily expression by regular expressions as $(AB)^*$, where A refers to anchor's announcement and B refers to news reports. The regular expression can be explained as that a varying number of anchor's announcement and news reports repeats in the program. Adopting regular expression seems to be a feasible method to express complex program structures. Consequently, more complex inference techniques shall be studied to adapt to the case of regular expressions, like identifying regular pattern during the processing of grammatical inference [68], or enhancing the grammar generalization after an initial grammars using regular expressions [35].

9.2.3 Content-based segmentation

Besides improving the techniques of grammatical inference, we should also improve the way we segment new episodes using the grammars. The quality of grammar models refers to not only the nature of the grammars, but also the ability of segmenting new episodes, which is the most important purpose of having a grammar model for programs.

The model derived from the grammar was still limited to time considerations to find the boundaries of the structural elements for the new episodes. Not using a content-based model for segmentation task is justified by the fact that the structural elements might have little, if any, commonalities at the content level. Yet, simple rules could be used to design a content-based model of some structural elements, e.g., separators or dialogues, based on the set of broad scope detectors.

In particular, after directly applying the grammar model to the episodes to be segmented, as stated in Chapter 8, time-based boundaries could be predicted. The proposed segmentation strategy did not account for the content of new episodes. To improve that, simple rules could be used to design a content-based segmentation of some structural elements. We now suggest to apply the set of broad scope events detectors on the predicted episodes to simply discover their contents, estimating content-based boundaries based on

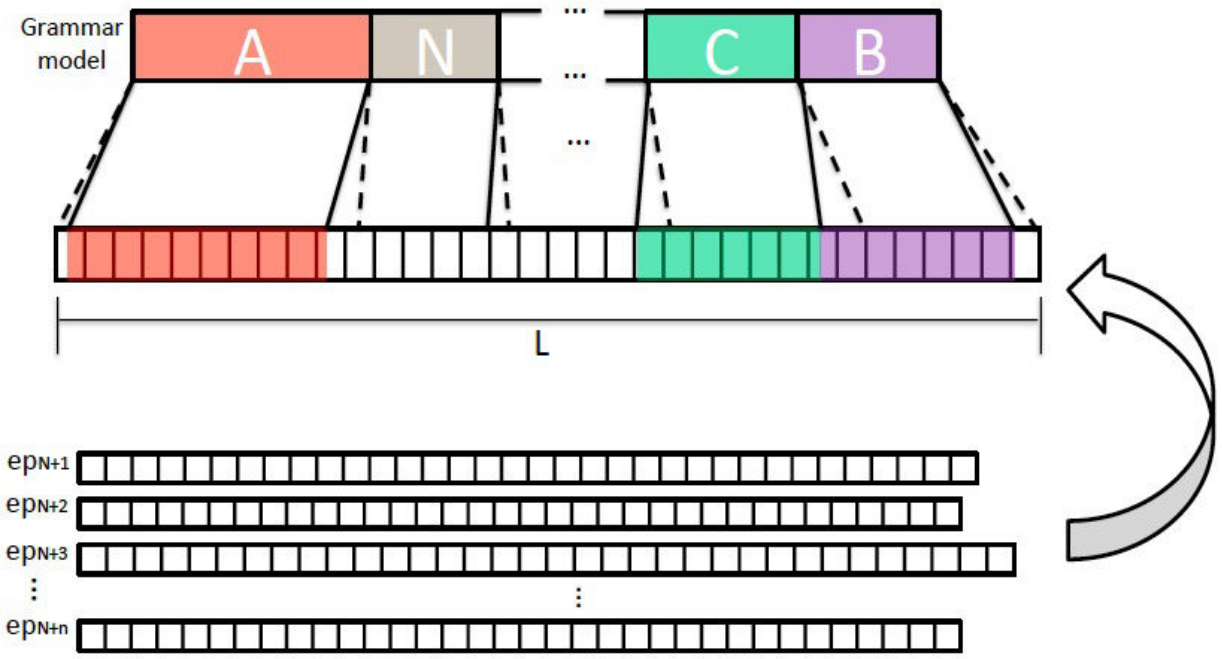


Figure 9.2: Content-based segmentation of new episodes

the event types as well as domain knowledge. Specifically, after the time-based segmentation, we could adjust the boundaries of structural elements by considering the content around the boundaries. On the one hand, by jointly considering the time-based boundaries and the content-based boundaries, the more accurate boundaries of structural elements could be found. On the other hand, the content discovery of the episodes also provides a verification of the prediction. An illustrative example in Figure 9.2 shows the content-based segmentation after a time-based segmentation. The dash lines represent the positions of the element considering just the time-based boundaries, while the solid lines represent the adjusted positions of elements considering both the time-based and the content-based boundaries.

In summary, the thesis is a big step of advance for content based TV program structure discovery. Due to time limitation, there exist several problems in the work to be further explored and ameliorated. Targeting the limitations, some potential solutions are proposed. The perspectives raise scientific challenges, leading to new research directions in further work.

Bibliography

[1] A. E. Abduraman. *Structuration intra-programme de contenus* TV. PhD thesis, Télécom ParisTech, 2013.

[2] A. E. Abduraman, S.-A. Berrani, and B. Merialdo. An unsupervised approach for recurrent TV program structuring. In *European Interactive TV Conference*, pages 123–126, 2011.

[3] A. E. Abduraman, S.-A. Berrani, and B. Merialdo. Audio/visual recurrences and decision trees for unsupervised TV program structuring. In *The 8th International Conference on Computer Vision Theory and Applications*, pages 701–708, 2013.

[4] V. Alfred. Algorithms for finding patterns in strings. *Algorithms and Complexity*, 1:255, 2014.

[5] N. Ancona, C. Cicirelli, A. Branca, and A. Distante. Goal detection in football by using support vector machines for classification. In *International Joint Conference on Neural Networks*, volume 1, pages 611–616, 2001.

[6] A. Barjatya. Block matching algorithms for motion estimation. *IEEE Transactions Evolution Computation*, 8(3):225–239, 2004.

[7] M. Ben and G. Gravier. Unsupervised mining of audiovisually consistent segments in videos with application to structure analysis. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2011.

[8] S.-A. Berrani, G. Manson, and P. Lechat. A non-supervised approach for repeated sequence detection in tv broadcast streams. *Signal Processing: Image Communication*, 23(7):525–537, 2008.

[9] M. Bertini, A. Del Bimbo, and P. Pala. Content-based indexing and retrieval of TV news. *Pattern Recognition Letters*, 22(5):503–516, 2001.

[10] Z. Botev, J. Grotowski, and D. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010.

[11] M. Broilo, A. Basso, and F. G. De Natale. Unsupervised anchorpersons differentiation in news video. In *International Workshop on Content-Based Multimedia Indexing*, pages 115–120. IEEE, 2011.

[12] R. Brunelli, O. Mich, and C. M. Modena. A survey on the automatic indexing of video data. *Journal of visual communication and image representation*, 10(2):78–112, 1999.

[13] V. Brunie, J. Carrive, and L. Vinet. Ingénierie des documents audiovisuels: le projet FERIA: Une approche centrée sur la description des contenus. *Technique et Science Informatiques*, 25(4):469–496, 2006.

[14] L. Catanese, N. Souviraà-Labastie, B. Qu, S. Campion, G. Gravier, E. Vincent, and F. Bimbot. MODIS: an audio motif discovery software. In *Annual Conference of the International Speech Communication Association*, pages 2675–2677, 2013.

[15] W. Chai. Semantic segmentation and summarization of music: methods based on tonality and recurrent structure. *IEEE Signal Processing Magazine*, 23(2):124–132, 2006.

[16] Y.-F. Chang, P. Lin, S.-H. Cheng, K.-H. Chan, Y.-C. Zeng, C.-W. Liao, W.-T. Chang, Y.-C. Wang, and Y. Tsao. Robust anchorperson detection based on audio streams using a hybrid i-vector and DNN system. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–4. IEEE, 2014.

[17] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk, and B. Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *IEEE International Conference on Image Processing*, pages 2609–2612. IEEE, 2011.

[18] S.-C. Chen, M.-L. Shyu, M. Chen, and C. Zhang. A decision tree-based multimodal data mining framework for soccer goal detection. In *IEEE International Conference on Multimedia and Expo*, volume 1, pages 265–268, 2004.

[19] K. Choroś. Automatic fast detection of anchorperson shots in temporally aggregated tv news videos. In *Intelligent Information and Database Systems*, pages 339–348. Springer, 2015.

[20] G. E. Crooks, G. Hon, J.-M. Chandonia, and S. E. Brenner. Weblogo: a sequence logo generator. *Genome research*, 14(6):1188–1190, 2004.

[21] M. Delakis, G. Gravier, and P. Gros. Audiovisual integration with segment models for tennis video parsing. *Computer vision and image understanding*, 111(2):142–154, 2008.

[22] A. Dielmann. Unsupervised detection of multimodal clusters in edited recordings. In *IEEE International Workshop onMultimedia Signal Processing*, pages 177–182, 2010.

[23] N. Dimitrova, H.-J. Zhang, B. Shahraray, I. Sezan, T. Huang, and A. Zakhor. Applications of video-content analysis and retrieval. *IEEE multimedia*, 9(3):42–55, 2002.

[24] Y. Dong, G. Qin, G. Xiao, S. Lian, and X. Chang. Advanced news video parsing via visual characteristics of anchorperson scenes. *Telecommunication Systems*, 54(3):247–263, 2013.

[25] E. Dumont and G. Quénot. Automatic story segmentation for tv news video using multiple modalities. *International Journal of Digital Multimedia Broadcasting*, 2012.

[26] S. Eickeler and S. Muller. Content-based video indexing of TV broadcast news using hidden Markov models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 2997–3000, 1999.

[27] S. Eickeler, F. Wallhoff, U. Lurgel, and G. Rigoll. Content based indexing of images and video using face detection and recognition methods. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1505–1508, 2001.

[28] A. Ekin, A. M. Tekalp, and R. Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, 2003.

[29] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[30] X. Gao and X. Tang. Unsupervised video-shot segmentation and model-free anchorperson detection for news video story parsing. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(9):765–776, 2002.

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

[32] B. Gu, A. Mu, A. M. Tekalp, et al. Temporal video segmentation using unsupervised clustering and semantic object tracking. *Journal of Electronic Imaging*, 7(3):592–604, 1998.

[33] A. Guéziec. Tracking pitches for broadcast television. *Computer*, 35(3):38–43, 2002.

[34] A. Hanjalic, R. Lagendijk, and J. Biemond. Template-based detection of anchorperson shots in news programs. In *IEEE International Conference on Image Processing*, pages 148–152, 1998.

[35] J. E. Hopcroft. *Introduction to automata theory, languages, and computation*. Pearson Education India, 1979.

[36] Z. A. A. Ibrahim and P. Gros. TV stream structuring. *ISRN Signal Processing*, 2011, 2011.

[37] A. Jacobs. Using self-similarity matrices for structure mining on news video. In *Advances in Artificial Intelligence*, pages 87–94. Springer, 2006.

[38] G. Jaffré, P. Joly, et al. Costume: A new feature for automatic video content indexing. In *Recherche d'Information Assistée par Ordinateur*, pages 314–325, 2004.

[39] D. B. Jayagopi, S. Ba, J.-M. Odobez, and D. Gatica-Perez. Predicting two facets of social verticality in meetings from five-minute time slices and nonverbal cues. In *International Conference on Multimodal interfaces*, pages 45–52, 2008.

[40] P. Ji, L. Cao, X. Zhang, L. Zhang, and W. Wu. News videos anchor person detection by shot clustering. *Neurocomputing*, 123:86–99, 2014.

[41] E. Kijak, G. Gravier, P. Gros, L. Oisel, and F. Bimbot. HMM based structuring of tennis videos using visual and audio cues. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages III–309, 2003.

[42] E. Kijak, G. Gravier, L. Oisel, and P. Gros. Audiovisual integration for tennis broadcast structuring. *Multimedia Tools and Applications*, 30(3):289–311, 2006.

[43] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. A. McGettigan, and H. McWilliam. Clustal W and clustal X version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[44] P. Letessier, O. Buisson, and A. Joly. Scalable mining of small visual objects. In *ACM International Conference on Multimedia*, 2012.

[45] H. Li, J. Tang, S. Wu, Y. Zhang, and S. Lin. Automatic detection and analysis of player action in moving background sports video sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(3):351–364, 2010.

[46] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining periodic behaviors for moving objects. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 1099–1108, 2010.

[47] B. Logan and S. Chu. Music summarization using key phrases. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II749–II752. IEEE, 2000.

[48] L. Lu, M. Wang, and H.-J. Zhang. Repeating pattern discovery and structure analysis from acoustic music data. In *ACM International Workshop on Multimedia Information Retrieval*, pages 275–282, 2004.

[49] A. Mittal, L.-F. Cheong, and L. T. Sing. Robust identification of gradual shot-transition types. In *International Conference on Image Processing*, volume 2, pages II–413, 2002.

[50] A. Muscariello, G. Gravier, and F. Bimbot. An efficient method for the unsupervised discovery of signalling motifs in large audio streams. In *Content-Based Multimedia Indexing*, pages 145–150, 2011.

[51] A. Muscariello, G. Gravier, and F. Bimbot. Unsupervised motif acquisition in speech via seeded discovery and template matching combination. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(7):2031–2044, 2012.

[52] X. Naturel, G. Gravier, and P. Gros. Fast structuring of large television streams using program guides. In *Adaptive Multimedia Retrieval: User, Context, and Feedback*, pages 222–231. Springer, 2007.

[53] C. Panagiotakis and G. Tziritas. A speech/music discriminator based on RMS and zero-crossings. *IEEE Transactions on Multimedia*, 7(1):155–166, 2005.

[54] N. V. Patel and I. K. Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30(4):583–592, 1997.

[55] J.-P. Poli. *Structuration automatique de flux télévisuels*. PhD thesis, Université Paul Cézanne-Aix-Marseille III, 2007.

[56] J.-P. Poli and J. Carrive. Modeling television schedules for television stream structuring. In *Advances in Multimedia Modeling*, pages 680–689. Springer, 2006.

[57] K. M. Pua, J. M. Gauch, S. E. Gauch, and J. Z. Miadowicz. Real time repeated video sequence identification. *Computer Vision and Image Understanding*, 93(3):310–327, 2004.

[58] B. Qu, F. Vallet, J. Carrive, and G. Gravier. Content-based inference of hierarchical structural grammar for recurrent TV programs using multiple sequence alignment. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2014.

[59] B. Qu, F. Vallet, J. Carrive, and G. Gravier. Using grammar induction to discover the structure of recurrent TV programs. In *International Conferences on Advances in Multimedia*, pages 112–117, 2014.

[60] B. Qu, F. Vallet, J. Carrive, and G. Gravier. Content-based discovery of multiple structures from episodes of recurrent TV programs based on grammatical inference. In *ACM International Conference on Multimedia Modeling*, pages 140–154, 2015.

[61] Z. Rasheed and M. Shah. Detection and representation of scenes in videos. *IEEE Transactions on Multimedia*, 7(6):1097–1105, 2005.

[62] S. Ray and R. H. Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *The 4th International Conference on Advances in Pattern Recognition and Digital Techniques*, pages 137–143. India, 1999.

[63] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[64] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[65] C. G. Snoek and M. Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25(1):5–35, 2005.

[66] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Improved sensitivity of profile searches through the use of sequence weights and gap excision. *Computer Applications in the Biosciences*, 10(1):19–29, 1994.

[67] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

[68] K. Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

[69] B. A. Trakhtenbrot and Y. M. Barzdin. *Finite automata*. American Elsevier Publishing Company, 1973.

[70] F. Vallet. *Structuration automatique de talk shows télévisés*. PhD thesis, Paris, Télécom ParisTech, 2011.

[71] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4:34–47, 2001.

[72] 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*, 10(3):393–408, 2008.

[73] L. Xie, P. Xu, S.-F. Chang, A. Divakaran, and H. Sun. Structure analysis of soccer video with domain knowledge and hidden Markov models. *Pattern Recognition Letters*, 25(7):767–775, 2004.

[74] X.-F. Yang, Q. Tian, and P. Xue. Efficient short video repeat identification with application to news video structure analysis. *IEEE Transactions on Multimedia*, 9(3):600–609, 2007.

[75] M. M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *Proceedings., International Conference on Image Processing*, volume 1, pages 338–341. IEEE, 1995.

[76] X. Yu, L. Li, and H. W. Leong. Interactive broadcast services for live soccer video based on instant semantics acquisition. *Journal of Visual Communication and Image Representation*, pages 117–130, 2009.

[77] H. Zhang, Y. Gong, S. W. Smoliar, and S. Y. Tan. Automatic parsing of news video. In *The International Conference on Multimedia Computing and Systems*, pages 45–54, 1994.

[78] J. Zhang, J. Qiu, X. Wang, and L. Wu. Representation of the player action in sport videos. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–4. IEEE, 2013.

[79] T. Zlitni, B. Bouaziz, and W. Mahdi. Automatic topics segmentation for tv news video using prior knowledge. *Multimedia Tools and Applications*, pages 1–28, 2015.

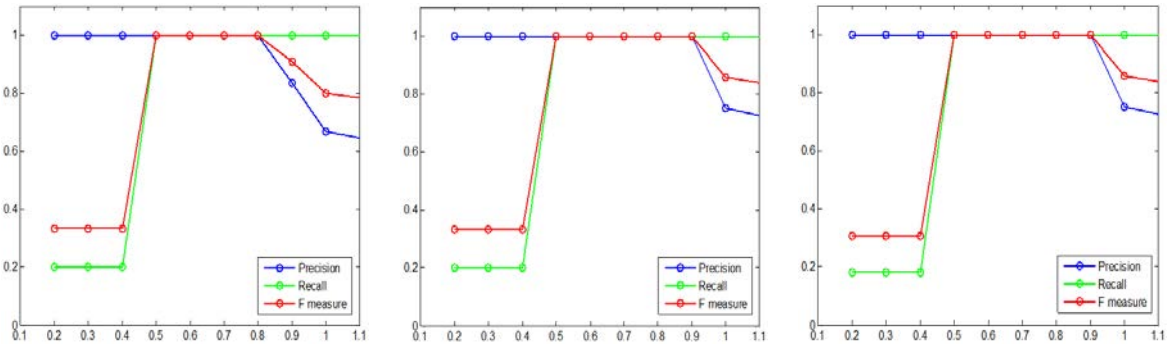
Appendix A

Appendix

A.1 Analysis of the threshold of density function - the results for individual data sets

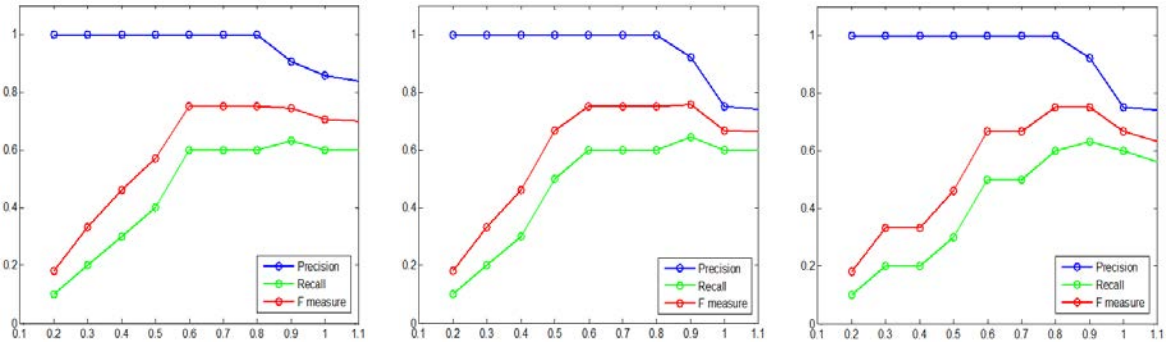
Figures A.1, A.2, A.3 and A.4 show the experiments for analysis of the threshold applied on the density function on the four programs, i.e., NEWS, GAME, TALK and MAGZ, respectively. As the number of episodes also produces influences on the estimated density function, hence we fix several numbers of episodes used and observe the effect of the parameter n. On the vertical axis, the obtained values for the precision, recall and F measure are illustrated. On the horizontal axis, the different values corresponding to the parameter n are shown.

As shown, from the results of the four data sets, the increase of the parameter n (decrease of the threshold) results in an increase of recall and a decrease of precision, as more sporadic occurrences are involved in. Consequently, F measure exhibits a decrease after an initial increase. When F measure goes to its maximum, it implies a best trade-off between precision and recall. In Figure A.1, evidently the maximal F measure lies between 0.5 to 0.8 in the three cases of episode numbers. For GAME, in the Figure A.2, the maximal F measure can be found between 0.6 to 0.9 when the number of episodes corresponds to $6 \sim 9$; when the number of episodes equals to 12, the maximal F measure is between 0.8 to 0.9. For TALK, when n lies between 0.5 to 0.9, F measure goes to its maximum for the number of episodes being 6. When the number of episodes is 9 or 12, the two results are very similar, n between 0.5 and 0.6 corresponds to the best F measure. The results for



(a) ep = 6 (b) ep = 9 (c) ep = 12

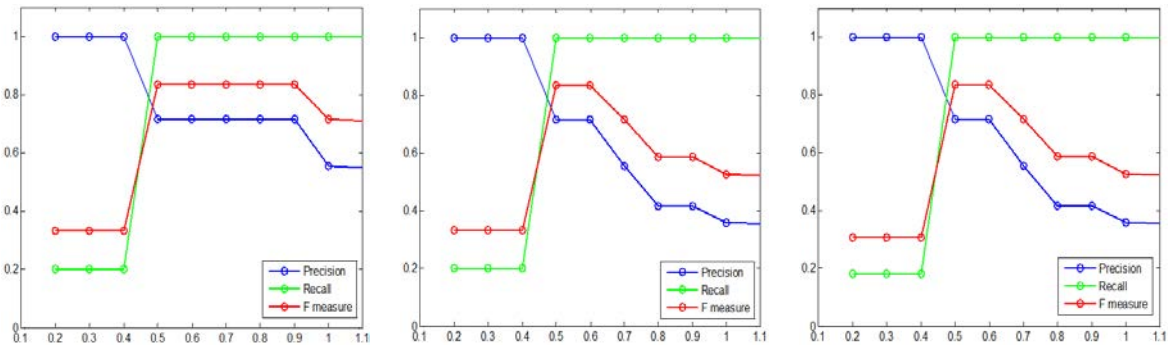
Figure A.1: Precision, recall and F measure for NEWS (ep: number of episodes involved for the evaluation)



(a) ep = 6 (b) ep = 9 (c) ep = 12

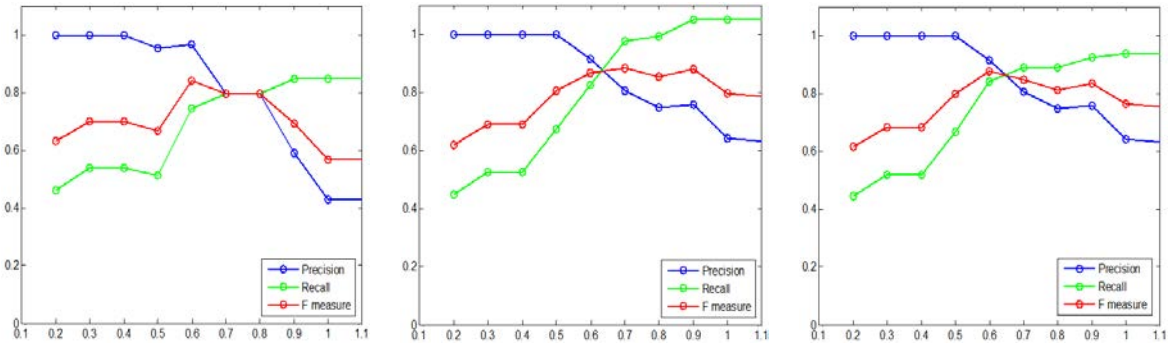
Figure A.2: Precision, recall and F measure for GAME (ep: number of episodes involved for the evaluation)

MAGZ are less evident to fix the best n, yet it seems that when n is between 0.6 and 0.9, the F measures in the three cases reach highest values.



(a) ep = 6 (b) ep = 9 (c) ep = 12

Figure A.3: Precision, recall and F measure for TALK (ep: number of episodes involved for the evaluation)



(a) ep = 6 (b) ep = 9 (c) ep = 12

Figure A.4: Precision, recall and F measure for MAGZ (ep: number of episodes involved for the evaluation)

List of Figures

List of Tables

Résumé

La structuration des programmes télévisés est une des grandes thématiques des dix dernières années pour la tâche d'indexation de haute qualité. La structuration des programmes focalise sur la recherche des instantes clés des événements d'intérêt et la compréhension des structures des programmes. Dans la littérature, certaines études travaillent sur le problème de la détection des événements ou de la segmentation des scènes, qui ne sont pratiquement pas utilisable pour des tâches d'indexation en raison du manque de la compréhension sémantique sur la structure du programme. Alternativement, certaines approches structurent des programmes avec des interprétations sémantiques, mais demandant énorme connaissances préalables sur la structure du programme. Par conséquent, des études récentes abordent le problème sur la structuration des programmes avec minimum de connaissances préalables mais plus de interprétations sémantiques sur la structure du programme.

Dans cette thèse, suivant la dernier direction, on aborde le problème de la structuration des programmes télévises de manière non supervisée à partir du point de vue de l'inférence grammaticale, focalisant sur la découverte de la structure des programmes récurrents à partir d'une collection homogène. Programme récurrent se réfère à des programmes avec plusieurs épisodes diffusés périodiquement, par exemple, quotidienne ou hebdomadaire. Prenant des journaux télévisés comme exemple: Les journaux commencent généralement par un bref titre des nouvelles, suivi par une alternance de l'annonce du présentateur et des nouvelles correspondante; La plupart des journaux télévisés finit avec des segments d'interview, les évènements sportifs ou bandes-annonces de programme. En conséquence, les épisodes des programmes récurrents se présentent deux propriétés, la répétitivité et la stabilité temporelle des éléments structuraux, c'est-à-dire, à travers des épisodes les mêmes éléments structuraux se trouvent dans presque le même ordre et avec une durée similaire. L'inférence grammaticale sur des programmes récurrents permet d'utiliser minimum des connaissances a priori dans le domaine pour atteindre la découverte de la structure des programmes. En particulier, on suppose qu'il n'y a aucune connaissance préalable sur les éléments structuraux qui pourraient être présent dans un programme et une connaissance très

limitée sur le type de programme. Avec cette hypothèse, on propose un cadre non supervisé se déroulant en deux phases.

Premièrement, on vise à découvrir les éléments structuraux qui sont pertinents à la structure du programme. Afin d'atteindre cet objectif, des détecteurs multimodaux sont implémentés et appliqués sur une collection d'épisodes d'un programme pour détecter des événements généraux, qui sont potentiellement liés à la structure du programme. En suite, un filtrage par la densité temporelle des événements généraux est utilisé afin de déterminer les éléments structuraux de programme. En adoptant la propriété de répétition, la détermination des éléments structuraux est réalisée d'une manière non supervisée, où seulement minimum des connaissances du domaine sont impliquées. Par conséquent, l'approche proposée n'est pas spécifique à un certain type de programmes, mais répond à une large catégorie de programmes récurrents.

Deuxièmement, ayant les éléments structuraux pour les épisodes dans la collection, on s'intéresse à l'inférence grammaticale de la structure des programmes, qui vise à trouver un alignement optimal sur des éléments structuraux entre les épisodes afin de apporter à régularités légers et en déduire un modèle commun partagé par les épisodes. Au-delà, à travers les épisodes, la structure d'un programme peut varier selon l'application éditoriaux, qui conduisent à multiples structures pour un programme récurrent. Prenant l'exemple de journaux télévisés, les épisodes où il y a des invités présents, ils finissent généralement avec une interview. Il y a aussi un certain nombre d'épisodes qui se terminent par des bandes-annonces de films, par exemple, quand un nouveau film est sorti. Des multiples structures sont fréquemment observées dans les téléviseurs. Donc l'identification de multiples structures et des grammaires sont indispensables pour éviter l'ambiguïté de la structure. Par conséquent, la tâche la grammaire d'inférence est liée à l'identification de multiples structures pour un programme récurrent. Afin d'atteindre l'inférence grammaticale, deux méthodes différentes, technique d'alignement des séquences multiples et technique d'alignement par ré-échantillonnage uniforme, sont adoptées afin de découvrir les grammaires pour ces épisodes. Enfin, des modèles structuraux issus des grammaires sont générés pour segmenter des nouveaux épisodes de même programme.

L’expérimentation est appliquée sur quatre programmes de différents types pour évaluer la performance des grammaires et des modèles. Focalisant sur la détermination des éléments structuraux, on analyse l'effet sur le nombre d'éléments détectés, afin de déterminer le seuil appliqué sur la fonction de densité du filtrage ainsi que le nombre des épisodes dans la collection. Pour l’inférence de la grammaire, on discute de la qualité des grammaires obtenues et montrons qu'ils reflètent fidèlement la structure du programme. On démontre également que les modèles obtenus par inférence grammaticale peuvent prédire la structure des nouveaux épisodes, effectuant une évaluation quantitative et comparative entre deux méthodes par segmenter les nouveaux épisodes.

Enfin, on discute des questions ouvertes sur la découverte de la structure, et indique trois nouvelles directions de recherche pour aborder dans les travaux futurs: L'exploitation des petites objets est adressée pour découvrir des structures de programmes plus complets; Expression rationnelle est proposé d'améliorer la capacité d'extraction de modèle plus généralisé pour les épisodes partageant le même modèle de structure, mais avec la différence évidente; La segmentation basée sur le contenu vise à améliorer l'efficacité de la structuration de nouveaux épisodes en utilisant des modèles de grammaire dans des utilisations pratiques.