LEMMS: Label Estimation of Multi-Feature Movie Segments

LEMMS: Label Estimation of Multi-feature Movie Segments

Bartolomeo Vacchetti
Polytechnic of Turin
Turin, Italy
bartolomeo.vacchetti@polito.it

Dawit Mureja Argaw
KAIST
Daejeon, South Korea
dawitmureja@kaist.ac.kr

Tania Cequtelli
Politechnic of Turin
Turin, Italy
tania.cequitelli@polito.it

Abstract

In the last few years, there has been an increasing amount of methods and algorithms that approach and automate different video and image editing tasks. A task that so far has not been investigated too much in depth is the analysis of video editing patterns. In this work, we present LEMMS *(Label Estimation of Multi-feature Movie Segments), a methodology to analyze and label 30-second-long movie editing patterns based on the following editing features: shot size, shot subject, editing pace, and editing trend.* LEMMS *can identify more or less fine-grained editing classes using a multi-clustering approach. To evaluate the robustness of* LEMMS *in assigning correct labels the performance of an LSTM classifier is analyzed. For our study, we extracted 24 363 segments of movie scenes from the AVE [1] dataset. The performance of* LEMMS *in semi-automatic label identification for 30-second long movie segments is accurate, as the proposed approach has an overall accuracy of 92.8% for 50 classes.*

1. Introduction

Recently more and more creative and astonishing results have been achieved in different creative areas related to the video and image editing context. On one hand traditional image and video editor software developers such as Adobe are implementing deep learning methodologies to improve their editing tools or create new ones (Firefly), while on the other hand, new algorithms, such as the latent diffusion models [19], allows by themselves amazing results. These new approaches vary from the generation and editing of single video clips from an image and a prompt [14] to image and video creation from a single textual prompt [24] and more(see Section 2). Most of these models and algorithms require massive data and resources to be trained. On the other hand, a context that so far has not been investigated too much in depth is the editing structure of videos. With editing structure, we mean the concatenation of shots chosen by the director to represent the scene. Many movie critics, such as [15], have always highlighted how the editing process heavily defines the movie or video, as it deeply impacts the viewer's perception. By being able to label and classify editing patterns more powerful video editing assist tools could be developed. For instance imagine a text to video model that also uses basic editing patterns. Instead of creating a video made of a single clip it would be possible to create a video with multiple clips and coherent cuts. To analyze the editing structures of videos we present the LEMMS methodology. Additionally, our approach works with metadata and light models and hence does not require high computing resources. In order to study video editing we represent the movie scenes as sequences of symbols that encode different features. Since every scene is unique, instead of analyzing them we analyze scene segments. Our intuition is the following: similar situations can happen in different scenes at different moments, but they will rely on similar shots. For instance, a dialogue can happen in different scenes and in different moments of a scene, but most likely close-ups and medium shots will be used to represent it. Depending on the mood of the moment, these shots will be concatenated in a different order and with a different frequency. If something else is happening, a different editing pace and different shots will be used. This is not surprising since the shots are chosen in relation to what the director wants to show. Consider the example in Figure 1 to illustrate better what we mean. The top segment is from the movie "Thoroughbreds" by Cory Finley. The second segment is from the movie "The Birdcage" by Mike Nichols. In both of these segments, the focus is on the interaction of people with objects. The reasons why the directors chose to have people interact with objects are different, but the way they portrayed it is similar. This is not the only way to show people interacting with objects, since there are many. The "shot-reverse shot" technique shown in Figure2 is a more familiar example. The top segment is from "The Great Lebowski" by Joel and Ethan Coen, while the bottom is from "Pulp Fiction" by Quentin Tarantino. Again, the same set of shots was used to describe a similar context, namely two people talking to each other, but the reason why

they talk to each other and what they say to each other is different.

Keeping these elements into account, we have devised LEMMS, a semi-unsupervised methodology to label scene segments according to the editing pace adopted, the shot size used, and the subjects shown. Additionally, we define the scene segment trend, a feature that encodes information concerning how the shot sizes in the segments are concatenated (see section 3). To sum it up, we present the following contributions:(i)Segmented AVE: a dataset created from AVE by rearranging the 5,591 scenes into 30-second long segments characterized by shot size, shot size, and editing pace, and trend; (ii)LEMMS: a semi-unsupervised approach that relies on clustering algorithms[2] and Levenshtein[10] distance to estimate the labels of the scene segments; (iii) a preliminary quantitative and qualitative analysis of the identified segments and labels on the new dataset. The robustness of LEMMS was validated by building an LSTM classifier on the results of the groups of movie segments. The model was trained on 50 classes and tested with a 10-fold stratified validation strategy, achieving an overall accuracy of 92.8%.

The paper is organized as follows. Section 2 discusses the related works. Section 3 illustrates the step that we have used to obtain a sequenced version of the AVE Dataset, while Section 4 presents the LEMMS methodology that we have developed to estimate the scene segment labels. In Section 5 the experimental results along with the qualitative and quantitative analysis are discussed. Finally, Section 6 presents the conclusions and future research directions.

2. Related Works

In the last years there have been different studies focusing on a multitude of task concerning machine learning, movies and video editing. Old studies focus on more classical machine learning tasks, such as image classification, but applied to shot sizes or features. For instance, the authors in [4] [21] [25] [11] addressed the shot size classification. Some of them implement a more fine-grained classification task [21] [25] relying on convolutional neural networks[23]. In [11], the classification of camera movement is performed using motion vector fields. A different research studies video editing tasks' automation or partial automation. For instance, in [6] they cut the video using as input the dialogue text, while in [18], the task addressed is video clip segmentation. In more recent years a lot of improvements have been made in the video generation field. For instance, in Dreamix [14] the authors propose a framework that generates a video from an image and a textual prompt. They can do so by implementing a video diffusion model based on the popular and performing diffusion model[28]. In [24], always relying on diffusion models, the authors develop a methodology to train a video generator using only image-text data and not video data. The model is then able to generate short videos from a textual prompt. All of these works present astonishing results. However, these studies do not take into account editing structures. In fact, they generate single shots. While not many, there have been some works in the past that focused on video editing features or structures. In [13] the authors perform a preliminary next shot prediction task and focus on editing pace. Instead, in [5], the focus is moved on how the concatenation of different shots influences the viewer's perception. In [26], the focus is on identifying editing patterns in 30-second long shot sequences extracted from movies. Our methodology takes inspiration from the methodology presented in [26], however, while the task at hand is similar, i.e. unsupervised movie editing classification, the data is different since multiple features characterize it. Additionally due to the imbalance of specific values in the feature space we have included a multi step clustering strategy. In [9] the authors perform movie style analysis showing how high-level features, such as camera motion and pose estimation, are a better fit for this task with respect to low level features, such as average shot length and color histogram. In [16] the authors investigate how visual audio video patterns trigger cuts between different shots. Other studies focus on video analysis on a larger scale. For instance, in [27], the authors present the Long Video Understanding (LVU) dataset and develop a methodology to perform different tasks. The main goal is long-form video understanding, while the specific task addressed range from content understanding, such as classifying the relationships among characters, to movie metadata predictions, such as director and genre. Also, in [8] the authors perform movie metadata prediction and other tasks on the LVU dataset and MovieNet [12]. They also present two new datasets: MovieCL30K for movie metadata classification and the Mature Content Dataset for video moderation. Other movie datasets that can be used to perform different tasks concerning movie and video analysis have been released in recent years. The Condensed Movie dataset [3] contains the main scenes from different movies with metadata. Unfortunately, the shot sizes are not taken into account. In [17] the authors present a video cut dataset and perform cut transition recognition. A different dataset is Cinescale [20]which contains 120 movies from six different directors. The movies are divided into frames, and each frame has a label according to its shot size. Last year also, the Anatomy of Video Editing Dataset (AVE)[1] was released. It is a dataset that contains 5,591 scenes from different movies. Each shot in every scene is characterized by multiple features such as shot size, shot subject, and shot duration. Due to the multiple features available for every scene, we have decided to use this dataset in our study. While previous studies focus on different interesting applications only a few take into account video editing patterns.

Figure 1. Camera movement showing people interacting with objects from "Thoroughbreds" (top), and "The Birdcage"(bottom).

Figure 2. Shot reverse shot extracted from "Pulp Fiction"(top) and "The Big Lebowski"(bottom).

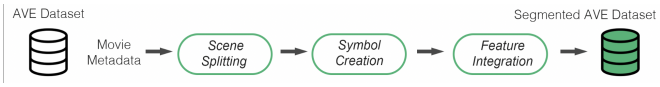

Figure 3. data preprocessing steps.

None of those studies consider all the features we use to analyze editing patterns in this study.

3. Creation of Sequenced Ave

We have chosen the AVE Dataset [1] to extrapolate movie editing sequences. In this dataset, each shot is characterized by multiple features that can be grouped into the following macro-categories; camera attributes: *shot size, shot angle, shot type, shot motion*, set attributes *shot location, shot subject, number of people* and additional attribute*sound source, start time, end time*. In order to have a realistic representation of the editing patterns without increasing the data variability too much, we selected some specific features while discarding others. *Start time* and *end time* were kept to compute the editing pace, while *sound source* wasn't since it is less relevant to the task at hand. For the remaining features, the following consideration was made. They can be divided into set attributes(*subject, number of people, location*) and camera attributes (*size, angle, movement, type*). To represent the editing sequences without increasing the complexity of the data representation too much, we have chosen to represent them using one attribute per set, specifically the *shot size* for the camera attributes and the *shot subject* for the set attributes. Hence the *filtered metadata* that we take from AVE is the scenes characterized in terms of sequences of shots defined by size, subject, start, and end time. An overview of the adopted pre-processing steps can be seen in 3.

Symbol-Size(abbr)	number of samples
0 - other(O)	1,977
1 - extreme close-up (ECU)	405
2 - close-up (CU)	26,692
3 - medium shot (MS)	142,314
4 - wide shot (WS)	22,852
5 - extreme wide shot (EWS)	1,936

Table 1. Shot Sizes

Sequence Splitting. Since, rather than the scenes themselves, which are unique and have varying lengths, we want to analyze the editing patterns in scene segments, similar to [26], we have divided the scenes into 30-second long segments. Each segment is made of 30 shot tokens, one per second. In this study, each token contains the shot size, the shot subject, and the duration obtained from start and

Symbol-Subject(abbr)	number of samples
0 - other (ot)	3 000
1 - face (f)	5 671
2 - human (h)	97 696
3 - location (l)	69 959
4 - object (ob)	19 855
5 - animal (a)	8 995

Table 2. Shot Subjects

end times. By only splitting the scenes into 30-second segments, we would have lost most of the ending parts of the scenes. Thus, instead of discarding the remaining seconds from every scene, we have resampled the ending part of every scene starting from the last 30 seconds. By doing so, we obtained 24,363 movie scene segments from 5,591 movie scenes. Additionally by selecting a fixed time range the editing pace and trend features can be easily extracted.

Symbol Creation Before applying the LEMMS methodology one more step is required, which is to convert the textual features from text to numerical symbols. All the shot sizes classes were kept, while some minor subject sizes classes (*text, limb, cartoon, other*) were merged with the class *other*. Tables 1 and 2 show the number of samples per shot and subject, the numeric symbol used to represent it, and the abbreviation used later on.

Additional Features To represent the editing pace we count the number of different shots in a 30-second segment. Depending on the number of shots the segment belongs to one of the following editing pace classes; slow, medium, and fast. Table 3 shows the number of samples per editing class and how the number of different shots influences the editing pace. To encode this information in the sequences, we added a symbol to each token representing the shot. If it is equal to 1, it means that the frame belongs to a new shot, 0 otherwise. The trend token that composes the editing segments is the last token that we add to every shot token. This token is necessary because we are using numbers as symbols. If we were to use numbers, we would obtain that a *medium shot* (symbol 3) is three times an *extreme close up*, or that the extreme close up (symbol 1) is an average between a close up(symbol 2) and *other* (symbol 0). However, the shot size indicates on a loose scale how close the camera's point of view is to the action. Also, keeping track of whether we are getting closer or further from the subject in a selected segment is important. Hence the addition of the trend token. The token can have different values:(i) 0: which means that the shot size of the next shot is the same;(ii) 1 means that the shot size of the next frame is closer to its subject;(iii) 2 means that the shot size of the next frame is further from its subject. The segments were then labeled into four trend classes depending on their behavior. Table 4 shows the trend classes.

Class	n_{shot} in segment	number of samples
Fast	$n_{shot} > 10$	8,101
Medium	$5 < n_{shot} < 11$	9,354
Slow	$n_{shot} < 6$	6,908

Table 3. Editing paces: definition and cardinality.

Class	trend token	number of samples
Stable	0	8,410
Mixed	0, 1, 2	12,857
Further	2, 0	1,334
Closer	1, 0	1,762

Table 4. Trend classes: composition and cardinality.

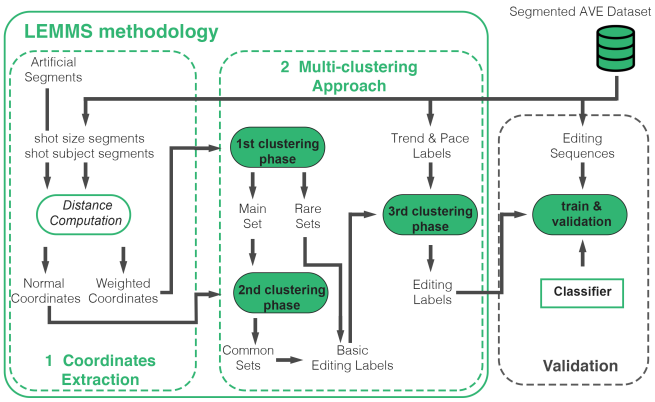

Figure 4. LEMMS methodology:main building blocks.

Each segment is 30 seconds long and is represented by 30 tokens, one per second. Each token represents a shot and is composed of four symbols, each symbol representing a different feature, using the following scheme:

$$Shot_{Size}, Shot_{Subject}, token_{SameShot}, token_{Trend}$$

4. Methodology

Now that we have a segment representation that keeps track of multiple features we can use the LEMMS methodology to estimate the labels of the movie segments. The goal LEMMS is to find labels representative of the editing patterns that characterize the segments in terms of shot sizes used, shot subjects, editing pace and trend. The methodology overview is shown in Figure 4. In the first block through a distance metric we represent the segments as points characterized by size and subject coordinates. Then in the multiple steps clustering block we extract the final labels of the segments. The first two phases identify clusters in terms of shot sizes and subjects, while in the third one we obtain the final labels characterized also in terms of editing pace and trend.

4.1. Coordinates Extraction

In a first analysis step, we model the shot that makes up each scene segment as tokens based on the shot size and shot subject. After recomposing the sequences in terms of shot sizes and subjects, the next step is to compute a distance metric to characterize the segments. We have used the Levenshtein distance[10] taking inspiration from [26]. The Levenshtein distance measures the number of substitutions, additions, or subtractions required to transform a sequence of characters A into a sequence of characters B. Since the sequences have all the same length in our case, it indicates only the number of substitutions required to transform one sequence into another. Since the Levenshtein distance measures the substitution needed to transform a string into another, in our case it is not so useful to apply it directly on our segments. Hence we create artificial sequences. Each artificial sequence has 30 characters that are all set on the same value, one value for each artificial segment. The characters used are the same symbols used to represent shot sizes and subjects. From every segment, we extract two segments, one containing all the shot size tokens and one containing all the shot subject tokens of the segment. Then for both segments, we compute the Levenshtein distance with respect to the artificial segments and we use those distances all together as coordinates. The resulting set of coordinates has the following shape: **[Ds0, Ds1, Ds2, Ds3, Ds4, Ds5, Dsb0, Dsb1, Dsb2, Dsb3, Dsb4, Dsb6]**. The first 6 distances represent the distance of the sequence n with respect to the artificial sequences, while the other six contain the distance for the shot subject. These are the *normal coordinates*. Additionally, we create a set of *weighted coordinates*. These coordinates are weighted to make segments with rarer shot sizes or subjects stand out. The rarer the token, the higher will be the weight associated with its coordinate.

4.2. Multi-step clustering

Since the different shot sizes and subjects that compose the segments do not have a balanced distribution also the resulting segments will have a majority of segments representing the most used combinations of size and subject. However in this context when the less common shots are used it is important to keep track, hence we have relied on a multiple step clustering approach. The first clustering step is intended to separate the most specific patterns from the most common ones; hence we will use the weighted coordinates. With most specific patterns, we intend those editing sequences that have less common shots, such as extreme close-ups. To this aim, we use a KMeans++ [2] on the weighted coordinates to identify the clusters that contain rare patterns. To automatically identify the number of clusters, we use the Silhouette index score [22], which measures the cohesion and separation of the identified clusters. With this strategy we identify the *Main Set*, which is the cluster that contains the majority of the samples, and the *Rare Sets*, smaller clusters that contain data points that represent segments with the less common editing shots and subjects. The *Main Set* on the other hand contains the data points representing the segments with more common subjects and shots. We can move to the second clustering phase after identifying the *Main Set* and the *Rare Sets* clusters.

performance	trend and pace	no trend	no pace
Silhouette	0.75%	0.99%	0.97%
Macro average	88%	76%	84%
weighted average	93 %	95%	94%

Table 5. LSTM performance in different scenarios.

Next, we take into account only the *Main Set*. In order to identify the different clusters instead of using the weighted set of coordinates we use the regular ones. Like in the previous step, we apply a KMeans++ clustering algorithm to the set of coordinates in order to identify the different classes, while assessing the most ideal number of clusters by consulting the Silhouette index score. At this point we have defined the *basic editing sets*. These clusters contain the points representing the sequences in terms of subjects and sizes. Thus each cluster is characterized by a set of shot sizes and subjects, but there is no information concerning how the shots are concatenated. This information is encoded in the editing pace and segment trend labels.

In the last phase of the multiple steps clustering, we join the editing pace and trend labels to the cluster id of *basic editing sets*. As a consequence every data point is characterized by three labels. On this new data configuration, we apply the KMeans++ clustering algorithm once more while consulting the silhouette index score. The clusters identified at the end of the multiple steps clustering represent the data points in terms of shot size and shot subject used, editing pace and sequence trend. Each of these data points represents a scene segment.

4.3. Validation

At the end of the previous phase, we obtained clusters of data points representing different features of the scene segments. We can use a classifier to evaluate the robustness of LEMMS in correctly identifying well-separated and well-connected groups of sequences. A classifier is built on the original input data enriched with the labels provided by LEMMS, and the ability of the model to correctly classify each 30-second sequence is evaluated. Good prediction metrics are indicators of a good robustness of LEMMS in correctly performing the data labeling task. In our case, since the segments are made of tokens that are made of symbols, we can treat the symbols as text and use an LSTM model[7] for sequence classification. We have used the se-

Phase	Cluster	Number of samples	main sizes	main subjects
1	1	33	ECU	Ot-F
1	2	227	O	Ot
1	3	93	ECU	Ot-Ob
2	4	1498	MS - WS	H
2	5	1659	WS - MS	l
2	6	1166	MS - CU	f-h
2	7	8441	MS	h
2	8	1713	MS - WS	h-ob
2	9	1781	CU	l
2	10	922	MS-WS	a
2	11	6830	MS	l

Table 6. The different clusters identified after the first two clustering phases.

quences as training data and the cluster id of points representing the segments as labels.

5. Experimental Validation

In this section, we present a case with 50 editing classes. In these experiments, we have used as the clustering algorithm the KMeans++. To assess the number of clusters, we have analyzed how the silhouette index changed in relation to the number of clusters.

In the first clustering phase, we stopped at 4 clusters. By increasing the number of clusters to 5 the silhouette index drops from 0.89 to 0.86, with 6, it drops to 0.76. One cluster contains the majority of the segments, the *Main Set*, while the other clusters contain more specific patterns, the *Rare Sets*.

In the second cluster phase, we focus on the *Main Set*, which has 24,010 samples. This time we use the normal coordinates. The clustering algorithm used is once more the KMeans++ algorithm. After trying with a wide range of number of clusters (2-60) we chose the number 8. The silhouette index is lower with respect to the previous clustering phase, but it reaches its best value (0.63) with 8 clusters. It makes sense for the silhouette to be lower since although these clusters identify sequences with a predominance of certain shot sizes and subjects, they also contain the other sizes and subjects in lower quantities. We call *Basic Editing Sets* the set of clusters made of the union of the *Main Sets* with the *Rare Sets*.

We use the *Basic Editing Sets* with the editing pace and trend labels in the third clustering step. These labels are treated as categorical variables and fed to a Kmeans++ model. This time we stop at 50 classes with a silhouette index of 0.91. We could further increase the number of classes; however, if we increase the number of clusters too much, the corresponding labels are too specific and do not contain enough samples. To validate our representation now, we use an LSTM classifier. To validate our model we have used a 10-fold stratified cross-validation. The overall accuracy is 92.8%.

Cluster	final classes (n sample)
1	*44(11), 8(22)*
2	*30(14), 44(81), 8(132)*
3	*14(25), 16(28), 21(23), 30(13), 42(4)*
4	*10(72), 14(510), 16(99), 21(496), 30(141), 42(180)*
5	1(541), 31(301), 34(265), 41(163), *10(189), 32(200)*
6	20(382), 25(438), 47(138), 46(113), *13(19), 22(2), 32(65), 48(4)*
7	15(1381), 19(372), 45(131), 49(131), 6(2356), 7(1421), *2(1421), 22(764), 13(220), 35(201), 48(96)*
8	23(521), 3(714), 38(164), 43(133) *2(1), 22(1), 29(16), 35(79), 4(83), 48(1)*
9	17(614), 24(160), 28(289), 39(236), *29(95), 37(197), 4(190)*
10	11(301), 26(381), 36(25), 9(31), *5(49), 12(19), 37(102), 40(8)*
11	0(1207), 27(457), 33(258), 18(651), *9(234), 36(321),* *40(226), 5(2196), 12(1284)*

Table 7. On the right are the clusters identified from the first two phases; on the right are the final labels identified for those classes.

layer(type)	output shape	number of parameters
Embedding	(None, 30, 30)	300,000
LSTM	(None, 100)	52,400
Dense	(None, 50)	5,050

Table 8. Total trainable parameters 357,450

5.1. Experimental Settings

The clustering algorithm used was the KMeans++, the number of initialization was set to 20. The LSTM model for sequence classification used was an adaptation of the code available here[7]. The number of words is 10,000, while the embedding vector length was set to 30. Table 5.1 shows the architecture of the LSTM model used. The optimizer used was Adam, while the loss function is the categorical cross-entropy. The batch size is 32 and the number of epochs is 20. The code was developed in Python using the SciKit Learn and Keras Libraries.

5.2. Preliminary Analysis

Table 5 shows the composition of the *Basic Editing Sets* and which final classes are formed from them. The classes in italics are the classes that are created from different *Ba-*

	Precision	Recall	f1-score
macro average	85.7%	85,8%	85.5%
weighted average	92.0%	92.8%	92.4%
Accuracy			92.8%

Table 9. The different cluster identified after the first two clustering phases.

sic Editing Sets. Of the 50 final classes, 28 are created from just one class, while the remaining 22 are created from multiple *Basic Editing Sets*. Table 5.2 shows the results of the LSTM classifier in terms of Precision(P), Recall(R), and f1-score(f1). Mavg is the macro-average, i.e., the f1-score is calculated for each label and then averaged without taking into account the proportion of different classes, while Wavg, the weighted average, does. All scores are averaged over the 10 scores from the 10 different folds. Overall, the performance of the classifier is good enough, but there are some areas where the performance drops. For classes 8 and 44, it is not able to detect a single sample. First of all, both classes do not have many samples and are mixed classes. So they also contain samples from different classes. The problem with mixed classes is that they group samples from different basic processing patterns based on similarities in trend and editing pace. One way to solve this problem is to increase the number of clusters in the third clustering stage. However, this degrades the performance of the classifier, which in turn complicates the analysis of the segments. For example, if one increases the number of classes to 70, the overall accuracy remains in the same range, while the macro-average of the f1 score is slightly worse at 84%, which means that the classifier recognizes a lower percentage of classes. If the number of classes is increased to 90, the f1 score drops to 75%. On the other hand if we reduce the number of classes the performance is better but the classes are less defined. In table 4.3 we have reported the performance of an LSTM trained with 30 classes. In this table we have also included the results of the LSTM classifier trained without the editing pace and without the editing trend scenarios. While accuracy and weighted f1-score marginally benefit from the elimination of the trend and editing pace attributes the macro average significantly drops, especially when the trend attribute is removed.

The identified classes characterize more or less specific patterns. If we look at two segments coming from two classes defined by different basic editing patterns, editing paces, and trends, the differences are evident. For instance, 6 is a segment taken from ”StarTrek:Insurrection” by Jonathan Frakes. There is a fast editing pace and a mix of mainly medium and wide shots in which the characters run and shoot across the location. A completely different structure is shown in segment 2 7, taken from ”Children of a Lesser God” by Randa Haines. This segment is taken from class 46, a class that derives from a basic editing pattern that is characterized by close-ups and medium shots of

	precision	recall	f1-score	support
0	0.87	0.93	0.90	121
1	0.90	0.87	0.89	54
2	0.99	0.99	0.99	142
3	0.93	0.89	0.91	71
4	0.82	0.85	0.84	27
5	0.98	1.00	0.99	224
6	0.94	0.97	0.95	236
7	0.93	0.97	0.95	137
8	0.00	0.00	0.00	15
9	0.92	0.85	0.88	27
10	0.76	0.85	0.80	26
11	0.89	0.81	0.85	31
12	0.97	0.98	0.98	130
13	0.86	1.00	0.92	24
14	0.86	0.80	0.83	54
15	0.97	1.00	0.99	138
16	0.82	0.69	0.75	13
17	0.90	0.93	0.92	61
18	0.91	0.95	0.93	65
19	1.00	0.92	0.96	37
20	0.88	0.97	0.93	39
21	0.88	0.87	0.87	52
22	1.00	0.99	0.99	77
23	0.90	0.87	0.88	52
24	1.00	1.00	1.00	16
25	0.90	0.80	0.84	44
26	0.88	0.92	0.90	38
27	0.95	0.85	0.90	46
28	0.96	0.93	0.95	29
29	0.77	0.91	0.83	11
30	0.81	0.76	0.79	17
31	0.87	0.67	0.75	30
32	0.81	0.81	0.81	27
33	1.00	0.85	0.92	26
34	0.86	0.67	0.75	27
35	0.86	0.86	0.86	28
36	0.88	0.80	0.84	35
37	0.84	0.90	0.87	30
38	0.92	0.69	0.79	16
39	0.96	1.00	0.98	24
40	0.67	0.96	0.79	23
41	1.00	1.00	1.00	16
42	0.62	0.89	0.73	18
43	0.93	1.00	0.96	13
44	0.00	0.00	0.00	9
45	1.00	0.92	0.96	13
46	0.88	0.64	0.74	11
47	0.61	1.00	0.76	14
48	0.89	0.80	0.84	10
49	0.92	0.85	0.88	13

Figure 5. Precision Recall and f1 score

Figure 6. Segment from StarTrek. For a better visualization https://www.youtube.com/watch?v=-1gCG8m1SHU, from the 0:30 to 1:00

Figure 7. Segment from Children of a lesser god. For a better visualization https://www.youtube.com/watch?v=1pJywLQLzcA, from 1:30 to 2:00

Figure 8. Segment from Thoroughbreds https://www.youtube.com/watch?v=1bBOUr7rAHw, from the 0:30 to 1:00

humans, usually representing dialogues. In this scene, we have a much slower pace with respect to segment 1 and the focus is more on the characters rather than the global context. If we take two segments from different classes that come from the same basic editing pattern cluster we notice fewer differences. The segments shown in 1 are extracted from class 43, which is entirely created from basic editing set 8. The complete segments are reported in Figure 8 and in Figure 9. The scene segment represented in both cases contains mainly medium shots of people interacting with objects. The fact that the two segments end in the same way is partly coincidental. By this, we mean the segments represented by a class can have varying structures as long as they stay within the limits indirectly defined by the editing pace and trend labels.

5.3. Limitations

In some cases, the sequences identified by the same label, while they look alike from a segment representation visually, are different. We can show the reason behind this issue with a unique segment 10. This segment is represented with all medium shots. However, the shot sizes vary. This is because the medium shots shown are at the extremities of the medium shot class. Hence one looks more like a close-up while the other one looks more like a long shot. To overcome these issues, a more fine-grained shot size classification system could be implemented while also including more features from the AVE dataset, such as *number of people* and *camera movement*. These countermeasures would also enrich the data representation allowing us to characterize more complex editing patterns. Another issue is that even if LEMMS can identify more defined labels, there are not enough data examples for the LSTM classifier to learn and thus properly validate them. Thus, if we want to identify more labels and their corresponding segments, we need to increase the amount of data to be analyzed.

Figure 9. Segment from the birdcage. For a better visualization https://www.youtube.com/watch?v=1pJywLQLzcA, from 1:00 to 1:30

Figure 10. Segment from Pacific Rim. For a better visualization https://www.youtube.com/watch?v=-7Sow81yi24, from 1:30 to 2:00

Finally, although the trend attribute has proven useful in characterizing editing sequences a smarter strategy should be implemented to better characterize the mixed trends.

6. Conclusion and future work

Although the qualitative and quantitative analysis revealed some limitations of the current approach, overall the LEMMS methodology achieves good results. We identified and distinguished 50 classes of editing patterns, characterized by shot size, shot subject, and editing pace representing real movie scene segments. To validate the identified classes, we successfully trained an LSTM classifier with an overall accuracy of 92.8%. Aside from overcoming current limitations, interesting future research directions include the integration of the LEMMS methodology with video editing tools and additional features, such as the textual scene labels provided in the Condensed Movies dataset[3].

References

[1] Dawit Mureja Argaw, Fabian Caba Heilbron, Joon-Young Lee, Markus Woodson, and In-So Kweon. The anatomy of video editing: A dataset and benchmark suite for ai-assisted video editing. In *European Conference on Computer Vision*, 2022.

[2] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[3] Max Bain, Arsha Nagrani, Andrew Brown, and Andrew Zisserman. Condensed movies: Story based retrieval with contextual embeddings, 2020.

[4] Hui-Yong Bak and Seung-Bo Park. Comparative study of movie shot classification based on semantic segmentation. *Applied Sciences*, 10:3390, 05 2020.

[5] Sergio Benini, Mattia Savardi, Katalin Balint, Andras Balint Kovacs, and Alberto Signoroni. On the influence of shot scale on film mood and narrative engagement in film viewers. *IEEE Transactions on Affective Computing*, 13(2):592–603, 2022.

[6] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics*, 31:1–8, 07 2012.

[7] Jason Brownlee. Sequence classification with lstm recurrent neural networks in python with keras, 2016.

[8] Shixing Chen, Chundi Liu, Xiang Hao, Xiaohan Nie, Maxim Arap, and Raffay Hamid. Movies2scenes: Using movie metadata to learn scene representation. 2022.

[9] Robin Courant, Christophe Lino, Marc Christie, and Vicky Kalogeiton. High-Level Features for Movie Style Understanding. In *ICCV 2021 - Workshop on AI for Creative Video Editing and Understanding*, pages 1–5, online, France, Oct. 2021.

[10] Rishin Haldar and Debajyoti Mukhopadhyay. Levenshtein distance technique in dictionary lookup methods: An improved approach. *Computing Research Repository - CORR*, 01 2011.

[11] M. A. Hasan, M. Xu, X. He, and C. Xu. Camhid: Camera motion histogram descriptor and its application to cinematographic shot classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(10):1682–1695, 2014.

[12] Qingqiu Huang, Yu Xiong, Anyi Rao, Jiaze Wang, and Dahua Lin. Movienet: A holistic dataset for movie understanding. *ArXiv*, abs/2007.10937, 2020.

[13] Yuya Matsuo, Miki Amano, and Kuniaki Uehara. Mining video editing rules in video streams. pages 255–258, 01 2002.

[14] Eyal Molad, Eliahu Horwitz, Dani Valevski, Alex Rav Acha, Y. Matias, Yael Pritch, Yaniv Leviathan, and Yedid Hoshen. Dreamix: Video diffusion models are general video editors. *ArXiv*, abs/2302.01329, 2023.

[15] Walter Murch. *In the Blink of an Eye*. Silman-James Press, 2001.

[16] Alejandro Pardo, Fabian Caba, Juan León Alcázar, Ali K Thabet, and Bernard Ghanem. Learning to cut by watching movies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6858–6868, 2021.

[17] Alejandro Pardo, Fabian Caba Heilbron, Juan León Alcázar, Ali Thabet, and Bernard Ghanem. Moviecuts: A new dataset and benchmark for cut type recognition. In *European Conference on Computer Vision*, pages 668–685. Springer, 2022.

[18] Jian Ren, Xiaohui Shen, Zhe Lin, and Radomír Měch. Best frame selection in a short video. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3201–3210, 2020.

[19] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021.

[20] Mattia Savardi, András Bálint Kovács, Alberto Signoroni, and Sergio Benini. Cinescale: A dataset of cinematic shot scale in movies. *Data in Brief*, 36, 2021.

[21] Mattia Savardi, Alberto Signoroni, Pierangelo Migliorati, and Sergio Benini. Shot scale analysis in movies by convolutional neural networks. pages 2620–2624, 10 2018.

[22] Ketan Rajshekhar Shahapure and Charles Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 747–748, 2020.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[24] Uriel Singer, Adam Polyak, Thomas Hayes, Xiaoyue Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. *ArXiv*, abs/2209.14792, 2022.

[25] Bartolomeo Vacchetti and Tania Cerquitelli. Cinematographic shot classification with deep ensemble learning. *Electronics*, 11(10):1570, 2022.

[26] Bartolomeo Vacchetti and Tania Cerquitelli. Movie lens: Discovering and characterizing editing patterns in the analysis of short movie sequences. In *European Conference on Computer Vision*, pages 660–675. Springer, 2022.

[27] Chao-Yuan Wu and Philipp Krähenbühl. Towards long-form video understanding. *CoRR*, abs/2106.11310, 2021.

[28] Ling Yang, Zhilong Zhang, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Ming-Hsuan Yang, and Bin Cui. Diffusion models: A comprehensive survey of methods and applications. *ArXiv*, abs/2209.00796, 2022.