Finding the Right Moment: Human-Assisted Trailer Creation via Task Composition

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata

Abstract—Movie trailers perform multiple functions: they introduce viewers to the story, convey the mood and artistic style of the film, and encourage audiences to see the movie. These diverse functions make trailer creation a challenging endeavor. In this work, we focus on finding trailer moments in a movie, i.e., shots that could be potentially included in a trailer. We decompose this task into two subtasks: narrative structure identification and sentiment prediction. We model movies as graphs, where nodes are shots and edges denote semantic relations between them. We learn these relations using joint contrastive training which distills rich textual information (e.g., characters, actions, situations) from screenplays. An unsupervised algorithm then traverses the graph and selects trailer moments from the movie that human judges prefer to ones selected by competitive supervised approaches. A main advantage of our algorithm is that it uses interpretable criteria, which allows us to deploy it in an interactive tool for trailer creation with a human in the loop. Our tool allows users to select trailer shots in under 30 minutes that are superior to fully automatic methods and comparable to (exclusive) manual selection by experts.

Index Terms—Computer vision, machine learning, natural language processing, neural networks, user interaction, video.

I. INTRODUCTION

TRAILERS are short videos used for promoting movies and are often critical to commercial success. While their core function is to market the film to a range of audiences, trailers are also a form of persuasive art and promotional narrative, designed to make viewers want to see the movie. Even though the making of trailers is considered an artistic endeavor, the film industry has developed strategies guiding trailer construction. According to one school of thought, trailers must exhibit a narrative structure, consisting of three acts.1 The first act establishes the characters and setup of the story, the second act introduces the main conflict, and the third act raises the stakes and provides teasers from the ending. Another school of thought is more concerned with the mood of the trailer as defined by the ups and downs of the story.2 According to this approach, trailers should have medium intensity at first in order to captivate viewers, followed by low intensity for delivering key information about the story, and then progressively increasing intensity until reaching a climax at the end of the trailer.

In this work we aim to automatically identify moments in a movie that are suitable for including in a trailer. For this, we need to perform low-level tasks such as person identification, action recognition, and sentiment prediction, but also more high-level ones such as understanding connections between events and their causality, as well as drawing inferences about the characters and their actions. Given the complexity of the task, directly learning all this knowledge from movie–trailer pairs would require many thousands of examples, whose processing and annotation would be a challenge. It is thus not surprising that previous approaches ([1], [2], [3]) have solely focused on audiovisual features and depend on ill-defined criteria, such as identifying the “trailerness” of a movie shot.

Following previous work ([1], [2], [3]), we formulate moment identification as the task of selecting shots for presentation in a trailer, under the assumption that a segmentation of the movie into moments (i.e., shots that represent events) is available. Inspired by the creative process of human trailer editors, we adopt a bottom-up approach, decomposing our task into two simpler subtasks. The first one is narrative structure analysis, i.e., retrieving the most important events of the movie. A commonly adopted theory in screenwriting ([4], [5], [6]) suggests that there are five types of key events in a movie, known as turning points (TPs; see their definitions in Fig. 1). The second subtask is sentiment prediction, which we view as an approximation of the intensity flow between shots and the emotions they evoke.

We identify trailer moments following an unsupervised graph-based approach. We model movies as graphs whose nodes are shots and whose edges denote important semantic connections between shots (see Fig. 2). In addition, nodes bear labels denoting whether they are key events (i.e., TPs) and scores signaling sentiment intensity (positive or negative). Our algorithm traverses this movie graph to retrieve sequences of movie shots that can be used in a trailer. In contrast to prior work, we exploit

Manuscript received 12 July 2022; revised 6 June 2023; accepted 24 September 2023. This work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), under grant FA8650-17-C-9118. The work of Pinelopi Papalampidi and Mirella Lapata was supported by the UK Engineering and Physical Sciences Research Council under grant EP/W002876/1. Recommended for acceptance by M. Moens. (Corresponding author: Pinelopi Papalampidi.)

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3323030, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3323030

https://www.studioli.com/blog/how-to-make-a-movie-trailer

1See https://www.ieee.org/publications/rights/index.html for more information.

0162-8828 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We propose a contrastive learning regime, where we take advantage of screenplays as privileged information \cite{7}, i.e., information available at training time only. Screenplays reveal how the movie is segmented into scenes, who and where the characters are, when and who they are speaking to, what they are doing (in a screenplay, “scene headings” explain where the action takes place while “action lines” describe what the camera sees). Specifically, we build two networks, a multimodal network based on movie videos and an auxiliary, textual network based on screenplays, and train them jointly using contrastive losses in order to distill information from screenplays to videos. The auxiliary network can be additionally pretrained on large collections of screenplays via self-supervised learning, without processing the corresponding movies, overcoming data scarcity issues. Experimental results show that contrastive training is beneficial for knowledge distillation, leading to trailers which are judged favorably by annotators in terms of their content and attractiveness.

Finally, we explore how our algorithm can be used in an interactive setting where users select shots to be included in trailers from a set of automatically identified candidates. We provide an interactive tool for trailer creation\footnote{https://movie-trailers-beta.herokuapp.com} and assess its functionality against fully manual and fully automatic methods. Our study reveals that interactive selection improves the quality of the trailers and is comparable to selecting shots via manual inspection, while reducing the time needed from 2–3 days to under 30 minutes. Our contributions in this work can be summarized as follows:

- We propose an unsupervised approach for identifying shot-level trailer moments in movies which operates over sparse graphs and decomposes the task into narrative structure analysis and sentiment prediction.
- We propose a contrastive regime for distilling knowledge from screenplays to movies in lieu of collecting and processing full-length movie videos.
- We develop an interactive tool for building trailers semi-automatically and demonstrate they are better than fully automatic ones and of similar quality to trailers created exclusively by human experts.

II. RELATED WORK

Movie understanding approaches have mainly focused on isolated video clips, and tasks such as the alignment between movie scenes and book chapters \cite{8}, question answering (\cite{9,10}), video captioning for movie shots \cite{11} or clips from TV episodes \cite{12}, and text-to-video retrieval (\cite{12,13,14}). Although this work exploits multimodal information (i.e., mainly video and language), it does not target full-length movies. Bain et al. \cite{13} adopt a more holistic approach and learn from movie clips, corresponding textual descriptions, and other metadata, such as genre information and bounding boxes for characters. Follow-on work \cite{15} creates a large-scale dataset based on full-length movies, their trailers, posters, textual synopses, scripts, action recognition tags, and character bounding boxes. Unfortunately, this dataset is not publicly available, limiting further research. Some recent work (\cite{16,17,18,19}) attempts to identify narrative structure and summarize entire TV episodes and movies, but focuses exclusively on the textual modality (i.e., screenplays). Our approach exploits information from multiple dimensions, i.e., video, audio, text, whilst processing full-length movies.

Trailer moment identification exploits superficial audiovisual features, such as background music or visual changes between shots (\cite{1,2}). Other work creates “attractive” trailers with a graph-based model for shot selection \cite{20} or uses a human in the loop in conjunction with a model trained on horror movies via audiovisual sentiment analysis \cite{21}. The Trailer Moment
Detection Dataset [3] consists of full-length movies paired with official trailers and annotations for key moments, but is not publicly available and does not include screenplays. Wang et al. [3] propose a state-of-the-art trailer generation model that again only focuses on the visual modality. To the best of our knowledge, we are the first to exploit all input modalities for identifying trailer moments in full-length movies.

Video highlight detection ([22], [23], [24]), i.e., identifying frames or video segments that can function as highlights is also relevant to our work. Most previous approaches focus on short videos with simple semantics (e.g., actions in YouTube videos), do not exploit the textual modality, and cannot easily transfer to the task of trailer moment identification. An adjacent line of work focuses on video editing ([25], [26], [27], [28], [29]), in order to create cohesive and visually appealing videos. Although transitions between shots are stylistically important, our model does not take this into account when selecting shots for inclusion in the trailer. However, it is possible for humans to post-process the trailer through our interactive tool.

Knowledge distillation ([30], [31]) was originally proposed for distilling information from a larger teacher model to a smaller student one. Generalized distillation [7] provides a framework for using privileged information available at training time only. Most related to our work is the use of different modalities/views of the same content ([32], [33]), such as transcribed narrations to learn visual representations in videos. We leverage screenplays as a source of privileged information and distill knowledge about events, characters, and scenes in a film, which we exploit for identifying trailer-worthy shots in video.

III. PROBLEM FORMULATION

We are interested in automatically identifying important movie content that should be included in a trailer. We define trailer moment identification as the task of selecting \(L\) shots from a full-length movie with \(M\) shots \((L \ll M)\). A shot is the continuous sequence of frames between two edits or cuts in a movie. The selected shots do not constitute the final trailer, several post-processing steps might be required: trimming them, changing their order, adding music, voice-over, and other information (e.g., release date), however we leave this to future work.

Movies are typically complex stories, they contain distinct subplots or events that unfold non-linearly, while redundant events, called “fillers” enrich the main story. We therefore cannot assume that consecutive shots are semantically related. To better explore relations between events, we represent movies as graphs [18]. Let \(G = (V, E)\) denote a graph where vertices \(V\) are shots and edges \(E\) represent their semantic similarity. We further consider the original temporal order of shots in \(G\) by only allowing directed edges from previous to future shots. \(G\) is described by an upper-triangular transition matrix \(T\), which records the probability of transitioning from shot \(i\) to every future shot \(j\).

We propose an algorithm for traversing \(G\) and selecting sequences of shots to be used in a trailer. The algorithm operates over graph structures exemplified in Fig. 2: \(G\) represents relations between movie shots describing key events (TPs; thick circles in Fig. 2); in addition each node in the graph has a sentiment score which can be positive or negative (different shades of green/red depending on the sentiment intensity in Fig. 2). In the following, we first describe our algorithm (Section III-A) assuming graph \(G\) is given and then discuss how \(G\) is learned and key events are detected [16] (Section III-B). Finally, we explain how shot-based sentiment scores are predicted (Section III-F).

A. GRAPHTRAILER: Movie Graph Traversal

Algorithm 1 retrieves trailer sequences by performing random walks in graph \(G\). We start by selecting a node identified as a first TP (i.e., Opportunity, see Fig. 1), i.e., the first major event that changes the course of the plot and serves as a catalyst for character development; we assume that first TPs should be included in a trailer as they reflect what the movie is about. Note that TPs extend over \(C\) shots and as a result, our algorithm can produce \(C\) different paths as proposal trailers (see line 4 in Algorithm 1). Given node \(i\), we decide where to go next by considering its \(K\) immediate neighbors \(N_i\). We select node \(j\) from \(N_i\) as the \(k\)th shot \(n_k\) to add to the path based on the following criteria: (1) normalized probability of transition \(e_{ij}\) from \(i\) to \(j\) based on matrix \(T\) (i.e., semantic similarity between shots), (2) normalized distance \(e_{ij} = |j - i|/M\) between \(i\) and \(j\) (i.e., temporal proximity), (3) normalized shortest path\(^4\) from node \(j\) to next major event \(d_{j,TP}\) (i.e., relevance to the storyline), and (4) difference between sentiment flow \(p_{ij}\) (from shot \(i\) to \(j\) and expected flow \(f_k\) at the \(k\)th step in the path (see Fig. 2 and Appendix, Section 1.3, available online):

\[
n_k = \arg\max_{j \in N_i} s_{ij}
\]

\[
s_{ij} = \lambda_1 e_{ij} - \lambda_2 t_{ij} - \lambda_3 d_{j,TP} - \lambda_4 |p_{ij} - f_k|
\]

\(^4\)Computed using Dijkstra’s [34] algorithm.
where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are hyperparameters used to combine the different criteria (tuned on the development set based on gold-standard trailer labels). Note that these criteria are interpretable and can be easily altered by a user (e.g., by adding new ones or defining a different flow $f$). Our approach can also be used for interactive trailer creation, where a user iteratively decides which shot to include in the trailer from a limited set of options (see Section VI for details).

We select $L$ shots in total (depending on a target trailer length) and retrieve a proposal trailer sequence as depicted in Fig. 2 (bold line). At each step, we keep track of the sentiment flow created and the TPs identified thus far (lines 10 and 13–14 in Algorithm 1, respectively). A TP event is selected for presentation in the trailer if a shot or its immediate neighbors have been added to the path.

B. Graph Construction and TP Identification

In the previous section, we discussed how we can identify important shot-level trailer moments given movie graph $G$ and a set of shots that act as key events (i.e., TPs). We now discuss how we learn $G$ and identify TPs in movies in tandem. We hypothesize that the training signal provided by TP labels (i.e., narrative structure) also encourages exploring more fine-grained semantic connections between shot-level events via the graph that is learned [18]. A neural network model first creates $G$ that represents relations between shots in the latent space and then computes the probability $p(y_{it}|h_i, F, \theta_1)$, where $y_{it}$ is a binary label denoting whether shot $h_i$ represents TP $t \in [1, T]$ and $\theta_1$ are network parameters. This network is depicted on the right side of Fig. 3 and is trained end-to-end on TP identification.

**Movie Input:** Let $F$ denote a full-length movie consisting of $M$ shots $F = \{h_1, h_2, \ldots, h_M\}$. For each shot $i$, we consider visual (i.e., sequence of frames), audio (i.e., audio segments), and textual (i.e., subtitles) information and compute a combination vector $h_i$ of all modalities (see step (1), right part of Fig. 3). First, we compute the textual representation $t_i$ for the $i$th shot, via the bi-directional attention flow ([35], [36]) between subtitles $t_i$ and audio $a_i$ and video frames $v_i$:

$$S_{t_i, a_i} = a^T_i t_i, \quad S_{t_i, v_i} = v^T_i t_i \quad (3)$$

$$a_{i, att} = \text{softmax}(S_{t_i, a_i} a_i), \quad v_{i, att} = \text{softmax}(S_{t_i, v_i}) v_i \quad (4)$$

$$t'_i = a_{i, att} + v_{i, att} + t_i \quad (5)$$

The final audio $a'_i$ and visual $v'_i$ representations are obtained analogously. Next, vectors $t'_i, a'_i,$ and $v'_i$ are projected to a lower dimension via a fully-connected linear layer and L2 normalization. Finally, we compute the multimodal representation $h_i$ for shot $i$ via a non-linear projection: $h_i = f([t'_i; v'_i; a'_i])$, where $f(\cdot)$ is a fully-connected layer followed by the ReLU non-linearity.

**Graph Structure:** We construct a fully-connected graph described by matrix $E$. Each cell in this matrix denotes the similarity $e_{ij}$ between multimodal shot vectors $h_i$ and $h_j$:

$$e_{ij} = \tanh(W_i h_i + b_i) \tanh(W_j h_j + b_j) + b_{ij} \quad (6)$$

![Fig. 3. Two networks process different views of the movie with different degrees of granularity. Our main network (right side) takes as input multimodal fine-grained shot representations based on the movie’s video stream. The auxiliary text-based network (left side) processes textual scene representations which are coarse-grained and based on the movie’s screenplay. The networks are trained jointly on TP identification with losses enforcing prediction and representation consistency between them.](image-url)
We then normalize similarities $e_{ij}$ using the softmax function (row-wise normalization in matrix $E$). We thus obtain a complete directed graph, where edge $e_{ij}$ records the probability that $h_i$ is connected to $h_j$. In order to avoid dense connections which lead to worse contextualization and computational overhead, we sparsify the graph by selecting a small but variable-length neighborhood per shot [18]. During sparsification, we also constrain the adjacency matrix of the graph to be upper triangular (i.e., allowing only future connections between shots). We select the top-$k$ neighborhood $P_i$ per shot $h_i$ as $P_i = \arg\max_{j \in [1,M], |i| \leq k} e_{ij}$. Instead of deciding on a fixed number of $k$ neighbors for all shots and movies, we determine a predefined set of options for $k$ (e.g., integers contained in a set $O$) and learn to select $k$ via a parametrized function: $z_i = \text{softmax}(W_ne_i + b_n)$, where $z_i$ is a probability distribution over the neighborhood size options for shot $h_i$, $W_n \in \mathbb{R}^{MxO}$, and $e_i$ is a vector of similarities between shot $i$ and all other shots. Hence, the final neighborhood size for shot $h_i$ is: $k_i = \arg\max_{k \in O} z_{it}$ (see step (2) in the right part of Fig. 3). We address discontinuities in our model (i.e., top-$k$ sampling, neighborhood size selection) by utilizing the Straight-Through Estimator [37]. During the backward pass we compute the gradients with the Gumbel-softmax reparameterization trick ([38], [39]).

**TP Identification:** We contextualize *all shots* with respect to the *entire movie* via a transformer encoder [40] to compute global shot representations (since transformers can be viewed as fully-connected graphs); we additionally encode the *graph neighborhood* of each shot via a one-layer Graph Convolution Network (GCN; [41], [42], [43]) for local representations that only depend on a small set of shots (since the learnt graph is sparse). Finally, we combine global and local shot representations (see step (3) in the right part of Fig. 3) to compute probability $p(y_{it}|h_i,F,\theta_1)$. The network is trained by minimizing the binary cross-entropy loss per TP (see Section IV for details on the dataset we used). After training, we use graph $G$ and predicted TP shots as input to Algorithm 1.

### C. Auxiliary Text-Based Network

Movie screenplays provide a wealth information in addition to subtitles, e.g., about characters and their role in a scene, their actions and emotions. Such information, typically conveyed by lines describing what the camera sees, is difficult to accurately infer from video (e.g., via person identification, action recognition, event localization), while it can be easily mined from the screenplay. Moreover, corpora of screenplays are relatively easy to collect leading to orders of magnitude larger datasets compared to collections of full-length videos.

We exploit screenplays by training our main network (described in the previous section) together with an auxiliary text-only network. The latter has an architecture similar to the main network, modulo two key differences: (1) it only considers textual information and (2) it processes movies at scene-level; although a scene is the smallest unit of a story in a screenplay, it is far more coarse-grained than shots (a scene may last several minutes). The auxiliary network thus creates a scene-level graph and estimates scene-level probabilities $q(y_{it}|s_i,D,\theta_2)$ which quantify the extent to which scene $s_i$ corresponds to the $t$th TP. We represent scenes with a small transformer encoder which operates over sequences of sentence vectors. As with our main network, we compute contextualized scene representations; a transformer encoder over the entire screenplay yields global representations, while local ones are obtained with a one-layer GCN over a sparse scene-level graph.

### D. Knowledge Distillation

We now describe our joint training regime for the two networks which encapsulate different views of the movie in terms of data streams (multimodal vs. text-only) and *their segmentation into semantic units* (shots vs. scenes), while assuming a (one-to-many) mapping from screenplay scenes to movie shots (see Section IV for details). Traditionally, in knowledge distillation ([30], [31]) the teacher model (here the auxiliary network) is trained first, and the knowledge is then asynchronously distilled in a later step to the student network (here the main network). We propose to *jointly* train the two networks, since they have complementary information and can benefit from each other.

**Prediction Consistency Loss:** We aim to enforce some degree of agreement between the TP predictions of the two networks. For this reason, we train them jointly and introduce additional constraints in the loss objective. Similarly to knowledge distillation settings ([30], [31]), we utilize the KL divergence loss between the auxiliary-based posterior distribution $q(y_{it}|D)$ and the distribution $p(y_{it}|F)$ of our main network (upper part in Fig. 3).

While in standard knowledge distillation settings both networks produce probabilities over the same units, in our case, our main network predicts TPs for shots and the auxiliary one for scenes. We obtain scene-level probabilities $p(y_{it}|F)$ for the main network by aggregating shot-level ones via max pooling and re-normalization. We then calculate the prediction consistency loss between the two networks as:

$$ P = \frac{1}{T} \sum_{t=1}^{T} D_{KL} \left( p(y_{it}|F) \parallel q(y_{it}|D) \right) \quad (7) $$

**Representation Consistency Loss:** We further use a second regularization loss between the two networks in order to enforce consistency between the two graph-based representations (i.e., over video shots and screenplay scenes). The purpose of this loss is twofold: to improve TP predictions for the two networks, as shown in previous work on contrastive representation learning ([44], [45], [46]), and also to help learn more accurate connections between shots (recall that the shot-based graph serves as input to our graph traversal algorithm; Section III-A). In comparison with screenplay scenes, which describe self-contained events, video shots are only a few seconds long and rely on surrounding context for their meaning. We hypothesize that by enforcing the graph neighborhood for a shot to preserve semantics similar to the corresponding screenplay scene, we will encourage the selection of appropriate neighbors in the shot-level graph (middle part of Fig. 3).

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We again first address the problem of varying granularity in the representations of the two networks. We compute an aggregated scene-level representation \( \overline{h}_j \) based on shots \( h_1, \ldots, h_{i+k} \) via mean pooling and calculate the noise contrastive estimation (NCE; [47], [48]) loss for the \( j \)th scene:

\[
\mathcal{R} = -\frac{1}{N} \sum_{j=1}^{N} \log \frac{e^{\langle \overline{h}_j, s_j \rangle / \tau}}{e^{\langle \overline{h}_j, s_j \rangle / \tau} + \sum_{k \neq j}^{N} e^{\langle \overline{h}_k, s_k \rangle / \tau}}
\]  

(8)

where \( N \) is the number of scenes, \( s_j \) is the scene representation calculated by the GCN in the auxiliary network, \( \overline{h}_j \) is the (average) scene representation calculated by the GCN in the main network, \( s(\cdot) \) is a similarity function (here the scaled dot product), and \( \tau \) is a temperature hyperparameter.

**Joint Training:** Our final joint training objective takes into account the individual losses \( S \) and \( V \) of the auxiliary and main networks, respectively (see Appendix, Section 1.1 for details, available online), and the two consistency losses \( \mathcal{P} \) (for prediction) and \( \mathcal{R} \) (for representation):

\[
\mathcal{L}_{TP} = S + V + a\mathcal{P} + b\mathcal{R}
\]

(9)

where \( a, b \) are hyperparameters modulating the importance of prediction vs. representation consistency. Fig. 3 provides a high-level illustration of our training regime.

**E. Self-Supervised Pretraining**

We further pretrain the auxiliary network on more textual data, which is easier to acquire than videos (e.g., fewer copyright issues and less computational overhead), in order to learn better scene representations. We hypothesize that this knowledge can then be transferred to our main network via the consistency losses in the joint training regime (7) and (8).

Pretraining takes place on Scriptbase [49], a dataset which consists of \( \sim 1,100 \) full-length screenplays (approx. 140K scenes). We adopt a self-supervised task, namely Contrastive Predictive Coding (CPC; [44]), to our setting: given a (contextualized) scene representation, we learn to predict a future representation in the screenplay. We consider a context window of several future scenes, rather than just one. This is an attempt to account for non-linearities in the screenplay, which can occur because of unrelated intervening events and subplots. Given the representation of an anchor scene \( g_i \), a positive future representation \( c_i^+ \) and a set of negative examples \( \{c_{i1}, \ldots, c_{i(N-1)}\} \), we compute the InfoNCE [44] loss as:

\[
\mathcal{L}_{self} = -\frac{1}{N} \sum_{j=1}^{N} \log \frac{e^{\langle g_i, c_i^+ \rangle / \tau}}{e^{\langle g_i, c_i^+ \rangle / \tau} + \sum_{k=1}^{N-1} e^{\langle g_i, c_{ik} \rangle / \tau}}
\]

(10)

We obtain scene representations \( g_i \) based on \( s_i \) provided by the one-layer GCN. Starting from the current scene, we perform a random walk of \( k \) steps and compute \( g_i \) from the retrieved path \( p_i \) in the graph via mean pooling (more details can be found in the Appendix, Section 1.2, available online).

**F. Sentiment Prediction**

Finally, our model takes into account how sentiment flows from one shot to the next. We predict sentiment scores per shot with the same joint architecture and training regime we use for TP identification (Section III-B–III-D). The only difference for sentiment prediction is in the downstream objective (losses \( S \) and \( V \) in (9)). The main network is trained on shots with sentiment labels (i.e., positive, negative, neutral; cross-entropy loss), while the auxiliary network is trained on scenes with sentiment labels (Section IV explains how the labels are obtained). After training, we predict a probability distribution over sentiment labels per shot to capture sentiment flow and discriminate between high- and low-intensity shots (see Appendix for details, available online).

**IV. EXPERIMENTAL SETUP**

**Datasets:** Our model was trained on TRIPOD ⊕, an expanded version of the TRIPOD dataset ([16], [18]) which contains 122 screenplays with silver-standard TP annotations (scene-level)5 and the corresponding videos.6 For each movie, we further collected as many trailers as possible from YouTube, including official and (serious) fan-based ones, or modern trailers for older movies. To evaluate the trailers produced by our algorithm, we also collected a new held-out set of 41 movies. These movies were selected from the Moviescope7 dataset [50] which contains official movie trailers. The held-out set does not contain any additional information, such as screenplays or TP annotations. Detailed statistics of TRIPOD ⊕ are presented in Table I.

**Movie Processing:** The modeling approach put forward in previous sections assumes a (one-to-many) mapping from screenplay scenes to movie shots. We obtain this mapping by automatically aligning dialogue in screenplays to subtitles that contain timestamps using Dynamic Time Warping (DTW;
We first segment the video into scenes based on this mapping, and then segment each scene into shots using PySceneDetect. Shots with less than 100 frames in total are too short for processing and displaying as part of the trailer and are therefore discarded.

Moreover, for each shot we extract visual and audio features. We consider three different types of visual features: (1) we sample one key frame per shot and extract features using ResNeXt-101 [52] pre-trained for object recognition on ImageNet [53]; (2) we sample frames with a frequency of 1 out of every 10 frames (we increase this time interval for shots with larger duration since we face memory issues) and extract motion features using the two-stream I3D network pre-trained on Kinetics [54]; and (3) we use Faster-RCNN [55] implemented in Detectron2 [56] to detect person instances in every key frame and keep the top four bounding boxes per shot which have the highest confidence alongside with the respective regional representations. We project all individual representations to the same lower dimension and perform L2-normalization. A shot is represented as their sum. For the audio modality, we use YAMNet pre-trained on the AudioSet-YouTube corpus [57] to classify audio segments into 521 audio classes (e.g., tools, music, explosion); for each audio segment contained in a scene, we extract features from the penultimate layer. Finally, we extract textual features [18] from subtitles and screenplay scenes using the Universal Sentence Encoder (USE; [58]).

**Trailer Labels:** For evaluation purposes, we need to know which shots in the movie are trailer-worthy. We obtain these binary labels as follows. We segment the corresponding trailer into shots and compute for each shot its visual similarity to all shots in the movie. Movie shots with highest similarity values receive positive labels (i.e., they should be in the trailer). However, since trailers also contain shots that are not in the movie (e.g., black screens with text, or material that did not make it into the final cut), we do not map trailer shots to movie shots unless their similarity is above a certain threshold (set to 0.85 in our experiments).

**Sentiment Labels:** Since TRIPOD does not contain sentiment annotations, we instead obtain silver-standard labels via COSMIC [59], a commonsense-guided framework with state-of-the-art performance for sentiment and emotion classification in natural language conversations. Specifically, we train COSMIC on MELD [60], which contains dialogues from episodes of the TV series Friends and is more suited to our domain than other sentiment classification datasets (e.g., [61], [62]). After training, we use COSMIC to produce sentence-level sentiment predictions for the TRIPOD screenplays. The sentiment of a scene corresponds to the majority sentiment of its sentences. We project scene-based sentiment labels onto shots using the same one-to-many mapping employed for TPs.

### V. RESULTS AND ANALYSIS

#### A. Knowledge Distillation for TP Identification

Before evaluating the performance of our model on trailer moment identification, we first investigate whether our joint training scheme, which distills information from screenplays to movie videos, improves **TP identification**, which is the task that our main network is directly trained on. We split the set of movies with gold-standard scene-level TP labels into development and test set and select the top 5 (i.e.,) and top 10 (i.e.,) shots per TP in a movie. As evaluation metric, we use Partial Agreement (PA), which measures the percentage of TPs for which a model correctly identifies at least one gold-standard shot from the 5 or 10 shots selected from the movie (see Appendix for details, available online).

Table II summarizes our results on the test set. We consider the following comparison systems: **Random** selects shots from evenly distributed sections (average of 10 runs); **Theory** assigns TP to shots according to screenwriting theory (e.g., “Opportunity” occurs at 10% of the movie, “Change of plans” at 25%, etc.); **Distribution** selects shots based on their average position in the training data; **GraphTP** is the original model of Papalampidi et al. [18] trained on screenplays (we identify scenes as TPs and then project scene-level predictions to shots given our scene-to-shots mapping); **Ours w/o graph structure** is a base transformer model without the graph-related information. We also use several variants of our own model (**Ours with graph structure**): without and with the auxiliary text-based network, trained asynchronously (i.e., first training the auxiliary network and then transferring the knowledge to the main network) only with the prediction consistency loss (P), both prediction and representation losses (P + R), and our contrastive joint training regime.

We observe that our approach outperforms all baselines, and the equivalent transformer-based model without the graph information. Although transformers can encode long-range dependencies between shots, directly encoding sparse connections learned in the graph is additionally beneficial. Moreover, asynchronous knowledge distillation via the prediction consistency loss (P) further improves performance, whereas adding the representation consistency loss (P + R) decreases PA. The proposed training approach (contrastive joint) performs best. We observe that encouraging the main network to compute representations close to the (fixed) auxiliary ones hurts performance leading to information loss, since the two networks capture complimentary information with visual and audio cues being richer in certain cases. In contrast, joint training improves both networks’ representations. Finally, pretraining brings further

---

8[https://github.com/Breakthrough/PySceneDetect](https://github.com/Breakthrough/PySceneDetect)
and random TP selection leads to better performance than supervised approaches (first block in Table III) including: Random selection among all shots and among TPs predicted by our main network, and two graph-based systems based on a fully-connected graph, where nodes are shots and edges denote the degree of similarity between them. This graph has no knowledge of TPs, it is constructed by calculating the similarity between generic multimodal representations. TEXT RANK [63] operates over this graph to select shots based on their centrality, while GRAPH TRAILER without TP traverses the graph with TP and sentiment criteria removed. For the unsupervised systems which introduce stochasticity and produce proposals (i.e., Random, GRAPH TRAILER), we consider the best trailer out of 10 proposals. The second block of Table III presents supervised approaches which use trailer labels for training. These include CCANet [3], which is the state-of-the-art in trailer moment identification; it considers only visual information and computes the cross-attention between movie and trailer shots; Supervised GRAPH TRAILER without graph is trained for the binary task of identifying whether a shot should be in the trailer without considering graph information, screenplays, sentiment or TPs; Supervised GRAPH TRAILER is our main network (Section III-B) trained on trailer moment identification with trailer-specific labels. GRAPH TRAILER performs best among unsupervised methods. Interestingly, TEXT RANK is worse than random, which shows that trailer moment identification is fundamentally different from vanilla summarization problems. GRAPH TRAILER without TP still performs better than TEXT RANK and random TP selection.\textsuperscript{9} With regard to supervised approaches, we find that adding graph-related information (Supervised GRAPH TRAILER w/o graph) leads to better performance than sophisticated models using visual similarity (CCANet). By adding graph-related information (Supervised GRAPH TRAILER), we obtain further improvements.

We performed two ablation studies on the development set for GRAPH TRAILER. We first assessed how the different training regimes of the dual network influence downstream performance (on the trailer moment identification task). Table IV shows that asynchronous training does not offer any discernible improvement over the base model. However, when we jointly train the main and auxiliary networks (using prediction and representation consistency losses), performance increases by nearly 3%. A further small increase is observed when the auxiliary network is pre-trained on more data. Our second ablation study concerned the criteria used for performing random walks on the graph $G$ (2). As shown in Table V, when we enforce the nodes in the selected path to be close to key events (similarity + TPs) performance improves. When we rely solely on sentiment (similarity + sentiment), performance drops slightly. This suggests that in contrast to previous approaches which mostly focus on superficial visual attractiveness ([3], [20]) or audiovisual sentiment analysis [21], sentiment information on its own is not sufficient and may promote outliers that do not fit well in a trailer. Inspection of trailers produced using the sentiment criterion alone revealed that it tends to promote shots that contain intense visuals and audio (e.g., music, explosions, etc.) but are loosely connected to the plot. When sentiment information is combined with knowledge about narrative structure (similarity + TPs + sentiment), trailer

\textsuperscript{9}Performance on the test set is lower because we only consider labels from official trailers, while the development set contains multiple trailers. We refrained from using more trailers for the test set in order to match the Moviescope dataset and avoid noise from fan-based videos.
shots are not only interesting but also related to important events in the movie. This further validates our hypothesis that the two theories about creating trailers are complementary and can be combined.

Finally, since we have multiple trailers per movie (in the dev set), we can measure the overlap between their shots (Upper bound). The average overlap is 86.14%, demonstrating good agreement between trailer makers and a big gap between and automatic models and human performance.

C. Human Evaluation of Trailer Moment Identification

We also conducted a human evaluation study to assess the quality of the selected trailer shots. For human evaluation, we include Random selection without TPs as a lower bound, the two best performing unsupervised models (i.e., GraphTRAILER with and without TPs), and two supervised models: CCANet, which is the previous state-of-the-art for trailer moment identification, and the supervised version of our model, which is the best performing model according to automatic metrics.\(^{10}\) We generated trailers for all movies in the held-out set by concatenating the identified trailer shots. We then asked Amazon Mechanical Turk (AMT) crowd workers to watch all trailers for a movie, answer questions relating to the information provided (Q1) and the attractiveness (Q2) of the trailer, and select the best and worst trailer (see details in Appendix, Section 6, available online). We collected assessments from five different judges per movie.

Table VI shows that GraphTRAILER with TPs provides on average more informative (Q1) and attractive (Q2) trailers than all other systems (pairwise differences are significant using a $\chi^2$ test). Although GraphTRAILER without TPs and Supervised GraphTRAILER are more often selected as best, they are also chosen equally often as worst. When we compute standardized scores ($z$-scores) using best-worst scaling [164], GraphTRAILER with TPs achieves the best performance (note that is also rarely selected as worst) followed by Supervised GraphTRAILER. Interestingly, GraphTRAILER without TPs is most often selected as best (24.40%), which suggests that the overall approach of modeling movies as graphs and performing random walks instead of individually selecting shots helps create coherent trailers. However, the same model is also most often selected as worst, which shows that this naive approach on its own cannot guarantee good-quality trailers.

\(^{10}\)We do not include gold-standard trailers in the human evaluation, since they are post-processed (i.e., montage, voice-over, music) and thus not directly comparable to automatic ones.

|                | Q1 | Q2 | Best | Worst | BWS |
|----------------|----|----|------|-------|-----|
| Random selection w/o TPs | 38.2 | 45.6 | 19.1 | 25.9 | -1.26 |
| GraphTRAILER w/o TPs | 37.2 | 44.5 | 24.4 | 25.9 | -0.84 |
| GraphTRAILER w/ TPs | 41.4 | 48.2 | 20.8 | 11.6 | 1.40 |
| CCANet | 37.7 | 46.6 | 14.3 | 15.2 | -0.14 |
| Supervised GraphTRAILER | 37.7 | 47.1 | 21.4 | 21.4 | 0.84 |

Percentage of yes answers for: does the trailer contain sufficient information (Q1) and is it attractive (Q2). Percentage of times each system was selected as best or worst, and standardized best-worst scaling score.

\textbf{Spoiler Alert:} Our model does not explicitly avoid spoilers when selecting trailer shots. We experimented with a spoiler-related criterion when traversing the movie graph in Algorithm 1. Specifically, we added a penalty when selecting shots that are in “spoiler-sensitive” graph neighborhoods. We identified such neighborhoods by measuring the shortest path from the last two TPs, which are by definition the biggest spoilers in a movie. However, this variant of our algorithm resulted in inferior performance according to automatic metrics and we thus did not pursue it further. We believe that such a criterion is not beneficial for selecting trailer shots, since it discourages the model from selecting exciting shots from the latest parts of the movie. These high-tension shots are important for creating interesting trailers and are indeed included in real-life trailers. More than a third of professionally created trailers in our dataset contain shots from the last two TPs (“Major setback”, “Climax”). We discuss this further in the Appendix (see Sections 4 and 5), available online.

VI. GraphTRAILER as an Interactive Tool

One of the advantages of our algorithm is that it uses interpretable shot selection criteria which can be easily turned on and off depend on user preferences. In the following, we describe how our algorithm works via an example, highlighting its human-in-the-loop functionality (Section VI-A). We also provide a use-case where trailers are created semi-automatically using GraphTRAILER proposals (Section VI-B).

\textbf{A. Method}

We present in Fig. 4 an example of how GraphTRAILER operates over a sparse (shot-level) graph for the movie “The Shining”. We begin with shots that have been identified as TPs (i.e., “Opportunity”; introductory event for the story). We sample a shot (bright green nodes in graph) and initialize our path. For the next steps (2–5; in reality, we execute up to 10 steps, but we omit a few for the sake of brevity), we only examine the immediate neighborhood of the current node and select the next shot to be included in the path based on the following criteria: (1) semantic coherence, (2) time proximity, (3) key events, and (4) sentiment intensity (see details about how we formalize these criteria in Section III-A). We observe that our algorithm manages to stay close to important events (colored nodes) while creating the path, which means that we reduce the probability of selecting random shots with no relevance to the main story. In “Final Step” (Fig. 4), we assemble the trailer sequence by concatenating all selected shots in the path. We also illustrate the path in the graph (i.e., red line; “Final Step” of Fig. 4).

A similar procedure is followed when GraphTRAILER is used as an interactive tool. (see our demo\(^ {11}\) and the example in Fig. 5). Specifically, given the immediate neighborhood at each step (e.g., Step 2 in Fig. 4), a user selects a shot for the trailer given a small set of options with corresponding metadata (i.e., key events, sentiment intensity, semantic coherence), which are easy to review. Moreover, the user can also decide when to finish the trailer, and move back and forth depending on the set of options.

\(^{11}\)https://movie-trailers-beta.herokuapp.com
Fig. 4. Run of GraphTrailer algorithm for the movie “The Shining”. Step 1 illustrates the shot-level graph (pruned for better visualization) with colored nodes representing the different types of TPs predicted in the movie (i.e., TP1, TP2, TP3, TP4, TP5). Our algorithm starts by sampling a shot identified as TP1 (Step 1). For each next step, we only consider the immediate neighborhood of the current shot (i.e., 6–12 neighbors) and select the next shot based on the following criteria: (1) semantic similarity, (2) time proximity, (3) narrative structure, and (4) sentiment intensity (Steps 2–5 or beyond). Finally, we assemble the proposal trailer (Final step) by concatenating the shots in the path. When our algorithm is used as an interactive tool, it allows users to review candidate shots at each step and manually select the best one while taking into account our criteria. Users create trailers by only reviewing around 10% of the movie.

B. Semi-Automatic Trailer Creation

We next evaluate whether our interactive tool can provide good quality outputs while minimizing the time a user spends on the task. By semi-automatically creating trailers, we can easily correct mistakes of the automatic process (e.g., avoid unwanted spoilers, better combine shots), while minimizing the time required. We measure this trade-off between automatic methods and human involvement by comparing shots selected via our interactive tool against those selected by an expert or an automatic system.

For assembling trailers semi-automatically, we recruited two non-experts and asked them to first familiarize themselves with the tool and then select sequences of shots for inclusion in a trailer for all movies in our held-out set (41 movies in total). Users were encouraged to select shots by freely going back and forth until they were satisfied with the result, without however devoting more than 30 minutes to each movie. On average, they spent 22 minutes per movie (going backwards and forward three times).

To assess the quality of the semi-automatic trailers, we conducted a human evaluation study similar to the one presented in Section V-C. Specifically, Crowdworkers were asked to watch a pair of trailers for the same movie and indicate which one was most informative (Q1), attractive (Q2), whether it contained spoilers (Q3), and which one was best overall. Participants compared semi-automatic trailers created as discussed above against fully automatic ones obtained from GraphTrailer or gold-standard labels (see Section IV). We elicited preferences that are presented later in the creation process. Finally, when selecting a shot, it is possible to trim it to adjust imperfections that might appear due to automatic shot segmentation. We provide more details and examples of our interactive tool in the Appendix (Section 7), available online. Overall, our approach drastically reduces the amount of shots that need to be reviewed during trailer creation to 10% of the movie. Moreover, our criteria allow users to explore different sections of the movie, and create diverse trailers.
from five different judges per movie who provided judgments for the following combinations: (1) semi-automatic trailers vs. GRAPHTRAILER, and (2) semi-automatic trailers vs. gold-standard selection.

We present the results of the human evaluation study in Fig. 6 as the relative difference between semi-automatic trailers and either GRAPHTRAILER or gold-standard selection. We observe that AMT participants prefer semi-automatic trailers against GRAPHTRAILER 60% of the time, indicating that having a human in the loop increases the quality of trailer shot selection. Moreover, semi-automatic trailers are also preferred 52% of the time against gold-standard selection, suggesting that the quality of the trailers for both methods is comparable, even though we instructed our users to make a trailer in less than 30 minutes!

Finally we also present the relative difference between methods for the percentage of Yes-answers to Q1 (Informativeness), Q2 (Attractiveness), and Q3 (Spoilers). We observe that the semi-automatic approach increases both the informativeness and attractiveness of the trailers, while reducing spoilers by 20%. Semi-automatic and gold-standard selection are comparable in terms of informativeness and attractiveness. Semi-automatic trailers, however, contain 40% more spoilers than gold-standard ones. Although this could be improved in the future, it does not seem to significantly affect how judges overall rate semi-automatic trailers, i.e., they still prefer them to GRAPHTRAILER and consider them as good as gold-standard trailers which, incidentally, also contain spoilers (5.85% of the time).

VII. CONCLUSION

In this work, we proposed an approach to trailer moment identification which adopts a graph-based representation of movies and uses interpretable criteria for selecting shots. We have also shown that privileged information from screenplays can be leveraged via contrastive learning, resulting in a model that identifies turning points and trailer moments (see Appendix, Section 6, available online for discussion on how TPs generalize to other tasks and datasets). Finally, we showcased how our algorithm can be converted into an assistive tool for trailer creation with a human in the loop. Semi-automatic trailer creation drastically reduces the number of shots that need to be reviewed, leading to better quality output compared to fully automatic methods.

In the future we would like to focus on methods for predicting fine-grained emotions (e.g., grief, loathing, terror, joy) in movies. In this work, we consider positive/negative sentiment as a stand-in for emotions, due to the absence of in-domain labeled datasets. Previous efforts have focused on tweets [65], YouTube opinion videos [66], talkshows [67], and recordings of human interactions [68]. Preliminary experiments revealed that transferring fine-grained emotion labels from other domains to ours leads to unreliable predictions compared to sentiment which is more stable and improves trailer moment identification. Avenues for future work include emotion datasets for movies and emotion detection models based on textual and audiovisual cues.

VIII. ETHICS STATEMENT

The human evaluation studies using Mechanical Turk reported in Sections V-C and VI-B were approved by the IRB of the US Air Force Research Laboratory, protocol number FWR20180142X. The human-in-the-loop study reported in Section VI-B was conducted by two employees of the University of Edinburgh. The School of Informatics ethics committee confirmed that such studies are exempt from ethical approval.

REFERENCES

[1] G. Irie, T. Satou, A. Kojima, T. Yamasaki, and K. Aizawa, “Automatic trailer generation,” in Proc. 18th ACM Int. Conf. Multimedia, 2010, pp. 839–842.
[2] A. F. Smeaton, B. Lehane, N. E. O’Connor, C. Brady, and G. Craig, “Automatically selecting shots for action movie trailers,” in Proc. 8th ACM Int. Workshop Multimedia Inf. Retrieval, 2006, pp. 231–238.
[3] L. Wang, D. Liu, R. Puri, and D. N. Metaxas, “Learning trailer moments in full-length movies with co-contrastive attention,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 300–316.
[4] J. E. Cutting, “Narrative theory and the dynamics of popular movies,” Psychol. Bull., vol. 23, no. 6, pp. 1713–1743, 2016.
[5] M. Hauge, Storytelling Made Easy: Persuade and Transform Your Audiences, Buyers, and Clients—Simply, Quickly, and Profitably, Oceanside, CA, USA: Indie Books International, 2017.
[6] K. Thompson, Storytelling in the new Hollywood: Understanding classical narrative technique. Cambridge, MA, USA: Harvard Univ. Press, 1999.
[7] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, “Unifying distillation and privileged information.” 2015, arxiv:1511.03643.
[8] M. Tapaswi, M. Bauml, and R. Stiefelhagen, “Book2movie: Aligning video/scenes with book chapters,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1827–1835.
[9] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, “MovieQA: Understanding stories in movies through question-answering,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 4631–4640.
[10] J. Lei, L. Yu, M. Bansal, and T. Berg, “TVQA: Localized, compositional video question answering,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2018, pp. 1369–1379.
[11] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, “A dataset for movie description,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 3202–3212.
[64] J. Louviere, T. Flynn, and A. A. J. Marley, *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2015.

[65] M. Abdul-Mageed and L. Ungar, “EmoNet: Fine-grained emotion detection with gated recurrent neural networks,” in *Proc. 55th Annu. Meeting ACL*, Vancouver, Canada, 2017, pp. 718–728. [Online]. Available: https://aclanthology.org/P17-1067

[66] A. Bagher Zadeh, P. P. Liang, S. Poria, E. Cambria, and L.-P. Morency, “Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph,” in *Proc. 56th Annu. Meeting ACL*, Melbourne, Australia, 2018, pp. 2236–2246.

[67] M. Grimm, K. Kroschel, and S. Narayanan, “The Vera am Mittag German audio-visual emotional speech database,” in *Proc. IEEE Int. Conf. Multimedia Expo*, 2008, pp. 865–868.

[68] S. Bilakhia, S. Petridis, A. Nijholt, and M. Pantic, “The MAHNOB mimicry database: A database of naturalistic human interactions,” *Pattern Recognit. Lett.*, vol. 66, pp. 52–61, 2015.

Pinelopi Papalampidi is currently working toward the PhD degree with the School of Informatics, the University of Edinburgh and member of EdinburghNLP working on structure-aware narrative understanding and summarization from multiple views.

Frank Keller is a professor with the School of Informatics, the University of Edinburgh. He is affiliated with EdinburghNLP, the Natural Language Processing Group, the University of Edinburgh. His research focuses on how people solve complex tasks, such as understanding language or processing visual information.

Mirella Lapata is professor with the School of Informatics, the University of Edinburgh. Her research focuses on computational models for the representation, extraction, and generation of semantic information from structured and unstructured data, involving text and other modalities, such as images, video, and large scale knowledge bases.