THE UNIVERSITY *of* EDINBURGH

Structure-aware Narrative Summarization From Multiple Views

Pinelopi Papalampidi

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2022

Abstract

Narratives, such as movies and TV shows, provide a testbed for addressing a variety of challenges in the field of artificial intelligence. They are examples of complex stories where characters and events interact in many ways. Inferring what is happening in a narrative requires modeling long-range dependencies between events, understanding commonsense knowledge and accounting for non-linearities in the presentation of the story. Moreover, narratives are usually long (i.e., there are hundreds of pages in a screenplay and thousands of frames in a video) and cannot be easily processed by standard neural architectures. Movies and TV episodes also include information from multiple sources (i.e., video, audio, text) that are complementary to inferring high-level events and their interactions. Finally, creating large-scale multimodal datasets with narratives containing long videos and aligned textual data is challenging, resulting in small datasets that require data efficient approaches.

Most prior work that analyzes narratives does not consider the above challenges all at once. In most cases, *text-only* approaches focus on full-length narratives with complex semantics and address tasks such as question-answering and summarization, or *multimodal* approaches are limited to short videos with simpler semantics (e.g., isolated actions and local interactions). In this thesis, we combine these two different directions in addressing narrative summarization. We use *all input modalities* (i.e., video, audio, text), consider *full-length narratives* and perform the task of narrative summarization both in a video-to-video setting (i.e., video summarization, trailer generation) and a video-to-text setting (i.e., multimodal abstractive summarization).

We hypothesize that information about the narrative structure of movies and TV episodes can facilitate summarizing them. We introduce the task of *Turning Point identification* and provide a corresponding dataset called TRIPOD as a means of analyzing the narrative structure of movies. According to screenwriting theory, turning points (e.g., change of plans, major setback, climax) are crucial narrative moments within a movie or TV episode: they define the plot structure and determine its progression and thematic units. We validate that narrative structure contributes to extractive screenplay summarization by testing our hypothesis on a dataset containing TV episodes and summary-specific labels.

We further hypothesize that movies should not be viewed as a sequence of scenes from a screenplay or shots from a video and instead be modelled as sparse graphs, where nodes are scenes or shots and edges denote strong semantic relationships between them. We utilize *multimodal* information for creating movie graphs in the latent

space, and find that both graph-related and multimodal information help contextualization and boost performance on extractive summarization.

Moving one step further, we also address the task of trailer moment identification, which can be viewed as a specific instiatiation of narrative summarization. We decompose this task, which is challenging and subjective, into two simpler ones: narrative structure identification, defined again by turning points, and sentiment prediction. We propose a graph-based unsupervised algorithm that uses interpretable criteria for retrieving trailer shots and convert it into an interactive tool with a human in the loop for trailer creation. Semi-automatic trailer shot selection exhibits comparable performance to fully manual selection according to human judges, while minimizing processing time.

After identifying salient content in narratives, we next attempt to produce *abstractive* textual summaries (i.e., video-to-text). We hypothesize that multimodal information is directly important for generating textual summaries, apart from contributing to content selection. For that, we propose a parameter efficient way for incorporating multimodal information into a pre-trained textual summarizer, while training only 3.8% of model parameters, and demonstrate the importance of multimodal information for generating high-quality and factual summaries. The findings of this thesis underline the need to focus on *realistic* and *multimodal* settings when addressing narrative analysis and generation tasks.

Lay Summary

Nowadays, we have access to a vast amount of movies and TV shows in platforms, such as Netflix. Navigating through this content (e.g., to refresh our memory or for deciding what to watch next) is becoming increasingly challenging, and as a result, developing automatic or semi-automatic methods for summarizing such content would greatly facilitate users. Although summarizing movies and TV shows comes naturally to humans, it is especially challenging for machines, which have to combine different input sources (i.e., video, audio, subtitles) each encompassing complementary information, process very long videos of 1-2 hours and their transcripts, and learn from a handful of examples, since collecting and processing such videos is hard.

Given the challenges of this multimodal summarization task, most prior work does not consider all facets of the computational problem at once but instead focuses on either processing multiple but short input sources or long text-only narratives. In contrast, we aim at summarizing *full-length* movies and TV episodes while considering all input sources for creating video and textual summaries.

We propose bottom-up methods for addressing narrative summarization, where we first identify the narrative structure of movies and TV episodes based on screenwriting theory. Next, we demonstrate how such information can contribute to narrative summarization by creating video summaries and explore ways to learn interactions between events in movies based on both textual and audiovisual information. Moving a step further, we address the task of trailer generation and propose an algorithm for selecting trailer moments in movies based on interpretable criteria such as the narrative importance and sentiment intensity of events. We further demonstrate how we can convert our algorithm into an interactive tool for trailer creation with a human in the loop. After successfully identifying salient content in narratives, we finally produce textual summaries from full-length TV episodes given important content from the corresponding videos and transcripts.

We overall highlight the importance of identifying the structure of a narrative in terms of key moments and more fine-grained interactions between events, and the combination of *all* (multimodal) input sources for identifying salient content and producing textual summaries of movies and TV episodes.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors, Mirella Lapata and Frank Keller, for their support, constructive feedback, and professional and personal guidance. Our discussions shaped my research interests, way of thinking, and long-term professional goals. Apart from the valuable knowledge I gained from them, they were my mentors in all things considered; research, academia, career, life. I am also thankful to my internship mentors and peers, Tomas Kocisky and Kris Cao from DeepMind, and Marcus Rohrbach, Florian Metze, and Christoph Feichtenhofer from Meta. Both the Language team at DeepMind and the Computer Vision team at Meta Menlo Park offered me amazing experiences and working environment and helped me broaden my knowledge and interests by working on exciting projects. I also thank my examiners, Mohit Iyyer and N. Siddharth, for their feedback and for making my viva an exceedingly positive experience

Next, I would like to thank the colleagues and friends I made during these years in Edinburgh, with whom I had so many research- and life-related conversations: Ivana Balazevic and Maria Gorinova for all the walks, coffees, discussions, and for letting me nag about weather and life, Christos Perivolaropoulos and Marina Efstratiou, for their valuable friendship, Nicola De Cao for the in-person and online conversations, and Janie Sinclair, Tom Sebourne and David Hodges, for being the best officemates that one could ask for and for their day-to-day support through this journey. I would also like to thank all members of EdinburghNLP and the research groups that I participated in during the PhD years for many fruitful discussions. A special thanks goes to my Greek friends that are all around the world, but they still support me in life from a distance and always believe in me: Konstantina Chorti, Elli Galata, Efimia Panagiotaki, Katerina Margatina, Evgenia Chroni, Alexandra Chronopoulou, Christos Frisiras, and Elena Messini.

Moreover, I would like to express my gratitude to a very special person, Christos Baziotis, for always and unconditionally being there for me, and for supporting me by giving me research- and life-related advise, helping me in any way possible, and making my life infinitely better. Finally, I am forever grateful to the people that have supported me the most my entire life in all my decisions and through school, undergrad, and PhD: my beloved parents, Vasilis Papalampidis and Olga Anyfanti, and sister, Liana Papalampidi. I would not be able to learn so much, cultivate my love for research, and achieve anything without them and their constant support and love.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Pinelopi Papalampidi*)

Table of Contents

1 Introduction **1**

1.1 Thesis Statement 4

1.2 Contributions 6

1.3 Thesis Outline 7

1.4 Published Work 10

2 Background **11**

2.1 Narrative Theory 11

2.1.1 Narrative Structure 12

2.1.2 Turning Points 13

2.1.3 Examples of Narrative Structure 15

2.1.4 Trailer Creation 20

2.2 Neural Networks 21

2.2.1 Recurrent Neural Networks 21

2.2.2 Transformers 24

2.3 Automatic Narrative Understanding 27

2.3.1 Text-based Narrative Analysis 28

2.3.2 Video-based Narrative Analysis 34

2.4 Summary of Chapter 39

3 Turning Point Identification: Task Definition and Dataset **41**

3.1 Related Work 45

3.2 TRIPOD Description 46

3.3 Automatic TP Identification 49

3.3.1 Task 1: Automatic TP Identification in Plot Synopses 50

3.3.2 Task 2: Coarse to Fine Projection of TP Labels 54

3.3.3 Task 3: End-to-end TP Identification 56

3.4 Experimental Setup 57
3.5 Experiments and Results 58
3.5.1 Task 1: TP Identification on Synopses 58
3.5.2 Tasks 2 and 3: Annotation Projection and TP Identification on Screenplays 61
3.5.3 Discussion 64
3.6 Summary of Chapter 66

4 Screenplay Summarization Using Latent Narrative Structure 71
4.1 Related Work 75
4.2 Problem Formulation 76
4.2.1 Unsupervised Screenplay Summarization 76
4.2.2 Supervised Screenplay Summarization 78
4.2.3 Narrative Structure 79
4.3 Experimental Setup 82
4.4 Results and Analysis 85
4.4.1 Is Narrative Structure Helpful? 85
4.4.2 What Does the Model Learn? 88
4.4.3 Do Humans Like the Summaries? 89
4.4.4 Which Narrative Sections Are More Important? 90
4.5 Summary of Chapter 92

5 Multimodal Movie Summarization via Sparse Graph Construction 95
5.1 Related Work 97
5.2 Problem Formulation 98
5.3 A Turning Point Graph Model 99
5.3.1 Graph Construction 99
5.3.2 Graph Convolutional Networks 102
5.3.3 Model Training 103
5.4 Experimental Setup 104
5.5 Results and Analysis 106
5.5.1 Which Model Identifies TPs Best? 106
5.5.2 Which Information Matters? 108
5.5.3 How Good Are the Summaries? 109
5.5.4 What Do the Graphs Mean? 111
5.5.5 Discussion 115

5.6 Summary of Chapter 116

6 Human-assisted Trailer Creation via Task Composition 123

6.1 Related Work 127
6.2 Problem Formulation 128
6.2.1 GRAPHTRAILER: Movie Graph Traversal 129
6.2.2 Graph Construction and TP Identification 131
6.2.3 Auxiliary Text-based Network 133
6.2.4 Knowledge Distillation 134
6.2.5 Self-supervised Pre-training 136
6.2.6 Sentiment Prediction 137
6.3 Experimental Setup 137
6.4 Results and Analysis 141
6.4.1 Knowledge Distillation for TP Identification 141
6.4.2 Automatic Results on Trailer Moment Identification 144
6.4.3 Human Evaluation on Trailer Moment Identification 146
6.5 GRAPHTRAILER as an Interactive Tool 148
6.5.1 Method 148
6.5.2 Interactive Tool Example 150
6.5.3 Semi-automatic Trailer Creation 152
6.6 Task Decomposition Analysis 155
6.7 Summary of Chapter 157

7 Long Video-to-text Summarization of TV Episodes 159

7.1 Related Work 161
7.2 The SummScreen^{3D} Dataset 163
7.3 Video-to-Text Summarization 165
7.3.1 Backbone Textual Model 165
7.3.2 Multimodal Augmentation 166
7.3.3 Self-supervised Auxiliary Guidance 168
7.3.4 Hierarchical3D Adapters 168
7.4 Content Selection 170
7.5 Experimental Setup 171
7.5.1 Dataset Pre-processing 171
7.5.2 Implementation Details 172
7.5.3 Evaluation Metrics 172

7.6 Results . . . 173
7.7 Examples of Generated Summaries . . . 179
7.8 Summary of Chapter . . . 180

8 Conclusions and Future Work **187**
8.1 Conclusions . . . 187
8.2 Future Work . . . 190

A Annotation Instructions for TRIPOD **195**
A.1 Annotation Scheme Explanation . . . 195
A.2 Examples . . . 200
A.2.1 "Drive" . . . 200
A.2.2 "It's Complicated" . . . 203
A.3 Annotation procedure . . . 205
A.4 Further guidelines . . . 208
A.5 How to annotate the plot files . . . 209
A.6 Example of Annotation Interface . . . 209

B TRIPOD Dataset Details **211**

C Instructions for Human Evaluation **219**
C.1 Human Evaluation on TP Identification in Plot Synopses . . . 219
C.2 Human Evaluation on Screenplay Summarization . . . 220
C.3 Human Evaluation on Movie Summarization . . . 222
C.4 Human Evaluation on Trailer Generation . . . 223

D Model Comparison between GRAPHTP and GRAPHTRAILER **231**

Bibliography **233**

Chapter 1

Introduction

Summarizing narratives, such as movies and TV shows, and creating a condensed version of their stories has a variety of applications in real life. Narrative summaries can either serve as a *preview* or a *recap* of a movie or TV episode. In the former case, viewers are unfamiliar with the content and want to assess whether they should watch the full video. In order to make this decision they would typically watch a trailer (e.g., the official trailer for the movie "Juno"[1]) or a video preview (e.g., as used in platforms such as Netflix[2]; see example in Figure 1.1) or read a short textual summary that provides an introduction to the story (e.g., in IMDb[3] or Rotten Tomatoes[4]). On the other hand, recaps present all important events in the story from beginning to end. In this case, users either have previously watched the full-length video and want to refresh their memory or catch up on prequels in order to directly watch newly released content. Such complete summaries may again appear in video form (e.g., YouTube videos that summarize previous seasons in TV shows or key events in movies[5]) or textual form (e.g., Wikipedia[6] or extended synopses in IMDb[7]).

The narrative content available to users nowadays via platforms such as Netflix is vast and is becoming increasingly difficult to navigate (e.g., to remember previously watched movies and shows or decide what to watch next). Netflix alone has over 222 million subscribers worldwide, with more than 17,000 movies, series, and shows available. Hence, developing automatic or semi-automatic methods to summarizing

[1] https://www.youtube.com/watch?v=QuN0Z65sp5ct=17s
[2] https://www.netflix.com/browse
[3] https://www.imdb.com
[4] https://www.rottentomatoes.com
[5] https://www.youtube.com/c/MOVIECLIPS
[6] https://en.wikipedia.org/wiki/Juno_(film)
[7] https://www.imdb.com/title/tt0467406/plotsummary?ref_=tt_stry_pl#synopsis

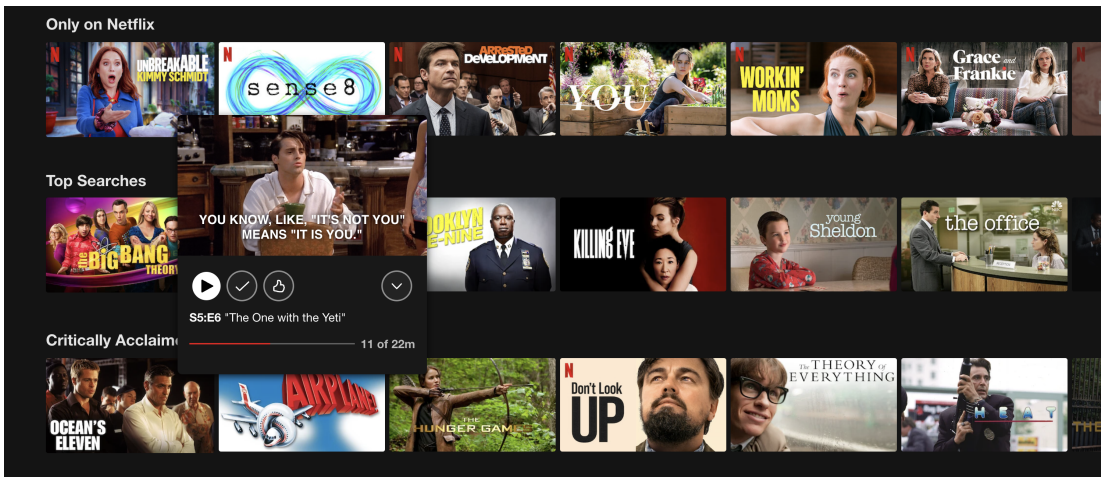


Figure 1.1: Example of the Netflix interface, where users are able to watch short clips as previews of movies and TV shows while browsing content.

narrative content would significantly facilitate navigating through the ever-growing lists of movies and TV shows. Netflix has already incorporated into their platform the presentation of short preview clips from movies and shows during navigation as illustrated in Figure 1.1.

Apart from the importance of real-life applications, narrative summarization also constitutes an interesting research topic from a computational point of view. Narratives, such as movies and TV shows, provide an appropriate testbed for addressing a variety of important challenges in the field of artificial intelligence. They have very complex semantics with interactions between characters and events. Inferring what is happening in a narrative requires encoding long-range dependencies between events, understanding commonsense knowledge, and accounting for non-linearities in the presentation of the story. Moreover, narratives are usually long (i.e., hundreds of pages in a screenplay and thousands of frames in a video; see illustration in Figure 1.3) and cannot be easily processed by standard neural architectures (Vaswani et al., 2017).

Movies and TV episodes also include information from multiple sources (i.e., video, audio, text) that are complementary. For example, in order to summarize all important events from the movie "The Shining" (see example in Figure 1.2), we need to understand what the characters are saying (e.g., screenplay or subtitles) but also process non-verbal cues, such as expressions, emotions (e.g., Wendy is *afraid* for her and her son's life; frames 1 to 3 in Figure 1.2), and actions (e.g., Jack starts to *act crazy* and *runs* with an axe in the house; frames 4 and 5 in Figure 1.2, and Wendy *escapes* from the bathroom window; frame 6 in Figure 1.2). Finally, creating large-scale multimodal

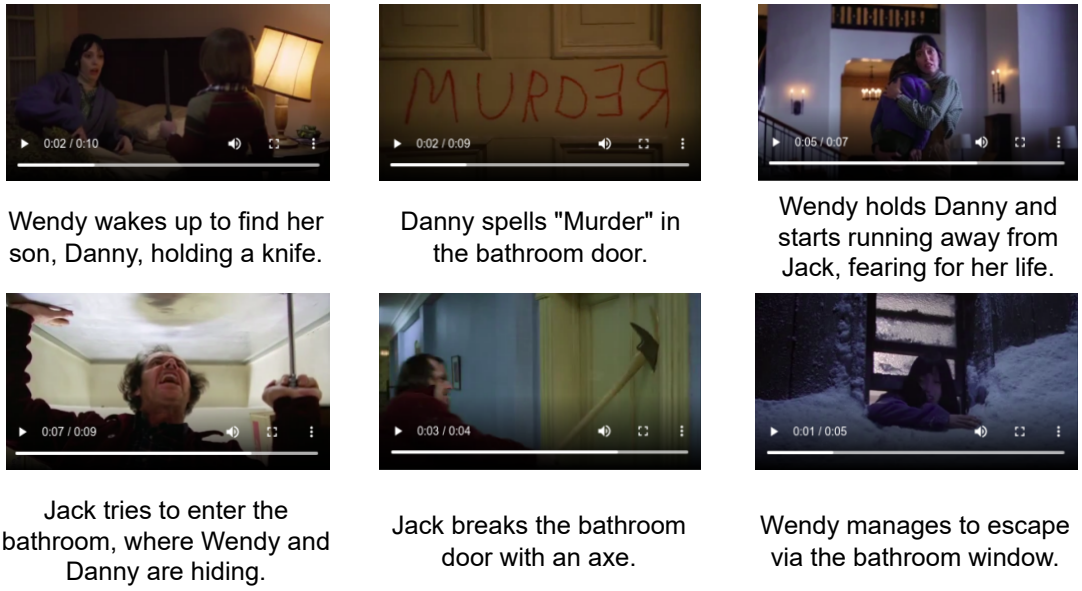


Figure 1.2: Examples of frames from the movie “The Shining” that convey important information about the narrative and can mainly be inferred by the video and audio and not by the dialogue parts of the movie. Descriptions of frames are provided for ease of understanding and are not in the screenplay.

datasets with narratives containing long videos, aligned textual data, and fine-grained annotations over videos (see example in Figure 1.3) is challenging and time consuming, resulting in small datasets (e.g., Huang et al. 2020; Lei et al. 2020b; Wang et al. 2020b) that require sample efficient approaches.

Most prior work that analyzes narratives does not consider all the above challenges at once. Most previous approaches are either text-only focusing on full-length narratives with complex semantics or multimodal but limited to short videos with simpler semantics (e.g., isolated actions and local character interactions). Previous work on textual narrative analysis most commonly analyzes short stories (Bamman et al., 2013; Iyyer et al., 2016) or addresses tasks such as question-answering (QA; Kočiskỳ et al. 2018; Xu et al. 2022) and summarization (Gorinski and Lapata, 2015; Chen et al., 2022a) on books and screenplays using generic QA and summarization approaches, respectively. Video-based approaches that utilize multimodal information from movies and TV shows are limited to isolated video clips which only last a couple of minutes on average and answer low-level questions (e.g., localized actions, object in a frame; Tapaswi et al. 2016; Lei et al. 2018; Liu et al. 2020) or produce single-sentence textual descriptions with simple vocabulary (Lei et al., 2020b). We present a more detailed analysis on previous methods and tasks around narrative understanding in Chapter 2.

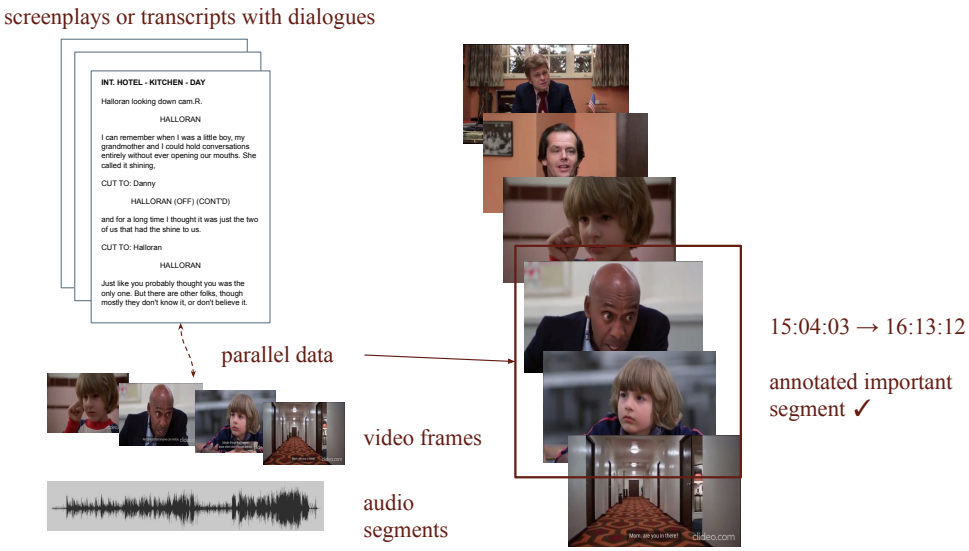


Figure 1.3: Examples of data needed per movie for addressing the task of multimodal movie summarization. Screenplays or transcripts contain dozens or hundreds of pages of dialogue. At the same time the full-length video contains thousands of frames and audio segments with corresponding timestamps that should be aligned with the textual information. Finally, annotations of important moments are needed in the full-length movie. Collecting such datasets is very time-consuming and challenging resulting in a few hundred samples.

1.1 Thesis Statement

In this thesis, we focus on narrative summarization, and we use *all input modalities* (i.e., video, audio, text), consider *full-length narratives*, and address the task of summarization both in a video-to-video (i.e., video summarization, trailer generation) and video-to-text setting (i.e., multimodal abstractive summarization).

We first hypothesize that we should identify narrative structure in movies or TV episodes in order to retrieve key events and create informative summaries. We propose a task and dataset for automatically identifying narrative structure initially in a text-only setting (Chapter 3). We draw inspiration from screenwriting theory (Hague, 2017), wich defines narrative structure by turning points (e.g., opportunity, change of plans, climax) which in turn determine the progression of the plot and segment the narrative into meaningful thematic units. We therefore introduce the task of turning point (TP) identification as a means of identifying narrative structure in movies.

Next, we validate our hypothesis that narrative structure can improve screenplay summarization and further find that the definition of narrative structure we use trans-

fers well to different narrative types (Chapter 4). We then address video-to-video summarization in movies by using information about their narrative structure and modeling them as sparse graphs for creating informative video summaries (Chapter 5) and trailers (Chapter 6). Finally, we transition to a video-to-text setting, where given salient content on TV episodes, we explore ways to generate abstractive textual summaries (Chapter 7) by considering multimodal information (i.e., text, video, audio).

Throughout this thesis, we formulate the following hypotheses for addressing multimodal narrative summarization in both video-to-video and video-to-text settings:

HYPOTHESIS I: Knowledge about the narrative structure of movies and TV shows can facilitate summarizing them. By identifying key events and segmenting the narrative into meaningful thematic units, we can then address summarization and provide key information in visual and textual summaries.

HYPOTHESIS II: On the surface, movies and TV episodes are a *sequence* of scenes or shots. However, they often contain non-linearities in the story, where events may be presented in a non chronological order (e.g., flashbacks, incremental presentation of a story such as in the movies "Memento" and "Slumdog Millionaire"), important scenes may be interrupted by redundant events called "fillers", and distinct sub-plots may intervene. Given this observation, we assume that modeling movies as *graphs* would better capture the complex relationships between events offering better contextualization and improved performance on summarization.

HYPOTHESIS III: Extending HYPOTHESIS II, we hypothesize that modeling movies as *sparse* graphs will lead to more interpretable approaches. By utilizing sparse graphs, we hypothesize that we can better navigate a movie, analyze the topology of the graphs depending on the narrative type, and develop interactive approaches to summarization.

HYPOTHESIS IV: Finally, we hypothesize that incorporating information from full-length *video* and *audio*, facilitates the inference of high-level events that are difficult to be captured solely based on dialogue (contained in screenplays or transcripts). For example, multimodal information should be helpful for understanding non-verbal actions, expressions, emotions, and the characters' whereabouts on screen.

1.2 Contributions

The main contributions of the thesis can be summarized as follows:

Narrative Analysis and Summarization We introduce the following tasks for analyzing and summarizing the content of movies and TV shows:

- *Turning Point (TP) identification*[8] as a means of identifying narrative structure in movies and TV episodes (Chapter 3). We demonstrate that the identified narrative structure contributes to better summarizing narratives[9] (Chapter 4).
- *Long video-to-video movie summarization*, where we generate video summaries or trailers given all input modalities of full-length movies (Chapters 5 and 6).
- *Long video-to-text summarization*, where we generate (multiple-sentence) textual summaries given all input modalities of full-length TV episodes (Chapter 7).

Dataset Creation We introduce two datasets for facilitating research on multimodal approaches with long input sequences (i.e., hour-long videos; thousands of words). Both datasets have been created for different settings of narrative summarization (i.e., video-to-video vs. video-to-text summarization):

- *TRIPOD*[10] contains movie screenplays with turning point (TP) annotations over corresponding plot synopses and aligned full-length movie videos (Chapter 3).
- *SummScreen^{3D}* contains transcripts from TV episodes alongside full-length videos and multiple textual reference summaries (Chapter 7). We create this dataset by extending SummScreen (Chen et al., 2022a), a recently proposed dataset for long dialogue summarization.

Modelling We develop modeling approaches for addressing narrative summarization in video-to-video and video-to-text multimodal settings. Overall, our methods consider structured representations of narratives and are interpretable and parameter-efficient. Specifically, we propose:

[8] https://github.com/ppapalampidi/TRIPOD
[9] https://github.com/ppapalampidi/SUMMER
[10] https://datashare.ed.ac.uk/handle/10283/3820

- Modeling movies as *sparse graphs*, where nodes are scenes from a screenplay or shots from the video and edges denote strong semantic relationships between them[11]. We learn the graph structure for a movie in the latent space and use graph-related information for addressing the tasks of video summarization (Chapter 5) and trailer generation (Chapter 6). We show that graph-based methods can improve accuracy in identifying salient content, while being more interpretable and allowing further analysis.

- An *interpretable* unsupervised algorithm for identifying trailer moments in movies (Chapter 7). We create informative *and* attractive trailers for movies, and most importantly, convert our algorithm into an interactive tool for trailer creation with a human in the loop[12]. We show that semi-automatically selecting sequences of trailer shots provides good quality trailers while minimizing human involvement.

- A *parameter-efficient* way for incorporating multimodal information from video into a pre-trained textual summarizer, that has strong generation capabilities and the correct inductive bias for the task (Chapter 7). We demonstrate that while training only 3.8% of model parameters, multimodal information offers better and more factual textual summaries in comparison with more memory-heavy and fully fine-tuned textual summarizers. Our results underline the importance of considering information from full-length video for summarizing narratives such as TV episodes.

1.3 Thesis Outline

The remainder of the thesis is organized as follows:

Chapter 2 presents background knowledge useful for our work. We introduce theories from screenwriting theory for formulating and identifying narrative structure in movies and TV episodes. Our work is inspired by these theories, that are also used in practice by screenwriters for writing their plays. Next, we give details about the neural network models used in the thesis for encoding narrative content, such as sequences of sentences in screenplays and sequences of video shots in movies. Finally, we provide a detailed analysis of different tasks introduced in previous work in the narrative analysis

[11] https://github.com/ppapalampidi/GraphTP
[12] https://movie-trailers-beta.herokuapp.com

domain. We discriminate between text-based methods that mostly utilize short stories, screenplays or transcripts, and video-based work from the Computer Vision community that uses multimodal information (i.e., most often video frames and subtitles) for addressing tasks, such as video captioning, and question answering in short isolated video clips from movies and TV shows.

Chapter 3 lays the foundations of how we define and identify narrative structure in movies. We present the task of Turning Point (TP) identification as a means of identifying narrative structure in movie screenplays and construct the TRIPOD dataset. As stated in HYPOTHESIS I, we assume that identifying key events and segmenting the narrative into thematic units can boost performance in narrative summarization. We describe how we collect the dataset, that contains movie screenplays, corresponding full-length videos and turning point annotations over plot synopses. Moreover, we present a method for projecting turning points from synopses to full-length screenplays and develop a model that considers both the synopses and screenplays for identifying turning points in movies as a baseline for assessing the difficulty of the task, which considers textual information only.

Chapter 4 examines and validates the hypothesis made in the previous chapter. Our main objective here is to understand whether turning points are general enough to be transferred to different narrative types and most importantly, whether information about narrative structure can boost performance on summarization. For that, we address the task of screenplay summarization, where we extract an optimal sequence of scenes to be presented as the summary, while again considering only textual information. We propose to explicitly incorporate the underlying structure of narratives into general unsupervised and supervised extractive summarization models. We formalize narrative structure in terms of turning points and treat it as latent in order to summarize screenplays. Experimental results on the CSI corpus of TV screenplays (Frermann et al., 2018), which contain scene-level summarization labels, show that latent turning points correlate with important aspects of a CSI episode and improve summarization performance over general extractive algorithms, leading to more complete and diverse summaries.

Chapter 5 addresses the task of multimodal movie summarization. In this chapter, we again consider screenplays as our main source of information, but also take into

account *multimodal* information from *full-length videos*. We first expand the TRIPOD dataset to the multimodal setting by collecting the movie videos and then address the task of movie summarization (i.e., extraction of optimal sequence of scenes) by identifying TPs and directly assembling them into video summaries. Moreover, we propose to discard the naive view of movies as a *sequence* of scenes and instead model them as sparse graphs, where nodes are scenes and edges between scenes denote strong semantic relationships. We find that modeling movies as graphs offers better contextualization, leading to more informative and diverse summaries. Moreover, multimodal information is especially useful for creating latent movie graphs, which are interpretable, displaying different topology for different movie genres.

Chapter 6 overcomes the limitations of the previous chapter that considers screenplays as the main source of information and is only concerned with creating informative video outputs. In this chapter, we operate directly on *video shots* from movies and consider additional criteria for automatically or semi-automatically creating trailers. Trailers have a different functionality from video summaries and directly contribute to the user decision making. Moreover, by directly operating on shots instead of scenes, we are able to compress the videos from 15 to 2 minutes long, which is more appropriate for trailers. We propose a contrastive training approach for distilling information from screenplays and present an unsupervised algorithm for selecting trailer shots that uses interpretable criteria. We finally show that our approach can be converted into an interactive tool for trailer creation, providing better shot selection while minimizing human involvement in the process.

Chapter 7 moves from *extractive* to *abstractive* summarization of TV episodes. After identifying salient content in narratives, we explore ways to incorporate multimodal information into a pre-trained textual summarizer for producing long textual summaries. We propose a parameter-efficient way for incorporating such knowledge by utilizing adapter modules augmented with a hierarchical structure and training only 3.8% of model parameters. We demonstrate that multimodal information significantly contributes to video-to-text summarization, apart from improving content selection in the narratives.

Chapter 8 concludes the thesis and discusses directions for future work.

1.4 Published Work

Parts of the thesis have been presented in Papalampidi et al. (2019) (Chapter 3), Papalampidi et al. (2020) (Chapter 4), Papalampidi et al. (2021b) (Chapter 5). Chapters 6 and 7 are currently under review and are published as preprints (Papalampidi et al., 2022a; Papalampidi and Lapata, 2022).

Chapter 2

Background

In this chapter, we will discuss how screenwriting theory analyzes narratives and identifies narrative structure. In this context, we will also separately focus on the art of trailer creation, where there is no firm theoretical basis as in screenwriting theory, but empirical rules of thumb for creating commercial trailers.

Next, we will move on to automatic methods for processing narratives and will first introduce two main neural network architectures used for processing long input sequences, either sequences of tokens or shots in video. Finally, we will discuss about prior work on automatic narrative analysis and discriminate between text-based and video-based methods and tasks. This thesis aims to bring these two directions closer by jointly considering different modalities of a narrative as they manifest themselves in a movie or TV show (e.g., screenplays, videos, audio), while processing the full-length input.

2.1 Narrative Theory

In this section we present a short overview of the definition of narrative structure as derived from screenwriting theory. Our analysis includes different schemes and variants used by playwrights and screenwriters for composing exciting stories (Section 2.1.1). We also briefly discuss empirical theories used by filmmakers when creating movie trailers (Section 2.1.4). In the thesis, we base our approaches to narrative summarization and trailer generation on such theories for identifying salient narrative content.

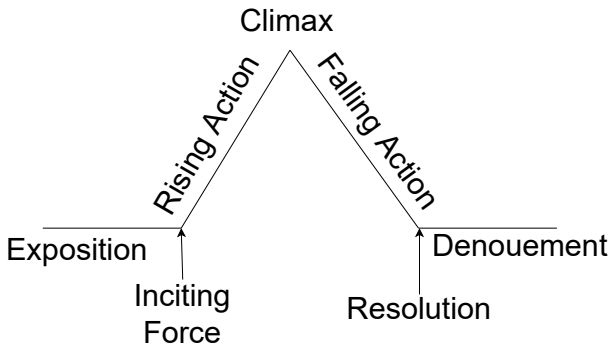


Figure 2.1: Freytag's pyramid for analyzing the narrative structure of plays. There are five acts and two narrative instances of high importance. The peak of the pyramid (i.e., "Climax") is the moment of highest tension and conflict in the story.

2.1.1 Narrative Structure

Our approach to narrative summarization draws inspiration from narratology. Narratology is the study of stories and story structure and the ways these affect how we perceive and experience narratives (Cutting, 2016). Narrative structure, also referred to as a storyline or plotline, describes the framework of how one tells a story and has its origins to Aristotle who defined the basic triangle-shaped plot structure representing the beginning (protasis), middle (epitasis), and end (catastrophe) of a story (Pavis, 1998).

Much later, in the first century BCE, the Roman poet Horace suggested that no play should be longer or shorter than five acts (Kline, 2005). The five-act scheme was formally encoded in the 19^{th} century by the German novelist and playwright Gustav Freytag, who modified Aristotle's structure by transforming the triangle into a pyramid (Freytag, 1896). In his scheme, which is called Freytag's pyramid and is illustrated in Figure 2.1, there are five acts:

1. *Exposition*: The characters and main setting of the story are introduced.

2. *Rising Action*: The story develops towards the main goal.

3. *Climax*: The peak of the conflict in the story.

4. *Falling Action*: The story progresses with decreasing intensity from the climax.

5. *Denouement*: The ending of the story is the same as "catastrophe" in the original triangle structure of Aristotle.

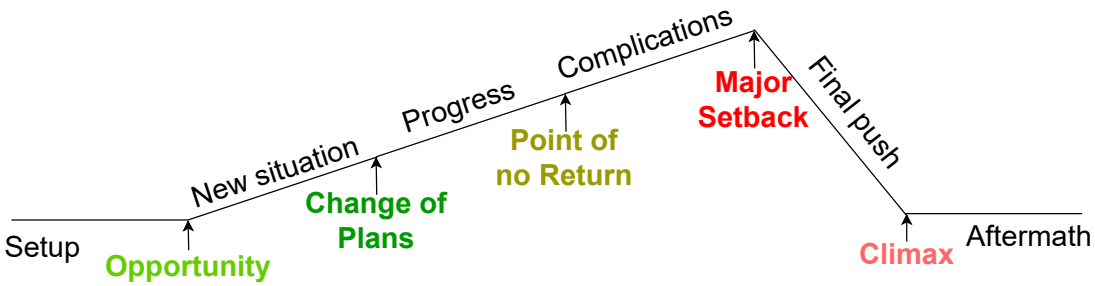


Figure 2.2: Hague (2017)'s modern adaptation of Freytag's pyramid. Each movie or TV show consists of six stages separated by five turning points (colored moments in the scheme).

Freytag also introduced two points in the story, viewed as important narrative instances. The first one is called *exciting force* (or *complication*) and happens right after the first act, and the second one is called *force of final suspense* and occurs in the end of the "Return" act in order to keep high tension until the ending of the story.

These narrative instances, first introduced by Freytag, were named later "turning points" by Thompson (1999) and acquired a more central role in the analysis of the narrative structure. According to Thompson, turning points are narrative moments from which the plot goes to different directions. By definition turning points occur between the acts offering a semantic segmentation of the narratives. Moreover, turning points are also important events that describe the plot of the narrative. Since the initial pyramid scheme by Freytag and the later introduction of turning points as narrative instances of high importance, there are several variations, that consider more or less acts, provide slightly different definitions, and are used today for narrative analysis and screenwriting (Egri, 1972; Frijda et al., 1986; Field, 2005; Lavandier, 2005).

2.1.2 Turning Points

Modern schemes based on Freytag's pyramid that describe the narrative structure are also used today as guides for screenwriters. We base our work on narrative analysis and understanding on such a scheme which was introduced by Hague (2017) and serves as a guide for writing screenplays for Hollywood movies and TV shows[1].

According to this scheme, every movie or TV show can be broken down to *six acts* separated by *five turning points* independently of the individual story presented or its genre. We present the arc for this scheme in Figure 2.2. Interestingly, in this scheme

[1] https://www.storymastery.com

the pyramid is not balanced as Freytag initially suggested; the highest tension event does not happen in the middle of the story but towards the end. Focusing first on the definition of the acts, the narrative is divided into the following six stages:

1. *Setup*: The main characters and the setting of the story are introduced during the first stage.

2. *New Situation*: An opportunity is introduced for the protagonist(s), which brings a new situation. The story starts to evolve now.

3. *Progress*: The goal of the story is determined by this point. The protagonist has now started to work towards that goal. At this point, the audience believes that the goal will be achieved. Although obstacles may be presented, the protagonist seems to find solutions.The main action of the movie has begun.

4. *Complications*: The protagonist fully commits to the goal, and now the main complications of the story are presented. More obstacles start to appear, while the protagonist responds to them.

5. *Final Push*: The major obstacles appear and the protagonist reacts to them. Events presented during this stage describe the protagonist's actions that tries to reach their goal. More complications may appear in this point, while the protagonist's efforts reach a peak.

6. *Aftermath*: The protagonist finally either achieves their goal or fails. After the resolution of the protagonist's objective, the new life/situation is presented. This is the situation, where the protagonist is, after the end of the main story.

However, we are mostly interested in the definition and functionality of *turning points*, since these are the boundaries between the different sections, and most importantly are the events that define the plot and offer a skeleton of the important points that occur in the story:

1. *Opportunity*: This is the introductory event to the story. This event presents an opportunity to the protagonist to change their life.

2. *Change of Plans*: This is the event that defines the main goal of the story. From then on the action begins to increase and there is a clear objective for the protagonist.

3. *Point of No Return*: This is the event that pushes the protagonist to fully commit to their goal. From this point on there is no turning back to the pre-story setting.

4. *Major Setback*: This is the biggest obstacle that the protagonist faces for achieving their goal. At this point everything fall apart either temporarily or permanently.

5. *Climax*: This is the final event of the story, the moment of resolution.

These events occur in between the acts of the story and define the progression of the plot. Hague (2017) suggests that screenwriters can use the turning points for defining the plot structure when creating screenplays. In our work, we use the turning points and their definition in order to identify the narrative structure of movies and TV shows and determine the most important events that can serve as a summary of the story.

2.1.3 Examples of Narrative Structure

We next provide examples of the segmentation of two movies belonging to different genres into acts with intervening *turning points*: "Drive" (crime, drama), and "It's Complicated" (comedy, drama, romance).

2.1.3.1 Drive

Genre: Crime, Drama
Release year: 2011
Metadata from: Wikipedia[2]

The unnamed Driver (Ryan Gosling), who lives in an Echo Park, Los Angeles apartment, works as a mechanic and a part-time movie stuntman. Managed in both jobs by auto shop owner Shannon (Bryan Cranston), the duo also provide a getaway driver service. With Shannon organizing jobs, the Driver gives criminals a strict five-minute window to commit crimes and reach his vehicle (lest they be left behind).: Now we see the setup of the story. The main character and some basic information about his life.

Turning point 1 (Opportunity): *Meeting his new neighbor, Irene (Carey Mulligan), the Driver soon becomes close to her and befriends her young son, Benicio*

[2]https://en.wikipedia.org/wiki/Drive_(2011_film)

(Kaden Leos).: The event of meeting his neighbor is going to change his life even though neither he or the audience know it yet. This is the starting point of our story.

This is undone, however, when Irene's husband, Standard Gabriel (Oscar Isaac), is released from prison. Standard, while initially hostile toward the Driver, soon warms to him. Meanwhile, Shannon persuades Jewish mobsters Bernie Rose (Albert Brooks) and Nino (Ron Perlman) to purchase a stock car chassis and to build it for the Driver to race. Standard, owing protection money from his time in prison, is beat up by Albanian gangster Cook (James Biberi). Threatening both Standard and his family, Cook demands he rob a pawnshop for $40,000 to pay off the debt.: The life of the neighbor is presented here, as well as the progression of the life of the main character and how it is affected by his new friend.

Turning point 2 (Change of Plans): *The Driver, concerned for the safety of Irene and Benicio, steals a Ford Mustang and offers to act as the getaway driver for the pawnshop job.*: Here it becomes clear that the goal of the protagonist is to keep the neighbor and her child safe, for whom he deeply cares.

While waiting for Standard and Blanche (Christina Hendricks) to complete the heist, the Driver sees a Chrysler pull into the lot. As Blanche returns with a large bag, Standard is shot and killed by the pawnshop owner.: The main action starts here. The plan of keeping the neighbor and her son safe seems to working, while her husband is getting killed.

Turning point 3 (Point of No Return): *The Driver flees with Blanche and the money, but they are pursued by the Chrysler, which tries to force them off the road; eluding the other vehicle, the Driver hides with Blanche in a motel.*: Now the character is deeply involved in the story. They start to chase him, so he has no longer the option of returning back to his normal life, as he is now exposed.

Learning the money actually totals a million dollars, the Driver interrogates Blanche, who admits she and Cook planned to double-cross him and Standard, and that the Chrysler belongs to Cook. Minutes later, two of Cook's men attack them in the motel room, killing Blanche and injuring the Driver before he manages to kill them both. The Driver confronts Cook in his strip club, breaking his fingers with a hammer and

threatening to kill him; Cook reveals that Nino was behind the heist. The Driver offers to return Nino's money, but Nino declines and instead sends a hitman (Jeff Wolfe) to the Driver's apartment building.: Now the main obstacles are presented. The action of the movie builds on and the protagonist encounters more and more problems.

Turning point 4 (Major Setback): *Entering the apartment elevator with Irene, the Driver encounters the hitman in the elevator.*: This is the major setback of the movie. While he tries to keep Irene safe from the beginning of the movie, now he encounters a killer in a very contained space while he is with her. So, they are both in great danger now.

Spotting the hitman's pistol, the Driver kisses Irene before violently beating the hitman, killing him while Irene watches in horror. In his pizzeria, Nino explains to Bernie and Cook that the heist money belonged to a crime family and, since anyone tied to the robbery could lead the Mafia to them, they need to kill everyone involved. Nino further explains that it was his plan all along to steal the money from the crime family, and it was his idea to set up the $40,000 dummy robbery. Bernie then proceeds to murder Cook, before killing Shannon when he refuses to divulge the whereabouts of the Driver. The Driver, disguising himself with a mask, follows Nino to the Pacific Coast Highway and T-bones Nino's car onto a beach. With Nino injured and weakened, the Driver drowns him in the Pacific Ocean. The Driver, using Nino's phone, arranges to meet Bernie at a Chinese restaurant. The Driver makes a final phone call to Irene to tell her he is leaving, and says that meeting her and Benicio was the best thing that happened to him. At the restaurant, Bernie promises Irene's and Benicio's safety in exchange for the money, but he warns that he cannot guarantee the safety of the Driver himself. Outside the restaurant, the Driver gives Bernie the money, only for Bernie to stab him in the stomach.: In this point, we observe the efforts of the protagonist to overcome the problems and actually achieve his goal. This is the part where the main action of the protagonist is presented.

Turning point 5 (Climax): *The Driver retaliates by fatally stabbing Bernie in the neck; he then departs in his car, leaving the money with Bernie's corpse.*: The protagonist reaches his goal in this point, he kills the man who was threatening Irene's safety.

That evening, Irene knocks on the Driver's apartment door to no response; the Driver is then shown driving away into the night.: This is the closing part, it depicts what happened after the resolution.

2.1.3.2 It's Complicated

Genre: Comedy, Drama, Romance
Release year: 2009
Metadata from: Wikipedia[3]

Jane (Meryl Streep), who owns a successful bakery in Santa Barbara, California, and Jake Adler (Alec Baldwin), a successful attorney, divorced ten years ago. They had three children together, two girls and a boy, who are grown. Jake, who was cheating on Jane, married the much younger Agness (Lake Bell).: Here, the main characters alongside with their lives are presented.

Turning point 1 (Opportunity): *Jane and Jake attend their son Luke's college graduation from St. John's University in New York City.*: Here is the opportunity for the divorced couple to be in the same room, in a family gathering.

After a dinner together, the two begin an affair, which continues in Santa Barbara. Jane is torn about the affair; Jake is not. While Agness has Jake scheduled for regular sessions at a fertility clinic, Jake is secretly taking medication, a side effect of which reduces his sperm count. After one of his sessions he has a lunchtime rendezvous with Jane at a hotel. Jake collapses in the hotel room and a doctor is called. The doctor speculates that the reason for Jake's distress may be the medication and says he should stop taking it. Jake and Jane's children know nothing of the affair, but Harley (John Krasinski), who is engaged to their daughter Lauren, spots the pair and the doctor in the hotel but keeps silent. Adam (Steve Martin) is an architect hired to remodel Jane's home. Still healing from a divorce of his own, he begins to fall in love with Jane. On the night of Luke's graduation party in Santa Barbara, Jane invites Adam to the party. She is stoned when he picks her up because she has smoked a marijuana joint that Jake had given her earlier. Later at the party, Adam also smokes a joint with Jane.: The result of the opportunity is the formation of the new situation, which is the development of

[3] https://en.wikipedia.org/wiki/It%27s_Complicated_(film)

an affair for the divorced couple. In this stage, events derived from the new situation alongside with details about the other characters that are involved and maybe affected by the affair, are presented.

Turning point 2 (Change of Plans): *Jake becomes jealous observing them, but with some cajoling by Jane, he gets stoned with them as well.*: The affair that was something casual for both of them, seems in this point to actually affect their mental state.

Agness then observes Jake and Jane dancing together and becomes suspicious of their closeness. When they leave the party, Adam asks Jane if they could have something to eat. Jane takes him to her bakery and makes him chocolate croissants. This takes hours, and they enjoy their time together.: In this stage, we observe that the affair indeed starts to affect the other characters as well (Agness). At the same time the progress of the plot is continued.

Turning point 3 (Point of No Return): *Jake and Agness separate, although it is not clear who leaves whom.*: Now there is an actual consequence of the affair. Jake and Agness separate and things can no longer go back to the initial situation (stopping the affair and proceed with their lives as they were).

Eventually by a webcam in Jane's bedroom, Adam sees Jake naked and realizes that the two have been having an affair. Adam tells Jane he cannot continue seeing her because it will only lead to heartbreak.: Now we observe more complications and problems in the life of the protagonists. Adam also separate with Jane because of Jake.

Turning point 4 (Major Setback): *Jane's kids also find out, and they are not happy about Mom and Dad getting together again because they are still recovering from the divorce.*: Although the audience thinks that with all these complications in the protagonists' lives, they are going to end up together, here is the big obstacle: their children's objections.

Jane tells them she is not getting back with Jake.: This stage does not contain a lot of action. Practically, it's just Jane who gives up to the obstacle (hers children's objection).

Turning point 5 (Climax): *Jane and Jake talk and end their affair on amicable terms.*: Now we have the bigger spoiler of the movie: they are not going to end up together. So, this is the event of the final break up.

The film ends with Adam at Jane's house ready to commence the remodeling. Before the credits roll, Jane and Adam are seen laughing while walking into her house.: Here is the aftermath of the movie, the protagonists are not together, but they maintain a good friendly relationship.

2.1.4 Trailer Creation

Creating trailers can also be viewed as a form of art. Trailers, similarly to screenplays, should present a coherent story and there are theories that try to define their structure. Specifically, the film industry has developed strategies for constructing trailers, which are either based on the three-act scheme derived from plays or focus primarily on the emotions evoked in the audience.

According to one school of thought, trailers must exhibit a narrative structure similar to plays, which is defined by three acts[4]. During the first act, the characters and setup of the story should be established as in full-length plays, the second act introduces the main conflict, and the third act raises the stakes and provides teasers from the ending. We can observe that this scheme is very similar to the five-act schemes used in movies and TV shows, however more emphasis is given in the complications and obstacles, while spoilers from the last two sections are avoided. An example trailer that follows this structure has been created for the movie "The Matrix"[5], where in the beginning, the protagonist "Neo" is introduced and the concept of the "Matrix" world is explained. In the next (second) part, the trailer shows that Neo goes to the real world and attempts to destroy the "Matrix" (i.e., conflict). Finally, the trailer ends with high intensity and exciting shots from the later parts of the movie, which briefly present high-conflict moments for the protagonist.

Another school of thought is more concerned with the mood of the trailer as defined by the ups and downs of the story[6]. This scheme is relevant to emotion-related analy-

[4]https://www.studiobinder.com/blog/how-to-make-a-movie-trailerhttps://www.studiobinder.com/blog/how-to-make-a-movie-trailer

[5]https://www.youtube.com/watch?v=vKQi3bBA1y8t=33s

[6]https://www.derek-lieu.com/blog/2017/9/10/the-matrix-is-a-trailer-editors-dream

sis of narratives (e.g., Reagan et al. 2016) that defines them based on the emotions that evoke in the audience. For example, narratives may be categorized differently depending on their ups and downs as ”rag-to-riches” or ”riches-to-rags” stories. 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 (i.e., cliffhanger). The trailer from “The Matrix” that we previously analyzed can also be interpreted based on this theory. The initial shots of the trailer are of high intensity and do not convey any information. The main goal of these shots is to excite viewers to keep watching the trailer. After the first 14 seconds, intensity drops, and the video delivers key information about the movie (i.e., who “Neo” is, what the “Matrix” is, etc.). Finally, the trailer closes with high intensity shots from the later parts of the movie that raise the stakes and excite viewers to watch the full-length film.

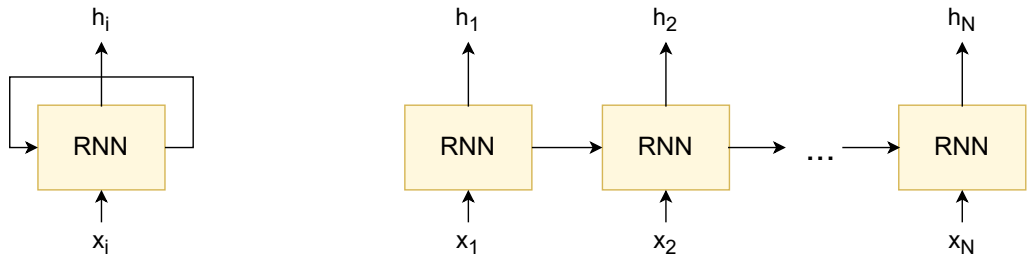
2.2 Neural Networks

In this section we describe the main architectures used for encoding sequences, such as a sequence of textual tokens or a sequence of shots in a movie. In the first chapters of the thesis, we mainly use Recurrent Neural Networks (RNNs; Section 2.2.1) and later we transition to Transformer networks (Vaswani et al., 2017) for encoding narratives (Section 2.2.2).

2.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are used mainly for encoding time series or sequences of input representations. There are several variants of RNNs, such as the Long-Short Term Memory (LSTM; Hochreiter and Schmidhuber 1997) and the Gated Recurrent Unit (GRU; Cho et al. 2014). In this thesis, we use LSTMs and hence we will provide a short description here.

First, we consider a vanilla RNN network. An RNN processes a sequence of input representations $[x_1, x_2, ..., x_N]$ step-by-step via a loop, that allows information to be propagated. We illustrate a high-level overview of RNNs in Figure 2.3. The RNN processes the input autoregressively, and for each time step t it computes a hidden representation h_t given the input representation x_t and the hidden representation of the



(a) RNN with loops (b) unfolded version of RNN

Figure 2.3: A Recurrent Neural Network (RNN) overview. An RNN contains loop for preserving information while processing input sequences. At each timestep the representation h_i of the input x_i also conditions on the previous hidden representation h_{i-1}.

previous time step h_{t-1}:

$$h_t = f(Ux_t + Wh_{t_1}) \tag{2.1}$$

where U, W are learnt parameters of the network and $f(\cdot)$ is a non-linear activation, such as a $\tanh(\cdot)$. Finally, for training RNNs, we use Backpropagation Through Time (BPTT; Werbos 1990), which is a gradient-based training technique used in recurrent networks and time series. In short, BPTT works by unrolling all input timesteps. Each timestep produces an output given the input at the specific timestep and the current weights of the network. Next, the loss is calculated and accumulated for all timesteps. Finally, the weights of the network are updated once after all errors are computed.

However, there is an important limitation of RNNs. While theoretically, they can process sequences of unlimited length, in practice they do not pay enough attention to inputs that are further distant. One important factor that contributes to this phenomenon is the vanishing gradient problem. More recent variants of the RNN have been proposed for overcoming this issue. We will focus on a very popular variant, called Long-Short Term Memory (LSTM; Hochreiter and Schmidhuber 1997) network.

In comparison with RNNs that they have a single function (i.e., $\tanh(\cdot)$) at each time step, LSTMs introduce different gating functions for controlling the information that should be passed to the next time step. Specifically, given the input representation x_t and the hidden representation h_{t-1} of the previous time step, first there is a forget gate layer that decides which information should not be used:

$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \tag{2.2}$$

where W_f and b_f are learnable weights and bias of the forget gate layer, respectively, σ is the sigmoid function and $[\cdot;\cdot]$ denotes the concatenation of the two vectors.

Next, an LSTM unit decided which information to store in its cell state C_t. For this, the input gate layer decides which values to be updated (i_t) and new candidate values $\tilde{C}_t$ are computed. Next, the cell state C_t is updated given the forget gate value f_t, the input gate value i_t, and the new candidate values $\tilde{C}_t$:

$$i_t = \sigma(W_i[h_{t-1};x_t] + b_i) \tag{2.3}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1};x_t] + b_C) \tag{2.4}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{2.5}$$

Finally, we compute the hidden representation h_t for the t^{th} time step, which depends on the cell state C_t. For that, we decide which values of the cell state will be used (o_t) and then again use a non-linear activation function, i.e., $\tanh(\cdot)$, over the cell state:

$$o_t = \sigma(W_o[h_{t-1};x_t] + b_o) \tag{2.6}$$

$$h_t = o_t \tanh(C_t) \tag{2.7}$$

LSTMs process the input autoregressively from left to right. This means that a hidden representation at time step t will only depend on the previously seen representations $x_{<t}$. However, there are cases, where we wish to encode each input with respect to the entire sequence. In this cases, we use the Bidirectional LSTM (BiLSTM), that combines a forward and a backward LSTM layer. We then compute the final hidden representation at time step t by concatenating the two intermediate vectors calculated by the forward and backward LSTMs, respectively: $h_t = [\overrightarrow{h_t};\overleftarrow{h_t}]$.

We compute a sequence of contextualized hidden representations $[h_1,h_2,...,h_N]$ given the input sequence $[x_1,x_2,...,x_N]$ via a (Bi)LSTM network. However, there are cases where we want to compute one final representation for the entire input sequence (e.g., computing a sentence representation given a sequence of tokens as input). In these cases, we aggregate the hidden representations via max-, mean-, or last-pooling.

However, not all elements of the input sequence are as important to contribute equally to the final sequence representation. For example, for computing a sentence representation from a sequence of tokens, there are some tokens that convey very little information (e.g., function words), whereas other tokens are more important for the meaning of the sentence (e.g., content words).

A more sophisticated way of computing the final sequence representation is by using an attention mechanism (Bahdanau et al., 2015) that learns a weight of importance per input element. In this case, the final representation is a weighted average of the hidden representations based on the learnt attention weights. We compute attention scores per hidden representation as follows:

$$y_t = \text{MLP}(h_t) \tag{2.8}$$

$$a_t = \frac{e^{y_t}}{\sum_{i=1}^{T} e^{y_i}} \tag{2.9}$$

where MLP is a multi-layer perceptron consisting of feed-forward layers and non-linear transformations. After computing the attention scores, we can then compute the final representation of the entire input sequence:

$$h = \sum_{i=1}^{T} a_i h_i \tag{2.10}$$

2.2.2 Transformers

Transformer networks were introduced by Vaswani et al. (2017) and are widely used initially in the Natural Language Processing community (e.g., Devlin et al. 2019) and later in the Computer Vision community as well (e.g., Dosovitskiy et al. 2020). Transformers were proposed as a sequence-to-sequence architecture consisting of an encoder and a decoder (Bahdanau et al., 2015).

They are widely popular, since they address important limitations of RNNs and can support parallel (instead of recurrent) training. Transformers overcome the vanishing gradient problem present in RNNs by employing a "self-attention" mechanism, where each element of a sequence attends directly to each other element, avoiding one fixed bottleneck representation for all preceding/following context. Moreover, a Transformer network can process a sequence in parallel speeding up the training process and allowing efficient training on orders of magnitude more data. However, in comparison with RNNs, the complexity of transformer networks grows quadratically with respect to the input sequence length, prohibiting processing very long sequences. A more detailed description of how a transformer network works follows.

Given an input sequence $[x_1, x_2, ..., x_N]$, a transformer encoder computes the contextualized hidden representations $[h_1, h_2, ..., h_N]$. Then, a transformer decoder can be used for sequence-to-sequence generation tasks, such as abstractive summarization. We present an illustration of the transformer architecture in Figure 2.4

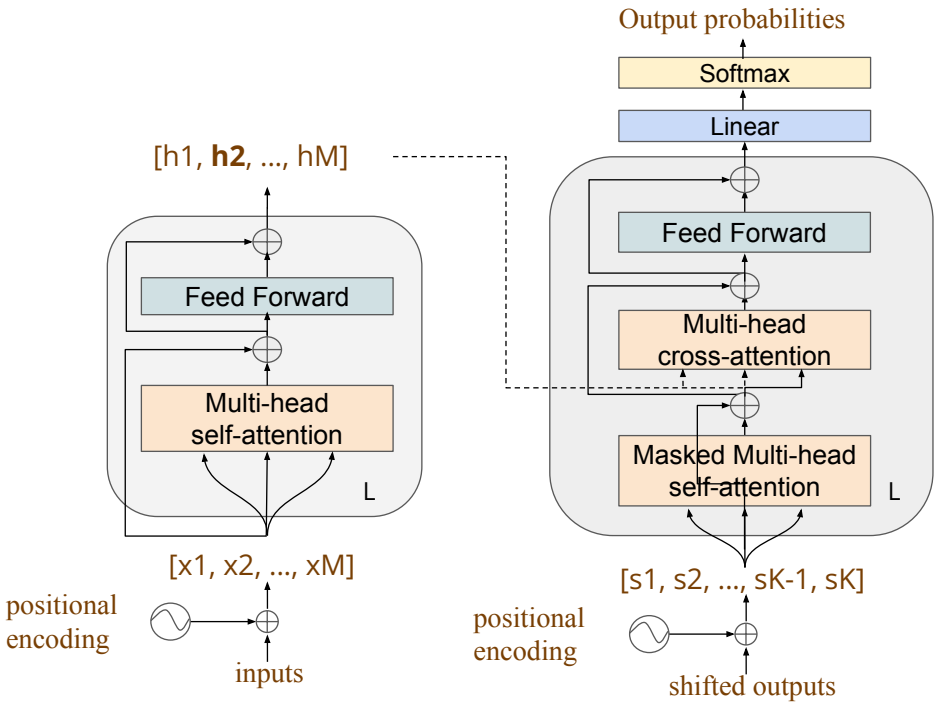


Figure 2.4: A transformer encoder-decoder architecture. The encoder (left side) consists of L stacked identical layers. Each encoder layer consists of a multi-head self-attention block and a feed-forward layer. The decoder (right side) again consists of L stacked identical layers. Each decoder layer consists of a masked multi-head self-attention, a multi-head cross-attention for attending to the encoder outputs and a feed-forward layer.

2.2.2.1 Transformer Encoder

A transformer encoder computes a sequence of contextualized hidden representations $[h_1, h_2, ..., h_N]$ given a sequence of input representations $[x_1, x_2, ..., x_N]$ and it consists of L stacked transformer encoder layers with identical structure.

Each transformer encoder layer takes as input a sequence of (embedded or hidden) representations $[x_1, x_2, ..., x_N]$ and consists of a *multi-head self-attention block* (explained in detail below) and a *feed forward (FFN) layer*. The role of self-attention in the transformer encoder is to contextualize each element of the input sequence with respect to each other element.

In the self-attention layer, we use three different projection matrices, W_k, W_q, W_v for projecting the input representations x_i to keys, queries, and values, respectively (e.g., $k_i = W_k x_i$). Next, we compute the pairwise similarity between the queries and keys and accordingly computed the weighted sum over the values. As pairwise simi-

larity between queries and keys, we consider the scaled dot product:

$$\text{Attention}(Q,K,V) = \text{softmax}\frac{QK^T}{\sqrt{d_k}}V \tag{2.11}$$

where d_k is the dimension of the keys and queries representations used for scaling the dot product values.

We can perform this attention function h times in parallel while learning different projection matrices W_{kj}, W_{qj}, W_{vj}. The different functions which use different parameters are called heads of the attention. When combining the outputs of the heads, we use multi-head self-attention:

$$\text{MultiHead}(Q,K,V) = [\text{Attention}(Q_1,K_1,V_1);...;\text{Attention}(Q_h,K_h,V_h)]W_o \tag{2.12}$$

where W_o is a final linear projection of the concatenated output of the multi-head attention. Finally, a residual connection and a layer normalization are added to the outputs. The last step of each encoder layer is a feed-forward layer that takes as input the output of the multi-head attention and performs two linear transformations with a non-linear (ReLU) activation in between:

$$\text{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.13}$$

After the FFN layer, we again add a residual connection and a layer normalization for computing the final hidden representations as the output of the current encoder layer.

2.2.2.2 Positional Encoding

The transformer encoder does not include any recurrence or convolution and therefore is permutation invariant. However, when encoding sequences, we need to add information about the input order to the model. We achieve that in the transformer by directly injecting information regarding the absolute or relative position of the inputs. Specifically, we first encode the position of the inputs, for example by using sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{2.14}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2.15}$$

We then add the positional embeddings to the embedding input representations and use a layer normalization.

2.2.2.3 Transformer Decoder

While the transformer encoder is used when we want to encode a sequence of input representations (similarly to our description for the (Bi)LSTM), there is also a decoder architecture for generating a new sequence while conditioning on the outputs of the encoder in conditional generation tasks, such as abstractive summarization.

The architecture of the transformer decoder is similar to the encoder. Again, it consists of L stacked identical layers. Each decoder layer consists of the multi-head self-attention and the feed-forward layer, as described in the previous section. We further again add positional encodings to the embedding input representations of the decoder. However, there are two key differences here compared to the encoder structure.

First, when computing the attention scores in the multi-head self-attention, we mask part of the input in order to prevent inputs from attending to subsequent inputs. Practically, this means that we enforce the attention matrix ($\frac{QK^T}{\sqrt{d_k}}$) to be upper triangular. Next, there is one extra block in each decoder layer in between the self-attention and the feed-forward layer in order to attend to the outputs of the encoder. This *cross-attention* layer is identical to the self-attention described in the previous section. The only difference here is that while the queries (Q_j) are computed given the decoder inputs, the keys and values (K_j, V_j) are computed from the encoder outputs. Finally, we again add residual connections and layer normalization in between all blocks included in a decoder layer.

2.3 Automatic Narrative Understanding

In this section we present prior work on automatic narrative understanding. We discriminate between text-based narrative understanding, that only considers text for analyzing narratives (e.g., short stories, screenplays, transcripts), and video-based narrative understanding, that considers primarily the video of movies and TV episodes (Figure 2.5). We also present a comprehensive list of narrative-related tasks in Table 2.1 categorized based on the input/output modalities and input/output lengths. Based on the table and the analysis that follows, we conclude that we differ from prior work in considering all input modalities of a narrative (i.e., video, audio, text) *and* the full-length movie or TV episode for addressing narrative tasks, i.e., summarization and trailer generation.

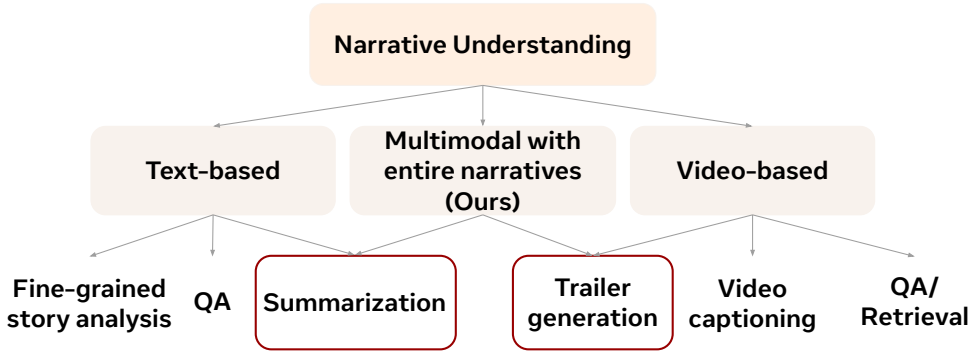


Figure 2.5: We discriminate between text-based and video-based tasks for narrative understanding. In comparison with prior work, we use multimodal information (i.e., video, audio, text) and consider the entire input narrative for addressing the tasks of summarization (either extractive or abstractive) and trailer generation.

2.3.1 Text-based Narrative Analysis

In this section, we focus on prior work on analyzing narratives given textual input, such as screenplays, transcripts of TV shows, books, and short stories (upper part of Table 2.1). We discriminate between different broader tasks that focus on narrative understanding: *fine-grained story analysis*, *closed-book question-answering (QA)*, and *summarization* (see left side of Figure 2.5).

Fine-grained story analysis Most prior work focuses on the entities that participate in the narrative for analyzing its structure. Entities are central in narratives and crucial for the development of the story (Jannidis, 2009; Frow, 2014). According to one school of thought (Fludernik, 2002), there can even be narratives without plot, but not without entities. Going one step further, there are also theories that study character archetypes which connect to actions and events within a story (Fludernik, 2002; Jung, 2014). This view of narratives (see for example Figure 2.6) is orthogonal to our approach, which is based on turning points, focuses on broader events and explores a high-level semantic segmentation of the narrative.

In the context of computational lingustics, there is prior work that tries to induce character types or "personas" in movie plot summaries (Bamman et al., 2013, 2014). A "persona" is defined as a set of mixtures over latent lexical classes, which capture actions and attributes that describe common types of characters across different films. For example, according to their analysis the "Joker" character from "The Dark Knight" and "Drakula" from "Van Helsing" share common attributes and actions and therefore

Input modality	Output modality	Input length	Output length	Tasks
			Prior work	
Text	Text	Long	Short	**Fine-grained story analysis** (e.g., Bamman et al. 2013; Black and Wilensky 1979), **Question-Answering** (e.g., Kočiskỳ et al. 2018; Xu et al. 2022)
Text	Text	Long	Long	**Summarization** (e.g., Gorinski and Lapata 2015, 2018; Chen et al. 2022a)
Multimodal	Text	Short	Short	**Fine-grained movie clip analysis** (e.g., Vicol et al. 2018; Sadhu et al. 2021), **VQA** (e.g., Tapaswi et al. 2016; Lei et al. 2018), **Video captioning** (e.g., Rohrbach et al. 2015; Lei et al. 2020b)
Audiovisual	Multimodal	Long	Short	**Trailer generation** (e.g., Smith et al. 2017; Wang et al. 2020b)
			Ours	
Multimodal	Multimodal	Long	Long/ Short	**Video summarization** (Papalampidi et al., 2020, 2021b), **Trailer generation** (Papalampidi et al., 2021a)
Multimodal	Text	Long	Long	**Multimodal abstractive summarization** (under review)

Table 2.1: Narrative understanding tasks given input/output modalities and input/output lengths. Our work takes *all modalities* (i.e., video, audio, text) and the *full-length narrative* into account for producing video/textual summaries, and trailers.

can be clustered to the same "persona". Other work focuses more on understanding *relationships* between characters (Iyyer et al., 2016; Chaturvedi et al., 2017). They focus on learning dynamic relations between characters and find that they can learn descriptors of events (e.g., marriage or murder) and interpersonal states (love, sadness).

Moving towards event-centric approaches, there is also prior work on analyzing narrative structure in a fine-grained fashion. Black and Wilensky (1979) evaluate the functionality of story grammars in story understanding. Story grammars attempt to formalize a sequence of events in a story given specific rules and states; for example, Mandler and Johnson (1977) describe six major categories of narrative information: setting, beginning, reaction, attempt, outcome, and ending. Then, rules of

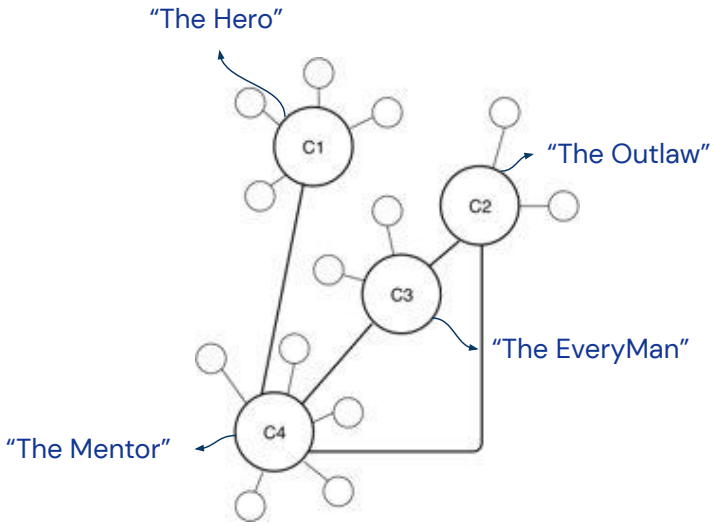


Figure 2.6: Fine-grained character-centered analysis of stories: characters follow specific archetypes and all fine-grained events revolve around them and their relationships.

the story grammar specify temporal (fine-grained) relationships between these categories. Moreover, Elson and McKeown (2009) develop a platform for representing and reasoning over narratives with human annotators, where given a sentence, they construct elaborate predicates, their types, properties and relationships. Finally, Chambers and Jurafsky (2009) learn fine-grained chains of events, which they call "narrative schemas". The arguments in these events are filled with participant semantic roles defined over words; a simplified example of a set of such events would be formulated as: $L = \{(Xpleads), (Xadmits), (convictedX), (sentencedX)\}$, when describing a trial. These sets of events are also enriched with temporal ordering information and argument types (e.g., "police", "government").

More recently, Bamman et al. (2019), Sims et al. (2019), and Bamman et al. (2020) introduce and augment LitBank, a dataset of fiction stories containing annotations for entity categories, literary events to be predicted, coreferences, and quotations. Finally, Thai et al. (2022) propose a new task, i.e., literary evidence retrieval, where given a excerpt of literary analysis they have to automatically retrieve the appropriate quotation from a large set of passages.

However, these approaches focus on *fine-grained events* and are typically studied in *shorter narratives*, which are easier to analyze, in comparison with entire screenplays, TV episode transcripts or books. We advocate turning points as a precursor to more fine-grained analysis that unveils character attributes and their relationships. Identi-

Relevant snippet from plot synopsis:	...Peter's former girlfriend Dana Barrett has had a son, Oscar...
Relevant snippet from screenplay:	*DANA (setting the wheel brakes on the buggy)* Thank you, Frank. I'll get the hang of this eventually. She continues digging in her purse while Frank leans over the buggy and makes funny faces at the baby, OSCAR, a very cute nine-month old boy. *FRANK (to the baby)* Hiya, Oscar. What do you say, slugger? *FRANK (to Dana)* That's a good-looking kid you got there, Ms. Barrett.
Question:	How is Oscar related to Dana?
Answer:	**her son** ✅

Figure 2.7: Example of the closed-book QA task given textual narratives (i.e., screenplays, books). Kočiskỳ et al. (2018) aim at correctly answering questions by identifying the correct supporting passages from full-length screenplays and/or plot synopses.

fying the most important narrative events and segmenting the narrative into broader thematic units can later facilitate a more fine-grained analysis.

Closed-book Question Answering (QA) is a popular task applied on short and long narratives (Bajgar et al., 2016; Lal et al., 2021). Kočiskỳ et al. (2018) introduce a dataset consisting of question-answer pairs over full narratives of screenplays and books (see example in Figure 2.7). Although the question-answer pairs are based on narratives, most prior work uses standard QA approaches for answering questions without any explicit modeling of the narratives (Tay et al., 2019; Kočiskỳ et al., 2018; Frermann, 2019; Mou et al., 2020). Specifically, since the input narratives are long, a standard approach for the task is to use a retriever module, such as tf*idf (Robertson et al., 1995) or BM25[7] (Robertson and Zaragoza, 2009) for retrieving relevant passages from the full-length input and a reader for processing the retrieved passages and extracting answer spans given a question as query. However, it has been demonstrated that such generic retrieve-then-read approaches are not as effective in the narrative do-

[7]BM25 is a widely known retrieval model similar to tf*idf. It assigns each passage a "relevance" score by comparing it against a query.

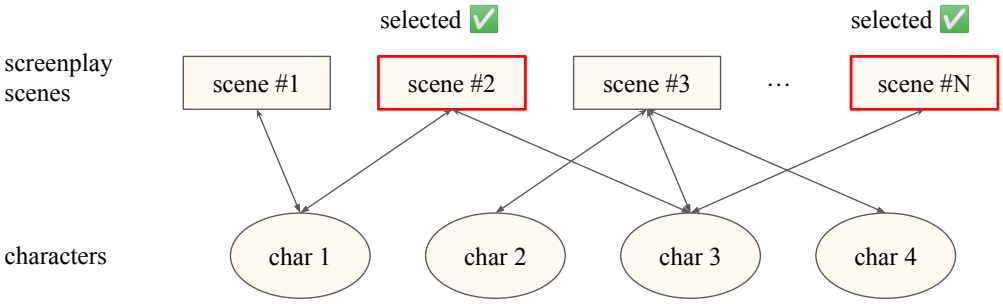


Figure 2.8: Extractive screenplay summarization: given a sequence of scenes from the screenplay Gorinski and Lapata (2015) try to identify a optimal sequence of a few scenes that present the storyline. They extract scenes for the summary via a graph-based approach centered around characters.

main (Angelidis et al., 2019; Mou et al., 2021), since (1) the narrative writing style differs from formal writing, (2) retrieval is more challenging since all candidate passages are semantically related, (3) there is difficulty in obtaining supervision over retrieval, (4) there is also a summarization need for providing answers given the whole narrative, and (5) different passages have logical relations among them.

More recently, Xu et al. (2022) introduce a new QA dataset comprising of question-answer pairs over children-friendly stories. In this dataset, the authors try to distinguish between different narrative elements in the questions and answers (i.e., characters, actions, settings, feelings, causal relationships, resolution, and prediction) and formulate QA as a generation task. Moreover, they provide an analysis of the strengths and weaknesses of existing generic models for the task. Although this is a step towards more explicitly modeling narratives in QA, there is still room for incorporating narrative-specific knowledge into the generic QA approaches.

Summarization is another task used in the narrative domain, where the aim is to summarize stories or long transcripts and allow readers to go through the scripts faster. In the context of narrative summarization, Gorinski and Lapata (2018) automatically generate an overview of the movie's genre (e.g., crime, thriller), mood (e.g., suspenseful, captivating), and artistic style (e.g., strong female presence) by encoding screenplays and learning to classify them according to their attributes. Gorinski and Lapata (2015) summarize full length screenplays by extracting an optimal chain of scenes (see example of extractive screenplay summarization in Figure 2.8). This chain contains a small number of scenes from the original screenplay that presents the most important

Transcript snippet:

Sheldon : What color would you like to be ?
Leonard : Well , I 'd like to be green , but you know you always take it . Sheldon : That 's not true . Any color 's fine with me . Yeah , I could be a - a combination of blue and yellow .
Leonard : Blue and yellow make green .
Sheldon : Well , then it 's settled .
Penny : Hi . Ready to go ?
Sheldon : Oh , good news , we ordered lunch , so we can all stay here and play Lord of the Rings Risk .
Amy : Sheldon , we said that we would play games with you tonight . Sheldon : Oh , no , we 'll still be playing it tonight , this game can easily take eight hours .
Penny : Sweetie , you really thought I 'd want to do this ? Leonard : No .
Penny : Well , did you tell him that ?
Leonard : Yes .
Penny : Did you say it out loud with words ?
Leonard : No .
Penny : I do n't want to spend the whole day playing a board game . ...

Abstractive summary snippet:

Sheldon and Leonard are happy playing a board game until Amy and Penny say they are tired of doing what the guys want ...

Figure 2.9: Abstrctive summarization of TV trascripts: given the full-length transcript of a TV episode, Chen et al. (2022a) create a dataset with corresponding human-written summaries that include the most important events.

events of the story. They extract such optimal chains of scenes via a graph-based approach centered around the characters of the movie and by utilizing criteria such as the logical progression of the story, and the diversity and importance of the selected scenes.

More recently, Chen et al. (2022a) introduce a large-scale dataset, called SummScreen, with transcripts from TV episodes accompanied by human-written summaries for addressing the task of abstractive summarization (see example in Figure 2.9). Although the dataset consists of long narratives for abstractive summarization, previous approaches treat it mainly as a long dialogue summarization dataset without explicitly focusing on the narrative domain. Specifically, most prior work on SummScreen is concerned with efficient methods for processing the entire (long) input. Notably, Beltagy et al. (2020) propose Longformer, an efficient transformer-based architecture for processing long input sequences, which seems to offer competitve performance against a select-then-summarize approach based on unsupervised retrievers, such as tf*idf and BM25 (Chen et al., 2022a). Zhong et al. (2022) also suggest a dialog-specific pre-

training for Longformer, where they mask part of the input and use auxiliary self-supervised objectives, such as identifying the speaker, predicting the next utterance, etc. However, it is still unclear whether there is a significant advantage of such approaches over content selection for this domain (Zhang et al., 2021; Shaham et al., 2022). Moreover, different content selection methods are under-explored for this task, since unsupervised retrievers present low performance in the narrative domain for the QA task as well. Finally, Zhang et al. (2022) follow a different direction and propose a multiple-stage hierarchical approach, where they first summarize the input chunk-by-chunk and then produce the final episode-level summary.

An important limitation of all prior work is that they only consider the *textual modality* for addressing summarization losing important information that can be found in the video and audio modalities. Given only the dialogues of TV episodes, there is information that is difficult or impossible to be inferred, such as who is talking to whom, who else is in the same location during a conversation, and non-verbal information, such as actions and emotions. This information loss is our motivation for considering *all input modalities* for addressing the task of narrative abstractive summarization (Chapter 7).

2.3.2 Video-based Narrative Analysis

Narrative understanding is a popular task in the Computer Vision (CV) community as well, where most prior work focuses on analyzing movies or TV episodes that naturally comprise of multiple modalities (i.e., video, audio, text). In this section, we discriminate between the following tasks (lower part of Table 2.1): *visual question answering (VQA) and retrieval*, *video captioning*, and *trailer generation*. Most previous work in this domain considers multimodal input (i.e., most commonly the video and subtitles/captions), which is however restricted to short clips rather than entire narratives. This restriction naturally results in simpler semantics, where there is no need to infer long-range dependencies between events; instead most approaches focus on localized events, action recognition, and fine-grained interactions. The only exception in existing video-based tasks is trailer generation, where the full-length movie video is considerred as input, but no textual information is given.

Fine-grained movie clip analysis Prior work analyzes actions, events, character relationships and interactions in short movie clips via a variety of different tasks. For

Question	What instrument is Raj playing when Raj and Howard have their show?
Answer 0	Raj is playing flute.
Answer 1	Raj is playing guitar.
Answer 2	Raj is playing drums.
Answer 3	Raj is playing trumpet.
Answer 4	Raj is playing keyboard.

Figure 2.10: Example of visual QA on the TVQA dataset[8](Lei et al., 2018). Given a video clip with subtitles, the goal is to correctly answer to questions given multiple choices.

example, Tapaswi et al. (2015a) and Tapaswi et al. (2015b) align different views of narratives (e.g., book chapters to video clips from the corresponding movie), and Xiong et al. (2019) use a graph-based approach to align paragraphs from plot synopses to movie segments. For performing this alignment, they first construct graphs from the text/video segments that encapsulate the relationships and actions of characters and then identify the most similar graphs between the textual and video modalities for performing the alignment. Chen et al. (2022b) focus on learning a similarity measure between video clips from movies and textual descriptions from plot synopses. A similar task on video-text retrieval is also recently addressed by Sun et al. (2022). Vicol et al. (2018) introduce the MovieGraphs dataset and describe video clips of movies with character-centered graphs. More recently, Sadhu et al. (2021) use Semantic Role Labeling (SRL) for understanding actions and character interactions in 10-second movie clips. Building on top of this work, Xiao et al. (2022) propose an hierarchical encoder for learning video representations and addressing tasks such as video retrieval and action recognition, while using a self-supervised objective.

Other work creates animated story-boards using the action descriptions of screen-

Query: Raj takes a drink of water while Penny walks in.

Candidate clips:

Figure 2.11: Example of visual retrieval on the TVR dataset[9]. Given a textual query, the goal is to retrieve the relevant video clip from a set of candidate clips.

plays (Ye and Baldwin, 2008), extracts social networks from screenplays (Agarwal et al., 2014a), or creates *xkcd* movie narrative charts (Agarwal et al., 2014b). Haurilet et al. (2016) and Sang et al. (2022) attempt to recognize characters in clips from TV episodes, and Kukleva et al. (2020) learn relationships and interactions between characters in movies. In more lengthy and complex videos, such as TV episodes, the majority of computer vision approaches also focus on low level analysis, namely recognizing and tracking characters in videos (Bauml et al., 2013; Bojanowski et al., 2013; Ramanathan et al., 2014; Sivic et al., 2009). On the other hand, Gu et al. (2018) mostly focus on recognizing *actions* in movie clips rather than relationships between characters, while applying CV-based approaches to the input video.

Finally, Bain et al. (2020) try to present a more holistic view on movie understanding by providing movie clips accompanied by textual descriptions and other metadata, such as bounding boxes for the participating characters, and the genre of the movie. This holistic approach is further extended by Huang et al. (2020), that provide full-length movies alongside trailers, posters, textual synopses, scripts, action recognition tags and character bounding boxes in the video frames. Unfortunately, although this dataset could have a large impact and facilitate future research, at the time of writing it is not publicly available.

Visual question answering and retrieval QA over narratives is also a popular task in the CV community, although the task formulation differs from NLP approaches. Most existing datasets and approaches focus on isolated movie/TV episode clips for creating and answering question-answer pairs, instead of considering the full-length

[8] https://tvqa.cs.unc.edu

[9] https://tvr.cs.unc.edu/tvc_explore.html

Captions:

- Sheldon is holding a large rock in his right hand. video-only
- Sheldon is talking to a rock that he is holding with his right hand and threatens to cast it far away. video-text
- Sheldon is holding a rock with his right hand and talking to it. video-only
- Sheldon talks to a rock outside while holding it in his hand. video-only

Figure 2.12: Example of video captioning on the TVC dataset[10](Lei et al., 2020b). Given a video clip and/or aligned subtitles, the goal is to generate a one-sentence textual description.

narrative (see example in Figure 2.10). Tapaswi et al. (2016) introduce a multimodal dataset (MovieQA) consisting of questions over movies, varying from the simple ”who” and ”where” to the more complicated ”why”. However, since the questions focus on isolated clips, the semantics in this case are simpler in comparison with text-based datasets such as NarrativeQA (Kočiskỳ et al., 2018). Next, Frermann et al. (2018) attempt to answer a single question, namely who is the perpetrator in episodes of the well-known crime series CSI, again based on multimodal information. Lei et al. (2018) also create question-answer pairs for video clips from TV episodes (TVQA) following a similar format to MovieQA, where they focus on short clips and localised events with simpler semantics (e.g., low-level actions, situations, and visual characteristics in the clips).

The TVQA dataset is also extended to address retrieval (TVR), where the aim is to identify video clips matching textual queries (Lei et al., 2020b) (see example in Figure 2.11). A similar dataset to TVR, for retrieving video clips based on textual queries is VIOLIN (Liu et al., 2020), that comprises of TV episodes and movie clips.

[10]https://tvr.cs.unc.edu/tvc_explore.html

Video captioning Another popular task for analyzing narrative content is via video captioning. In this case, most prior work considers a short video clip comprising of different modalities (i.e., video and language) as input and aims at generating short, single-sentence textual descriptions (see example in Figure 2.12). Rohrbach et al. (2015) introduce the Movie Description Dataset for video captioning on movie clips given multimodal input. Based on this dataset, there is also a Large Scale Movie Description Challenge[11], which focuses on different aspects of video captioning, such as identifying and mentioning the correct characters in the descriptions. More recently, the TVQA/TVR dataset has also been extended for the video captioning task (Lei et al., 2020b), where text-based, video-based, and multimodal-based descriptions are provided for the video clips contained in the original TVQA. Current methods to multimodal video captioning mostly focus on training strong language-and-vision encoders with massive pre-training (e.g., Li et al. 2020; Luo et al. 2020; Xu et al. 2021; Lei et al. 2020a; Li et al. 2021), while the decoder is typically shallow and under-trained. Although such approaches offer good performance for generating short descriptions, they cannot maintain fluency in long outputs with rich vocabulary. This is an important difference between video captioning of isolated clips and full-length summarization, where producing fluent multi-sentence text with rich semantics is more challenging.

Trailer generation Although movie summarization is not as popular in CV as in the NLP community, trailer generation has been addressed in prior work. Trailer generation can be viewed as a specific instantiation of movie summarization with some key differences. In contrast to summarization, the main goal of trailer generation is not to illustrate the storyline, but to identify attractive and stylistic shots that introduce the viewer to the story while concealing some important information (spoilers). The task of trailer generation is formulated as identifying a sequence of shots from the full-length movie that can be used in trailer creation (see example in Figure 2.13). Most traditional approaches to trailer generation focus on the affective content analysis of sequences of shots within the movie and are mainly based on superficial audiovisual features, such as the background music or the visual changes between sequential shots (Irie et al., 2010; Smeaton et al., 2006). Xu et al. (2015) also focus solely on the attractiveness of the generated trailers by proposing a graph-based attractiveness model for shot selection and Smith et al. (2017) publicly release a semi-automatic generated trailer as part of their publication. Specifically, they train a model on horror movies via audiovisual

[11] https://sites.google.com/site/describingmovies/

Figure 2.13: Example of the setup for trailer generation. Given a full-length feature film, the goal is identify a small subsequence of shots to be included in the trailer, that acts as a preview of the movie.

sentiment analysis and post-process the output via human involvement. Finally, Wang et al. (2020b) introduce a new trailer generation dataset, which unfortunately is not publicly available. They again focus solely on the visual modality and try to learn patterns for quantifying the "trailerness" of a movie shot. In comparison with all previous work, we generate trailers while using *all input modalities* and try to identify *specific and well-defined criteria* that can be used for selecting trailer shots inspired by theories of narrative structure and industrial trailer creation.

2.4 Summary of Chapter

Narratives provide an appropriate testbed for addressing a variety of different challenges: (1) multimodal input with complementary information, (2) long input both in terms of video frames and number of textual tokens, (3) data scarcity, since most existing and publicly available datasets comprise of a few hundred of samples, and (4) very complex semantics and interactions between characters and events (e.g., non-lineartities, intervening substories, etc.). However, most existing approaches that analyze narratives do not consider all challenges at once. In most cases, we either have text-only approaches that focus on the full-length narratives with complex semantics

and address tasks such as question-answering and summarization, or multimodal approaches that are limited to short input length and simpler semantics (e.g., isolated actions and local interactions).

In this thesis, we combine the two different directions for addressing the tasks of narrative summarization and trailer generation. We use *all input modalities* (i.e., video, audio, text), consider *full-length narratives* and address the task of narrative summarization both in a video-to-video setting (i.e., video summarization, trailer generation) and a video-to-text setting (i.e., multimodal abstractive summarization).

Chapter 3

Turning Point Identification: Task Definition and Dataset

In the previous chapter, we discussed about narratology and how it defines narrative structure in the context of movies and TV episodes. Moreover, we observed that prior work has focused on either character-centric approaches to narrative understanding (Bamman et al., 2013, 2014; Iyyer et al., 2016; Chaturvedi et al., 2017; Gorinski and Lapata, 2015) or fine-grained analysis of the structure via story grammars (Black and Wilensky, 1979) and chains of events (Chambers and Jurafsky, 2009).

In this chapter and in contrast to prior work, we are interested in a high-level analysis of narrative structure, as suggested first by Aristotle with the basic triangle-shaped plot structure (Pavis, 1998) and re-formulated later by the German novelist and playwright Gustav Freytag as a pyramid (Freytag, 1896). Specifically, we adopt a modern variant commonly employed by screenwriters as a practical guide for producing successful screenplays (Hague 2017; Chapter 2). According to this scheme, there are six stages (acts) in a film, namely *the setup*, *the new situation*, *progress*, *complications and higher stakes*, *the final push*, and *the aftermath*, separated by five *turning points* (TPs). TPs are narrative moments from which the plot goes in a different direction (Thompson, 1999), and by definition they occur at the junctions of acts. Aside from changing narrative direction, TPs define the movie's structure, tighten the pace, and prevent the narrative from drifting. The five TPs and their definitions (Hague, 2017) are given in Table 3.1.

We propose the task of turning point identification in movies as a means of analyzing their narrative structure. TP identification provides a sequence of key events in the story and segments the screenplay into thematic units. Common approaches to

Turning Point	Description
1. Opportunity	Introductory event, that occurs after the presentation of the setting and the background of the main characters.
2. Change of Plans	Event where the main goal of the story is defined. Upon this point, the action begins to increase.
3. Point of No Return	Event that pushes the main character(s) to fully commit to their goal.
4. Major Setback	Event where everything fall apart (temporarily or permanently).
5. Climax	Final event of the main story, moment of resolution and the "biggest spoiler".

Table 3.1: Turning points and their definitions according to Hague (2017).

summarization and QA of long or multiple documents (Chen et al., 2017; Yang et al., 2018; Kratzwald and Feuerriegel, 2018; Elgohary et al., 2018) include a retrieval system as the first step, which selects a subset of relevant passages for further processing. However, Mou et al. (2021) and Chen et al. (2022a) demonstrate that these approaches do not perform equally well for answering questions and summarizing long narratives, respectively, since individual passages are very similar and the same entities are referred to throughout the story. We argue that this challenge can be addressed by TP identification, which finds the most important events and segments the narrative into thematic units. Downstream processing for summarization or question answering can then focus on those segments that are relevant to the task. In the rest of the thesis, we specifically focus on the downstream task of summarization and finally use such content selection for producing abstractive textual summaries (Chapter 7).

Although, we ultimately want to use all input modalities (i.e., video, text, audio) for identifying narrative structure in a movie, in this chapter we only focus on the *textual modality* as a first step towards defining the task and establishing baseline methods on TP identification. In accordance with prior work (Gorinski and Lapata, 2015, 2018), we consider movie screenplays as textual input. Screenplays are a very rich source of information, since they are typically divided into scenes (i.e., complete semantic units involving a fixed set of characters, taking place in a specific location, and related to a specific topic) and contain the dialogue parts of the movie as well as character names and descriptions of actions, emotions and situations. An example of a screenplay scene is provided in Figure 3.1.

```
EXT.  WEST 83RD STREET - DAY

Race across a field of PEDESTRIANS to pick up three women
hurrying down the sidewalk.  LYDIA LYNCH, a real estate
broker, vaults down the sidewalk, she's got a hell of a
stride.  MEG ALTMAN, thirtyish, struggles to keep up with
her, she's tall, wafer-thin, pale as a ghost.  SARAH, a
nine year old girl, flat out runs to keep up, dribbling a
basketball as she goes.  The kid's athletic, much tougher
than Meg, who she resembles.

Lydia reads from a sheet she carries in her bouncing hands.

                    LYDIA
          -- seventeen feet wide, fifty-
          five feet deep, forty-two hundred
          square feet, four floors with a
          rentable basement apartment, so
          five altogether, courtyard in
          back --

                    MEG
          Could you slow down a little?
              (looking back over her
               shoulder)
          Or we could wait for the car...
```

Figure 3.1: Example of a screenplay scene from the movie “The Panic Room”. A scene contains the dialogue parts of a movie, but also character names (e.g., LYDIA, MEG) and descriptions of actions (e.g., looking back over her shoulder), emotions (e.g., pale as a ghost), and situations. A screenplay consists of 133 scenes and 23,000 words on average based on the statistics of our dataset (TRIPOD).

However, problematically for modeling purposes, TPs are latent in screenplays, there are no screenwriting conventions (like character cues or scene headings) to denote where TPs occur, and their exact manifestation varies across movies (depending on genre and length), although there are some rules of thumb indicating where to expect a TP (e.g., the Opportunity occurs after the first 10% of a screenplay, Change of Plans is approximately 25% in). To enable automatic TP identification, we develop a new dataset which consists of screenplays, plot synopses, and turning point annotations. To save annotation time and render the labeling task feasible, we collect TP annotations at the plot synopsis level (synopses are a few paragraphs long compared to screenplays which are on average 120 pages long). An example is given in Figure 3.2. We then project the TP annotations via distant supervision onto screenplays and propose an end-to-end neural network model which identifies TPs in full length screenplays.

The contributions of this chapter can be summarized as follows:

1. We introduce TP identification as a new task for the computational screenplay analysis that can benefit applications such as QA and summarization.

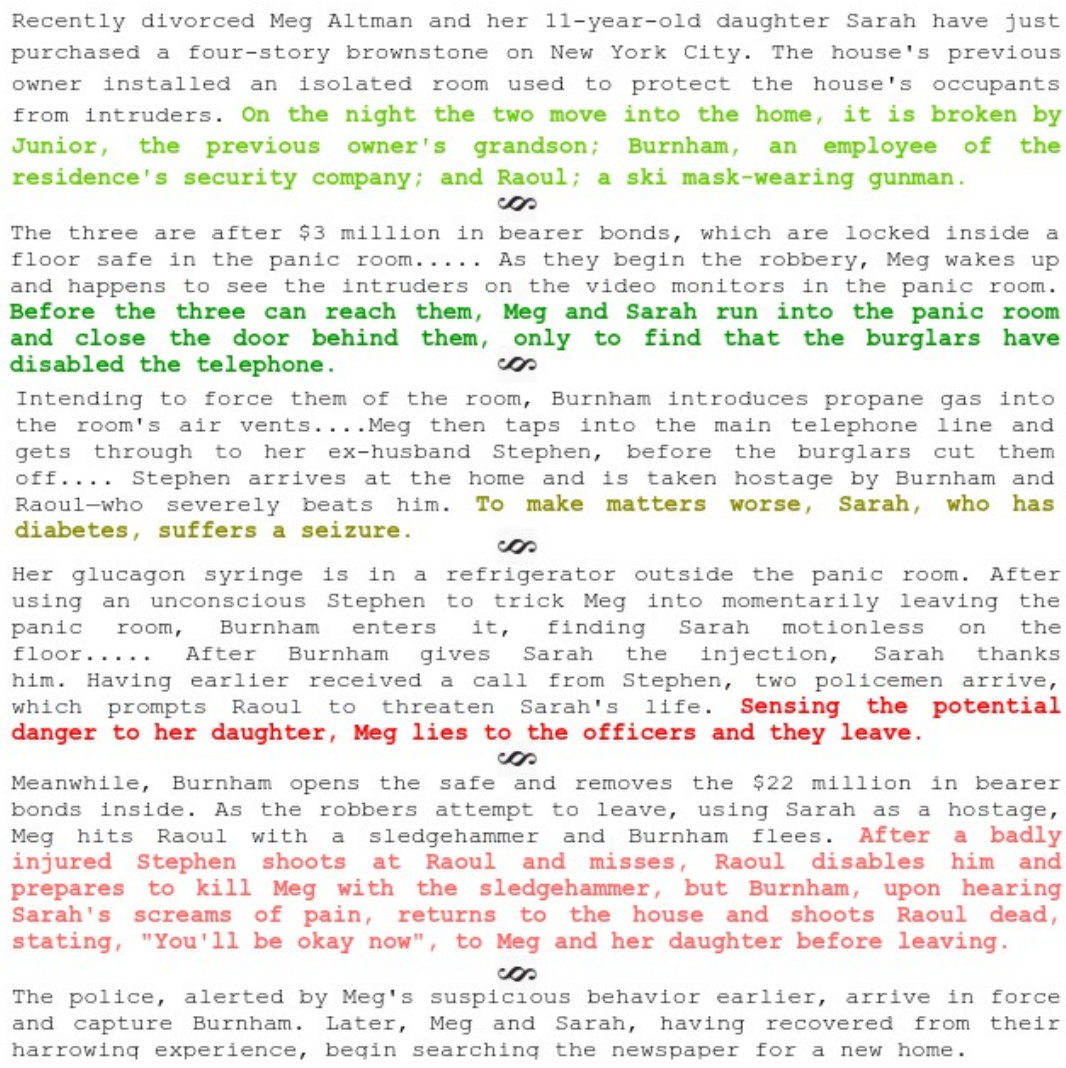


Figure 3.2: Example of turning point annotation (TP1, TP2, TP3, TP4, TP5, respectively) for the synopsis of the movie “Panic Room”.

2. We create and make publicly available the **T**u**R**n**I**ng **PO**int **D**ataset (TRIPOD) of 122 movie screenplays annotated with TPs[1]. We also collect and pre-process the corresponding full-length videos, which we will use in later chapters when addressing multimodal summarization.

3. We present an end-to-end neural network model that identifies turning points in plot synopses and projects them onto scenes in screenplays, outperforming strong baselines based on the expected position of TPs.

[1] https://github.com/ppapalampidi/TRIPOD

3.1 Related Work

Recent years have seen increased interest in the automatic analysis of long and complex narratives. Specifically, Machine Reading Comprehension (MRC) and Question Answering (QA) tasks are transitioning from investigating single short and clean articles or queries (Rajpurkar et al., 2016; Nguyen et al., 2016; Trischler et al., 2016) to large scale datasets that consist of complex stories (Tapaswi et al., 2016; Frermann et al., 2018; Kočiskỳ et al., 2018; Joshi et al., 2017; Lal et al., 2021; Angelidis et al., 2019; Mou et al., 2021; Chen et al., 2022a) or require reasoning across multiple documents (Welbl et al., 2018; Wang et al., 2018; Dua et al., 2019; Yang et al., 2018; Kwiatkowski et al., 2019; Clark et al., 2020). Tapaswi et al. (2016) and Lei et al. (2020b) introduce multi-modal datasets consisting of questions over movies and TV episodes, while Frermann et al. (2018) attempt to answer a single question, namely who is the perpetrator in 39 episodes of the well-known crime series CSI, again based on multi-modal information. Finally, Kočiskỳ et al. (2018) and Xu et al. (2022) recently introduced datasets consisting of question-answer pairs over movie screenplays, books, and stories.

Previous approaches have focused on fine-grained story analysis, such as inducing character types (Bamman et al., 2013, 2014) or understanding relationships between characters (Iyyer et al., 2016; Chaturvedi et al., 2017). Various approaches have also attempted to analyze the goal and structure of narratives. Black and Wilensky (1979) evaluate the functionality of story grammars in story understanding, Elson and McKeown (2009) develop a platform for representing and reasoning over narratives, and Chambers and Jurafsky (2009) learn fine-grained chains of events.

In the context of movie summarization, Gorinski and Lapata (2018) automatically generate an overview of the movie's genre, mood, and artistic style based on screenplay analysis. Gorinski and Lapata (2015) summarize full length screenplays by extracting an optimal chain of scenes via a graph-based approach centered around the characters of the movie. A similar approach has also been adopted by Vicol et al. (2018), who introduce the MovieGraphs dataset consisting of 51 movies and describe video clips with character-centered graphs. Finally, Chen et al. (2022a) introduce recently a long dialogue summarization dataset for producing abstractive summaries from transcripts of TV episodes. Other work creates animated story-boards using the action descriptions of screenplays (Ye and Baldwin, 2008), extracts social networks from screenplays (Agarwal et al., 2014a), or creates *xkcd* movie narrative charts (Agarwal et al., 2014b).

Our work also aims to analyze the narrative structure of movies, but we adopt a

high-level approach. We advocate TP identification as a precursor to more fine-grained analysis that unveils character attributes and their relationships. Our approach identifies key narrative events and segments the screenplay accordingly; we argue that this type of preprocessing is useful for applications which might perform question answering and summarization over screenplays. Although our experiments in this chapter focus solely on the textual modality, turning point analysis is also relevant for multimodal tasks such as trailer generation and video summarization, which we will address in Chapters 5 and 6.

3.2 TRIPOD Description

We created the TRIPOD dataset which consists of Wikipedia plot synopses, screenplays, IMDb casting lists and TP annotations at the plot synopsis level for 122 movies. The movies alongside with their info, i.e., synopses, screenplays, and IMDb casting lists, were selected from the Scriptbase dataset (Gorinski and Lapata, 2015), which contains movie screenplays and their metadata. Our movie selection was based on: (a) maintaining a variation across movie genres (e.g., action, romance, comedy, drama), and (b) including screenplays faithful to the movies and their corresponding plot synopses.

Our motivation for annotating the data at the plot synopsis level (coarse-grained), instead of at the screenplay level (fine-grained) is: (a) a decrease in annotation time, which will allow us to scale the process in the future, and (b) an increase in inter-annotator agreement, since broad descriptions (synopsis sentences) instead of fine-grained events (scenes) are annotated. As an annotation example, consider the plot synopsis of the movie “Panic Room” in Figure 3.2. Each TP is colored differently and both the chain of key events (overall colored text) and the resulting segmentation is illustrated.

During the first pilot study (PS1), the author and two additional annotators identified TPs in the plot synopses of the movies. The annotators selected exactly one synopsis sentence per TP, assuming that all TPs are present. During PS1, we devised annotation instructions and an annotation tool which presents the plot synopsis sentence by sentence. We provide the full annotation instructions and an example of the annotation interface in Appendix A. After annotating 30 movies, we recruited two new annotators for the rest of the dataset. They were trained using the annotation instructions and in a second pilot study (PS2) double-annotated five movies. The remaining movies in our dataset were then single annotated by the new annotators.

We examined inter-annotator agreement using two different metrics: (a) total agreement (Total Agr), i.e., the percentage of TPs that two annotators agree upon by selecting the exact same sentence; (b) annotation distance, i.e., the distance $d[p_i, tp_i]$ between the two annotations for a given TP, normalized by synopsis length:

$$d[p_i, tp_i] = \frac{1}{N}|p_i - tp_i| \tag{3.1}$$

where N is the number of synopsis sentences and tp_i and p_i are the sentence indices of TP i annotated by the two annotators. The mean annotation distance D is then computed by averaging the distances $d[p_i, tp_i]$ across all annotated TPs. We use two different agreement metrics in order to evaluate the exact agreement between annotators (i.e., Total Agr), but also the degree of their disagreement (i.e., annotation distance).

The Total Agr between the annotators in PS2 was 64.00% and the mean annotation distance was 4.30%, SD 3.43%. The annotation distance per TP is presented in Table 3.5 (last line), where it is compared with the automatic TP identification results (to be explained later). This indicates that annotators agree on the exact same sentence 64% of times. However, even in cases where the annotators do not agree, they still annotate neighboring sentences in the synopsis as representative of each TP.

Next, we asked annotators to annotate the screenplays (rather than synopses) of a subset of 38 movies from our dataset. This subset serves as our gold standard test set. The annotators were given plot synopses with TPs already annotated and were instructed to indicate for each TP which scenes in the screenplay correspond to it. Six of the 38 movies were double annotated, so that we could measure agreement. Since annotators were allowed to choose a variable number of scenes for each TP, this changes the agreement metrics:

Total Agreement (Total Agr) now is the percentage of TP scenes the annotators agree on:

$$\text{Total Agr} = \frac{1}{T \cdot V} \sum_{i=1}^{T \cdot V} \frac{|S_i \cap G_i|}{|S_i \cup G_i|} \tag{3.2}$$

where T is the number of TPs per screenplay, V is the number of screenplays and S_i and G_i are the indices of the scenes selected for TP i by the two annotators.

Partial Agreement (Part Agr) is the percentage of TPs where there is an overlap of

	Train	Test
movies/scenes	84/11,320	38/5,830
synopsis vocabulary	13.0k	6.8k
screenplay vocabulary	45.3k	28.3k
per plot synopsis		
tokens	729.8 (165.5/321/1122)	698.4 (187.4/345/1060)
sentences	35.4 (8.4/13/56)	33.9 (9.9/15/59)
sentence tokens	20.6 (9.5/1/80)	20.6 (9.3/1/82)
per movie		
scenes	133.0 (61.1/27/385)	153.4 (54.0/42/299)
sentences	3.0k (0.9/0.6/5.5)	2.9k (0.6/1.6/4.5)
tokens	23.0k (6.6/0.5/41.9)	21.5k (4.0/12.7/30.5)
video length (secs)	6.8k (1.1/4.2/10.3)	6.9k (1.3/3.5/10.2)
per scene		
sentences	22.2 (31.5/1/684)	19.0 (24.9/1/433)
tokens	173.0 (235.0/3/4875)	139.9 (177.5/5/2793)
sentence tokens	7.8 (6.0/1/220)	7.4 (6.0/1/106)
video length (secs)	88.1 (152.5/2/6317)	81.6 (114.8/2/1356)

Table 3.2: Statistics of TRIPOD dataset; means are shown with standard deviation/minimum/maximum in parentheses.

at least one scene between the two annotators[2]:

$$\text{Part Agr} = \frac{1}{T \cdot V} \sum_{i=1}^{T \cdot V} [S_i \cap G_i \neq \emptyset] \tag{3.3}$$

Annotation distance D then becomes the mean of distances $d[S_i, G_i]$ between the two annotators normalized by the screenplay length:

$$d[S_i, G_i] = \frac{1}{M} \min_{(s \in S_i, g \in G_i)} |s - g| \tag{3.4}$$

where M is the length of the screenplay.[3]

[2]We also compute Part Agr in addition to Total Agr for screenplays because *more than one* scene corresponds to a TP in comparison with the synopses, where *exactly one* sentence is representative of each TP.

[3]We compute the minimum distance between the two sets of scenes, since scenes further away from each other may be included in the same set. Hence, considering the center of the sets is not always representative of the TP scenes.

The Total Agr and Part Agr between the two annotators were 35.48% and 56.67%, respectively. The mean annotation distance was 1.48%, SD 2.93%. The Total Agr shows that the annotators rarely indicate the same scenes, even if they are asked to annotate an event in the screenplay that is described by a specific synopsis sentence. However, their scene annotations are close in the screenplay, as Part Agr and annotation distance reveal. This analysis validates our assumption that annotating the synopses first limits the degree of overall disagreement.

Finally, we also collected the full-length videos and corresponding subtitles for the TRIPOD movies[4]. Although we do not use multimodal information from the videos in this chapter, we will take advantage of the audiovisual information for the movies in Chapters 5 and 6. Table 3.2 presents the dataset statistics. We also provide further details about the dataset, such as the titles of the included movies and their distribution depending on genre, release year, and IMDb score, in Appendix B.

3.3 Automatic TP Identification

After defining the task and collecting the dataset with TP annotations, we examine Turning Point (TP) identification for three different settings. First, we explore how we can identify TPs directly in the plot synopses (Task 1; Section 3.3.1), which is much easier than considering the full-length screenplays and could provide us with a direct insight on the difficulty of the task. Next we automatically project the TP annotations from the synopses to the full-length screenplays in order to acquire fine-grained labels on screenplays (Task 2; Section 3.3.2). Finally, we explore how an end-to-end TP identification model, that first selects sentences from the synopsis (Task 1) and then projects them to the full-length screenplay (Task 2) performs on TP identification over screenplays (Task 3; Section 3.3.3).

Our method aims at:

1. Projecting TP annotations from synopses to screenplays automatically in order to acquire fine-grained labels in screenplays without additional human annotation.

2. Assessing the difficulty of the task, first at the synopsis-level, which is easier, and then at the screenplay-level, which is more challenging and closer to our main focus.

[4]We purchased and processed the DVDs for all TRIPOD movies.

3.3.1 Task 1: Automatic TP Identification in Plot Synopses

For the first task, the objective is to identify which sentences in the synopsis act as TPs. As the sequence, number, and labels of TPs are fixed (see Table 3.1), we treat TP identification as a binary classification problem with the labels "TP" and "no TP". An overview of the proposed models for this task is given in Figures 3.3 and 3.4.

3.3.1.1 Context-Aware Model (CAM)

We aim at contextualizing the synopsis sentences with respect to the whole synopsis. In order to identify whether an event, as described in a synopsis sentence, act as a TP, we need to encode the interaction of this event with other events described in the story. In the following, we describe ways of contextualizing the sentences with respect to the whole synopsis.

Sentence encoder We use a pre-trained sentence encoder (Devlin et al., 2019; Cer et al., 2018) in order to compute the semantic representation p of each sentence in the plot synopsis.

Synopsis encoder We employ a Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber 1997) network as the synopsis encoder, which takes as input the sentence representations of a synopsis and produces sentence representations $h_1, h_2, \ldots, h_N$, where h_i is the hidden state at time-step i, summarizing all the information of the synopsis up to the i-th sentence. We use a Bidirectional LSTM (BiLSTM) in order to get sentence representations that summarize the information from both directions. A BiLSTM consists of a forward LSTM $\overrightarrow{f}$ that reads the synopsis from p_1 to p_N and a backward LSTM $\overleftarrow{f}$ that reads it from p_N to p_1. We obtain the final representation cp_i for a given synopsis sentence p_i by concatenating the representations from both directions, $cp_i = h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}], \quad h_i \in \mathbb{R}^{2S}$, where $[.;.]$ denotes the concatenation operation and S the size of each LSTM.

Output layer We use the representation cp_i of the synopsis sentence i as feature vector for the classification task and feed it to a fully-connected layer with a single neuron, which outputs the probability that sentence i acts as a TP.

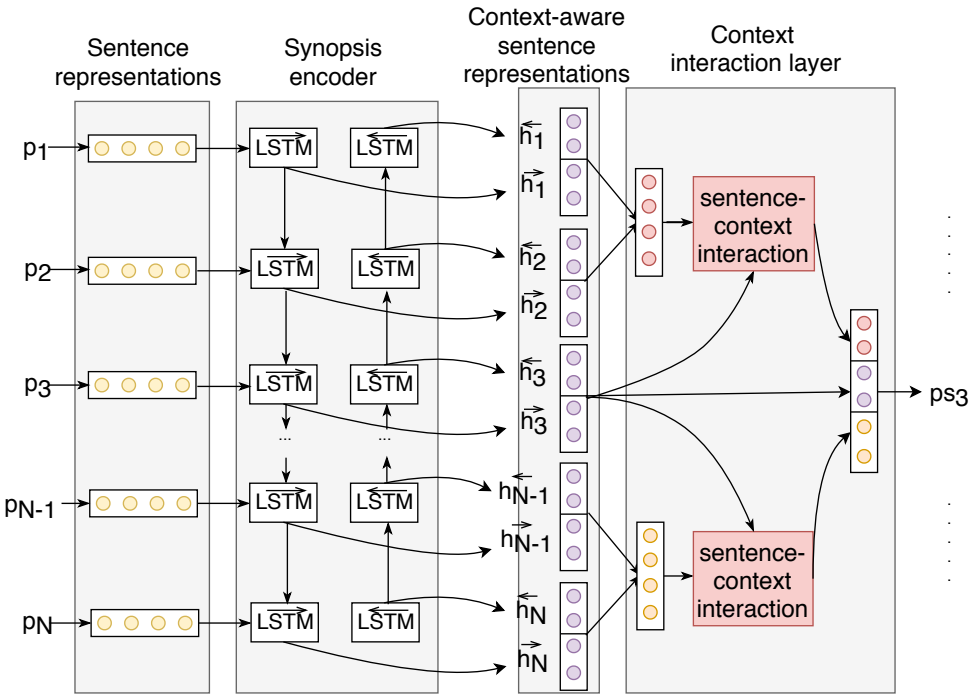


Figure 3.3: Overview of our models for Task 1. (a) Multi Context-Aware Model (MCAM): the synopsis sentence representations p_i are fed to the MCAM, they are contextualized via a synopsis encoder (BiLSTM layer) and after interacting with the left and right context windows in the context interaction layer, the final sentence representation ps_i is computed

3.3.1.2 Multi Context-Aware Model (MCAM)

TPs act by definition as *boundaries* between different semantic units of the movie. It therefore makes sense to enrich CAM with a context interaction layer (CIL), which calculates the similarity of the current sentence with the left and right context window, as illustrated in Figure 3.3. CIL is inspired by traditional segmentation approaches (Hearst, 1997) and measures the semantic similarity of the current sentence with a preceding and following context window in the synopsis. Hence, the final contextualized sentence representation encodes the degree to which each sentence can act as a topic boundary in the synopsis.

Context interaction layer (CIL) After calculating the contextualized sentence representations cp_i via the synopsis encoder, we compute the representation of the left context lc_i and the right context rc_i of the current sentence (right-most block in Figure 3.3). We select windows of fixed length l and calculate the lc_i and rc_i by averaging the sentence representations within each window. Next, we calculate the interaction (i.e., degree of similarity) of the current sentence with each of the context representa-

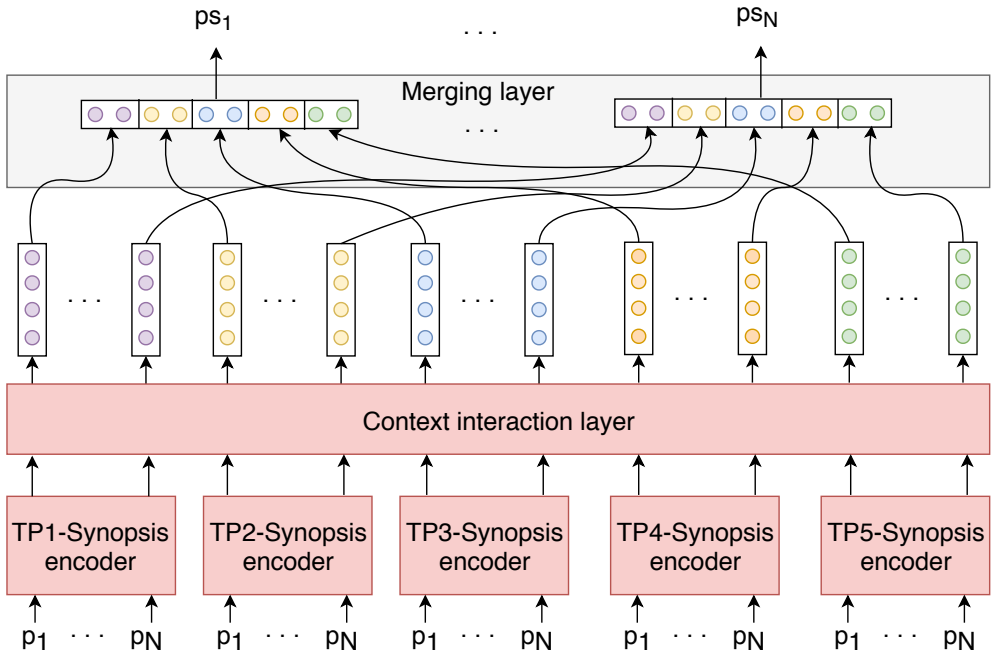


Figure 3.4: Overview of our models for Task 1. (b) TP-specific MCAM: five different synopsis encoders are utilized, one per TP, and these different views of a synopsis sentence p_i are combined in the merging layer.

tions in CIL:

$$b_i = cp_i \odot lc_i \qquad c_i = \frac{cp_i \cdot lc_i}{\|cp_i\| \|lc_i\|} \tag{3.5}$$

$$u_i = \frac{cp_i \cdot lc_i}{\max(\|cp_i\|_2 \cdot \|lc_i\|_2, \varepsilon)} \tag{3.6}$$

$$fl_i = [cp_i; lc_i; b_i; c_i; u_i], \tag{3.7}$$

where $\odot$ is the element-wise product and fl_i is the interaction representation of the sentence cp_i with the left context lc_i. The corresponding interaction representation fr_i for the right context rc_i is computed in the same fashion. We obtain the final representation of sentence i by concatenating cp_i and the interaction representations: $ps_i = [fl_i; fr_i; cp_i]$

3.3.1.3 TP-specific MCAM

Another variation of our model is to use TP-specific encoders instead of a single one (Figure 3.4). In this case, we employ five different encoders for calculating five different representations of the current synopsis sentence p_i, each one with respect to a specific TP ("TP-Synopsis encoder" blocks in Figure 3.4). These representations can

be considered multiple views of the same sentence. We calculate the interaction of each view with the left and right context window, as previously, via CIL.

Merging Layer (ML) Finally we compute the sentence representation ps_i by concatenating its individual context-enriched TP representations.

3.3.1.4 Entity Specific Information

We also enrich our model with entity-specific information. Entities are central in narratives and crucial for the development of the story (Jannidis, 2009; Frow, 2014). We hypothesize that adding entity-related representations to our model for the characters participating in the story will contribute to identifying key events, which naturally revolve around the protagonists of the story. For adding such information, we first apply co-reference resolution to the plot synopses using the Stanford CoreNLP toolkit (Manning et al., 2014) and substitute the mentions of the named entities if the entity is included in the IMDb cast list. Then we use a second sentence encoder to calculate entity-specific sentence representations.

Entity-specific (ES) encoder We use a word embedding layer to project the words $w_1, w_2, \ldots, w_T$ of the i^{th} synopsis sentence p_i to a continuous vector space R^E, where E is the size of the embedding layer. This layer is initialized with pre-trained entity embeddings. Next, we use a BiLSTM as described in the case of the synopsis encoder. On top of the LSTM, we add an attention mechanism, which assigns a weight a_i to each word representation h_i. We compute the entity-specific representation p_i^e of the i^{th} plot sentence as the weighted sum of all the word representations.

$$w_j^e = \tanh(W_h h_j + b_h), \quad e_j \in [-1, 1] \tag{3.8}$$

$$a_j = \frac{\exp(w_j^e)}{\sum_{t=1}^{T} \exp(w_t^e)}, \quad \sum_{j=1}^{T} a_j = 1 \tag{3.9}$$

$$p_i^e = \sum_{j=1}^{T} a_j h_j, \quad e \in R^{2S} \tag{3.10}$$

where W_h, b_h the attention layer's weights.

We compute the enriched synopsis sentence representation p_i' by concatenating the generic p_i and entity-specific p_i^e vectors: $p_i' = [p_i; p_i^e]$.

	TP1	TP2	TP3	TP4	TP5
theory	10.00	25.00	50.00	75.00	94.50
μ	11.39	31.86	50.65	74.15	89.43
σ	6.72	11.26	12.15	8.40	4.74

Table 3.3: Expected position of each TP based on screenwriting theory (theory) and its mean position μ (and SD σ) in the gold standard synopses of our training set.

3.3.2 Task 2: Coarse to Fine Projection of TP Labels

Here we assume we are given gold standard TP sentences from the synopsis and want to identify where the TPs are described in the screenplay. We consider the screenplay as a sequence of scenes (scene boundaries are already manually marked up in the screenplay). The scene is a suitable unit since it describes a self-contained event that takes place in one location, is about a specific topic, and includes some characters from beginning to end.

The objective of Task 2 is to identify the scenes of the screenplay that are semantically similar to the given TP sentences. We formulate this task as a binary classification problem, where a pair of TP sentence and scene is classified as either "relevant" or "irrelevant". We provide distant supervision by constructing noisy labels (Section 3.3.2.1). Our model is depicted in Figure 3.5 and described in detail in Section 3.3.2.2.

3.3.2.1 Noisy Screenplay-level Labels

Based on the screenwriting scheme of Hague (2017), we expect to find certain TPs in specific parts of a screenplay (e.g., the first TP often occurs after the first 10% of a screenplay; see the second row of Table 3.3). We can utilize this knowledge as a form of distant supervision. We calculate the mean position of each type of TP using the gold standard annotation of the plot synopses in our training set, normalized by the length of the synopses. The results are give in Table 3.3 (see μ and σ), together with the plot positions assumed by screenwriting theory. We observe that our estimates agree well with the theoretical predictions, but also that some of the TPs (e.g., TP2 and TP3) are more variable in their position than others (e.g., TP1 and TP5).

This leads us to the following *hypothesis:* Each TP is situated within a specific window in the screenplay. Scenes that lie within the window are semantically related to the specific TP, whereas all other scenes are unrelated to it.

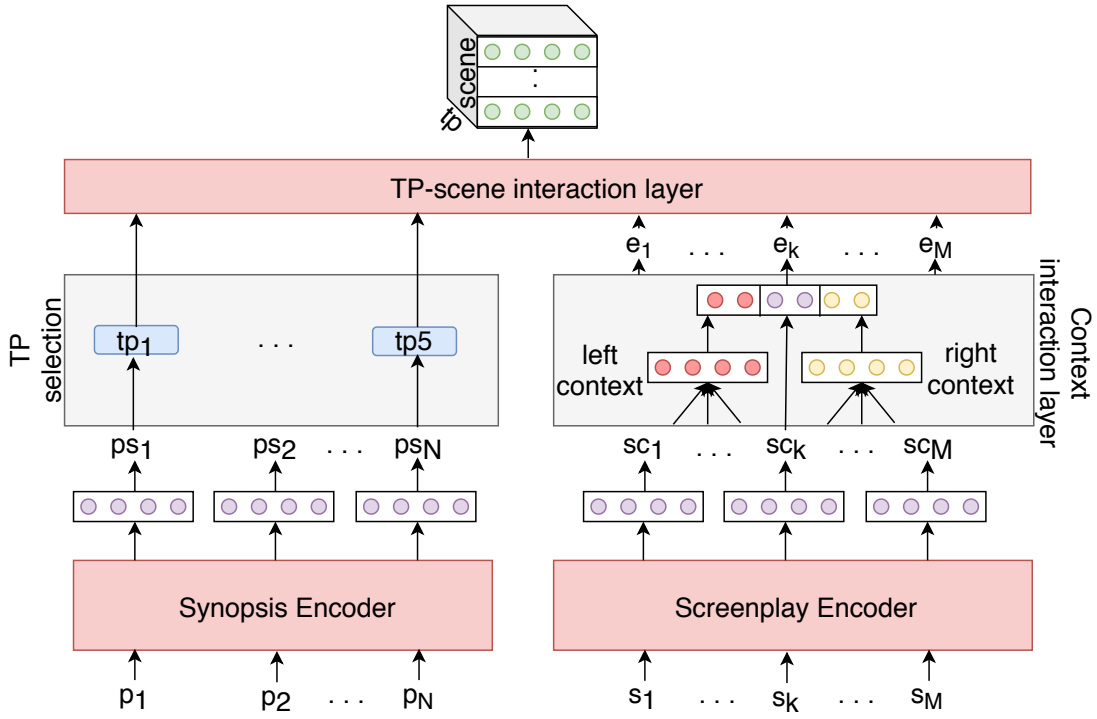


Figure 3.5: Overview of MCAM for Task 2: The synopsis and screenplay encoders contextualize the synopsis sentences p_i and scenes s_i of the screenplay, respectively. TPs are selected from the contextualized synopsis sentences ps_i and a richer representation sc_i is computed for s_i via the context interaction layer. The similarity between each sentence tp_i and each scene e_i is computed by the TP–scene interaction layer.

We can now assign noisy "relevant" and "irrelevant" labels to scenes in a screenplay with respect to a TP as follows: We assume that scenes that lie within the window $(\mu \pm \sigma)$ are relevant to this TP, whereas all other scenes are irrelevant, where μ and σ are estimated as in Table 3.3. Depending on the length of a screenplay, there might be an overlap between windows corresponding to different TPs (i.e., one scene might be deemed relevant to more than one TPs).

3.3.2.2 (Multi) Context-Aware Model

After creating the noisy labels as described in the previous section, we now propose a model for identifying scenes in the screenplay that are semantically similar to the annotated TP sentences from the synopsis. The model is depicted in Figure 3.5 and is trained based on the constructed noisy labels.

Encoder We again utilize a pre-trained *sentence encoder* for calculating sentence representations for plot synopses and screenplays. For the screenplay, in order to calculate the scene representation from the sequence of sentences that comprise it, we use a BiLSTM to contextualize the sentences of the scene and an attention mechanism in order to identify the most informative sentences in the scene (*scene encoder*), as described in Section 3.3.1. The final scene representation s is a weighted sum of the contextual representations of the scene sentences. Next, we utilize a *synopsis* and a *screenplay encoder*, each one in order to contextualize the synopsis sentences and scenes with information related to the whole synopsis or screenplay, respectively. The synopsis encoder is the same as in Task 1 (see Section 3.3.1.1). The screenplay encoder works in the same fashion, but has as its input the scene representations. Finally, we can optionally utilize an ES encoder (see Section 3.3.1.4) for the calculation of entity-specific representations for the synopsis and scene sentences. In this case, the generic and entity-specific representations are combined via concatenation.

TP–scene interaction layer After contextualizing all synopsis sentences via the synopsis encoder, we select only the representations of those that are marked as TPs for passing them to the TP–scene interaction layer (see "TP selection" block in the left part of Figure 3.5). In the case of the MCAM, we add information about the left and right context in the screenplay. Specifically, we compute the representations of the left lc_i and right rc_i context window of scene i in the screenplay as described in Section 3.3.1.2. Next, we compute the final representation e_i of scene i by concatenating the representations of the context windows lc_i and rc_i and the current scene sc_i: $e_i = [lc_i; sc_i; rc_i]$ (see "Context interaction layer" in the right part of Figure 3.5). We calculate the interaction representation t_{ij} between scene sc_i and TP tp_j via an interaction layer as the one described by Equations (3.5)–(3.7) (see "TP–scene interaction layer" in the top part of Figure 3.5).

Output layer We use the interaction representation t_{ij} as a feature vector for the classification task and feed it into a fully-connected layer with a single neuron, which outputs a probability of identifying scene i as relevant to TP j.

3.3.3 Task 3: End-to-end TP Identification

Our final task is to identify TPs in the full length screenplay without any gold standard information about their position in the plot synopsis. We address this with an end-to-

end model by combining the models for Tasks 1 and 2. We first predict the sentences that act as TPs in the synopsis using the best performing model for Task 1. We then feed these predictions to the MCAM model for Task 2 to identify TP scenes.

3.4 Experimental Setup

Pre-trained sentence encoder The performance of our models depends largely on the initial sentence representations (for both synopsis and screenplay), since Tasks 2 and 3 are distantly supervised. We experiment with using either the large BERT model (Devlin et al., 2019) or the Universal Sentence Encoder (USE; Cer et al. 2018) as the pre-trained sentence encoder in all tasks. Intuitively, we expect USE to be more suitable, since it is trained on textual similarity tasks which are relevant to ours. We validate our assumption via experiments in the development set. Specifically, for Task 2 the annotation distance D with respect to the screenplay length drops from 17.00% to 10.04% when we use the USE instead of the BERT model in the CAM version of our model for a smaller development set used during our preliminary experimentation.

Hyper-parameters Since the binary labels in both prediction tasks are imbalanced, we apply class weights to the loss function of our models. We weigh each class by its inverse frequency in the training set. We use the Adam algorithm (Kingma and Ba, 2015) for optimizing our networks. After experimentation, we chose an LSTM with 32 neurons (64 for the BiLSTM) for the synopsis encoder of Task 1 and one with 64 neurons for the synopsis and screenplay encoders of Tasks 2 and 3. For the context interaction layer, the window l is set to two sentences for Task 1 and 20% of screenplay length for Tasks 2 and 3. For the ES encoder, an embedding layer of size 300 is initialized with the Wikipedia2Vec pre-trained word embeddings (Yamada et al., 2018) and remains frozen during training. The LSTM of the encoder has 32 and 64 neurons for Tasks 1/2 and 3, respectively. Finally, we also add a dropout of 0.2. For developing our models we used PyTorch (Paszke et al., 2017).

Data augmentation When multiple annotations of the training set are available from the PS1 and PS2, conducted during the dataset creation, and considered as reliable, we use all of them during training. The reasons for this are: (a) it allows us to take into account the subjective nature of the task during training, and (b) we increase the size of our dataset, which contains only a limited number of movies. Specifically, 17 movies

are triple annotated, while 5 movies are double annotated. We add these annotations to our training data.

Inference During inference in Task 1, we select only one synopsis sentence per TP. Specifically, we select the five sentences with the highest posterior probabilities of being TPs and sequentially assign them TP labels based on their position. However, it is possible to have a cluster of neighboring sentences with high probability, even though they all belong to the same TP. We therefore also consider the expected position of each TP (Section 3.3.2.1) and select the sentence with the highest probability within a window (size $\mu_i \pm \sigma_i$) around this window (μ_i and σ_i were estimated from the training set, see Table 3.3).

For Tasks 2 and 3, we obtain a probability distribution over all scenes of the screenplay that indicates how relevant each scene is to each TP in the plot synopsis. We find the peak of each such distribution and select a neighborhood of relevant scenes around this peak as the TP-relevant ones. Based on the gold standard annotation, each TP corresponds to 1.77 relevant scenes on average (SD 1.23). We therefore consider a neighborhood of three relevant scenes per TP.

3.5 Experiments and Results

3.5.1 Task 1: TP Identification on Synopses

We split off a development set of 20 movies from the original training set. As for computing inter-annotation agreement, we use the evaluation metrics Total Agreement (Total Agr) and annotation distance D, normalized by synopsis length (Equation (3.1)). In Tables 3.4a and 3.4b, we report our experimental results for variants of our proposed model on the development and test sets, respectively. We also report the performance of two strong baselines on the test set: the theory baseline, i.e., we select the sentences that lie on the expected positions of TPs according to screenwriting theory, and the distribution baseline, i.e., we select the sentences that lie on the peaks of the empirical TP distributions in the training set (Section 3.3.2.1). For completeness, we also report the performance of a simple model based on state-of-the-art sentence representations, namely the vanilla model, on the development set. For the vanilla model, we again use the pre-trained encoder (BERT, USE) for computing synopsis sentence representations and we then feed the representations to a Multi-Layer Perceptron (MLP) in order to

	Total Agr ↑	D ↓
Vanilla model (BERT)	27.00	8.52
Vanilla model (USE)	29.00	8.13
CAM	33.00	7.44
MCAM	36.00	7.11
TP-specific MCAM	**39.00**	**6.52**
+ ES encoder	38.00	6.91

(a) Development set

	Total Agr ↑	D ↓
Random	2.63	35.02
Theory baseline	12.11	**8.35**
Distribution baseline	13.68	**8.38**
Vanilla model (USE)	21.05	9.42
TP-specific MCAM	28.95	9.01
+ ES encoder	**30.53**	8.84
Human agreement	*64.00*	*4.30*

(b) Test set

Table 3.4: Results for Task 1: Identification of TP sentences in the plot synopses. Evaluation metrics: mean Total Agreement (Total Agr) and annotation distance D, in percent. CAM stands for Context-Aware Model, MCAM: Multi Context-Aware Model, and ES encoder for entity-specific encoder.

perform binary sentence-level classification. For these experiments, we use the same settings as described in Section 3.4. We give more detailed results per TP in Table 3.5, compared against human agreement.

In Table 3.4a, we observe that the vanilla model presents the lowest performance among all variants of our model and according to all evaluation metrics. This observation suggests that the TP identification task requires information about the context of the synopsis and cannot be addressed only based on state-of-the-art sentence representations. Indeed, when using the synopsis encoder (BiLSTM) in the case of CAM, we observe an absolute increase of 4.00% in performance with respect to the Total Agr metric. Moreover, by adding the context interaction layer in MCAM we gain an absolute improvement in Total Agr of 3.00% compared to CAM. This indicates that the BiLSTM layer used in CAM does not provide sufficient information and we can compute more accurate contextualized sentence representations via the context interaction layer. Finally, the combination of different views of the same synopsis sentence using TP-specific encoders improves performance by a further 3.00%, reaching 39.00% Total Agr and reducing D to 6.52%. On the test set (see Table 3.4b), the TP-specific MCAM achieves 28.95% Total Agr compared to 13.68% for the distribution baseline and 21.05% for the vanilla model. Finally, although entity-specific information does not offer a benefit on the development set, we observe a 1.58% improvement in terms of Total Agr on the test set.

MCAM	TP1 ↓	TP2 ↓	TP3 ↓	TP4 ↓	TP5 ↓
+ TP views	**6.36**	**10.12**	**12.01**	9.89	6.67
+ entities	7.33	10.60	**12.14**	**9.07**	**5.08**
Human agreement	3.33	5.00	10.58	1.07	1.53

Table 3.5: Mean annotation distance D (test set) for different variations of the Multi Context-Aware Model (MCAM); results are shown per TP on the synopsis identification task.

When examining the performance of our model per TP and comparing it against human agreement in Table 3.5, we observe that they share a similar behavior: TPs 1, 4 and 5 are the easiest to distinguish, whereas TPs 2 and 3 are hardest and frequently placed at different points in the plot. Presumably, TPs 2 and 3 present the highest variance depending on the movie and the agreement is therefore lower (also note that we find the largest standard deviation for those TPs across movies in the annotated synopses according to Table 3.3).

We also conducted a human evaluation experiment on Amazon Mechanical Turk (AMT) in order to compare the performance of our model (MCAM) against the gold standard annotations (gold standard) and the distribution baseline (baseline). Specifically, for 15 of the movies of the test set, we formed "sets of highlights" of 5 sentences for each model/method and asked AMT workers to first read the whole synopsis of the respective movie and then rank these sets from best to worst based on the following criteria: (1) the quality of the plotline that they form, (2) whether they include the most important events and plot twists of the movie and (3) whether they provide some description of both the initial events and ending of the story. We provide details of the annotation instructions and interface in Appendix C.1. In Figure 3.6 we present the results of the human evaluation by plotting how often each set of highlights was ranked in each position, where a lower position indicates a better performance for the respective system. Overall, the average ranking positions for the gold standard, MCAM and distribution baseline are 1.87, 1.98 and 2.16, respectively.

Moreover, the human evaluation, presented in Figure 3.6, validates our observations. Specifically, as expected the gold standard sets of highlights are selected most often as the best plotline. Our model is most often ranked in the second place, whereas the distribution baseline is the least preferred set of highlights, even though it seems competitive with our model based on the distance evaluation metric. The difference in

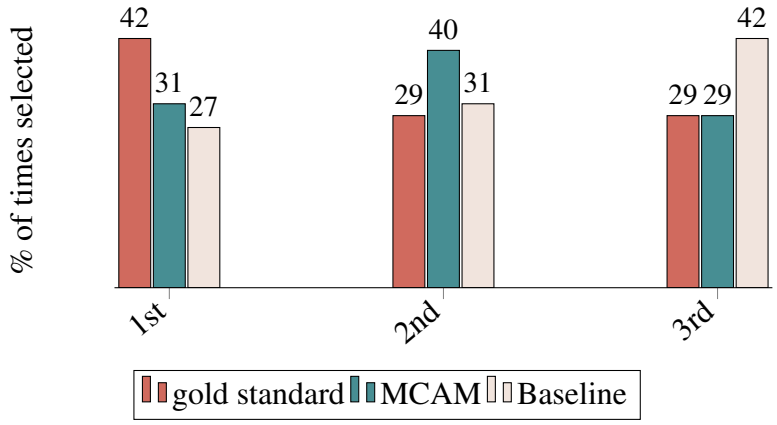


Figure 3.6: Human evaluation results for Task 1 on the test set. The performance of the gold standard annotations, the distribution baseline (based on the expected position of TPs) and our model (multi context-aware model; MCAM) is compared. X-axis: ranking place, y-axis: % of times that the system's outputs ranked at the respective place.

the average ranking positions between MCAM and the distribution baseline (1.98 vs. 2.16) verifies that the distribution baseline may indicate synopsis sentences that are on average close to the groundtruth TP ones but they are often semantically irrelevant.

3.5.2 Tasks 2 and 3: Annotation Projection and TP Identification on Screenplays

We perform five-fold crossvalidation over our original gold standard set to obtain a test-development split (recall we do not have gold standard annotations for training). Again we use the same evaluation metrics as for computing inter-annotator agreement (Section 3.2): Total Agreement (Total Agr), Partial Agreement (Part Agr) and annotation distance D, normalized by screenplay length (Equations (3.2)–(3.4)). Table 3.6 gives the crossvalidation results for our models and compares them against the same baselines as in Task 1. For this task, the vanilla model is altered as follows: we again compute the synopsis sentence and screenplay scene representations via the pre-trained encoder (USE) and use the scene encoder (see Section 3.3.2.2) in order to calculate the scene representations from the sequences of sentences that comprise them. Finally, we select the TP synopsis sentences (TP selection) and utilize the TP–scene interaction layer, as described in Section 3.3.2.2, in order to compute the feature vector that is fed into a fully-connected layer with a single neuron for the binary scene-level classifica-

	Total Agr ↑	Part Agr ↑	D ↓
Theory baseline	4.41	6.32	11.03
Distribution baseline	5.59	7.37	10.74
tf*idf similarity	1.18	2.11	26.42
tf*idf + distribution	2.06	3.68	14.40
Vanilla model	2.65	3.68	34.75
CAM	7.06	10.00	10.33
+ ES encoder	**8.53**	**12.63**	13.74
MCAM	7.94	11.58	**9.61**
+ ES encoder	7.35	11.58	10.92
MCAM End2end	6.76	8.42	10.98
Human agreement	*35.48*	*56.67*	*1.48*

Table 3.6: Results for Tasks 2 and 3 for five-fold crossvalidation (test/dev split) over our gold standard dataset. Evaluation metrics: Total Agreement (Total Agr), Partial Agreement (Part Agr), and mean annotation distance D, in percent. The vanilla model matches scenes to synopsis sentences without any synopsis- and screenplay-level contextualization of the sentences and scenes, respectively, CAM is the context-aware model, MCAM is the multi context-aware model and ES stands for an extra entity-specific encoder.

tion task.

For these tasks, we also test the performance of an IR baseline commonly used for retrieving relevant passages based on a query: the tf*idf similarity baseline. For this baseline, we consider the TP synopsis sentences as queries and search, in accordance with our experimental setup, for a neighborhood of three scenes in the screenplay that are semantically similar to each query. Specifically, we calculate the tf*idf similarity between the given query and the scenes of the script by computing the maximum similarity between the query and each sentence included in the scene. Finally, we also combine the tf*idf baseline with the position-related baselines by selecting scenes based on tf*idf similarity only within the windows determined by the position distributions ($\mu \pm \sigma$) for each TP. Based on the evaluation results we notice that the tf*idf baseline selects scenes in completely different sections of the screenplay, a problem that is reduced when we restrict the regions of selection in the screenplay. Overall, similar vocabulary across scenes and a lot of mentions to the same entities throughout

	TP1 ↓	TP2 ↓	TP3 ↓	TP4 ↓	TP5 ↓
CAM	5.63	10.14	16.82	12.86	6.24
+ entities	6.55	13.13	22.99	17.21	8.85
MCAM	**5.07**	**8.87**	**16.45**	**12.32**	**5.38**
+ entities	7.41	10.79	17.51	**12.32**	6.59
Human agreement	0.14	2.41	2.64	1.64	0.57

Table 3.7: Mean annotation distance D (test set); results are shown per TP for Task 2 (i.e., projection of TP labels to screenplays). CAM stands for the context-aware model and MCAM for the multi context-aware model.

the screenplay make tf*idf approaches insufficient for our tasks. We also experimented with common segmentation (i.e., TextTiling) approaches as baselines, but we do not report them in Table 3.6 due to poor performance.

In Table 3.6, we observe that the performance of the vanilla model is poor especially with respect to the average distance metric (34.75%), meaning that this model is not even able to indicate the region of the screenplay that is semantically related to a given TP. That suggests that state-of-the-art sentence representations are not enough for addressing the TP identification task and information related to the context of the screenplay is also required. Indeed, when we add the synopsis and screenplay encoders in the case of CAM, we notice a drop of 24.42% (absolute difference) in the average distance. Moreover, we observe that the context interaction layer reduces D to 9.61% compared to 10.33% for CAM, validating our assumption that TPs are not just isolated key events, but also mark boundaries between semantic sections. Finally, we observe that for the CAM version of our model, the entity-specific information improves Total Agr and Part Agr but increases mean annotation distance.

Based on the results of Tasks 1 and 2, we conclude that entity-specific encoders enhance our models with information that can either help identifying TP sentences/scenes or point to irrelevant regions in the synopsis/screenplay. Entity-specific information therefore seems to be too unreliable overall (i.e., entity-related representations boost performance for CAM based on Total Agr and Partial Agr but they significantly increase the annotation distance and do not offer any advantage for the MCAM version of our model), and we do not include it in our end-to-end experiments (Task 3). Specifically, Table 3.6 shows that the performance of the end-to-end MCAM model drops in comparison to the same model using gold standard synopsis annotations. However, it still remains competitive with the baselines.

Finally, we again present the mean annotation distance per TP for CAM and MCAM with or without entity-related information for the annotation projection task (i.e., Task 2) in Table 3.7. In accordance with Table 3.5 and Task 1, TPs 2, 3, and 4 are the hardest to identify on average, whereas the first and last TP present the lowest average distance. This indicates that given a sentence from the synopsis, it is easier to correctly identify which region in the screenplay describes it, if it is an introductory or concluding event of the story. Events in the middle of screenplay are presumably more semantically similar and hence, difficult to distinguish.

3.5.3 Discussion

In Figure 3.7, we visualize the posterior distribution of the models for Task 2 over the scenes of the screenplay for "Juno". The first panel shows the distribution baseline (Section 3.3.2.1) alongside the gold standard TP scenes for each TP (vertical lines). We observe that the distribution baseline provides a good approximation of the relevant TP position (which validates its use in the construction of noisy labels, Section 3.3.2.1), even though is not always accurate. For example, TPs 1 and 3 not only fail to appear at the peak of their respective distributions, but also lie outside the expected window in "Juno". The second and third panels present the distributions of the computed tf*idf similarity scores per TP when all scenes and only the scenes included in the windows determined by the distribution baseline are considered, respectively. For the simple tf*idf baseline, we observe that scenes located in entirely different parts of the screenplay present high similarity scores with respect to a given TP. This observation validates our initial assumption that since the tf*idf baseline is based on superficial textual clues, it cannot provide meaningful TP scene predictions and is dominated by the co-ocurrence of entities and key words that appear throughout the screenplay. When we combine the tf*idf similarity with the distribution baseline, we notice that we are able to alleviate this behavior. However, the performance of this system is again poor, since even though there are cases that it predicts some gold standard TP scenes (see TPs 4 and 5 in the illustrated example), it fails to provide good predictions when the position-based window is not accurate, leading to larger overall errors in comparison with the distribution baseline (13.33% average distance in comparison with 10.84% for the position-based baseline).

In the next panel we present the distributions predicted by the vanilla model. We again notice a similar behavior as in the simple tf*idf baseline: scenes in totally differ-

ent regions of the screenplay are selected as TP ones, since the scene representations are not contextualized with respect to the whole screenplay. In the last two panels we present the distributions predicted by CAM and MCAM. When adding the synopsis and screenplay encoders in the case of CAM, the distributions become smoother with higher probabilities of selecting TP scenes inside a distinct region in the screenplay. This observation indicates that contextualizing the scenes with respect to the whole screenplay is critical in order to acquire more accurate predictions. We also observe that the context interaction layer in the case of MCAM gives rise to even more accurate estimation of the TP distributions, with sharper peaks and higher confidence. Specifically, the maximum posterior probability of CAM is 0.7, while for MCAM it is close to one. This differentiation in the behavior of two variants of our model is consistent with our assumption that the TPs act as boundaries between different semantic sections.

Moreover, as mentioned in Section 3.5.1, we conduct a human evaluation experiment, where highlights are extracted by combining the five sentences labeled as TPs the synopsis. In Tables 3.8, 3.9, and 3.10, we present the highlights presented to the AMT workers for the movies ”Juno”, ”Panic Room”, and ”The Shining”, respectively. For each movie we show the gold standard annotations alongside with the predicted TPs for MCAM (+ TP views) and the distribution baseline, which is the strongest performing baseline with respect to the automatic evaluation results.

Overall, we observe that gold standard highlights describe the plotline of the movie, contain a first introductory sentence, some major and intense events, and a last sentence that describes the ending of the story. The distribution baseline is able to predict a few gold standard TPs by only considering the relative position of the sentences in the synopsis. This observation validates the screenwriting theory: TPs, or more generally important events that determine the progression of the plot, are consistently distributed in specific parts of a movie. However, when the distribution baseline cannot predict the exact TP sentence, it might select one that describes irrelevant events of minor importance (e.g., TP4 for ”Panic Room” is a detail about a secondary character instead of a major setback and highly intense event in the movie).

Finally, our own model seems to be able to predict some gold standard TP sentences, as demonstrated during the automatic evaluation. However, we also observe here that even when it does not select the gold standard TPs, the predicted ones describe important events in the movie that have some desired characteristics. In particular, for the movie ”Juno” the climax (TP5) is the moment of resolution, where Vanessa decides to adopt the baby after all the setbacks and obstacles. Even though our model

does not predict this sentence, it does select one that reveals information about the ending of the movie. An other such example is the movie ”Panic Room”, where the point of no return (TP3) is not correctly predicted, but the selected sentence refers to the same event.

3.6 Summary of Chapter

In this chapter we laid the foundations of how we define narrative structure in movies. Given screenwriting theory and modern methods that screenwriters use for composing their plays, we define turning points in movies as a means of identifying key events and segmenting the narrative into sections. We provide a dataset with TP annotations over movie synopses and propose a method for projecting them into screenplay scenes. Finally, we propose an end-to-end network for TP identification that considers both the synopses and the screenplays as a baseline for assessing the difficulty of the task.

The task of TP identification that we propose aims at facilitating downstream narrative understanding tasks, such as narrative summarization. We hypothesize that by identifying key events and segmenting the narrative into thematic units we can better summarize narratives, such as movies and TV shows. In the following chapter, we will closely examine this hypothesis by addressing screenplay summarization on another, out-of-domain dataset. Our goal is to validate that (a) narrative structure information can facilitate screenplay summarization, and (b) TPs are a general enough scheme and transferrable to other types of narratives.

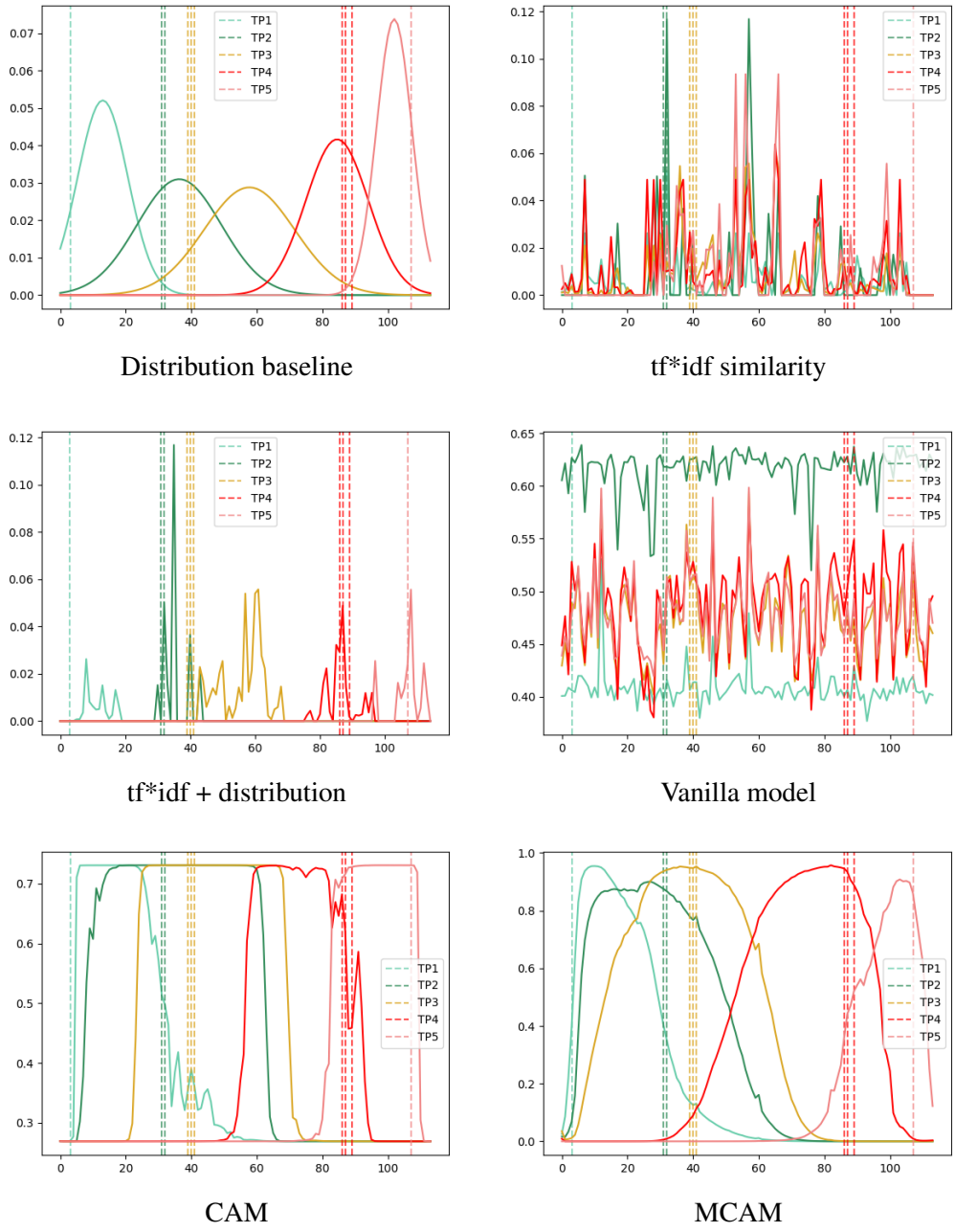


Figure 3.7: Probability distributions over the scenes of the screenplay for the movie “Juno”. X-axis: scene indices, y-axis: probability that the scene is relevant to a specific TP. Vertical dashed lines are gold standard TP-scenes. The distribution baseline identifies TPs based on their expected position in a movie, tf*idf is a standard information retrieval approach, the vanilla model matches scenes to synopsis sentences without any screenplay-level contextualization of the scenes, CAM is the context-aware model, and MCAM is the multi context-aware model.

gold standard
• Sixteen-year-old Minnesota high-schooler Juno MacGuff discovers she is pregnant with a child fathered by her friend and longtime admirer, Paulie Bleeker.
• All of this decides her against abortion, and she decides to give the baby up for adoption.
• With Mac, Juno meets the couple, Mark and Vanessa Loring (Jason Bateman and Jennifer Garner), in their expensive home and agrees to a closed adoption.
• Juno watches the Loring marriage fall apart, then drives away and breaks down in tears by the side of the road.
• Vanessa comes to the hospital where she joyfully claims the newborn boy as a single adoptive mother.
MCAM (+ TP views)
• Going to a local clinic run by a women's group, she encounters outside a school mate who is holding a rather pathetic one-person Pro-Life vigil.
• With Mac, Juno meets the couple, Mark and Vanessa Loring (Jason Bateman and Jennifer Garner), in their expensive home and agrees to a closed adoption.
• Juno and Leah happen to see Vanessa in a shopping mall being completely at ease with a child, and Juno encourages Vanessa to talk to her baby in the womb, where it obligingly kicks for her.
• Juno watches the Loring marriage fall apart, then drives away and breaks down in tears by the side of the road.
• The film ends in the summertime with Juno and Paulie playing guitar and singing together, followed by a kiss.
Distribution baseline
• Once inside, however, Juno is alienated by the clinic staff's authoritarian and bureaucratic attitudes.
• Juno visits Mark a few times, with whom she shares tastes in punk rock and horror films.
• Not long before her baby is due, Juno is again visiting Mark when their interaction becomes emotional.
• Juno then tells Paulie she loves him, and Paulie's actions make it clear her feelings are very much reciprocated.
• Vanessa comes to the hospital where she joyfully claims the newborn boy as a single adoptive mother.

Table 3.8: Highlights for the movie "Juno": gold standard annotations and predicted TPs for the multi context-aware model (MCAM + TP views) and distribution baseline. Correctly predicted highlights are marked with green.

gold standard
• On the night the two move into the home, it is broken into by Junior, the previous owner's grandson; Burnham, an employee of the residence's security company; and Raoul, a ski mask-wearing gunman recruited by Junior. • Before the three can reach them, Meg and Sarah run into the panic room and close the door behind them, only to find that the burglars have disabled the telephone. • To make matters worse, Sarah, who has diabetes, suffers a seizure. • Sensing the potential danger to her daughter, Meg lies to the officers and they leave. • After a badly injured Stephen shoots at Raoul and misses, Raoul disables him and prepares to kill Meg with the sledgehammer, but Burnham, upon hearing Sarah's screams of pain, returns to the house and shoots Raoul dead, stating, "You'll be okay now", to Meg and her daughter before leaving.
MCAM (+ TP views)
• On the night the two move into the home, it is broken into by Junior, the previous owner's grandson; Burnham, an employee of the residence's security company; and Raoul, a ski mask-wearing gunman recruited by Junior. • Before the three can reach them, Meg and Sarah run into the panic room and close the door behind them, only to find that the burglars have disabled the telephone. • Her emergency glucagon syringe is in a refrigerator outside the panic room. • As Meg throws the syringe into the panic room, Burnham frantically locks himself, Raoul, and Sarah inside, crushing Raoul's hand in the sliding steel door. • After a badly injured Stephen shoots at Raoul and misses, Raoul disables him and prepares to kill Meg with the sledgehammer, but Burnham, upon hearing Sarah's screams of pain, returns to the house and shoots Raoul dead, stating, "You'll be okay now", to Meg and her daughter before leaving.
Distribution baseline
• On the night the two move into the home, it is broken into by Junior, the previous owner's grandson; Burnham, an employee of the residence's security company; and Raoul, a ski mask-wearing gunman recruited by Junior. • Unable to seal the vents, Meg ignites the gas while she and Sarah cover themselves with fireproof blankets, causing an explosion which vents into the room outside and causes a fire, injuring Junior. • To make matters worse, Sarah, who has diabetes, suffers a seizure. • While doing so, he tells Sarah he did not want this, and the only reason he agreed to participate was to give his own child a better life. • As the robbers attempt to leave, using Sarah as a hostage, Meg hits Raoul with a sledgehammer and Burnham flees.

Table 3.9: Highlights for the movie "Panic Room": gold standard annotations and the predicted TPs for the multi context-aware model (MCAM + TP views) and distribution baseline. Correctly predicted highlights are marked with green.

gold standard
• Manager Stuart Ullman warns him that a previous caretaker developed cabin fever and killed his family and himself.
• Hallorann tells Danny that the hotel itself has a ”shine” to it along with many memories, not all of which are good.
• After she awakens him, he says he dreamed that he had killed her and Danny.
• Jack begins to chop through the door leading to his family’s living quarters with a fire axe.
• Wendy and Danny escape in Hallorann’s snowcat, while Jack freezes to death in the hedge maze.
MCAM (+TP views)
• Jack’s wife, Wendy, tells a visiting doctor that Danny has an imaginary friend named Tony, and that Jack has given up drinking because he had hurt Danny’s arm following a binge.
• Hallorann tells Danny that the hotel itself has a ”shine” to it along with many memories, not all of which are good.
• Danny starts calling out ”redrum” frantically and goes into a trance, now referring to himself as ”Tony”.
• When Wendy sees this in the bedroom mirror, the letters spell out ”MURDER”.
• Wendy and Danny escape in Hallorann’s snowcat, while Jack freezes to death in the hedge maze.
Distribution baseline
• Jack’s wife, Wendy, tells a visiting doctor that Danny has an imaginary friend named Tony, and that Jack has given up drinking because he had hurt Danny’s arm following a binge.
• Jack, increasingly frustrated, starts acting strangely and becomes prone to violent outbursts.
• Jack investigates Room 237, where he encounters the ghost of a dead woman, but tells Wendy he saw nothing.
• When Wendy sees this in the bedroom mirror, the letters spell out ”MURDER”.
• He kills Hallorann in the lobby and pursues Danny into the hedge maze.

Table 3.10: Highlights for the movie ”The Shining”: gold standard annotations and the predicted TPs for the multi context-aware model (MCAM + TP views) and distribution baseline. Correctly predicted highlights are marked with green.

Chapter 4

Screenplay Summarization Using Latent Narrative Structure

In the previous chapter, we developed a dataset and model for identifying the narrative structure of Hollywood movies, which we define in terms of turning points (TPs) based on screenwriting theory (Hague, 2017). Moreover, we hypothesized that knowledge about narrative structure can facilitate downstream narrative understanding tasks, such as question answering and summarization. In this chapter, we focus on extractive screenplay *summarization* and examine whether information about the narrative structure can facilitate this task.

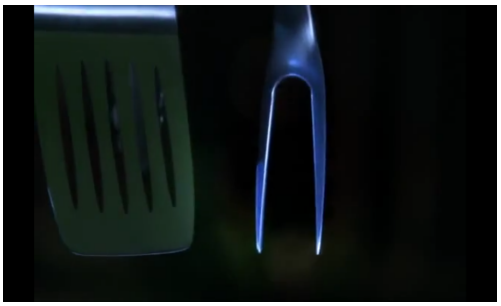
If the definition of TPs given by Hague (2017) and described in the previous chapter is strictly movie-specific and not transferable to different types of narratives, their usability will be limited. For evaluating whether TPs can transfer to different narrative types, we now focus on TV episodes from the well-known television program "*CSI: Crime Scene Investigation*" (Frermann et al., 2018) which revolves around a team of forensic investigators solving criminal cases. Each episode has a complex but well-defined structure: it opens with a crime, the crime scene is examined, the victim is identified, suspects are introduced, forensic clues are gathered, suspects are investigated, and finally the case is solved. We illustrate the structure of a CSI episode in Figure 4.1. This rigid structure will allow us to evaluate (1) whether the general definition of TPs can adapt to this specific scheme and (2) whether such information overall contributes to identifying important summary content.

As in the previous chapter, we only consider *full-length screenplays* for identifying text-only summary content. Moreover, we introduce a number of assumptions to make the task feasible. Firstly, our goal is to produce *informative* summaries, which serve

Crime scene: Investigators arrive to a fountain, where a dead body was found.

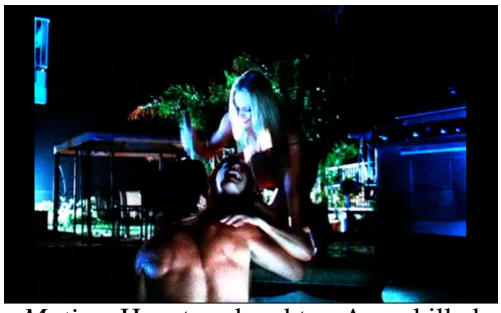
Victim: They identify the victim Vanessa Keaton.

Cause of death: They determine that Vanessa died from drowning.



Evidence: Investigators gather evidence for determining the perpatrator.

Perpetrator: They examine suspects, including her husband and step-daughter, until they conclude that the perpetrator is her step-daughter.



Motive: Her step-daughter, Amy, killed Vanessa out of jealousy when she saw her kissing another man named Tom, during a pool party.

Figure 4.1: Example of the structure of a "CSI: Crime Scene Investigation" episode. Here we present "Swap Meet" (Season 5, Episode 5) and illustrate snapshots from all important aspects.

as a surrogate to reading the full script or watching an entire TV episode. Secondly, we follow Gorinski and Lapata (2015) in conceptualizing screenplay summarization as the task of identifying a sequence of informative scenes.

We adapt general-purpose supervised and unsupervised extractive summarization algorithms (Nallapati et al., 2017; Zheng and Lapata, 2019) to identify informative scenes in screenplays and instill in them knowledge about narrative structure (Hague 2017; Cutting 2016; Freytag 1896; Chapter 3). In Figure 4.2, TPs are highlighted for a CSI episode. Instead of collecting new TP annotations for this dataset, we approximate narrative structure automatically by pre-training on the annotations of the TRIPOD

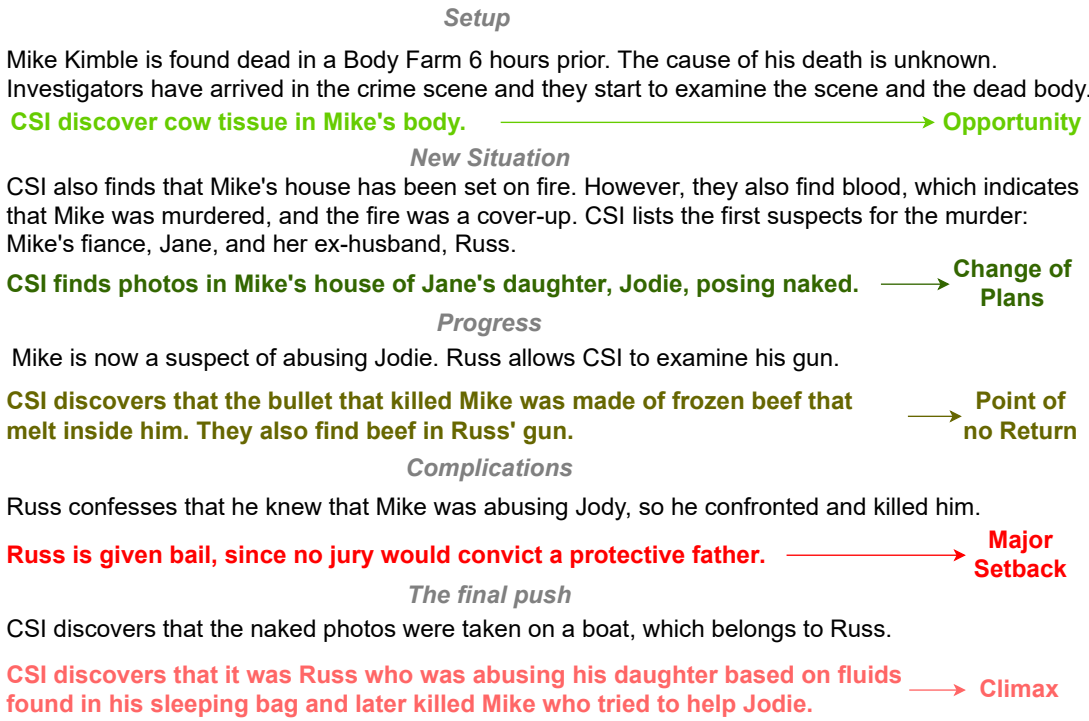


Figure 4.2: Example of narrative structure for episode "Burden of Proof" (Season 2, Episode 15)[1] from TV series " CSI: Crime Scene Investigation"; turning points are highlighted in color and narratives sections as defined in Hague (2017) are given in between.

dataset (Chapter 3) and employing a variant of the TP identification model proposed in the previous chapter. We find that narrative structure representations learned on TRIPOD, which contains feature-length films, transfer well across cinematic genres and computational tasks.

We propose a framework for end-to-end training in which narrative structure is treated as a latent variable for summarization. For unsupervised extractive summarization, we augment a directed version of TEXTRANK (Mihalcea and Tarau, 2004; Zheng and Lapata, 2019) with extra criteria related to narrative structure for computing centrality scores. For supervised extractive summarization, we augment a variant of SUMMARUNNER (Nallapati et al., 2017) with latent information of turning points and measure the relevance of all scenes with the identified storyline formed by the TPs. We extend the CSI dataset (Frermann et al., 2018) with binary labels indicating whether a scene should be included in the summary and present experiments with both supervised and unsupervised summarization models. An overview of our approach is shown in Figure 4.3.

[1] https://csi.fandom.com/wiki/Burden_of_Proof

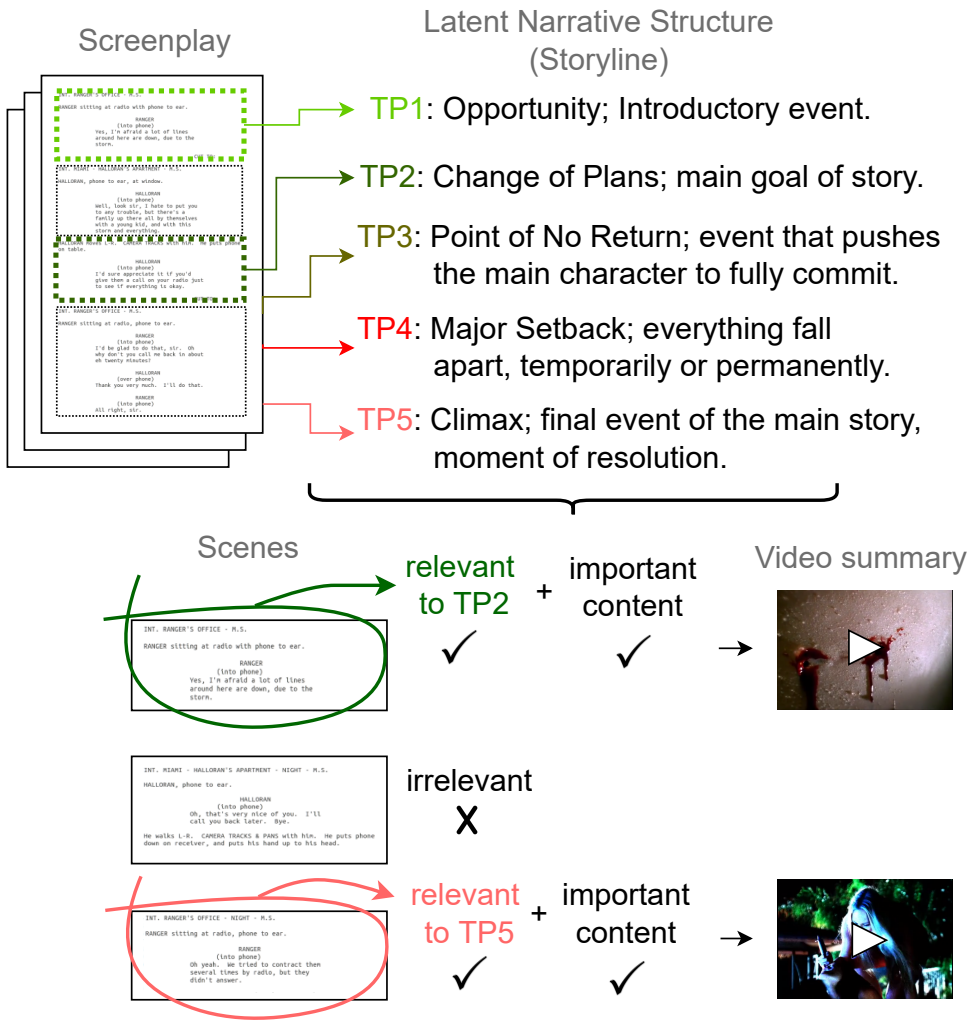


Figure 4.3: We first identify scenes that act as turning points (i.e., key events that segment the story into sections). Turning points can form the storyline of the episode. Next, we create a summary by selecting informative scenes, that are semantically related to turning points and present important content.

The contributions of this chapter can be summarized as follows:

1. We develop methods for instilling knowledge about narrative structure into generic supervised and unsupervised summarization algorithms.

2. We validate the hypothesis stated in the previous chapter that narrative structure can facilitate screenplay summarization.

3. We show that the general definition of TPs can be adapted to the specific structure and type of a given narrative. Our analysis shows that TPs identified in the latent space correlate with important summary content.

4.1 Related Work

A large body of previous work has focused on the computational analysis of narratives (Mani, 2012; Richards et al., 2009). Attempts to analyze how stories are written have been based on sequences of events (Schank and Abelson, 1975; Chambers and Jurafsky, 2009), plot units (McIntyre and Lapata, 2010; Goyal et al., 2010; Finlayson, 2012) and their structure (Lehnert, 1981; Rumelhart, 1980), as well as on characters or personas in a narrative (Black and Wilensky, 1979; Propp, 1968; Bamman et al., 2014, 2013; Valls-Vargas et al., 2014) and their relationships (Elson et al., 2010; Agarwal et al., 2014a; Srivastava et al., 2016).

However, work on summarization of narratives has had limited appeal, possibly due to the lack of annotated data for modeling and evaluation. Kazantseva and Szpakowicz (2010) summarize short stories by extracting features based on importance criteria (e.g., whether a segment contains protagonist or location information); they create extractive summaries to help readers decide whether they are interested in reading the whole story, without revealing its plot. Mihalcea and Ceylan (2007) summarize books with an unsupervised graph-based approach operating over segments (i.e., topical units). Their algorithm first generates a summary for each segment and then an overall summary by collecting sentences from the individual segment summaries.

Focusing on screenplays, Gorinski and Lapata (2015) generate a summary by extracting an optimal chain of scenes via a graph-based approach centered around the main characters. In a similar fashion, Tsoneva et al. (2007) create video summaries for TV series episodes; their algorithm ranks sub-scenes in terms of importance using features based on character graphs and textual cues available in the subtitles and movie scripts. Vicol et al. (2018) introduce the MovieGraphs dataset, which also uses character-centered graphs to describe the content of movie video clips. More recently, Chen et al. (2022a) address the task of abstractive summarization of transcripts from TV episodes by fine-tuning large pre-trained sequence-to-sequence models.

Our work synthesizes various strands of research on narrative structure analysis (Cutting, 2016; Hague, 2017), screenplay summarization (Gorinski and Lapata, 2015), and neural network modeling (Dong, 2018). We focus on *extractive* summarization and our goal is to identify an optimal sequence of key events in a narrative. We aim to create summaries which re-tell the plot of a story in a concise manner. Inspired by prior neural network-based approaches (Cheng and Lapata, 2016; Nallapati et al., 2017; Zhou et al., 2018b; Zheng and Lapata, 2019), we develop supervised and unsu-

pervised models for our summarization task based on neural representations of scenes and how these relate to the screenplay's narrative structure. Contrary to most previous work which has focused on characters, we select summary scenes based on events and their importance in the story. Although our definition of narrative structure is based on TPs (Chapter 3), the model architectures we propose are general and could be adapted to different plot analysis schemes (Field, 2005; Vogler, 2007). To overcome the difficulties in evaluating summaries for longer narratives, we also release a corpus of screenplays with scenes labeled as important (summary worthy). Our annotations augment an existing dataset based on CSI episodes (Frermann et al., 2018), which was originally developed for incremental natural language understanding.

4.2 Problem Formulation

Let $\mathcal{D}$ denote a screenplay consisting of a sequence of scenes $\mathcal{D} = \{s_1, s_2, \ldots, s_M\}$. Our aim is to select a subset $\mathcal{D}' = \{s_i, \ldots, s_K\}$ consisting of the most *informative* scenes (where $K < M$). Note that this definition produces extractive summaries; we further assume that selected scenes are presented according to their order in the screenplay. We next discuss how summaries can be created using both *unsupervised* and *supervised* approaches, and then move on to explain how these are adapted to incorporate narrative structure. We focus on both unsupervised and supervised methods in order to examine whether narrative structure can consistently help summarization in settings with or without summary-specific labels.

4.2.1 Unsupervised Screenplay Summarization

Our unsupervised model is based on an extension of TEXTRANK (Mihalcea and Tarau, 2004; Zheng and Lapata, 2019), a well-known algorithm for extractive single-document summarization. Given a document, i.e., a sequence of sentences, TEXTRANK creates an undirected fully-connected graph, where the nodes are representations of the sentences and edges denote the degree of similarity between sentences (see illustration in Figure 4.4). Next, TEXTRANK computes a centrality score per sentence based on its similarity with all other sentences in the document and rank the sentences from most central (i.e., important) to least.

In our setting, a screenplay is represented as a graph, in which nodes correspond to scenes and edges between scenes s_i and s_j are weighted by their similarity e_{ij}. A

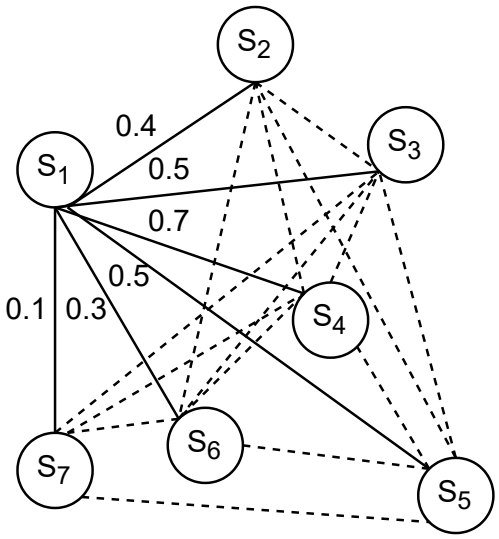


Figure 4.4: TEXTRANK creates an undirected fully-connected graph where nodes $s_1, ..., s_7$ are scene representations from the screenplay and edges between nodes denote their semantic similarity e_{ij}. For each scene, a centrality score is computed based on the weights of all edges between the current scene and all other scenes in the screenplay.

node's centrality (importance) is measured by computing its degree:

$$centrality(s_i) = \lambda_1 \sum_{j<i} e_{ij} + \lambda_2 \sum_{j>i} e_{ij} \tag{4.1}$$

where $\lambda_1 + \lambda_2 = 1$. The modification introduced in Zheng and Lapata (2019) takes directed edges into account, capturing the intuition that the centrality of any two nodes is influenced by their relative position. Also note that the edges of preceding and following scenes are differentially weighted by λ_1 and λ_2 (this modifies the graph of Figure 4.4 to an equivalent *directed* graph).

Although earlier implementations of TEXTRANK (Mihalcea and Tarau, 2004) compute node similarity based on symbolic representations such as tf*idf, we adopt a neural approach. Specifically, we obtain sentence representations based on a pre-trained encoder. In our experiments, we rely on the Universal Sentence Encoder (USE; Cer et al. 2018), however, other embeddings are possible.[2] We represent a scene by the mean of its sentence representations and measure scene similarity e_{ij} using cosine.[3] As in the original TEXTRANK algorithm (Mihalcea and Tarau, 2004), scenes are ranked

[2]USE (Cer et al., 2018) performed better than BERT (Devlin et al., 2019) in our experiments.

[3]We found cosine to be particularly effective with USE representations; other metrics, such as scaled dot product, are also possible.

based on their centrality and the M most central ones are selected to appear in the summary.

4.2.2 Supervised Screenplay Summarization

Most extractive models frame summarization as a classification problem. Following a recent approach (SUMMARUNNER; Nallapati et al. 2017), we use a neural network-based encoder to build representations for scenes and apply a binary classifier over these to predict whether they should be in the summary. For each scene $s_i \in \mathcal{D}$, we predict a label $y_i \in \{0,1\}$ (where 1 means that s_i must be in the summary) and assign a score $p(y_i|s_i,\mathcal{D},\theta)$ quantifying s_i's relevance to the summary (θ denotes model parameters). We assemble a summary by selecting K sentences with the top $p(1|s_i,\mathcal{D},\theta)$.

We calculate sentence representations via the pre-trained USE encoder (Cer et al., 2018); a scene is represented as the weighted sum of the representations of its sentences, which we obtain from a BiLSTM equipped with an attention mechanism. Next, we compute richer scene representations by modeling surrounding context of a given scene. We encode the screenplay with a BiLSTM network and obtain contextualized representations for scenes s_i by concatenating the hidden layers of the forward $\overrightarrow{h_i}$ and backward $\overleftarrow{h_i}$ LSTM, respectively: $s_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. The vector s_i therefore represents the *content* of the i^{th} scene. In contrast with the unsupervised approach (i.e., TEXTRANK), we in this case have summary-specific labels for training, and hence we can learn better contextualized scene representations by training BiLSTM networks.

We also estimate the *salience* of scene s_i by measuring its similarity with a global screenplay content representation g. The latter is the weighted sum of all scene representations $s_1, s_2, \ldots, s_M$. We calculate the semantic similarity between s_i and g by computing the element-wise dot product b_i, cosine similarity cos_i, and pairwise distance d_i between their respective vectors:

$$b_i = s_i \odot g \quad cos_i = \frac{s_i \cdot g}{\|s_i\|\|g\|} \tag{4.2}$$

$$d_i = \frac{s_i \cdot g}{\max(\|s_i\|_2 \cdot \|g\|_2)} \tag{4.3}$$

These similarity metrics are able to capture different properties of the vectors s_i and g. The salience of scene s_i is then the concatenation of the similarity metrics:

$$\text{salience}_i = [b_i; cos_i; d_i] \tag{4.4}$$

The content vector s_i and the salience vector $salience_i$ are concatenated and fed to a single neuron that outputs the probability of a scene belonging to the summary.[4]

4.2.3 Narrative Structure

We now explain how to inject knowledge about narrative structure into our summarization models. For both models, such knowledge is transferred via a network pre-trained on the TRIPOD[5] dataset introduced in the previous chapter. In the previous chapter, we presented an approach for projecting TP annotations from the plot synopses to the screenplays and creating silver-standard labels over full-length movie screenplays. In this chapter, we use this silver-standard dataset in order to pre-train a network which performs TP identification.

TP Identification Network We follow the model proposed in Chapter 3 and first encode screenplay scenes via a BiLSTM equipped with an attention mechanism. We then contextualize them with respect to the whole screenplay via a second BiLSTM. Next, we compute topic-aware scene representations t_i via a context interaction layer (CIL) as in the previous chapter. CIL is inspired by traditional segmentation approaches (Hearst, 1997) and measures the semantic similarity of the current scene with a preceding and following context window in the screenplay. Hence, the topic-aware scene representations also encode the degree to which each scene acts as a topic boundary in the screenplay.

In the final layer, we employ *TP-specific attention mechanisms* to compute the probability p_{ij} that scene t_i represents the j^{th} TP in the screenplay. Note that we expect the TP-specific attention distributions to be sparse, as there are only a few scenes which are relevant for a TP (recall that TPs are boundary scenes between sections). To encourage sparsity, we add a low temperature value τ (Hinton et al., 2015) to the softmax part of the attention mechanisms:

$$t'_{ij} = \tanh(W_j t_i + b_j), \quad t'_j \in [-1, 1] \tag{4.5}$$

$$p_{ij} = \frac{\exp(t'_{ij}/\tau)}{\sum_{r=1}^{M} \exp(t'_{rj}/\tau)}, \quad \sum_{i=1}^{M} p_{ij} = 1 \tag{4.6}$$

where W_j, b_j represent the trainable weights of the attention layer of the j^{th} TP.

[4] Aside from salience and content, Nallapati et al. (2017) take into account novelty and position-related features. We ignore these as they are specific to news articles and denote the modified model as SUMMARUNNER*.

[5] https://github.com/ppapalampidi/TRIPOD

Unsupervised SUMMER We now introduce our model, SUMMER (short for Screenplay Summarization with Narrative Structure). We first present an unsupervised variant which modifies the computation of scene centrality in the directed version of TEXTRANK (Equation (4.1)).

Specifically, we use the pre-trained network described above to obtain TP-specific attention distributions. We then select an overall score f_i for each scene (denoting how likely it is to act as a TP). We set $f_i = \max_{j\in[1,5]} p_{ij}$, i.e., to the p_{ij} value that is highest across TPs. We incorporate these scores into centrality as follows:

$$centrality(s_i) = \lambda_1 \sum_{j<i} (e_{ij} + f_j) + \lambda_2 \sum_{j>i} (e_{ij} + f_i) \tag{4.7}$$

Intuitively, we add the f_j term in the forward sum in order to incrementally increase the centrality scores of scenes as the story moves on and we encounter more TP events (i.e., we move to later sections in the narrative). At the same time, we add the f_i term in the backward sum in order to also increase the scores of scenes identified as TPs.

Supervised SUMMER We also propose a supervised variant of SUMMER following the basic model formulation in Section 4.2.3. We still represent a scene as the concatenation of a content and salience vector, which serve as input to a binary classifier. However, we now modify how salience is determined; instead of computing a general global content representation g for the screenplay, we identify a sequence of TPs and measure the semantic similarity of each scene with this sequence. Our model is depicted in Figure 4.5.

We utilize the pre-trained TP network (Figures 4.5(a) and (b)) to compute sparse attention scores over scenes. In the supervised setting, where gold standard binary labels provide a training signal, we fine-tune the network in an end-to-end fashion on summarization (Figure 4.5(c)). We compute the TP representations via the attention scores; we calculate a vector tp_j as the weighted sum of all topic-aware scene representations t produced via the context interaction layer (CIL; see TP identification network): $tp_j = \sum_{i\in[1,M]} p_{ij} t_i$, where M is the number of scenes in a screenplay. In practice, only a few scenes contribute to tp_j due to the τ parameter in the softmax function (Equation (4.6)).

A TP–scene interaction layer measures the semantic similarity between scenes t_i and latent TP representations tp_j (Figure 4.5(c)). Intuitively, a complete summary should contain scenes which are related to at least one of the key events in the screenplay. We calculate the semantic similarity $salience_{ij}$ of scene t_i with TP tp_j as in Equa-

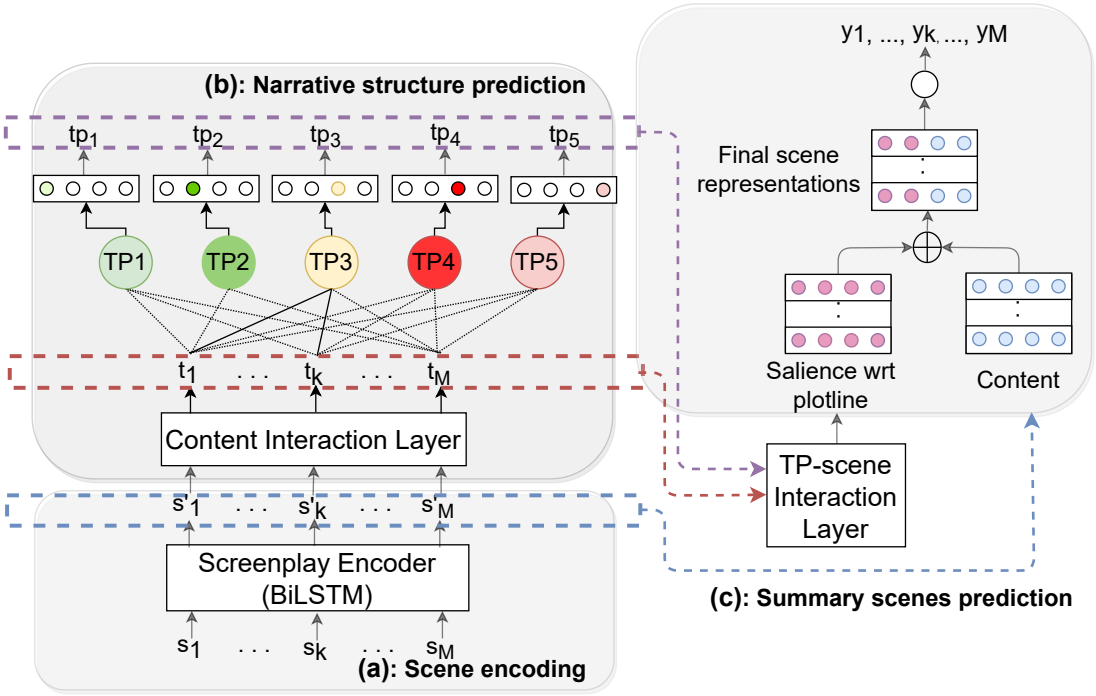


Figure 4.5: Overview of SUMMER. After contextualizing the scenes in the screenplay (a) with a screenplay encoder, we use one TP-specific attention mechanism per turning point in order to acquire TP-specific distributions over scenes (b). We then compute the similarity between TPs and contextualized scene representations. Finally, we perform max pooling over TP-specific similarity vectors and concatenate the final similarity representation with the contextualized scene representation (c).

tions (4.2) and (4.3). This vector measures the degree of salience of the i^{th} scene with respect to the j^{th} key event. We then perform max pooling over vectors $\text{salience}_{i1}, \ldots,$ salience_{iT}, where T is the number of TPs (i.e., five) and calculate a final similarity vector $\text{salience}'_i$ for the i^{th} scene.

The model is trained end-to-end on the summarization task using *BCE*, the binary cross-entropy loss function. We add an extra regularization term to this objective to encourage the TP-specific attention distributions to be orthogonal (since we want each attention layer to attend to different parts of the screenplay). We thus maximize the Kullback-Leibler (KL) divergence $\mathcal{D}_{KL}$ between all pairs of TP attention distributions $tp_i, i \in [1,5]$:

$$O = \sum_{i \in [1,5]} \sum_{j \in [1,5], j \neq i} \log \frac{1}{\mathcal{D}_{KL}\left(tp_i \| tp_j\right) + \varepsilon} \tag{4.8}$$

Furthermore, we know from screenwriting theory (Hague, 2017) that there are rules of thumb as to when a TP should occur (e.g., the Opportunity occurs after the first

overall	
episodes	39
scenes	1544
summary scenes	454
per episode	
# scenes	39.58 (6.52)
# crime-specific aspects	5.62 (0.24)
# summary scenes	11.64 (2.98)
# summary scenes (%)	29.75 (7.35)
# sentences	822.56 (936.23)
# tokens	13.27k (14.67k)
per episode scene	
# sentences	20.78 (35.61)
# tokens	335.19 (547.61)
# sentence tokens	16.13 (16.32)

Table 4.1: CSI dataset statistics; means and (std).

10% of a screenplay, Change of Plans is approximately 25% in). It is reasonable to discourage tp distributions to deviate drastically from these expected positions. Focal regularization F minimizes the KL divergence $\mathcal{D}_{KL}$ between each TP attention distribution tp_i and its expected position distribution th_i:

$$F = \sum_{i \in [1,5]} \mathcal{D}_{KL}(tp_i \| th_i) \tag{4.9}$$

The final loss $\mathcal{L}$ is the weighted sum of all three components, where a, b are fixed during training:

$$\mathcal{L} = BCE + aO + bF \tag{4.10}$$

4.3 Experimental Setup

Crime Scene Investigation Dataset We performed experiments on an extension of the CSI dataset[6] introduced by Frermann et al. (2018). The original purpose of this dataset is to predict who the perpetrator is as events unfold in the episode. It consists

[6] https://github.com/EdinburghNLP/csi-corpus

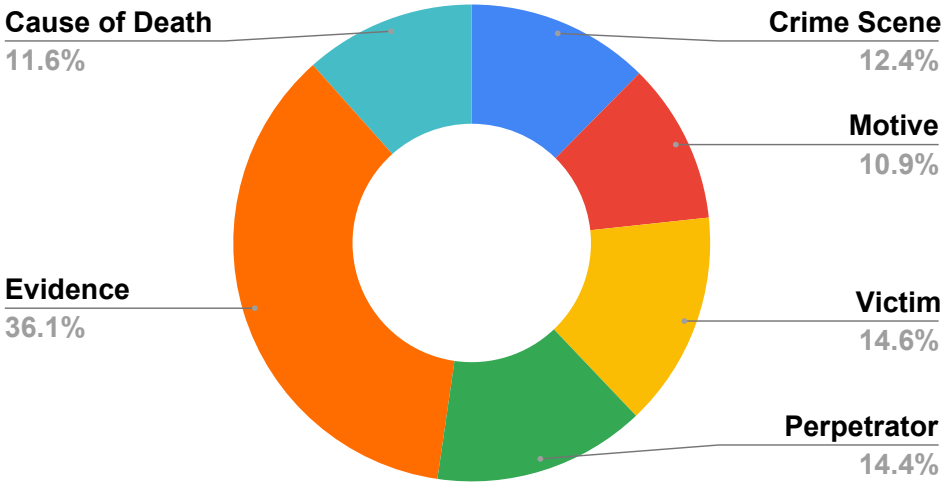


Figure 4.6: Average composition of a CSI summary based on different crime-related aspects.

of 39 CSI episodes, each annotated with *word-level* labels denoting whether the perpetrator is mentioned in the utterances characters speak. We further collected *scene-level* binary labels indicating whether episode scenes are important and should be included in a summary. Three human judges performed the annotation task after watching CSI episodes scene-by-scene. To facilitate annotation, judges were asked to indicate why they thought a scene was important, because it revealed: (i) the victim, (ii) the cause of death, (iii) an autopsy report, (iv) crucial evidence, (v) the perpetrator, and (vi) the motive or the relation between perpetrator and victim. Annotators were free to select more than one or none of the listed reasons where appropriate. We can think of these reasons as high-level *aspects* a good summary should cover (for CSI and related crime series). Annotators were not given any information about TPs or narrative structure; the annotation was not guided by theoretical considerations, rather our aim was to produce useful CSI summaries. Table 4.1 presents the dataset statistics.

In Figure 4.6 we illustrate the average composition of a summary based on the different aspects seen in a crime investigation (e.g., crime scene, victim, cause of death, perpetrator, evidence). Most of these aspects are covered in 10–15% of a summary, which corresponds to approximately two scenes in the episode. Only the "Evidence" aspect occupies a larger proportion of the summary (36.1%) corresponding to five scenes. However, there exist scenes which cover multiple aspects (an as a result are annotated with more than one label) and episodes that do not include any scenes related to a specific aspect (e.g., if the murder was a suicide, there is no perpetrator).

We should note that Frermann et al. (2018) discriminate between different cases

presented in the same episode in the original CSI dataset. Specifically, there are episodes in the dataset, where except for the primary crime investigation case, a second one is presented occupying a significantly smaller part of the episode. Although in the original dataset, there are annotations available indicating which scenes refer to each case, we assume no such knowledge treating the screenplay as a single unit — most TV series and movies contain sub-stories. We also hypothesize that the latent identified TP events in SUMMER should relate to the primary case.

Implementation Details In order to set the hyperparameters of all proposed networks, we used a small development set of four episodes from the CSI dataset. After experimentation, we set the temperature τ of the softmax layers for the TP-specific attentions (Equation (4.6)) to 0.01. In all unsupervised versions of TEXTRANK and SUMMER we used a threshold h equal to 0.2 for removing weak edges from the corresponding fully connected screenplay graphs (e.g., we make the graph presented in Figure 4.4 less dense by removing the edge with weight 0.1 for scene s_i). For the supervised version of SUMMER, where we use additional regularization terms in the loss function, we experimentally set the weights a and b for the different terms to 0.15 and 0.1, respectively. We used the Adam algorithm (Kingma and Ba, 2015) for optimizing our networks. After experimentation, we chose an LSTM with 64 neurons for encoding the scenes in the screenplay and another identical one for contextualizing them. For the context interaction layer, the window l for computing the surrounding context of a screenplay scene was set to 20% of the screenplay length as in the previous chapter. Finally, we also added a dropout of 0.2. For developing our models we used PyTorch (Paszke et al., 2017).

Since the binary labels in the supervised setting are imbalanced, we apply class weights to the binary cross-entropy loss of the respective models. We weight each class by its inverse frequency in the training set. Finally, in supervised SUMMER, where we also identify the narrative structure of the screenplays, we consider as key events per TP the scenes that correspond to an attention score higher than 0.05.

As shown in Table 4.1, the gold standard summaries in our dataset have a compression rate of approximately 30%. During inference, we select the top M scenes as the summary, such that they correspond to 30% of the length of the episode.

4.4 Results and Analysis

4.4.1 Is Narrative Structure Helpful?

We perform 10-fold cross-validation and evaluate model performance in terms of F1 score. Table 4.2 summarizes the results of unsupervised models. We present the following baselines: Lead 30% selects the first 30% of an episode as the summary, Last 30% selects the last 30%, and Mixed 30%, randomly selects 15% of the summary from the first 30% of an episode and 15% from the last 30%. We also compare SUMMER against TEXTRANK based on tf*idf (Mihalcea and Tarau, 2004), the directed neural variant described in Section 4.2.1 without any TP information, a variant where TPs are approximated by their expected position as postulated in screenwriting theory, and a variant that incorporates information about characters (Gorinski and Lapata, 2015) instead of narrative structure. Finally, we also experiment with adding character-related information into the general TEXTRANK algorithm. As mentioned in previous chapters, entities are central in narratives and crucial for the development of the story (Jannidis, 2009; Frow, 2014). In the previous chapter, we considered entity-specific information, but we found such information unreliable for identifying TP events. Here, we examine again the performance for a variant of TEXTRANK that considers character-related information. For the character-based TEXTRANK, called SCENESUM, we substitute the f_i, f_j scores in Equation (4.7) with character-related importance scores c_i. Character-related scores are computed as the fraction of the main characters participating in a scene over all characters that appear in the scene, similar to the definition in Gorinski and Lapata (2015):

$$c_i = \frac{\sum_{c \in C} [c \in S \cup main(C)]}{\sum_{c \in C} [c \in S]} \tag{4.11}$$

where S is the set of all characters participating in scene s_i, C is the set of all characters participating in the screenplay and $main(C)$ are all the main characters of the screenplay. We retrieve the set of main characters from the IMDb page of the respective episode. This serves as an upper bound, since in reality we have to employ a different model to determine who the main characters are (Gorinski and Lapata, 2015), which will introduce further errors into the algorithm. We also note that human agreement between annotators selecting summary scenes in an episode is 79.26 F1 score, as measured on a small subset of the corpus.

As shown in Table 4.2, SUMMER achieves the best performance (44.70 F1 score) among all models and is superior to an equivalent model which uses expected TP

Model	F1 ↑
Lead 30%	30.66
Last 30%	39.85
Mixed 30%	34.32
TEXTRANK, undirected, tf*idf	32.11
TEXTRANK, directed, neural	41.75
TEXTRANK, directed, expected TP positions	41.05
SCENESUM, directed, character-based weights	42.02
SUMMER	**44.70**

Table 4.2: Unsupervised screenplay summarization.

positions or a character-based representation. This indicates that the pre-trained network provides better predictions for key events than position and character heuristics, even though there is a domain shift from Hollywood movies in the TRIPOD corpus to episodes of a crime series in the CSI corpus. Moreover, we find that the directed versions of TEXTRANK are better at identifying important scenes than the undirected version. We found that performance peaks with $\lambda_1 = 0.7$ (see Equation (4.7)), indicating that higher importance is given to scenes as the story progresses.

In Table 4.3, we report results for supervised models. Aside from the various baselines in the first block of the table, we compare the neural extractive model SUMMARUNNER*[7] (Nallapati et al., 2017) presented in Section 4.2.2 with several variants of our model SUMMER in order to determine which parts of the model contribute to performance gains on summarization.

We experimented with randomly initializing the network for TP identification of Section 4.2.3 (denoted as $-$P) and with using a pre-trained network (denoted as $+$P). Recall that the TP identification model can optionally be pre-trained on the TRIPOD dataset which consists of movies with TP annotations, and then transferred to the CSI dataset. We also experimented with removing the regularization terms, O and F (Equations (4.8) and (4.9)) from the loss function (denoted as $-$R). By removing these regularization terms ($-$R), we do not encourage the model to compute sparse attention scores for TPs and to favor specific positions per TP in the screenplay.

We also assess the performance of SUMMER when we follow a two-step approach where we first predict TPs via the pre-trained network of Section 4.2.3 and then train a network on screenplay summarization based on *fixed* TP representations (fixed one-hot

[7] Our adaptation of SUMMARUNNER that considers content and salience vectors for scene selection.

Model	F1 ↑
Lead 30%	30.66
Last 30%	39.85
Mixed 30%	34.32
SUMMARUNNER*	48.56
SCENESUM	47.71
SUMMER, fixed one-hot TPs	46.92
SUMMER, fixed distributions	47.64
SUMMER, −P, −R	**51.93**
SUMMER, −P, +R	49.98
SUMMER, +P, −R	50.56
SUMMER, +P, +R	**52.00**

Table 4.3: Supervised screenplay summarization.

TPs), or alternatively use expected TP position distributions as postulated in screenwriting theory (fixed distributions; Hague 2017). This is in contrast to our proposed approach, where the pre-trained TP identification network is further fine-tuned on the CSI corpus in an end-to-end fashion for predicting TP events in the latent space. Finally, we incorporate character-based information into our baseline (SUMMARUNNER*) and create a supervised version of SCENESUM. We now utilize the character importance scores per scene (Equation (4.11)) as attention scores – instead of using a trainable attention mechanism – when computing the global screenplay representation d (Section 4.2.2).

Table 4.3 shows that all end-to-end SUMMER variants outperform SUMMARUNNER*. The best result (52.00 F1 Score) is achieved by pre-trained SUMMER with regularization, outperforming SUMMARUNNER* by an absolute difference of 3.44. The randomly initialized version with no regularization achieves similar performance (51.93 F1 score). For summarizing screenplays, explicitly encoding narrative structure seems to be more beneficial than general representations of scene importance. Finally, two-step versions of SUMMER perform poorly, which indicates that end-to-end training and fine-tuning of the TP identification network on the target dataset is crucial.

	Coverage of aspects ↑	# scenes per TP
Fixed one-hot TPs	63.11	1.00
Fixed distributions	67.01	1.05
Learnt, −P, −R	44.48	1.19
Learnt, −P, +R	51.96	1.14
Learnt, +P, −R	62.35	3.07
Learnt, +P, +R	**70.25**	1.20

Table 4.4: We report the percentage of aspect labels covered by *latent TP predictions* for SUMMER variants. Coverage of aspects measures diversity and # scenes per TP measures sparsity of the latent identified events.

4.4.2 What Does the Model Learn?

Apart from performance on summarization, we would also like to examine the quality of the TPs inferred by SUMMER (supervised variant). Problematically, we do not have any gold standard TP annotation in the CSI corpus. Nevertheless, we can implicitly assess whether they are meaningful by measuring how well they correlate with the reasons annotators cite to justify their decision to include a scene in the summary (e.g., because it reveals cause of death or provides important evidence). Specifically, we compute the extent to which these aspects overlap with the TPs predicted by SUMMER as:

$$C = \frac{\sum_{A_i \in A} \sum_{TP_j \in TP} [dist(TP_j, A_i) \leq 1]}{|A|} \tag{4.12}$$

where A is the set of all aspect scenes, $|A|$ is the number of aspects, TP is the set of scenes inferred as TPs by the model, A_i and TP_j are the subsets of scenes corresponding to the i^{th} aspect and j^{th} TP, respectively, and $dist(TP_j, A_i)$ is the minimum distance between TP_j and A_i in number of scenes.

The proportion of aspects covered is given in Table 4.4, middle column. We find that coverage is relatively low (44.48%) for the randomly initialized SUMMER with no regularization. There is a slight improvement of +7.48% when we force the TP-specific attention distributions to be orthogonal and close to expected positions. Pre-training and regularization provide a significant boost, increasing coverage to 70.25%, while pre-trained SUMMER without regularization infers on average more scenes representative of each TP. This shows that the orthogonality constraint also encourages sparse attention distributions for TPs.

Turning Point	Crime scene	Victim	Death Cause	Perpetrator	Evidence	Motive
Opportunity	**56.76**	**52.63**	15.63	15.38	2.56	0.00
Change of Plans	**27.03**	**42.11**	**21.88**	15.38	5.13	0.00
Point of no Return	8.11	13.16	9.38	**25.64**	**48.72**	5.88
Major Setback	0.00	0.00	6.25	10.25	**48.72**	**35.29**
Climax	2.70	0.00	6.25	2.56	**23.08**	**55.88**

Table 4.5: Percentage of aspect labels covered per TP for SUMMER, +P, +R.

Table 4.5 shows the degree of association between individual TPs and summary aspects. We observe that Opportunity and Change of Plans are mostly associated with information about the crime scene and the victim, Climax is focused on the revelation of the motive, while information relating to cause of death, perpetrator, and evidence is captured by both Point of no Return and Major Setback. Overall, the generic Hollywood-inspired TP labels are adjusted to our genre and describe crime-related key events, even though no aspect labels were provided to our model during training.

4.4.3 Do Humans Like the Summaries?

We also conducted a human evaluation experiment using the summaries produced by automatic methods for 10 CSI episodes.[8] We produced summaries based on the gold standard annotations (Gold), SUMMARUNNER*, and the supervised version of SUMMER. Since 30% of an episode results in lengthy summaries (15 minutes on average), we further increased the compression rate for this experiment by limiting each summary to six scenes (which accounts for 9.5 minutes on average). For the gold standard condition, we randomly selected exactly one scene per aspect. For SUMMARUNNER* and SUMMER we selected the top six predicted scenes based on their posterior probabilities. We then created video summaries by isolating and merging the selected scenes in the raw video.

We asked Amazon Mechanical Turk (AMT) workers to watch the video summaries for all systems and rank them from most to least informative. They were also presented with six questions relating to the aspects the summary was supposed to cover (e.g., Was the victim revealed in the summary? Do you know who the perpetrator was?). They

[8] https://github.com/ppapalampidi/SUMMER/tree/master/video_summaries

System	Crime scene	Victim	Death Cause	Perpe-trator	Evidence	Motive	Overall	Rank
SUMMARUNNER*	85.71	**93.88**	75.51	81.63	59.18	38.78	72.45	2.18
SUMMER	**89.80**	87.76	**83.67**	81.63	**77.55**	**57.14**	**79.59**	2.00
Gold	**89.80**	91.84	71.43	**83.67**	65.31	**57.14**	76.53	1.82

Table 4.6: Human evaluation: percentage of yes answers by AMT workers regarding each aspect in a summary. All differences in (average) Rank are significant ($p < 0.05$, using a χ^2 test).

could answer Yes, No, or Unsure. Five workers evaluated each summary. We provide details of the instructions and interface used for human evaluation in Appendix C.2.

Table 4.6 shows the proportion of times participants responded Yes for each aspect across the three systems. Although SUMMER does not improve over SUMMARUNNER* in identifying basic information (i.e., about the victim and perpetrator), it creates better summaries overall with more diverse content (i.e., it more frequently includes information about cause of death, evidence, and motive). This observation validates our assumption that identifying scenes that are semantically close to the key events of a screenplay leads to more complete and detailed summaries. Finally, Table 4.6 also lists the average rank per system (lower is better), which shows that crowdworkers like gold summaries best, SUMMER is often ranked second, followed by SUMMARUNNER* in third place.

4.4.4 Which Narrative Sections Are More Important?

We illustrate in Figure 4.7 the performance (F1 score) of the directed neural TEXTRANK and SUMMER, which also considers scores related to the narrative structure, in the unsupervised setting with respect to different λ_1 values. *Higher* λ_1 values correspond to *higher importance* for the *succeeding scenes* and respectively lower importance for the preceding ones, since λ_1 and λ_2 are bounded ($\lambda_1 + \lambda_2 = 1$).

We observe that performance increases when higher importance is attributed to screenplay scenes as the story moves on ($\lambda_1 > 0.5$), whereas for extreme cases ($\lambda_1 \rightarrow 1$), where only the later parts of the story are considered, performance drops. Overall, the same peak appears for both TEXTRANK and SUMMER when $\lambda_1 \in [0.6, 0.7]$, which means that slightly higher importance is attributed to the screenplay scenes that follow. Intuitively, initial scenes of an episode tend to have high similarity with all other

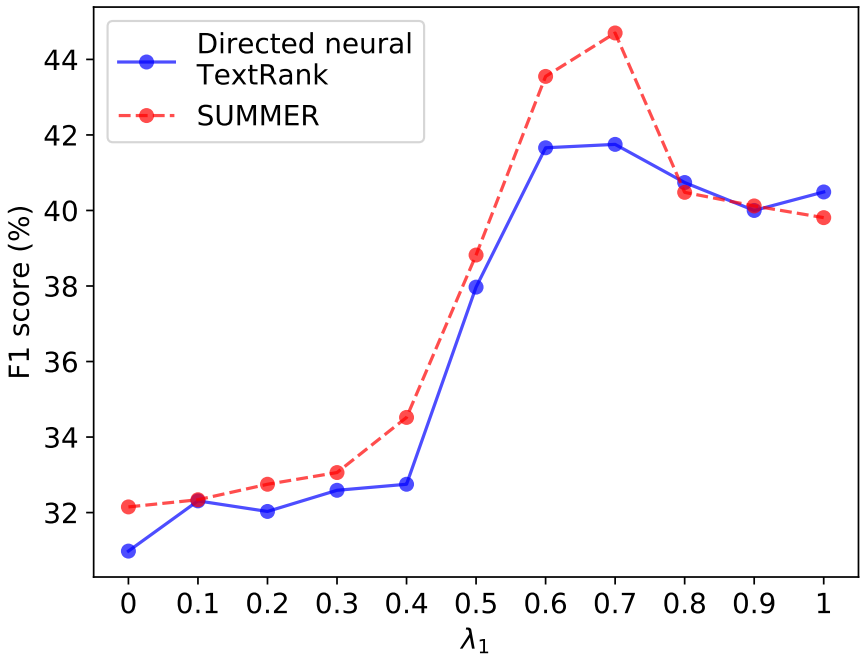


Figure 4.7: F1 score (%) for directed neural TEXTRANK and SUMMER for unsupervised summarization with respect to different λ_1 values. Higher λ_1 values correspond to higher importance in the next context for the centrality computation of a current scene.

scenes in the screenplay, and on their own are not very informative (e.g., the crime, victim, and suspects are introduced but the perpetrator is not yet known). As a result, the undirected version of TEXTRANK tends to favor the first part of the story and the resulting summary consists mainly of initial scenes. By adding extra importance to later scenes, we also encourage the selection of later events that might be surprising (and hence have lower similarity with other scenes) but more informative for the summary. Moreover, in SUMMER, where the weights change in a systematic manner based on narrative structure, we also observe that scenes appearing later in the screenplay are selected more often for inclusion in the summary.

As described in detail in Section 4.2.3, we also infer the narrative structure of CSI episodes in the supervised version of SUMMER via latent TP representations. During experimentation (see Section 4.4), we found that these TPs are highly correlated with different aspects of a CSI summary. In Figure 4.8 we visualize examples of identified TPs on CSI episodes during test time alongside with gold standard aspect-based summary annotations. Based on the examples, we empirically observe that different TPs tend to capture different types of information helpful for summarizing crime investi-

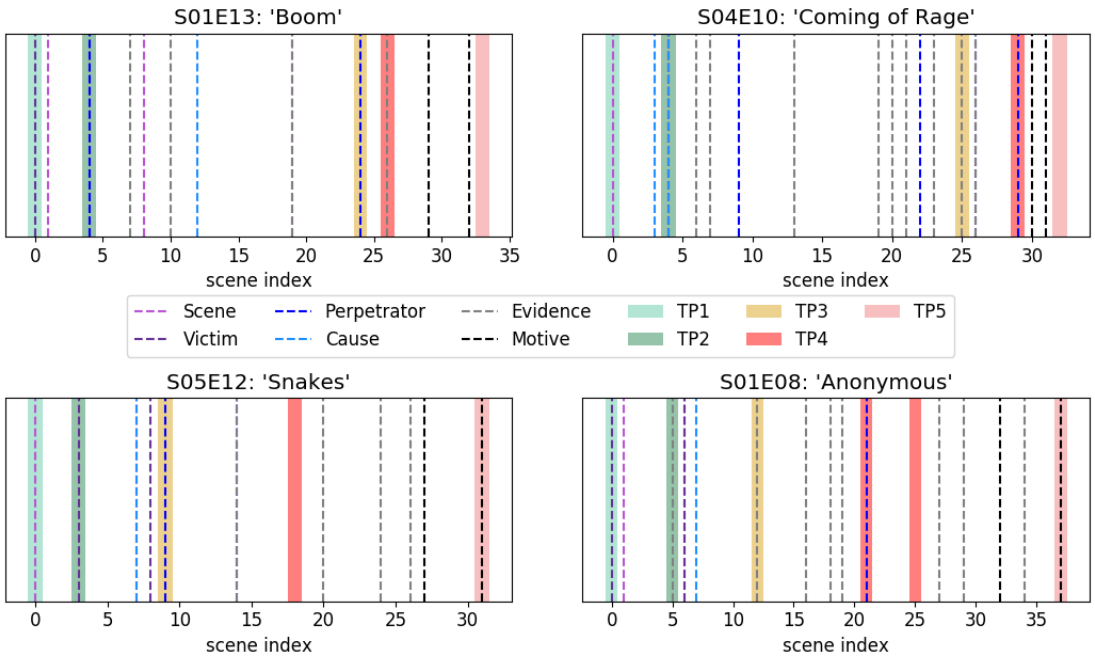


Figure 4.8: Examples of inferred TPs alongside with gold standard aspect-based summary labels in CSI episodes at test time. The TP events are identified in the latent space for the supervised version of SUMMER (+P, +R).

gation stories (e.g., crime scene, victim, perpetrator, motive).

4.5 Summary of Chapter

In this chapter, we validated our hypothesis that the underlying structure of narratives is beneficial for long-form summarization. We utilized knowledge about the narrative structure in terms of turning points as defined in the previous chapter and showed how this information can be integrated with supervised and unsupervised extractive summarization algorithms. Experiments on the CSI corpus (Frermann et al., 2018) showed that this scheme transfers well to a different genre (crime investigation) and that utilizing narrative structure boosts summarization performance, leading to more complete and diverse summaries. Moreover, analysis of model output further revealed that latent events encapsulated by turning points correlate with important aspects of a CSI summary.

Although currently our methods rely solely on textual information from screenplays, we believe that additional modalities, such as video and audio, could further facilitate the identification of key events in narratives. In the next chapter, we investigate the role of additional modalities on extractive movie summarization (i.e., video-

to-video). Moreover, we alleviate the need for summary-specific labels in movies, which is challenging and time consuming to gather. Instead, we hypothesize that we can create informative video summaries by directly considering TPs as key events that should appear in a movie summary.

Chapter 5

Multimodal Movie Summarization via Sparse Graph Construction

In this chapter, we transition from a text-only to a multimodal setting for addressing extractive movie summarization (i.e., video-to-video). As in the previous chapter, we again consider full-length screenplays, which are naturally segmented into scenes, as our main input and formalize extractive movie summarization as the selection of a few important scenes. However, we now additionally consider multimodal input (i.e., full-length video and audio) for the identification of key events.

Since collecting summary-related annotations similarly to the CSI dataset (Chapter 4) is challenging and time-consuming, we propose that automatic movie summarization can be reduced to turning point identification. Our assumption that turning point identification is sufficient for addressing summarization builds on earlier work (Lehnert, 1981; Lohnert et al., 1981; Mihalcea and Ceylan, 2007) which claims that high-level analysis is necessary for revealing concepts central to a story. Moreover, we have already validated in the previous chapter that turning points present key events and improve summarization. As demonstrated in Chapters 3 and 4, turning points are ideally suited to summarizing movies for at least three reasons. Firstly, they are intuitive, and can be identified by naive viewers (Chapter 3), so there is hope the process can be automated. Secondly, TPs have specific definitions and expected positions which facilitate automatic identification especially in low resource settings by providing prior semantic and positional knowledge (Chapter 3). Thirdly, they facilitate data efficiency: since the summarization problem is re-formulated as a scene-level classification task, no additional resources are required for creating the movie summaries over and above those developed for identifying turning points (as in Chapter 4). We present in Fig-

1. Opportunity
Introductory event.
Juno discovers she is pregnant with a child fathered by her friend and longtime admirer.
2. Change of Plans
Main goal of story.
Juno decides to give the baby up for adoption.
3. Point of No Return
Event that pushes the main characters to fully commit.
Juno meets a couple, and agrees to a closed adoption.
4. Major Setback
Everything falls apart, temporarily or permanently.
Juno watches the couple's marriage fall apart.
5. Climax
Final event of the main story, moment of resolution.
Juno gives birth and spouse from ex-couple claims newborn as single adoptive mother.

Figure 5.1: Turning points (*from the movie "Juno"*) and their definitions.

ure 5.1 an example of TPs for the movie "Juno" in order to illustrate how these key events describe the storyline of the movie.

Next, we model TP identification (and by extension summarization) as a supervised classification task. However, we depart from previous approaches to movie analysis which mostly focus on interactions between *characters* (Do et al., 2018; Tran et al., 2017; Gorinski and Lapata, 2015) and model connections between *events*. Moreover, we discard the simplifying assumption that a screenplay consists of a *sequence of scenes* (Gorinski and Lapata 2015; Chapter 3; Chapter 4) and instead represent interactions between scenes as a *sparse graph*. Specifically, we view the screenplay of a movie as a graph whose nodes correspond to scenes (self-contained events) and edges denote relations between them which we compute based on their linguistic and audiovisual similarity. In contrast to previous work on general-purpose summarization that relies on fully connected graphs (Mihalcea and Tarau, 2004; Zheng and Lapata, 2019; Wang et al., 2020a), we induce sparse graphs by selecting a subset of nodes as neighbors for a scene; the size of this subset is not set in advance but learnt as part of the network. Sparse graphs provide better contextualization for scenes and tend to be more informative, as different genres present different degrees of connectivity between important events. We rely on Graph Convolutional Networks (GCNs; Duvenaud et al.

2015; Kearnes et al. 2016; Kipf and Welling 2017) to encode relevant neighborhood information in the sparsified graph for every scene which in turn contributes to deciding whether it acts as a TP and should be included in the summary. Finally, we find that multimodal information from the movie video is especially informative for creating the sparse graphs and finding meaningful connections between events in movies.

The contributions of this chapter can be summarized as follows:

1. We approach movie summarization directly via TP identification which we argue is a well-defined and possibly less subjective task. This formulation alleviates the need for gathering summary-specific labels.

2. We propose a TP identification model which relies on sparse graphs constructed based on multimodal information. We show that the audiovisual information contributes to more meaningful graph structures which improve performance on summarization.

3. We find that the induced graphs are meaningful with differing graph topologies corresponding to different movie genres.

5.1 Related Work

We reviewed prior work on *textual* computational analysis and summarization of narratives in Chapters 3 and 4. However, work on video understanding has also looked at movies and TV shows. Existing datasets in this domain (Tapaswi et al., 2016; Rohrbach et al., 2015; Lei et al., 2018, 2020b) do not contain more than a few hundred movies or TV episodes and focus mostly on *isolated* video clips rather than *entire* narratives. For example, Tapaswi et al. (2015b) align movie scenes to book chapters and focus on computing the semantic similarity between different modalities (i.e., video and text), while Xiong et al. (2019) align movie segments to descriptions (i.e., synopses) using a graph-based approach for a similar task.

The most popular tasks in computer vision that consider movies and TV episodes are video captioning, question answering and video retrieval. However, these tasks again focus on isolated clips with simple semantics (e.g., actions, objects) rather than entire narratives. As an example, Rohrbach et al. (2015) introduce a dataset where video clips from movies are aligned to text descriptions in order to address video captioning. Tapaswi et al. (2015a) introduce a Question-Answering (QA) dataset based

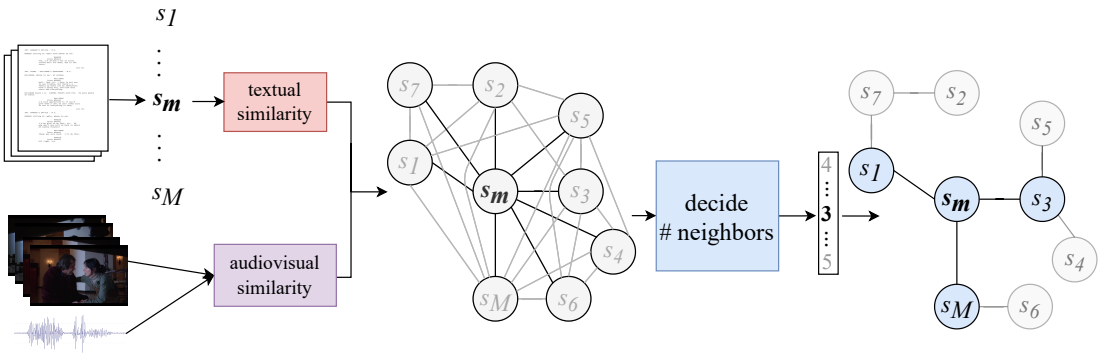
on movies, although the questions are again restricted to isolated video clips. Lei et al. (2018) create a similar QA dataset called TVQA for answering questions from TV episodes, while again considering only isolated clips. Later, Lei et al. (2020b) expand TVQA for retrieving video clips given textual queries and producing textual descriptions for the clips which have a maximum duration of a couple minutes. Liu et al. (2020) also introduce a similar dataset for video retrieval from movies and TV episodes.

Most prior work on video-based analysis on movies and TV shows focuses on methods for computing similarity or combining different modalities that contain complementary information, but they only consider short video clips with simple semantics. In this chapter, we propose a method for summarizing entire screenplays while also considering visual and audio information from the full-length videos, that have an average duration of 113 minutes.

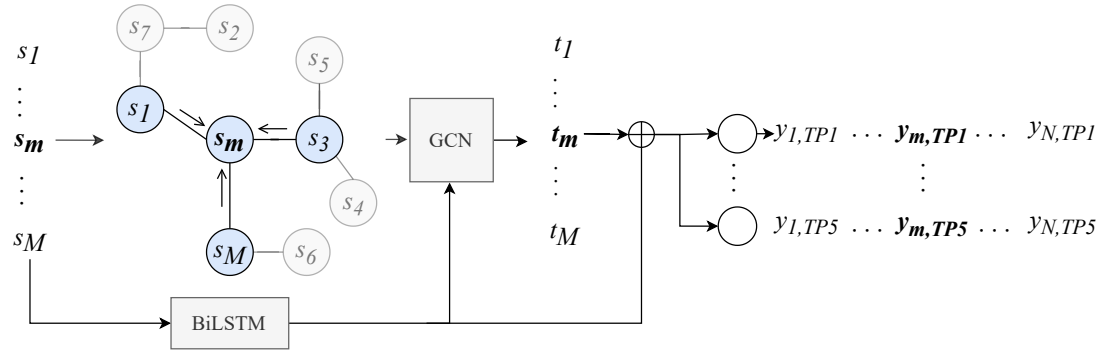
5.2 Problem Formulation

Let $\mathcal{D}$ denote a screenplay consisting of a sequence of scenes $\mathcal{D} = \{s_1, s_2, \ldots, s_M\}$. We aim at selecting a smaller subset $\mathcal{D}' = \{s_i, \ldots, s_K\}$ consisting of the most *informative* scenes describing the movie's storyline. Hence, our objective is to assign a binary label $y_i \in \{0,1\}$ to each scene s_i denoting whether it is part of the summary (where 1 means that s_i must be in the summary).

Furthermore, we hypothesize that we can construct an informative summary by identifying TPs directly. As we explained earlier, screenwriting theory (Hague, 2017) postulates that most movie narratives are delineated by five key events called turning points (see Figure 5.1). Hence, we re-formulate the summarization problem as follows: for each scene $s_i \in \mathcal{D}$ we assign a binary label y_{it} denoting whether it represents turning point t. Specifically, we calculate probabilities $p(y_{it}|s_i, \mathcal{D}, \theta)$ quantifying the extent to which s_i acts as the t^{th} TP, where $t \in [1,5]$ (and θ are model parameters). During inference, we compose a summary by selecting l consecutive scenes that lie on the peak of the posterior distribution $\text{argmax}_{i=1}^{M} p(y_{it}|s_i, \mathcal{D}, \theta)$ for each TP. We next describe in detail the graph-based model that we propose for TP identification.



(a) We compute the pairwise similarity between scenes based on both textual and audiovisual representations. We construct a fully-connected graph and then sparsify it by automatically selecting the k nearest neighbors per scene. During sparsification we first automatically decide on the number of neighbors per scene via a parameterized function.



(b) Next, we compute global scene representations based on the entire screenplay via a BiLSTM and local graph-based representations via an one-layer GCN. We finally combine the two representations for predicting which scenes represent each type of TP.

Figure 5.2: GRAPHTP model for TP identification based on multimodal information and graph-based scene representations.

5.3 A Turning Point Graph Model

5.3.1 Graph Construction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a directed screenplay graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. $\mathcal{G}$ consists of M nodes, each corresponding to a scene (M varies with screenplay size; some screenplays have many short scenes (for example up to 385 scenes), while others only a few long ones (for example 27 scenes)). We further represent $\mathcal{G}$ by an adjacency matrix $\mathcal{A} \in \mathcal{R}^{M \times M}$ where entry a_{ij} denotes the weight of the edge from node i to node j. We initially construct a dense complete graph $\mathcal{G}$ with edge weights representing the probability p_{ij} of scene i being a neighbor of scene j (see Figure 5.2a). We estimate

p_{ij} as:

$$p_{ij} = \frac{\exp(e_{ij}/\tau)}{\sum_{r=1}^{M} \exp(e_{ir}/\tau)} \tag{5.1}$$

where e_{ij} denotes the similarity between scenes s_i and s_j (explained in the next section), and τ is a temperature parameter for controling how sharp the distribution will be (where if $\tau < 1$ then there is higher sparsity and we assign higher probability mass to fewer neighbors).

Similarity Computation There are various ways to compute the similarity e_{ij} between two scenes. In addition to linguistic information based on the text of the screenplay, we wish to take advantage of other modalities, such as audio and video. Audiovisual cues might be relatively superficial; simply on account of two scenes sounding or seeming alike, it might not be possible to induce which events are being described and their relations. Nevertheless, we expect audiovisual information to contribute to the similarity computation by helping distinguish scenes which refer to the same sub-story or event, e.g., because they have the same background, the same characters, or similar noises. We thus express e_{ij} as a composite term based mostly on textual information but also modulated by audiovisual cues (see left part of Figure 5.2a):

$$e_{ij} = u_{ij}\left(\tanh(W_i n_i + b_i)^{\intercal} \tanh(W_j n_j + b_j)\right) + b_{ij} \tag{5.2}$$

where W_i and W_j are weight matrices, n_i and n_j are *textual* vectors representing the content of scenes s_i and s_j, and u_{ij} expresses the *audiovisual* similarity between s_i and s_j. Since the pairwise textual similarity is multiplied by u_{ij}, we create an AND gate, where high similarity scores are attributed to pairs of scenes that have both similar textual and audiovisual representations.

It is relatively straightforward to obtain textual representations for scenes. The latter contain mostly dialogue (lines the actors speak) as well as descriptions explaining what the camera sees. We first calculate representations for the sentences included in a scene via the same pre-trained transformer-based sentence encoder (Cer et al., 2018) as used in the previous chapters. We then obtain contextualized sentence representations using a BiLSTM equipped with an attention mechanism. A scene is represented as the weighted sum of the representations of its sentences.

We also assume that a scene corresponds to a sequence of audio segments extracted from the movie and a sequence of frames sampled (with a fixed sampling frequency) from the video. We first non-linearly project the features of each modality to a lower

dimension and obtain scene-level representations (as the attention-weighted average of the segments/frames in each scene). After computing scene-level unimodal representations, we concatenate the two vectors (i.e., for audio and video) and create a joint multimodal representation for the scene (late fusion; Frermann et al. 2018; Papasarantopoulos et al. 2019). Specifically, we first compute a scene-level representation per modality. Next, we concatenate all unimodal vectors and create a multimodal representation. The audiovisual similarity u_{ij} (applied in Equation (5.2)) between scenes s_i and s_j is the dot product of their fused representations.

Graph Sparsification Next, we sparsify graph $\mathcal{G}$ (or equivalently, matrix $\mathcal{A}$) by considering only k neighbors per scene (see right part of Figure 5.2a). Compared to fully connected graphs, sparse representations are computationally more efficient and also have shown better classification accuracy (Ozaki et al., 2011; Zhu, 2005). Moreover, we hypothesize that sparse graphs are crucial for our turning point identification task. We anticipate the screenplay graph to capture high-level differences and similarities between movies which would be difficult to discern when each scene is connected to every other scene.

The most common way to obtain a sparse graph is to construct a k-NN graph by introducing a threshold on the number of nearest neighbors k (Szummer and Jaakkola, 2003; Goldberg and Zhu, 2006; Niu et al., 2005). Specifically, we create sparse graph $\mathcal{G}'$ by selecting the set of neighbors $\mathcal{P}_i$ for each scene s_i as follows:

$$\mathcal{P}_i = \text{argmax}_{j\in[1,M],|\mathcal{P}_i|=k}\, p_{ij} \tag{5.3}$$

where p_{ij} is the normalized similarity between scenes s_i and s_j and is calculated as in Equation (5.1). After removing for each node the neighbors not included in the set $\mathcal{P}_i$, the new graph $\mathcal{G}'$ contains edges $|\mathcal{E}'| \ll |\mathcal{E}|$ which are unweighted[1].

Instead of a priori deciding on a fixed number of neighbors k for all scenes, which may cause false neighborhood assumptions, we treat k as a parameter to be learned as part of the network which computes $p(y_{it}|s_i,\mathcal{D})$, the probability of a scene being a TP. The right part of Figure 5.2a illustrates this neighborhood selection module. All unnormalized outgoing edge weights e_i from $s_i \in \mathcal{G}$ serve as input to a fully-connected layer which outputs a probability distribution w_i:

$$w_i = \text{softmax}(W_n e_i + b_n) \tag{5.4}$$

[1]We find that considering all remaining neighbors as equally important for the neighborhood representation provides better results than taking into account and re-normalizing the similarity-based weights of the edges

over a pre-defined set of neighborhood sizes $[1,C]$. We then select $k_i = \text{argmax}_{t\in[1,C]} w_{it}$ as the neighborhood size for scene s_i in the sparse graph $\mathcal{G}'$.

When deciding on the neighborhood size k_i and the set of neighbors $\mathcal{P}_i$ for scene s_i, we perform discrete choices, which are not differentiable. We address these discontinuities in our model by utilizing the Straight-Through Estimator (Bengio et al., 2013). During the backward pass we compute the gradients with the Gumbel-softmax reparametrization trick (Maddison et al., 2017; Jang et al., 2017). To better approximate the argmax selections (for k and $\mathcal{P}$) during backpropagation, we also add a low temperature parameter $\tau = 0.1$ (Hinton et al., 2015) in the softmax function, shown in Equation (5.1).

5.3.2 Graph Convolutional Networks

After creating the sparse movie graph $\mathcal{G}'$, we next want to encode the connections between scenes in the graph. For that, we rely on graph convolutional networks (GCNs; Duvenaud et al. 2015; Kearnes et al. 2016; Kipf and Welling 2017) to induce embeddings representing graph nodes. Our GCN operates over the sparsified graph $\mathcal{G}'$ and computes a representation for the current scene s_i based on the representation of its neighbors. We only encode information from the scene's *immediate* neighbors and thus consider one layer of convolution.[2] Moreover, in accordance with Kipf and Welling (2017), we add a self-loop to all scenes in $\mathcal{G}'$[3]. This means that the representation of scene s_i itself affects the neighborhood representation t_i:

$$t_i = f\left(\frac{1}{|\mathcal{P}_i \cup \{s_i\}|} \sum_{j\in P_i\cup\{s_i\}} \left(W_g c_j + b_g\right)\right) \tag{5.5}$$

where $f(.)$ is a non-linear activation function (i.e., ReLU), vectors c represent the contextualized *content* of a scene (in relation to the overall screenplay and its relative position), and $\mathcal{P}_i$ is the set of neighbors for scene s_i.

We encode the screenplay as a sequence $n_1, n_2, ..., n_M$ of textual scene representations with a BiLSTM network and obtain contextualized representations by concatenating the hidden layers of the forward $\overrightarrow{h}$ and backward $\overleftarrow{h}$ LSTM[4]. In other words,

[2] We empirically observe that performance deteriorates when stacking GCN layers.

[3] Kipf and Welling (2017) showed that adding self-loops improves accuracy in graph-based classification and encourages the computation of locally smooth features across the graph.

[4] We experimented with including multimodal information from the video and audio as well for contextualizing scenes via the BiLSTM, but this did not yield any improvement in performance. Presumably, audio and visual information does not improve contextualization, since screenplays already

graph convolutions are performed on top of LSTM states (Marcheggiani and Titov, 2017). The one-layer GCN only considers information about a scene's immediate neighbors, while contextualized scene representations capture longer-range relations between scenes. The left part of Figure 5.2b illustrates our GCN and the computation of the neighborhood representations t.

Finally, we concatenate the neighborhood representation t_i and the content representation c_i to obtain an encoding for each scene s_i: $[c_i;t_i]$. This vector is fed to a single neuron that outputs probabilities $p(y_{it}|s_i,\mathcal{D})$ (see right part of Figure 5.2b).

5.3.3 Model Training

Our description so far has assumed that TP labels are available for screenplay scenes. However, in practice, such data cannot be easily sourced (due to the time consuming nature of watching movies, reading screenplays, and identifying TP locations). In Chapter 3, we collected sentence-level TP annotations for plot synopses and proposed ways to project these annotations into screenplays for identifying which scenes act as TPs. Following this direction, we now first train a teacher model which takes as input *synopses marked with gold standard TP sentences* and *the corresponding screenplays D* and outputs the *probability* $q(y_{it}|s_i,D)$ *for scene* s_i *to convey the meaning of the* t^{th} *TP sentence*, where $t \in [1,5]$.

We use the same model as in Chapter 3 (i.e., Multi Context-aware Model from Task 2) as our teacher model to obtain probability distribution $q(y_t|D)$ over screenplay D. Instead of creating silver standard labels based on the TP identification model of Chapter 3, we utilize the posterior distributions of this network for training our model, which only takes scenes as input, similarly to knowledge distillation settings (Ba and Caruana, 2014; Hinton et al., 2015). We hypothesize that by allowing our model to access the soft TP-specific posterior distributions produced by the teacher model we will provide a stronger training signal in comparison with using (noisy) hard binary labels. Specifically, we utilize the KL divergence loss between the teacher posterior distributions $q(y_t|D)$ and the ones computed by our model $p(y_t|D)$:

$$O_t = \mathcal{D}_{KL}\left(p(y_t|D)\middle\|q(y_t|D)\right), t \in [1,T] \tag{5.6}$$

where T is the number of TPs. We further add a second objective to the loss function in order to control adjacency matrix S and hence the latent graph $\mathcal{G}'$. Intuitively,

contain important non-verbal information (e.g., actions, emotions, characters), but the multimodal information is still helpful for creating meaningful movie graphs.

we want to assign higher probabilities for scenes to be neighbors in $\mathcal{G}'$ if they are also temporally close in the screenplay. For this reason, we add a focal regularization term $\mathcal{F}$ to the loss function. Specifically, we assume a Gaussian distribution g_i over the screenplay centered around the current scene index i and try to keep the probability distribution in matrix S that corresponds to candidate neighbors for scene s_i close to the prior distribution: $\mathcal{F}_i = \mathcal{D}_{KL}(p_i \| g_i)$.[5] The loss function now becomes:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} O_t + \lambda \frac{1}{M}\sum_{i=1}^{M} \mathcal{F}_i \tag{5.7}$$

where λ is a hyperparameter, T is the number of different TPs (i.e., 5) and M is the number of scenes in the screenplay.

5.4 Experimental Setup

Multimodal TRIPOD In this Chapter, we use the TRIPOD dataset introduced and described in detail in Chapter 3. However, we now use the multimodal version of the dataset by taking into consideration the visual and audio information from the full-length video of the movies[6]. Table 5.1 revises the dataset statistics.

Data Preprocessing As in the previous chapters, we again used a pre-trained transformer encoder (USE; Cer et al. 2018) to obtain sentence-level representations. Following previous work (Tapaswi et al., 2015a), subtitles (and their timestamps on the movie video) were aligned to the dialogue parts of the screenplay using Dynamic Time Wrapping (DTW; Myers and Rabiner 1981). Subsequently, we obtained alignments of screenplay scenes to video segments. Finally, we segmented the video into scenes and extracted audiovisual features.

For the visual modality, we first sampled one out of every 50 frames within each scene. However, the length of a scene can vary from a few seconds to several minutes. For this reason, in cases where the number of sampled frames became too big for memory, we lowered the sampling frequency to one frame per 150. We employed ResNeXt-101[7] (Xie et al., 2017) pre-trained for object recognition on ImageNet (Deng

[5] We disregard τ (see Equation (5.1)) while recalculating probabilities p_{ij}, since we want to directly regulate the e_{ij} values.

[6] We make multimodal features for this dataset publicly available: https://datashare.ed.ac.uk/handle/10283/3819

[7] This is a modified version of ResNet (He et al., 2016) where convolutions inside the bottleneck block have been substituted by grouped convolutions. Specifically, the network is constructed by re-

	Train	Test
movies/scenes	84/11,320	38/5,830
synopsis vocabulary	13.0k	6.8k
screenplay vocabulary	45.3k	28.3k
	per movie	
scenes	133.0 (61.1/27/385)	153.4 (54.0/42/299)
sentences	3.0k (0.9/0.6/5.5)	2.9k (0.6/1.6/4.5)
tokens	23.0k (6.6/0.5/41.9)	21.5k (4.0/12.7/30.5)
video length (secs)	6.8k (1.1/4.2/10.3)	6.9k (1.3/3.5/10.2)
	per scene	
sentences	22.2 (31.5/1/684)	19.0 (24.9/1/433)
tokens	173.0 (235.0/3/4875)	139.9 (177.5/5/2793)
sentence tokens	7.8 (6.0/1/220)	7.4 (6.0/1/106)
video length (secs)	88.1 (152.5/2/6317)	81.6 (114.8/2/1356)

Table 5.1: Statistics of TRIPOD dataset; means are shown with standard deviation/minimum/maximum in parentheses.

et al., 2009) to extract a visual representation per frame. Similarly, for the audio modality, we used YAMNet[8] pre-trained on the AudioSet-YouTube corpus (Gemmeke et al., 2017) for classifying audio segments into 521 audio classes (e.g., tools, music, explosion); for each audio segment contained in the scene, we extracted features from the penultimate layer (out of the 28 layers of the model).

Implementation Details Following Chapter 3, we select $l = 3$ consecutive scenes to represent each TP in the summary. Moreover, we set the maximum size of neighbors C that can be selected for a scene in graph $\mathcal{G}'$ to 6, since we want to create a sparse and interpretable graph. Experiments with fixed-sized neighborhoods also showed that performance dropped when considering neighborhoods over 6 scenes. For training our model we set the hyperparameter λ in Equation (5.7), that regulates the contribution of the focal loss term to the objective, to 10. We used the Adam algorithm (Kingma

peating a building block and aggregates a set of transformations for image classification. By adding these repeated blocks, the network adds an extra dimension (on top of depth and width) called "cardinality" which is equal to the size of the set of transformations.

[8]YAMNet is a pre-trained network based on the MobileNet architecture (Howard et al., 2017) that uses efficient convolutions for audio classification. YAMNet is often also used as a feature extractor for audio representations.

and Ba, 2015) for optimizing our networks. We chose an LSTM with 64 neurons for encoding scenes in the screenplay and an identical one for contextualizing them. We also added a dropout of 0.2. Our models were developed in PyTorch (Paszke et al., 2019) and PyTorch geometric (Fey and Lenssen, 2019). For analyzing the movie graphs we used NetworkX (Hagberg et al., 2008).

5.5 Results and Analysis

Our experiments were designed to answer three questions: (1) Is the proposed graph-based model better at identifying TPs compared to less structure-aware variants? (2) To what extent are graphs and multimodal information helpful? and (3) Are the summaries produced by automatically identified TPs meaningful?

5.5.1 Which Model Identifies TPs Best?

Table 5.2 addresses our first question. We perform 5-fold cross-validation over 38 gold standard movies to obtain a test-development split and evaluate model performance in terms of three metrics: Total Agreement (Total Agr), i.e., the percentage of TP scenes that are correctly identified, Partial Agreement (Part Agr), i.e., the percentage of TP events for which at least one gold standard scene is identified, and Distance (D), i.e., the minimum distance in number of scenes between the predicted and gold standard set of scenes for a given TP, normalized by the screenplay length. We consider Total Agr and Part Agr as our main evaluation metrics as they measure the percentage of exact TP matches. However, apart from identifying important events, when producing a summary it is also important to display events from all parts of the movie in order to accurately describe its storyline. For this reason, we also employ the distance D metric which quantifies how well distributed the identified TP events are in the movie. Hence, a disproportionately large D suggests that the model fails to even predict the correct sections of a movie where TPs might be located let alone the TPs themselves. We have discussed the definition and purpose of these automatic evaluation metrics in more detail in Chapter 3.

The first block in the table compares our graph-based TP identification model (henceforth GRAPHTP) against the following baselines: a random selection of a sequence of three scenes from five evenly segmented sections of the movie (reported mean of five runs); the selection of a sequence of three scenes that lie on the expected

	Total Agr ↑	Part Agr ↑	D ↓
Random (evenly distributed)	4.82	6.95	12.35
Theory position	4.41	6.32	11.03
Distribution position	5.59	7.37	10.74
TEXTRANK	6.18	10.00	17.77
+ audiovisual	6.18	10.00	18.90
SCENESUM	4.41	7.89	16.86
+ audiovisual	6.76	11.05	18.93
TAM	7.94	9.47	**9.42**
+ audiovisual	7.36	10.00	10.01
GRAPHTP	6.76	10.00	9.62
+ audiovisual	**9.12**	**12.63**	9.77

Table 5.2: Five-fold crossvalidation. Total Agreement (Total Agr), Partial Agreement (Part Agr), and mean distance D. TEXTRANK and SCENESUM are unsupervised graph-based summarization models, TAM is the topic-aware model for TP identification which has similar architecture to MCAM of Chapter 3 but only considers screenplay-based information, and GRAPHTP is our proposed model for TP identification that also encodes graph-related information.

position of each TP event according to screenwriting theory (Hague, 2017); and the selection of a sequence of three scenes based on the position of gold standard TPs in the synopses of the TRIPOD training set. The second block includes the performance of two unsupervised summarization models: TEXTRANK (Mihalcea and Tarau, 2004) with neural input representations (Zheng and Lapata 2019; Chapter 4) and SCENESUM (Gorinski and Lapata 2015; Chapter 4), a variant of TEXTRANK that takes the characters participating in each scene into account. As discussed in the previous chapter, TEXTRANK creates a fully-connected graph where nodes are scenes and edges between scenes denote the degree of similarity. Next, it computes centrality scores per scene given all outgoing edges and ranks them from most to least central (i.e., informative). SCENESUM is a variation of TEXTRANK that also considers the characters participating in a scene (i.e., main characters over total characters that appear) for computing centrality scores per scene.

Finally, we report results for the Topic-Aware Model (TAM), which has a similar architecture to the Multi Context-Aware Model (MCAM; see Chapter 3) but only

considers screenplays without the corresponding plot synopses and gold-standard TP sentences (as in Tasks 2 and 3 of Chapter 3). As discussed in Chapter 3, MCAM, and hence TAM, are sequence-based supervised models which employ a sliding context window and compute the similarity between *sequential* contexts. In contrast to TAM, GRAPHTP does not assume a sequential order for the scenes in the screenplay. Instead of computing contextualized scene representations based on the few previous and following scenes in the screenplay via the sliding context window, GRAPHTP learns which the most relevant scenes are via a graph structure and then encodes these interactions via a GCN.

We also report the performance of a multimodal variant for all comparison systems (+audiovisual). For the unsupervised models, we add scene-level features as extra weights to the pairwise similarity calculation between scenes similarly to GRAPHTP. Hence, the fully-connected graphs for these models are created in a similar way to our model. For TAM, we add audiovisual information via early fusion. Specifically, we concatenate the scene-level vectors from all modalities (i.e., text, vision, audio) after applying L2 normalization.

The unsupervised summarization models (TEXTRANK, SCENESUM) have competitive performance in terms of Total Agr and Part Agr, but significantly higher average distance D. This suggests that they do not select events from all parts of a story but favor specific sections. For the supervised models (TAM and GRAPHTP), the average D is in general lower, which means that they are able to adapt to the positional bias and select events from all parts of a movie. Moreover, both models seem to benefit from multimodal information (see Total Agr and Part Agr metrics). Finally, GRAPHTP seems to perform best, by correctly identifying a higher number of gold standard TP events (based on both Total Agr and Part Agr metrics), whereas D is comparable for TAM and GRAPHTP.

5.5.2 Which Information Matters?

Table 5.3 answers our second question by presenting an ablation study on GRAPHTP. We observe that the performance of a similar model which uses a fully-connected graph drops across metrics. This is also the case when we do not take into account a graph or any other form of interaction between scenes (i.e., only content). We also test the model's performance when we remove the content representation and keep only the neighborhood interactions together with a vector that simply encodes the position of a

	Total Agr ↑	Part Agr ↑	D ↓
Fully connected graph	5.00	7.37	9.73
Content	5.59	7.37	9.98
Neighborhood & position	**9.71**	**13.16**	10.98
GRAPHTP	6.76	10.00	**9.62**
+ vision	6.18	7.89	9.84
+ audio	7.06	8.95	10.38
+ audiovisual	9.12	12.63	9.77

Table 5.3: GRAPHTP variants for TP identification. Total Agreement (Total Agr), Partial Agreement (Part Agr), and mean distance D.

scene in the screenplay (as a one-hot vector). This model may not be able to adapt to the positional bias as well (higher D), but is able to predict TPs with higher accuracy (high Total Agr and Part Agr). Finally, we find that audio and visual information boost model performance in combination but not individually. We hypothesize that combining audio and visual cues could lead to more meaningful graphs where scenes containing the same background audio and visuals are deemed as neighbors, whereas using the modalities individually may not correctly filter similar and dissimilar scenes.

5.5.3 How Good Are the Summaries?

We now answer our last question by evaluating the video summaries produced by our model. We conducted a human evaluation experiment using the videos created for 10 movies of the test set[9]. We produced summaries based on the gold scene-level annotations (Gold), SCENESUM, TAM, and GRAPHTP, which are the best performing models according to the automatic evaluation of Table 5.2. For all systems we used model variants which consider audiovisual information, since it consistently improves performance on TP identification, except for SCENESUM. Inclusion of audiovisual information for this model yields overly long summaries (in the excess of 30 minutes) for most movies. All other systems produce 15 minutes long summaries on average.

Our study was conducted on Amazon Mechanical Turk (AMT). We created textual summaries describing a movie's storyline from beginning to end. Specifically, we produced a shorter version of the Wikipedia plot synopsis for each movie keeping

[9]The videos used for human evaluation are available at https://github.com/ppapalampidi/GraphTP/blob/main/TRIPOD_video_summaries.csv.

	SCENESUM	TAM	GRAPHTP	Gold
TP1	28	28	**64**	66
TP2	36	54	**64**	62
TP3	38	18	**44**	54
TP4	26	34	**52**	56
TP5	8	16	**24**	48
Mean	27	30	**50**	57
Rating	2.63	2.68	**3.02**	3.58

Table 5.4: Human evaluation; proportion of TPs found in video summaries (shown as percentages) and average ratings attributed to each system (ratings vary from 1 to 5, with 5 being best). SCENESUM is unsupervised graph-based summarization model that builds on top of TEXTRANK and considers character-based information, TAM is the topic-aware model for TP identification which has similar architecture to MCAM of Chapter 3 but only considers screenplays (and not the corresponding synopses), and GRAPHTP is our proposed model for TP identification that also encodes graph-related information. All pairwise differences are significant ($p < 0.05$, using a χ^2 test).

only essential information. Next, we colored differently the text in the summary that corresponded to each TP in order to clearly demonstrate the important events in the movie.

We asked AMT workers to first read this textual summary paying extra attention to the descriptions in the colored text. Next, they were asked to watch a video summary for the movie and answer five questions. Each question examined whether a specific TP event was presented in the video summary. AMT workers answered with 'Yes' if they were certain it was present in the video, 'No' if the event was absent, and 'Unsure' otherwise. Examples of the summaries given to the AMT workers alongside with the corresponding questions are presented in Tables 5.6, 5.7, and 5.8. Finally, we asked AMT workers to provide an overall rating from 1 to 5, with 5 being the most informative summary. For the overall rating, we asked crowdworkers to take into account the questions answered previously, but also consider the quality of the summary (i.e., how condensed the summary was, whether it contained redundant events, overall information provided). We provide more details on the human evaluation experiment in Appendix C.3.

Table 5.4 shows the proportion of 'Yes' answers (per TP and overall mean) and the

Category	Movies
Comedy/Romance	'Juno', 'The Back-up Plan', 'The Breakfast Club', '500 Days of Summer', 'Crazy, Stupid, Love', 'Easy A', 'Marley & Me', 'No Strings Attached'
Thriller/Mystery	'Arbitrage', 'Panic Room', 'The Shining', 'One Eight Seven', 'Black Swan', 'Gothika', 'Heat', 'House of 1000 Corpses', 'Sleepy Hollow', 'The Talented Mr. Ripley', 'The Thing'
Action	'Die Hard', 'Soldier', 'The Crying Game', 'Total Recall', '2012', 'From Russia with Love', 'American Gangster', 'Collateral Damage','Oblivion'
Drama/Other	'Moon', 'Slumdog Millionaire', 'The Last Temptation of Christ', 'Unforgiven', 'American Beauty', 'Jane Eyre', 'The Majestic', 'A Walk to Remember'

Table 5.5: Movies from test set divided in four broad categories based on their genre.

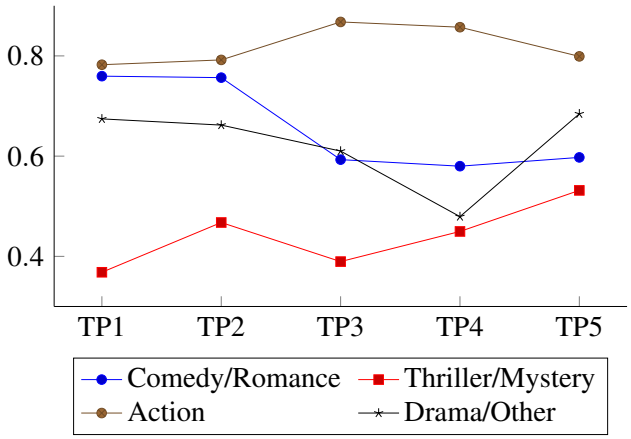


Figure 5.3: Average node connectivity per TP across movie genres (GRAPHTP (+audiovisual), test set.)

average system rating.[10] Perhaps unsurprisingly gold summaries are the most informative. Some key events might still be absent due to errors in the automatic alignment between the screenplay scenes and the video. GRAPHTP is the second best system overall (and across TPs), while SCENESUM and TAM have similar ratings. GRAPHTP manages to create more informative and diverse summaries, presenting important events from all parts of the story.

5.5.4 What Do the Graphs Mean?

We further analyzed the graphs induced by our model, in particular their connectivity. Figure 5.3 shows the average node connectivity per TP (i.e., minimum number of nodes that need to be removed to separate the remaining nodes into isolated subgraphs) for the

[10]We omit 'Unsure' from Table 5.4, since it only accounts for 4.1% of the answers.

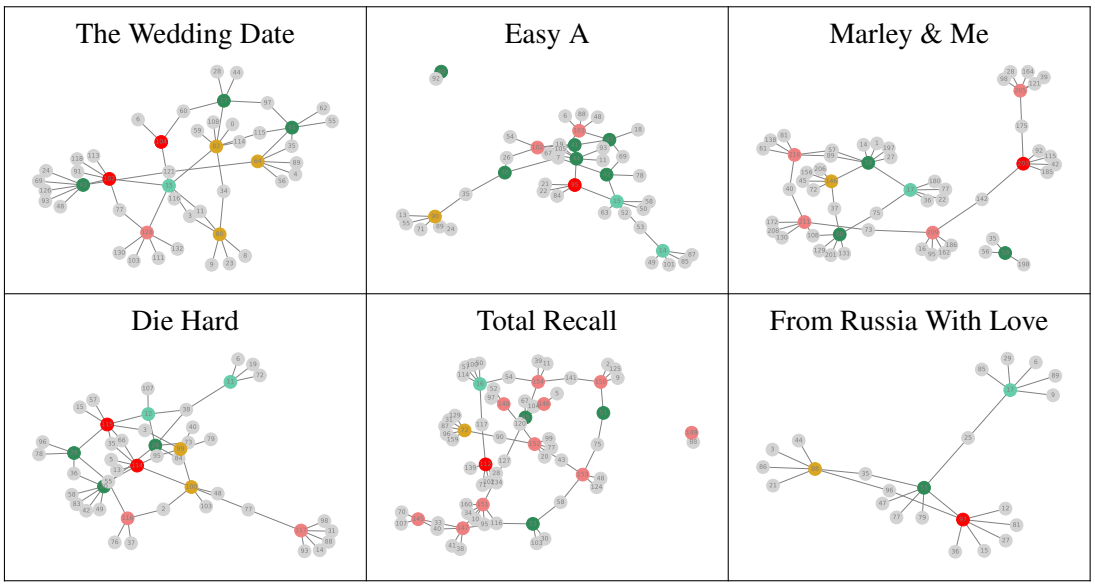


Figure 5.4: Examples of graphs produced by GRAPHTP (+audiovisual) for movies from the test set. Nodes (in color) are scenes which act as TPs and their immediate neighbors (in gray). Genre category per row: comedy/romance, action. All graphs are dense and present high connectivity.

movies in the test set. For this analysis, graphs were pruned to nodes which act as TPs and their immediate neighbors and movies were grouped in four broad genre categories (comedy/romance, thriller/mystery, action and drama/other). In Table 5.5, we present how the movies are categories into genres. When a movie represents more than one genres, we assign only the main one. Moreover, we consider the pruned graphs as a graphical description of a movie's storyline containing only the most important events and their semantic connections.

Based on Figure 5.3, we find that thrillers and mysteries correspond to more disconnected graphs followed by dramas, while comedies, romance and especially action movies display more connected graphs. This is intuitive, since comedies and action movies tend to follow predictable storylines, while thrillers often contain surprising events which break screenwriting conventions. Moreover, for comedies and dramas the introductory events (i.e., first two TPs) are the central ones in the graph and connectivity decreases as the story unfolds and unexpected events take place. We see the opposite trend for thrillers and action movies. Initial events present lower connectivity, while the last ones are now central when crucial information is revealed justifying earlier actions (e.g., see the last two TPs which correspond to 'major setback' and 'climax'). A similar picture emerges when visualizing the graphs (see Figures 5.4

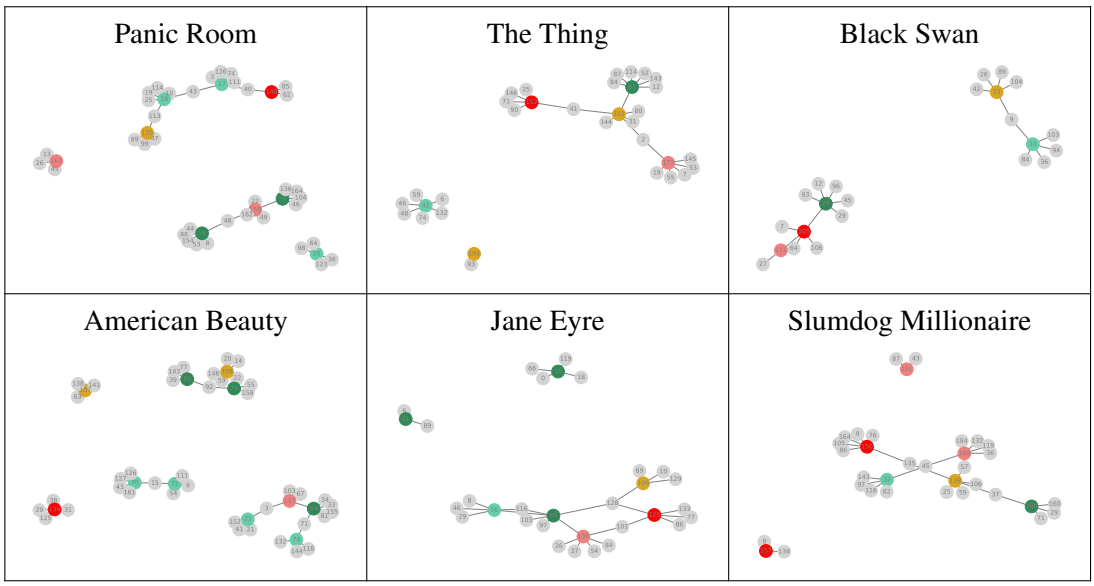


Figure 5.5: Examples of graphs produced by GRAPHTP (+audiovisual) for movies from the test set. Nodes (in color) are scenes which act as TPs and their immediate neighbors (in gray). Genre category per row: thriller/mystery, drama/other. The graphs here present low connectivity containing several disconnected subgraphs.

and 5.5). Figure 5.4 contains movies belonging to the comedy/romance and action genre categories per row, respectively. Similarly, Figure 5.5 presents illustrations of graphs for thriller/mystery and drama/other movies per row. We can empirically observe that comedies and action movies (Figure 5.4) tend to present denser graphs in comparison with thrillers and dramas (Figure 5.5) which contain several disconnected subgraphs.

Except for each movie's genre, we also consider the quality of the movie when analyzing its graph. Specifically, we hypothesize that movies with high ratings (e.g. according to Rotten Tomatoes[11]) often have less predictable and distinctive storylines and hence would present a different graph topology in comparison with those of poorly rated movies. In order to test this hypothesis, we first analyze the graph topology using several metrics from graph theory. Specifically, as when examining the graphs with regard to different genres, we again consider the average node connectivity and the node connectivity per TP. However, we now also compute four extra metrics: the TP pairwise node connectivity (i.e., the average node connectivity when considering only the nodes that act as TPs in order to directly measure the degree of connectivity between TP events), the number of disconnected components presented in the graph, its

[11] https://www.rottentomatoes.com/

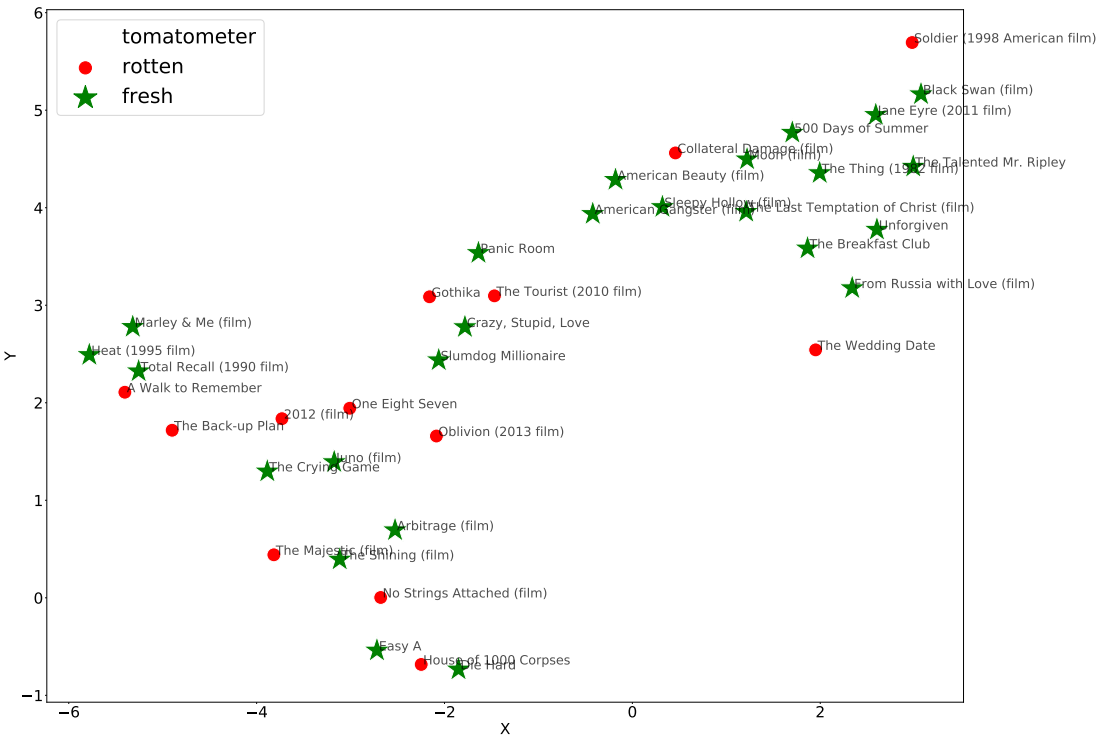


Figure 5.6: Analysis of the graphs for the movies of the test set with regard to their scores in Rotten Tomatoes. We computed 10 different metrics, such as node connectivity, diameter and triadic closure, that describe each movie graph. Next, we performed dimensionality reduction using UMAP given the values of these metrics and visualize all movies in two dimensions. A small distance between movies in the figure indicates structural similarity between their graphs. Regarding the scores, rotten are the movies with a score lower than 60%, otherwise they are fresh.

diameter (i.e., the greatest distance between any pair of nodes) and the triadic closure (i.e., tendency of edges to form triangles). Next, we perform dimensionality reduction using Uniform Manifold Approximation and Projection (UMAP)[12] (McInnes et al., 2018) in order to project the movies of the test set to two dimensions and visualize them.

We present the visualization of the movies in two dimensions in Figure 5.6. Movies are colored differently depending on the score that they received in Rotten Tomatoes (i.e. movies with a score equal or higher than 60% are considered fresh (green star), otherwise rotten (red circle)). Although movies are not perfectly divided into clus-

[12]UMAP is a non-linear dimensionality reduction technique that provides fast visualizations, scales well with the number of dimensions and dataset size, and tends to better preserve the global structure of the data in comparison with other techniques such as t-Distributed Stochastic Neighbor Embedding (t-SNE; Van der Maaten and Hinton 2008).

ters according to their score category ($< 60\%$ vs. $\geq 60\%$), we observe that there is a tendency for movies with high scores to concentrate in the upper right side of the figure, whereas a lot of poorly rated movies are located in the leftmost, down side. This suggests that although each movie has a unique storyline, and hence graph, the way that important events are connected plays a significant role to the quality of the movie. Notably, a lot of well-known unconventional movies with out-of-the-box storylines are located in the "fresh" cluster in the figure: e.g., "American Beauty", "Black Swan", "The Thing', "Moon".

5.5.5 Discussion

In Figure 5.7, we visualize the posterior distribution over the scenes of the screenplay for the movie "Juno" for the text-only and multimodal variants of SCENESUM, TAM, and GRAPHTP. This figure is equivalent to the visualization presented in Figure 3.7 of Chapter 3. The first two panels show the posterior distributions of textual and multimodal SCENESUM alongside the gold standard TP scenes for each TP (vertical lines). We observe that textual SCENESUM attributes high probability for inclusion to summary to scenes that are far away from all gold standard key events (i.e., TPs). For example, we observe multiple peaks between scenes 60 and 80, which are not related to either of the third and fourth TPs. In contrast, when we add multimodal information for creating the fully connected graph used for the centrality calculation, we observe that the peaks in the probability distribution are closer to all important events, whereas the region in the middle of the screenplay (i.e., scenes 60 to 80) that was considered important by textual SCENESUM now concentrates less probability mass.

In the next four panels we present TAM and GRAPHTP, which are both trained on TP identification, when considering only textual and multimodal information. We now observe that the probability distributions per TP for all models are similar following the position bias per TP. Interestingly, models seem to be more certain for the first and last TP and least certain for the second and third TP, which they fail to accurately predict. Comparing the text-only and multimodal variants, we observe that models have higher certainty for TPs 1, 4, and 5 when adding the multimodal information. Among all four variants, TAM with textual information seems to provide the least accurate predictions, especially for the third TP, which is the hardest to identify.

Finally, when comparing these distributions with the ones presented in Figure 3.7 of Chapter 3, we draw two conclusions. Firstly, all supervised models trained with

TP labels (TAM, GRAPHTP) learn the position bias as presented in the distribution baseline in Figure 3.7. Secondly, TAM and GRAPHTP have sharper probability distributions around the correctly identified TPs in comparison with both CAM and MCAM, which show smoother distributions for all TPs and attribute high probability to irrelevant scenes as well. For example, GRAPHTP with multimodal information correctly predicts TP4 with high certainty in comparison with MCAM in Figure 3.7.

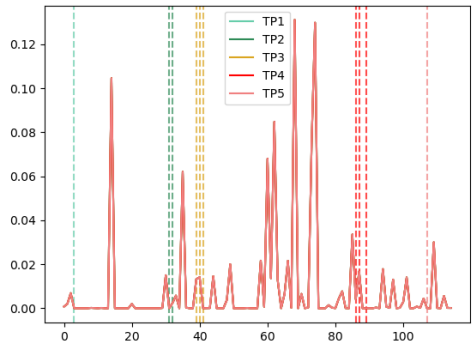
Given the comparisons in this example, we overall conclude that TP position bias is important for predicting key events (e.g., SCENESUM vs. GRAPHTP) and that multimodal information increases certainty for correctly identified TPs.

5.6 Summary of Chapter

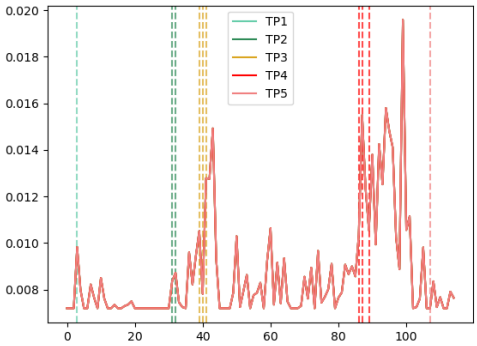
In this chapter we demonstrated that TP identification can be used directly for summarizing movies, alleviating the need for further manual annotation. Moreover, we extended the summarization task to a multimodal setting by also considering the full-length video and audio for the movies and proposed GRAPHTP, a model that operates over sparse graphs relying on the multimodal information. Based on our experimental results, we conclude that (1) modeling movies as sparse graphs instead of treating them as a sequence of scenes can offer better contextualization and leads to interpretable and meaningful graph structures, and (2) multimodal information is especially helpful in creating meaningful graphs in the latent space (see Table 5.3), possibly due to superficial similarity (e.g., same background visuals or audio) of events that belong to the same substory.

However, there are certain limitations to our current approach. First, while we now take into account audiovisual information from the full-length video, we still consider screenplays as our primary source of information. Although screenplays contain rich (verbal and non-verbal) information (i.e., dialogue, characters, emotions, descriptions, actions), they are not always available or sometimes deviate significantly from the actual movie. Second, we consider *scenes* as our unit, which is the natural unit in screenplays, for creating video summaries. Although scenes describe self-contained events, concerned with a fixed set of characters, location and topic, they can be several minutes long, which results in lengthy video summaries of 15 minutes on average. Finally, our main objective so far has been to deliver key information and present a movie storyline to viewers. Key information is crucial for high-quality movie summaries; however, there are additional important aspects to video summarization, such

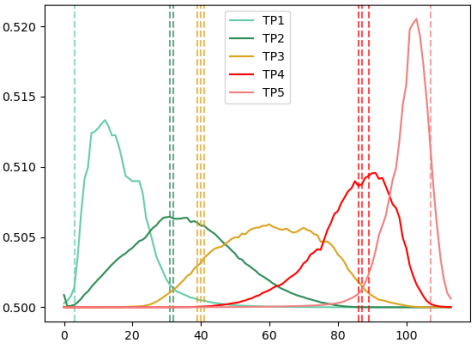
as the *attractiveness* and *coherence* of the output videos. In the next chapter, we will address these limitations by addressing a special instiation of movie summarization: trailer creation.



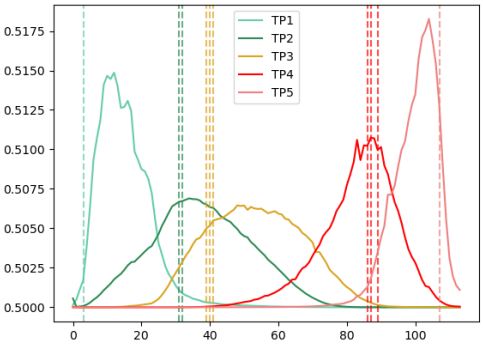
SCENESUM with textual information.



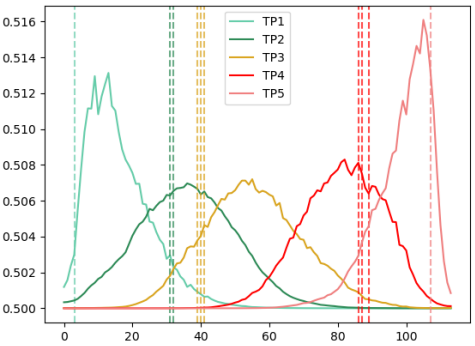
SCENESUM with multimodal information.



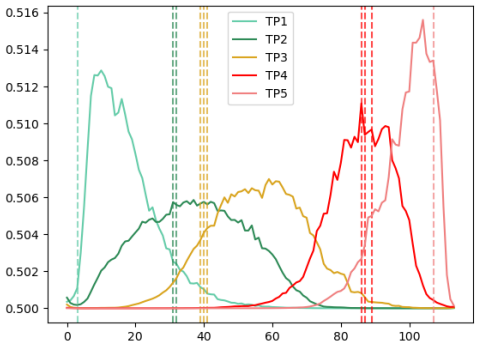
TAM with textual information.



TAM with multimodal information.



GRAPHTP with textual information.



GRAPHTP with multimodal information.

Figure 5.7: Probability distributions over the scenes of the screenplay for the movie “Juno”. X-axis: scene indices, y-axis: probability that the scene is relevant to a specific TP. Vertical dashed lines are gold standard TP-scenes. These distributions could be compared with the equivalent ones presented in Chapter 3.

Movie	Summary	Questions
Arbitrage	Sixty-year-old magnate Robert Miller manages a hedge fund with his daughter Brooke and is about to sell it for a handsome profit. However, unbeknownst to his daughter and most of his other employees, he has cooked his company's books in order to cover an investment loss and avoid being arrested for fraud. One night, while driving with his mistress Julie Cote, he begins to doze off and crashes; Julie is killed. Miller covers up Julie's death with Jimmy's help. The next day, he is questioned by police detective Bryer. Bryer is keen on arresting Miller for manslaughter and begins to put the pieces together. Jimmy is arrested and placed before a grand jury but still refuses to admit to helping Miller. The case against Jimmy is dismissed and the detective is ordered not to go near him. Miller's wife tries to blackmail him with a separation agreement getting rid of his wealth. In the final scene, Miller addresses a banquet honoring him for his successful business, with his wife at his side and his daughter introducing him to the audience but their false embrace on the stage signifies that he has lost the respect and admiration of his daughter.	1. Did the video summary show the accident and Julie's death? 2. Did the video summary show Bryer's suspicions towards Miller? 3. Did the video summary show Jimmy's arrest and the police's efforts to get Jimmy to admit helping Miller? 4. Did the video summary show that Miller's wife blackmailed him? 5. Did the video summary show Miller give a speech about his successful bussiness with his family on his side in the end?

Table 5.6: Examples of movies used for human evaluation. We present a short text summary per movie based on Wikipedia and the questions we asked for evaluating the information contained in each video summary. Different colors in the summary correspond to different TPs (i.e., TP1, TP2, TP3, TP4, TP5).

Movie	Summary	Questions
The Back-up Plan	Zoe has given up on finding the man of her dreams and decided to become a single mother and undergoes artificial insemination. The same day she meets Stan when they both try to hail the same taxi. After running into each other twice more, Stan asks Zoe on a date. At the end of the date Stan asks her to come to his farm during the weekend and Zoe finds out that she is pregnant. In the farm Zoe tells Stan that she is pregnant and he is confused and angry at first. However, Stan decides he still wants to be with her and they reconcile. After many misunderstandings and comedic revelations, Zoe and Stan are walking into the Market when they run into Stan's ex-girlfriend. Due to Stan's remark that the twins are not his, Zoe believes that he is not ready to become a father to them, and breaks off the relationship. At her grandmother's wedding, Zoe's water breaks and on the way to the hospital they make a pit stop at the Market. Zoe apologizes to Stan and they begin to work things out. He pulls out a penny, that he kept from their first acquaintance, and Zoe promises to trust him more.	1. Did the video summary show Zoe and Stan's first encounter in a taxi? 2. Did the video summary show Zoe discover that she is pregnant? 3. Did the video summary show Stan's commitment to Zoe after finding out that she is pregnant? 4. Did the video summary show Zoe and Stan's break up? 5. Did the video summary show the final reconciliation between Stan and Zoe?

Table 5.7: Examples of movies used for human evaluation. We present a short text summary per movie based on Wikipedia and the questions we asked for evaluating the information contained in each video summary. Different colors in the summary correspond to different TPs (i.e., TP1, TP2, TP3, TP4, TP5).

Movie	Summary	Questions
Juno	Sixteen-year-old Minnesota high-schooler Juno MacGuff discovers she is pregnant with a child fathered by her friend and longtime admirer, Paulie Bleeker. She initially considers an abortion, but finally decides to give the baby up for adoption. Juno meets a couple, Mark and Vanessa Loring, in their expensive home and agrees to a closed adoption. After some time, where Juno gets to know the couple better and not long before her baby is due, she is again visiting Mark when their interaction becomes emotional. A few moments later, she watches the Loring marriage fall apart, then drives away and breaks down in tears by the side of the road. Not long after, Juno goes into labor and is rushed to the hospital, where she gives birth to a baby boy. Vanessa comes to the hospital where she joyfully claims the newborn boy as a single adoptive mother.	1. Did the video summary show Juno discover that she is pregnant? 2. Did the video summary show Juno's decision to give the baby up for adoption? 3. Did the video summary show Juno's first visit to Mark and Vanessa's house? 4. Did the video summary show Mark and Vanessa's breakup? 5. Did the video summary show Vanessa adopt Juno's baby as a single mother in the end?

Table 5.8: Examples of movies used for human evaluation. We present a short text summary per movie based on Wikipedia and the questions we asked for evaluating the information contained in each video summary. Different colors in the summary correspond to different TPs (i.e., TP1, TP2, TP3, TP4, TP5).

Chapter 6

Human-assisted Trailer Creation via Task Composition

We now address all limitations discussed in the previous chapter by focusing on a slightly different task: trailer moment identification in movies. 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 chapter, we aim at *automatically* identifying moments in a movie that are suitable for including in a trailer. In contrast to previous chapters, we now consider the *video* as our main information source, which is naturally segmented into *shots* instead of scenes. A shot is a video segment from the moment that the camera starts rolling until the moment it stops and usually last only a few seconds. In accordance with

[1] https://www.studiobinder.com/blog/how-to-make-a-movie-trailer

[2] https://www.derek-lieu.com/blog/2017/9/10/the-matrix-is-a-trailer-editors-dream

prior work (Irie et al., 2010; Smeaton et al., 2006; Wang et al., 2020b), we formulate the task of trailer moment identification as identifying which *shots* should appear in the trailer, assuming an a priori segmentation of the movie into shots. By considering shots as our unit we can create further compressed output videos with an average duration of a couple minutes, which can better serve as trailers. Moreover, we still consider the rich information from screenplays, but only during training. Finally, in contrast to previous chapters, we do not exclusively focus on delivering key information, but consider additional criteria related to the attractiveness and coherence of the output videos, which are important properties when creating trailers.

For automatically identifying trailer-worthy moments, 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 (Irie et al., 2010; Smeaton et al., 2006; Wang et al., 2020b) have solely focused on audiovisual features and depend on ill-defined criteria, such as identifying the "trailerness" of a movie shot.

Inspired by the creative process of human trailer editors, we adopt a bottom-up approach to the task, which we decompose into two orthogonal, simpler and well-defined subtasks. The first is the identification of narrative structure, which we formulate in terms of turning points (TPs) as in previous chapters (Chapters 3, 4, 5; also see example from the movie "The Shining" in Figure 6.1). However, we now consider an additional task, sentiment prediction, which we view as an approximation of flow of intensity between shots and the emotions evoked.

For identifying sequences of shots that are suitable for trailers, we propose an unsupervised graph-based approach. In accordance with Chapter 5, we again model movies as sparse graphs whose nodes are now shots and edges denote important semantic connections between shots (see Figure 6.2). 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 *all modalities* (i.e., video, audio, text) for identifying trailer moments. More importantly, our method uses interpretable criteria (i.e., key events, sentiment scores) and therefore

1. Opportunity	Introductory event that occurs after presentation of setting and background of main characters. *Jack Torrance, his wife Wendy, and their five-year-old son Danny move into the hotel after Jack accepts the position as winter caretaker.*
2. Change of Plans	Main goal of story is defined; action begins to increase. *Danny, unknown to his parents, possesses psychic abilities referred to as "shining", which enable him to read minds and experience premonitions as well as clairvoyance.*
3. Point of No Return	Event that pushes the characters to fully commit to their goal. *Jack starts to develop cabin fever and becomes increasingly unstable, destroying a CB radio and sabotaging a snowcat, the only two links with the outside world the Torrances had.*
4. Major Setback	Event where everything falls apart, temporarily or permanently. *Jack attacks Wendy with one of the hotel's roque mallets, grievously injuring her, but she escapes to the caretaker's suite and locks herself in the bathroom.*
5. Climax	Final event of the main story, moment of resolution. *As Danny, Wendy, and Hallorann flee, the hotel-creature rushes to the basement in an attempt to vent the pressure, but it is too late and the boiler explodes, killing Jack and destroying the Overlook.*

Figure 6.1: Turning points and their definitions. We provide an example for each turning point from the movie "The Shining" in italics.

can be deployed as part of an interactive trailer creation tool with a human in the loop.

Both the tasks of TP identification and sentiment prediction stand to benefit from a lower-level understanding of movie content. Indeed, we could employ off-the-shelf modules for identifying characters and places, recognizing actions, and localizing semantic units. However, such approaches substantially increase pre-processing time and memory requirements during training and inference and suffer from error propagation. Instead, we propose a contrastive learning regime, where we take advantage of screenplays as *privileged information* (Lopez-Paz et al., 2015), i.e., information available at *training time* only. As demonstrated in previous chapters, screenplays reveal how the movie is segmented into scenes, who the characters are, when and who they are speaking to, where they are and what they are doing. Specifically, we build two individ-

ual networks, a *multimodal* network based on movie videos and an *auxiliary, textual* network based on screenplays (similar to the network of Chapter 5), and train them jointly using auxiliary contrastive losses in order to distill information from screenplays to videos. The auxiliary text-based network can additionally be pre-trained on large collections of screenplays via self-supervised learning, without having to collect and process the corresponding movies, overcoming data scarcity issues. Experimental results show that this contrastive training approach 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 interactively with human users for selecting sequences of shots to be included in trailers. We create a demo for interactive trailer creation[3] and test it via human evaluation against fully manual shot selection and fully automatic methods. Based on human judges' preferences we conclude that interactively selecting shots improves the quality of trailers and is comparable with fully manual selection, while reducing the time needed from 2-3 days to under 30 minutes.

The contributions of this chapter can be summarized as follows:

- We propose an *interpretable* unsupervised approach for identifying shot-level trailer moments in movies. For that, we model movies as sparse graphs and decompose the task of trailer moment identification into two simpler ones: narrative structure identification and sentiment prediction.

- We propose a *joint contrastive training regime* for *distilling knowledge* from screenplays to movie videos. Given the difficulty of collecting and processing full-length movie videos, our approach takes advantage of the information contained in screenplays and learns from more text-based samples.

- We extend our method to a *semi-automatic setting* by developing an interactive tool for trailer creation with a human in the loop. Human evaluation shows that trailers generated using our tool are of better quality than automatic shot selection and comparable to fully manual selection.

[3] https://movie-trailers-beta.herokuapp.com

6.1 Related Work

Movie understanding Previous approaches have mainly focused on isolated video clips, and tasks such as the alignment between movie scenes and book chapters (Tapaswi et al., 2015b), question answering (Tapaswi et al., 2016; Lei et al., 2018), video captioning for movie shots (Rohrbach et al., 2015) or clips from TV episodes (Lei et al., 2020b), and text-to-video retrieval (Bain et al., 2020; Lei et al., 2020b; Liu et al., 2020). Although this work utilizes multimodal information (i.e., mainly video and language), it does not use full-length movies. Bain et al. (2020) present a holistic approach to movie understanding by providing movie clips accompanied by textual descriptions and other metadata, such as bounding boxes for the characters and the genre of the movie. This holistic approach is further extended by Huang et al. (2020) who provide full-length movies alongside trailers, posters, textual synopses, scripts, action recognition tags and character bounding boxes in the video frames. Unfortunately, this dataset is not publicly available, limiting further research. On the other hand, recent work (Chen et al., 2022a) attempts to summarize entire TV episodes and movies focusing exclusively on the textual modality (i.e., screenplays, transcripts). We aim at using both the video, audio and textual information while considering full-length movies and alleviating the limitations of the previous chapters.

Trailer moment identification Existing approaches exploit superficial audiovisual features, such as background music or visual changes between sequential shots (Irie et al., 2010; Smeaton et al., 2006). Other work creates "attractive" trailers with a graph-based model for shot selection (Xu et al., 2015) or uses a human in the loop in conjunction with a model trained on horror movies via audiovisual sentiment analysis (Smith et al., 2017). The Trailer Moment Detection Dataset (Wang et al., 2020b) consists of full-length movies paired with official trailers and annotations for key moments, but it is not publicly available and does not include screenplays. Moreover, Wang et al. (2020b) 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 combine all input modalities for identifying trailer moments in full-length movies. As we demonstrate in this thesis, utilizing all modalities is crucial for selecting trailer shots.

Video highlight detection Relevant to our task is also the task of video highlight detection, where the purpose is to identify frames or video segments that can be used

as highlights of videos (Ye et al., 2021; Badamdorj et al., 2021; Chen et al., 2021). Problematically, most prior work on this domain focuses on short videos with simple semantics (e.g., actions in YouTube videos) and does not utilize the textual modality. It is there not straightforward to transfer these methods to the task of trailer moment identification.

Knowledge distillation (Ba and Caruana, 2014; Hinton et al., 2015) was originally proposed for distilling information from a larger teacher model to a smaller student one. Generalized distillation (Lopez-Paz et al., 2015) provides a framework for using privileged information, i.e., information which is available at training time only. Most related to our work is the use of different modalities or views of the same content (Miech et al., 2019, 2020), e.g., transcribed narrations to learn visual representations in instructional videos. We leverage screenplays as a source of privileged information and distill knowledge about events, characters, and scenes in a film, which we subsequently exploit for identifying trailer-worthy shots in video.

6.2 Problem Formulation

Trailer moment identification requires the selection of L shots from a full-length movie of V shots ($L \ll V$). After selecting the desired movie content, human creators can post-process it by trimming the selected shots, changing the order, adding music, voice-over, and other information, such the release date. We are interested in how we can *automatically* or *semi-automatically* identify important movie content that should be included in a trailer[4].

Movies present complex stories that may contain distinct subplots or events that unfold non-linearly, while redundant events, called "fillers" enrich the main story. Hence, we cannot assume that consecutive shots are necessarily semantically related. To better explore relations between events, we represent movies as graphs, similarly to Chapter 5. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph where vertices $\mathcal{V}$ are shots and edges $\mathcal{E}$ represent their semantic similarity. We further consider the original temporal order of shots in $\mathcal{G}$ by only allowing directed edges from previous to future shots. $\mathcal{G}$ is described by an upper-triangular transition matrix $\mathcal{T}$, which records the probability of transitioning from shot i to every future shot j.

[4]We leave further post-processing of trailers for future work.

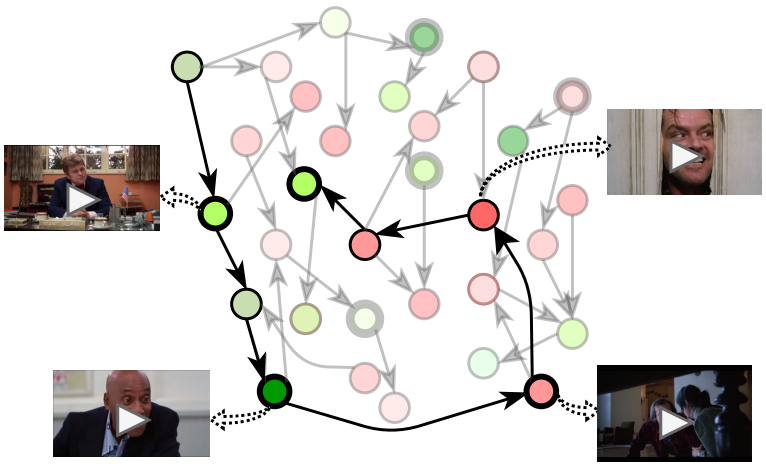


Figure 6.2: GRAPHTRAILER: a movie is a graph whose nodes are shots and edges denote relations between them. Each shot is characterized by a sentiment score (green/red shades for positive/negative values) and labels describing important events (thick circles). Our algorithm performs walks in the graph (bold line) to generate proposal trailer sequences.

First, we assume that the following information is given for a movie: the graph structure $\mathcal{G}$ that represents relations between shots in the movie (graph in Figure 6.2), the shots that describe key events (TPs; thick circles in Figure 6.2), and a sentiment (positive or negative) score for each shot in the graph (different shades of green/red depending on the sentiment intensity in Figure 6.2). Given this information, we propose an algorithm for traversing $\mathcal{G}$ and selecting sequences of shots to be used in a trailer. In the following, we first describe this algorithm (Section 6.2.1) and then discuss how the graph $\mathcal{G}$ is learned and key events are detected via TP identification (Chapters 3 and 5; Section 6.2.2). Finally, we explain how shot-based sentiment scores are predicted (Section 6.2.6).

6.2.1 GRAPHTRAILER: Movie Graph Traversal

Algorithm 1 retrieves trailer sequences by performing random walks in graph $\mathcal{G}$. We start by selecting a node identified as the first TP (i.e., Opportunity, see Figure 6.1), since the first TP is by definition an introductory event to the movie and therefore appropriate for a trailer. 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 $\mathcal{N}_i$. We select node j from $\mathcal{N}_i$ as the k^{th} shot n_k to add to the path based on the fol-

Algorithm 1 Graph traversal: Retrieve trailer path

Input: shot-level graph $\mathcal{G}$, sets of TP shots, sentiment scores for all shots

Output: proposal trailer path (Path)

procedure GRAPHTRAILER
    Path $\leftarrow \emptyset$, budget $\leftarrow L$, TPs_id $\leftarrow 0$, flow $\leftarrow \emptyset$
    events $\leftarrow [\mathcal{TP}_1, \mathcal{TP}_2, \mathcal{TP}_3, \mathcal{TP}_4, \mathcal{TP}_5]$
    i $\leftarrow$ sample($\mathcal{TP}_1$)
    add i to Path
    next_TP $\leftarrow$ events[TPs_id]
    while budget > 0 & TPs_id < 5 **do**
        next_node := $\underset{j \in \mathbb{N}_i}{\text{argmax}}(\text{score}_{ij})$ ▷ Eq. 6.2
        add next_node to Path
        add sentiment(next_node) to flow
        i $\leftarrow$ next_node
        budget -= 1
        if i $\in$ next_TP $\cup$ $\mathcal{N}_{next\_TP}$ **then**
            TPs_id++, next_TP $\leftarrow$ events[TPs_id]
    return Path

lowing criteria: (1) normalized probability of transition e_{ij} from i to j based on matrix $\mathcal{T}$ (i.e., *semantic similarity* between shots), (2) normalized distance $z_{ij} = |j-i|/V$ between shots i and j in the movie (i.e., *temporal proximity*), (3) normalized shortest path in the graph from node j to next major event $d_{j,\mathcal{TP}}$ (i.e., *relevance to the storyline*), and (4) variation between the sentiment difference l_{ij} from i to j and the *desired sentiment flow* f_k at the k^{th} step in the path (see Figure 6.2 and Section 6.3):

$$n_k = \underset{j \in \mathcal{N}_i}{\text{argmax}}(\text{score}_{ij}) \tag{6.1}$$

$$\text{score}_{ij} = \lambda_1 e_{ij} - \lambda_2 z_{ij} - \lambda_3 d_{j,\mathcal{TP}} - \lambda_4 |l_{ij} - f_k| \tag{6.2}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are hyperparameters used to combine the different criteria (tuned on the development set based on ground-truth trailer labels). Note that these criteria are interpretable and can be easily altered by a user (e.g., add/delete a criterion, define a different flow f). Our approach can also be used for interactive trailer creation, where a human user decides on the next shot at each step from a limited set of options (see Section 6.5 for details).

We select L shots in total (depending on a target trailer length) and retrieve a proposal trailer sequence as depicted in Figure 6.2 (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 has been selected for presentation in the trailer if a shot or its immediate neighbors have been added to the path.

6.2.2 Graph Construction and TP Identification

In the previous section, we discussed how we can identify important shot-level trailer moments by assuming a movie graph $\mathcal{G}$ and a set of shots that act as key events (i.e., TPs). We now discuss how we learn $\mathcal{G}$ and identify TPs in movies in tandem. In accordance with Chapter 5, 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. A neural network model, that is similar to GRAPHTP of Chapter 5, first creates $\mathcal{G}$ that represents relations between shots in the latent space and then computes the probability $p(y_{it}|m_i,\mathcal{F},\theta_1)$, where y_{it} is a binary label denoting whether shot m_i represents TP $t \in [1,T]$ and θ_1 are network parameters. However, in contrast to the previous chapter, we further directly use the learned sparse graph for retrieving trailer shots in a later step (Algorithm 1). The network is depicted in the right side of Figure 6.3.

Movie input Let $\mathcal{F}$ denote a full-length movie consisting of V shots $\mathcal{F} = \{m_1, m_2, \ldots, m_V\}$. For each shot i, we consider visual (i.e., sequence of frames), audio (i.e., audio segment), and textual (i.e., subtitles) information and compute a combination vector m_i of all modalities (see step (1), right part of Figure 6.3). First, we compute the textual representation $\text{textual}'_i$ for the i^{th} shot, via the bi-directional attention flow (Seo et al., 2017; Kim et al., 2020) between textual_i, audio_i, and visual_i:

$$S_{\text{textual}_i,\text{audio}_i} = \text{audio}_i^T \text{textual}_i, \quad S_{\text{textual}_i,\text{visual}_i} = \text{visual}_i^T \text{textual}_i \tag{6.3}$$

$$\text{audio}_{i,att} = \text{softmax}(S_{\text{textual}_i,\text{audio}_i})\text{audio}_i, \tag{6.4}$$

$$\text{visual}_{i,att} = \text{softmax}(S_{\text{textual}_i,\text{visual}_i})\text{visual}_i \tag{6.5}$$

$$\text{textual}'_i = \text{audio}_{i,att} + \text{visual}_{i,att} + \text{textual}_i \tag{6.6}$$

The final audio'_i and visual'_i representations are obtained analogously. Next, vectors $\text{textual}'_i$, audio'_i, and visual'_i are projected to a lower dimension via a fully-connected linear layer and L2 normalization. Finally, we compute the multimodal representation m_i for shot i via a non-linear projection: $m_i = f([\text{textual}'_i; \text{visual}'_i; \text{audio}'_i])$, where $f(\cdot)$ is a fully-connected layer followed by the ReLU non-linearity.

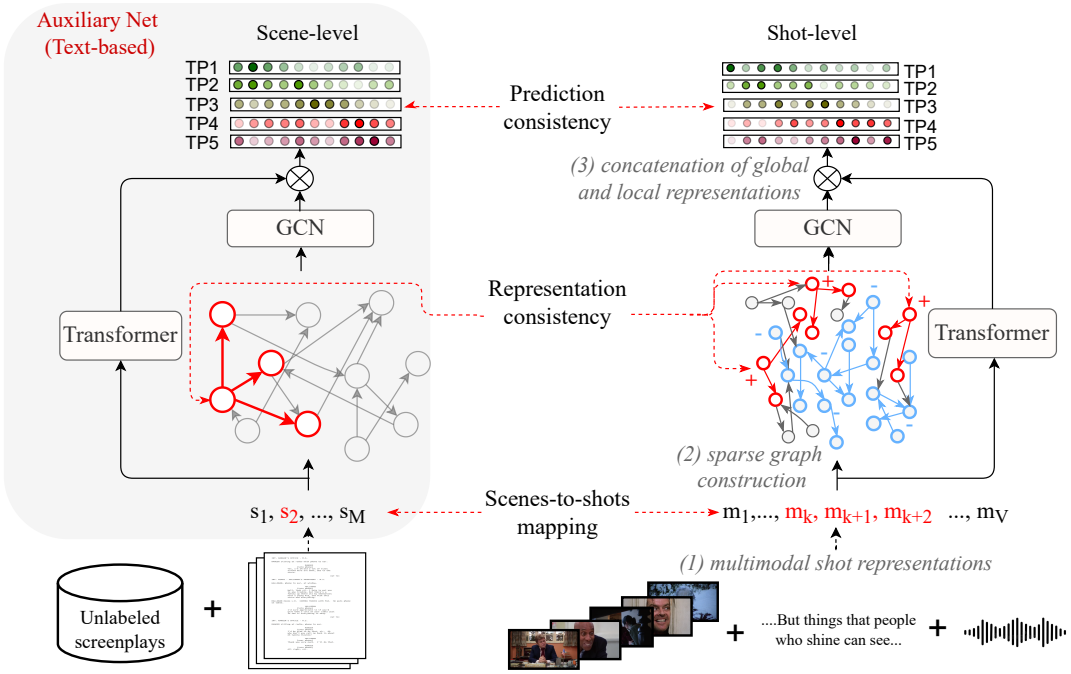


Figure 6.3: Two networks process *different views* of the movie with *different degrees of granularity*. Our main network (right side) takes as input *multimodal* fine-grained *shot* representations based on the movie's video stream. The auxiliary text-based network (left side) processes *textual scene* 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.

Graph structure Given the shot-level multimodal vectors m, we create a sparse graph $\mathcal{G}$ similarly as in Chapter 5. Specifically, we initially construct a fully-connected graph described by the adjacency matrix $\mathcal{A}$ by computing the pairwise similarity e_{ij} between all multimodal shot vectors $m_1, m_2, \ldots, m_V$:

$$e_{ij} = \tanh(W_i m_i + b_i) \tanh(W_j m_j + b_j) + b_{ij} \tag{6.7}$$

Similarities are normalized using the softmax function (row-wise normalization in matrix E). We thus obtain a fully-connected directed graph where edge p_{ij} records the probability that m_i is connected with m_j in the graph. In order to avoid dense connections in the graph, which lead to worse contextualization and computational overhead, we select a small but variable-length neighborhood per shot and we sparsify the graph accordingly. As we already explained in detail and experimentally validated in Chapter 5, modeling movies as *sparse* graphs offers better contextualization and performance improvements on TP identification. For this reason, we again select a small but

variable-length neighborhood per shot and we sparsify the graph accordingly. However, we now further constrain the adjacency matrix of the graph to be upper triangular (i.e., allowing only future connections between shots) during sparsification of the graph. This is important in order to avoid loops and select trailer shots in a sequential order during the graph traversal (Algorithm 1).

For sparsifying the graph, in accordance with Chapter 5 we select a top-k neighborhood $\mathcal{P}_i$ per shot h_i: $\mathcal{P}_i = \text{argmax}_{j\in[1,V],|\mathcal{P}_i|=k}\, p_{ij}$. 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 the appropriate k per shot via a parameterize function: $w_i = \text{softmax}(W_n e_i + b_n)$, where w_i is a probability distribution over the neighborhood size options for shot m_i, $W_n \in \mathbb{R}^{VxO}$, and e_i is a vector of similarities between shot i and all other shots. Hence, the final neighborhood size for shot m_i is: $k_i = \text{argmax}_{t\in O}\, w_{it}$ (see step (2) in the right part of Figure 6.3). We address discontinuities in our model (i.e., top-k sampling, neighborhood size selection) by utilizing the Straight-Through Estimator (Bengio et al., 2013). During the backward pass we compute the gradients with the Gumbel-softmax reparameterization trick (Maddison et al., 2017; Jang et al., 2017).

TP identification For identifying TPs given the multimodal representations and the sparse graph $\mathcal{G}$, we again follow Chapter 5. However, we now replace the BiLSTM for contextualizing events (i.e., scenes or shots) with respect to the *whole movie* with a Transformer encoder (Vaswani et al., 2017). Moreover, we again additionally encode the *graph neighborhood* of each shot via a one-layer Graph Convolution Network (GCN; Duvenaud et al. 2015; Kearnes et al. 2016; Kipf and Welling 2017). Finally, we combine the global (transformer-based) and local (graph-based) representations for each shot (see step (3) in the right part of Figure 6.3) to compute the probability $p(y_{it}|m_i, \mathcal{F}, \theta_1)$. The network is trained with the same objective as in the previous chapter (see Equation 5.5 in Section 5.3.3). After training, we use the graph $\mathcal{G}$ and predicted TP shots as input to Algorithm 1.

6.2.3 Auxiliary Text-based Network

Movie screenplays provide a wealth of additional information in comparison with subtitles, e.g., about characters and their roles in a scene, or their actions and emotions (conveyed by lines describing what the camera sees). This information is difficult to

be accurately inferred from the video, requiring low-level analysis such as person identification, action recognition, and event localization, while it is easily available within the screenplay. Moreover, unlabeled text corpora of screenplays are relatively easy to obtain leading to larger datasets with orders of magnitude more samples than when we collect full-length videos.

In this section, we describe how we can exploit the rich textual information provided by screenplays during training of our main network (described in the previous section). We use an *auxiliary text-based network* that takes as input the screenplay and has a similar architecture to the main network. There are two key differences in the architecture of the auxiliary network: 1. It only considers *textual* information. 2. It processes the movie at *scene-level*, which is the semantic unit of screenplays and is more coarse-grained than shots (one scene may last several minutes). Hence, this network creates a *scene-level graph* and estimates *scene-level probabilities* $q(y_{it}|s_i, \mathcal{D}, \theta_2)$ 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.

When comparing this auxiliary network with GRAPHTP of Chapter 5, there are two main architectural differences: 1. We now allow only *future* connections between scenes in the graph (constraint of corresponding adjacency matrix to be upper triangular). 2. We substitute the BiLSTMs used for contextualizing sentences with respect to a scene, and scenes with respect to the whole screenplay, with transformer encoders. We provide an ablation study of how the network modifications affect model performance on TP identification over screenplays in Appendix D.

6.2.4 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 6.3 for details of how we obtain this mapping). Traditionally, in knowledge distillation (Ba and Caruana, 2014; Hinton et al., 2015) 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 (Ba and Caruana, 2014; Hinton et al., 2015), we utilize the KL divergence loss between the auxiliary-based posterior distributions $q(y_t|\mathcal{D})$ and the distribution $p(y_t|\mathcal{F})$ of our main network (upper part in Figure 6.3).

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 $\overline{p(y_t|\mathcal{F})}$ 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:

$$\mathcal{P} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{D}_{KL}\left(\overline{p(y_t|\mathcal{F})} \,\middle\|\, q(y_t|\mathcal{D})\right) \tag{6.8}$$

Representation Consistency Loss We propose using a second regularization loss between the two networks in order to also 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 (Oord et al., 2018; Sun et al., 2020; Pan et al., 2020), 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 6.2.1). 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 Figure 6.3).

We again first address the problem of varying granularity in the representations of the two networks. We compute an aggregated scene-level representation $\overline{m_j}$ based on shots $m_i, \ldots, m_{i+k}$ via mean pooling and calculate the noise contrastive estimation

(NCE; Gutmann and Hyvärinen 2010; Wu et al. 2018) loss for the j^{th} scene:

$$\mathcal{R} = -\frac{1}{M}\sum_{j=1}^{M}\log\frac{e^{\text{sim}(\overline{m_j},s_j)/\tau}}{e^{\text{sim}(\overline{m_j},s_j)/\tau}+\sum_{\substack{k=1\\k\neq j}}^{M}e^{\text{sim}(\overline{m_j},s_k)/\tau}} \tag{6.9}$$

where M is the number of scenes in the screenplay, s_j is the scene representation calculated by the GCN in the auxiliary network, $\overline{m_j}$ is the (average) scene representation calculated by the GCN in the main network, $\text{sim}(\cdot)$ is a similarity function (we use the scaled dot product between two vectors), and τ is a temperature hyperparameter.

Joint Training Our final joint training objective takes into account the individual losses $\mathcal{S}$ and $\mathcal{M}$ of the auxiliary and main networks, respectively (see Chapter 5 for details), and the two consistency losses $\mathcal{P}$ (for prediction) and $\mathcal{R}$ (for representation):

$$\mathcal{L}_{TP} = \mathcal{S} + \mathcal{M} + a\mathcal{P} + b\mathcal{R} \tag{6.10}$$

where a, b are hyperparameters modulating the importance of prediction vs. representation consistency. Figure 6.3 provides a high-level illustration of our training regime.

6.2.5 Self-supervised Pre-training

We further pre-train 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 (Equations 6.8 and 6.9).

Pre-training takes place on Scriptbase (Gorinski and Lapata, 2015), a dataset which consists of ∼1,100 full-length screenplays (approximately 140k scenes). We adapt a self-supervised task, namely Contrastive Predictive Coding (CPC; Oord et al. 2018), 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 $\{c_{i1}^-, \ldots, c_{i(M-1)}^-\}$, we compute the InfoNCE (Oord et al., 2018) loss:

$$\mathcal{L}_{self} = -\frac{1}{M}\sum_{i=1}^{M}\log\frac{e^{\text{sim}(g_i,c_i^+)/\tau}}{e^{\text{sim}(g_i,c_i^+)/\tau}+\sum_{k=1}^{M-1}e^{\text{sim}(g_i,c_{ik}^-)/\tau}} \tag{6.11}$$

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.

Notice that we compute structure-aware scene representations g_i via random walks rather than stacking multiple GCN layers. We could in theory stack two or three GCN layers and thus consider many more representations (e.g., 100 or 1,000) for contextualizing g_i. However, this would result to over-smoothed representations, that converge to the same vector (Oono and Suzuki, 2019). Moreover, when trying to contextualize such large neighborhoods in a graph, the bottleneck phenomenon prevents the effective propagation of long-range information (Alon and Yahav, 2020) which is our main goal. Finally, in "small world" networks with a few hops, neighborhoods could end up containing the majority of the nodes in the graph (Barceló et al., 2019), which again hinders meaningful exploration of long-range dependencies. Based on these limitations of GCNs, we perform random walks, which allow us to consider a small number of representations when contextualizing a scene, while also exploring long-range dependencies.

6.2.6 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 (Sections 6.2.2, 6.2.3, 6.2.4). The only difference for sentiment prediction is the different downstream objective (losses $\mathcal{S}$ and $\mathcal{M}$ in Equation 6.10). 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 6.3 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.

6.3 Experimental Setup

Datasets Our model was trained on TRIPOD$\bigoplus$, an expanded version of the TRIPOD dataset (Chapter 3) which contains 122 screenplays with silver-standard TP annotations (scene-level)[5] and the corresponding videos[6]. For each movie, we further

[5]https://github.com/ppapalampidi/TRIPOD
[6]https://datashare.ed.ac.uk/handle/10283/3819

TRIPOD⊕	Train	Dev+Test	Held-out
No. movies	84	38	41
No. scenes	11,320	5,830	—
No. video shots	81,400	34,100	48,600
No. trailers	277	155	41
Avg. movie duration	6.9k (0.6)	6.9k (1.3)	7.8k (2.3)
Avg. scenes per movie	133 (61)	153 (54)	—
Avg. shots per movie	968 (441)	898 (339)	1,186 (509)
duration	6.9 (15.1)	7.1 (13.6)	6.6 (16.6)
Avg. valid shots per movie	400 (131)	375 (109)	447 (111)
duration	13.5 (19.0)	13.9 (18.1)	13.3 (19.6)
Avg. trailers per movie	3.3 (1.0)	4.1 (1.0)	1.0 (0.0)
duration	137 (42)	168 (462)	148 (20)
Avg. shots per trailer	44 (27)	43 (27)	57 (28)
duration	3.1 (5.8)	3.9 (14.0)	2.6 (4.0)

Table 6.1: Statistics of TRIPOD⊕ dataset (for movies, screenplays, trailers); standard deviation within parentheses, duration in seconds.

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 Moviescope dataset[7] (Cascante-Bonilla et al., 2019), which contains official movie trailers. The held-out set does not contain any additional information, such as screenplays or TP annotations. The statistics of TRIPOD⊕ are presented in Table 6.1.

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 the dialogue parts in screenplays with the subtitles that contain timestamps using Dynamic Time Warping (DTW; Myers and Rabiner 1981; Chapter 5). We first segment the video into scenes based on this mapping, and then segment each scene into shots using PySceneDetect[8]. Shots with less than 100 frames

[7]http://www.cs.virginia.edu/ pc9za/research/moviescope.html
[8]https://github.com/Breakthrough/PySceneDetect

in total are too short for both 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 (Xie et al., 2017) pre-trained for object recognition on ImageNet (Deng et al., 2009). (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 (Carreira and Zisserman, 2017). (3) We use Faster-RCNN (Girshick, 2015) implemented in Detectron2 (Wu et al., 2019) 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 first project all individual representations to the same lower dimension and perform L2-normalization. Next, we consider the visual shot representation as the sum of the individual vectors. For the audio modality, we use YAMNet pre-trained on the AudioSet-YouTube corpus (Gemmeke et al., 2017) for classifying audio segments into 521 audio classes (e.g., tools, music, explosion); for each audio segment contained in the scene, we extract features from the penultimate layer. Finally, we extract textual features from subtitles and screenplay scenes using the Universal Sentence Encoder (USE; Cer et al. 2018; Chapters 3 and 5).

Trailer Labels For evaluation purposes, we need to know which shots in the movie are trailer-worthy or not. For this, we segment the corresponding trailer into shots and compute for each shot its visual similarity with *all shots in the movie* (Wang et al., 2020b). As visual similarity we consider the cosine similarity between the concatenated visual shot-level representations (i.e., frame- and motion-level). 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 simply material that did not make it in the final movie), we set a threshold of 0.85 in cosine similarity, below which we do not map trailer shots to movie shots. In this way, we create binary shot-level trailer labels.

Sentiment Labels Since TRIPOD does not contain sentiment annotations, we instead obtain silver-standard labels via COSMIC (Ghosal et al., 2020), a commonsense-guided framework with state-of-the-art performance for sentiment and emotion classi-

fication in natural language conversations. Specifically, we train COSMIC on MELD (Poria et al., 2019), which contains dialogues from episodes of the TV series *Friends* and is more suited to our domain than other sentiment classification datasets (e.g., Busso et al. 2008; Li et al. 2017). 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.

Sentiment Flow in GRAPHTRAILER One of the criteria for selecting the next shot in our graph traversal algorithm (Section 6.2.1) is the sentiment flow of the trailer generated so far. Specifically, we adopt the hypothesis[9] that trailers are segmented into three sections based on sentiment intensity. The first section has medium intensity for attracting viewers, the second section has low intensity for delivering key information about the movie and finally the third section displays progressively higher intensity for creating cliffhangers and excitement for the movie (see also Chapter 2).

Accordingly, given a budget of L trailer shots, we expect the first $L/3$ ones to have medium intensity without large variations within the section (e.g., we want shots with average absolute intensity close to 0.7, where all scores are normalized to a range from -1 to 1). In the second part of the trailer (i.e., the next $L/3$ shots) we expect a sharp drop in intensity and shots within this section to maintain more or less neutral sentiment (i.e., 0 intensity). Finally, for the third section (i.e., the final $L/3$ shots) we expect intensity to steadily increase. In practice, we expect the intensity of the first shot to be 0.7 (i.e., medium intensity), increasing by 0.1 with each subsequent shot until we reach a peak at the final shot.

Hyperparameters Following Chapter 5, we project all types of features (i.e., textual, visual, and audio) to the same lower dimension of 128. We find that larger dimensions increase the number of parameters considerably and yield inferior results possibly due to small dataset size.

We contextualize scenes (with respect to the screenplay) and shots (with respect to the video) using transformer encoders. We experimented with 2, 3, 4, 5, and 6 layers in the encoder and obtained best results with 3 layers. For the feed forward (FF) dimension, we experimented with both a standard size of 2,048 and a smaller size of 1,024 and found the former works better. We use another transformer encoder to compute

[9]https://www.derek-lieu.com/blog/2017/9/10/the-matrix-is-a-trailer-editors-dream

the representation of a scene from a sequence of input *sentence* representations. This encoder has 4 layers and 1,024 FF dimension. Both encoders, employ 8 attention heads and 0.3 dropout.

During graph sparsification (i.e., selection of top-k neighbors), we consider different neighborhood options for the scene- and shot-based networks due to their different granularity and size. Following Chapter 5, we consider [1–6] neighbors for the scene network and we increase the neighborhood size to [6–12] for the shot network.

For training our dual network on TP identification, we use the $\mathcal{L}_{TP}$ objective described in Equation 6.10. We set hyperparameters a and b that determine the importance of the prediction and representation consistency losses in $\mathcal{L}_{TP}$ to 10 and 0.03, respectively. Moreover, while pre-training the auxiliary network on the Scriptbase corpus (Gorinski and Lapata, 2015) we select as future context window 10% of the screenplay. As explained in Section 6.2.5, we compute structure-aware scene representations by performing random walks of k steps in the graph starting from an anchor scene. We empirically choose 3 steps.

Finally, as described in Section 6.2.1, GRAPHTRAILER uses a combination of multiple criteria for selecting the next node to be included in the trailer path. The criteria are combined using hyperparameters which are tuned on the development set. The search space for each hyperparameter λ (Equation 6.2) is [0, 1, 5, 10, 15, 20, 25, 30] and we find that the best combination for λ_1 (semantic similarity), λ_2 (time proximity), λ_3 (narrative structure), and λ_4 (sentiment intensity) is [1, 5, 10, 10], respectively.

6.4 Results and Analysis

6.4.1 Knowledge Distillation for TP Identification

Before evaluating the performance of our approach on trailer moment identification, we first investigate whether our proposed joint training approach, which distills information from screenplays to movie videos, improves *TP identification*, which is the task that our main network is directly trained on. For that, we split the set of movies with gold standard scene-level TP labels into development and test set with the same distribution over different movie genres per set.

Next, we select the top 5 (@5) and top 10 (@10) shots per TP in a movie. As evaluation metric, we consider Partial Agreement (Part Agr; Equation 3.3 in Section 3.2), which measures the percentage of TPs for which a model correctly identifies at

	Part Agr@5↑	Part Agr@10↑
Random (evenly distributed)	21.67	33.44
Theory position	10.00	12.22
Distribution position	12.22	15.56
GRAPHTP (Chapter 5)	10.00	12.22
Ours w/o graph structure	22.22	33.33
Ours with graph structure	27.78	35.56
+ Auxiliary net, Asynchronous ($\mathcal{P}$)	28.89	41.11
+ Auxiliary net, Asynchronous ($\mathcal{P}+\mathcal{R}$)	21.11	35.56
+ Auxiliary net, Contrastive Joint ($\mathcal{P}+\mathcal{R}$)	33.33	47.78
+ pre-training	**34.44**	**50.00**

Table 6.2: Model performance on *TP identification* (test set). Our method shown with different training regimes. Evaluation metric: Partial Agreement (Part Agr) against top 5 (@5) and top 10 (@10) selected shots per TP and movie.

least one gold standard shot from the 5 or 10 shots selected from the movie. In comparison with the previous chapters, we can no longer use total agreement, since we evaluate against silver standard (rather than gold) labels for shots (rather than scenes) and as a result consider all shots within a scene equally important. We do not use the distance metric either since it yields very similar outcomes and does not help discriminate among model variants.

Table 6.2 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 positions in the training data; **GRAPHTP** is the original model of Chapter 5 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 use our own model (**Ours with graph structure**) in several variants for TP identification: 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 ($\mathcal{P}$), both prediction and representation losses ($\mathcal{P}+\mathcal{R}$), and our contrastive joint training regime.

	Part Agr@5↑	Part Agr@10↑
Ours (Contrastive Joint w/ pre-training)	**34.44**	**50.00**
Subtitles only	21.11	36.67
Video only	<u>33.33</u>	<u>46.67</u>
Audio only	27.78	44.44

Table 6.3: Ablation study on the contribution of different modalities (i.e., text, video, audio) on *TP identification*. Evaluation metric: Partial Agreement (Part Agr) against top 5 (@5) and top 10 (@10) selected shots per TP and movie. The best performance is marked with bold and the second best is underlined.

We observe that our approach outperforms all baselines, as well the equivalent transformer-based model without the graph information. Although transformers can encode long-range dependencies between shots, our approach additionally benefits from directly encoding sparse connections learned in the graph. Moreover, asynchronous knowledge distillation via the prediction consistency loss ($\mathcal{P}$) further improves performance, suggesting that knowledge contained in screenplays is complementary to what can be extracted from video. Notice that when we add the representation consistency loss ($\mathcal{P}+\mathcal{R}$), performance deteriorates by a large margin, whereas the proposed training approach (contrastive joint) performs best. Finally, pre-training offers further gains, albeit small, which underlines the benefits of the auxiliary text-based network.

We also perform an ablation study on the role of different modalities (i.e., text, video, audio) for identifying TPs. In the previous chapter, we demonstrated that although textual information from screenplays is very rich and leads to good scene contextualization, audiovisual cues are helpful in creating meaningful graphs in the latent space, which can lead to improved performance on TP identification. However, we now only consider information from the movie video and hence, the main network has only access to the textual information provided by the subtitles (i.e., dialogue parts) without the additional information related to the characters, their actions and expressions. We hypothesize that the contribution of visual and audio information to contextualization in this case will be larger for identifying important shots in the movie video. We perform an ablation study, where we use our full approach, where the main network is trained jointly with the pre-trained auxiliary one, while only considering unimodal information from the video (i.e., subtitles, video, or audio). We present the experimental

	Dev↑	Test↑
Random selection w/o TPs	14.47	5.61
Random selection with TPs	20.00	9.27
TEXTRANK (Mihalcea and Tarau, 2004)	10.26	3.66
GRAPHTRAILER w/o TPs	23.58	11.53
GRAPHTRAILER with TPs	**26.95**	**16.44**
CCANet (Wang et al., 2020b)	31.05	15.12
Supervised GRAPHTRAILER w/o graph	32.63	17.32
Supervised GRAPHTRAILER	**33.42**	**17.80**
Upper bound	86.41	—

Table 6.4: Performance of unsupervised (upper part) and weakly supervised (lower part) models on trailer moment identification: accuracy of correctly identified trailer shots. All systems have the same shot budget for trailer creation.

results in Table 6.3. We observe that the textual modality does not offer the strongest performance amongst unimodal variants. In contrast, the visual modality is the most informative for identifying key shots in the movie. However, combining all modalities provides the highest performance on TP identification, as expected.

6.4.2 Automatic Results on Trailer Moment Identification

We now evaluate our target task of trailer moment identification on the held-out set of 41 movies (see Table 6.1). As evaluation metric and similarly to prior work (Wang et al., 2020b), we use accuracy, i.e., the percentage of correctly identified trailer shots and we consider a total budget of 10 shots for the trailers in order to achieve the desired length ($\sim$2 minutes).

We compare GRAPHTRAILER against several unsupervised approaches (first block in Table 6.4) 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. **TEXTRANK** (Mihalcea and Tarau, 2004) operates over this graph to select shots based on their centrality (see also Chapters 4 and 5), while **GRAPHTRAILER without TPs** traverses the graph with TP and sentiment criteria removed (Equation 6.2). For the unsupervised systems which include stochas-

	Accuracy ↑
GRAPHTRAILER	22.63
+ Auxiliary net, Asynchronous ($\mathcal{P}$)	21.87
+ Auxiliary net, Asynchronous ($\mathcal{P}+\mathcal{R}$)	22.11
+ Auxiliary net, Contrast Joint ($\mathcal{P}+\mathcal{R}$)	25.44
+ pre-training	**25.79**

Table 6.5: Different training regimes for GRAPHTRAILER: accuracy (%) of correctly identified trailer shots. For direct comparison GRAPHTRAILER here only uses the narrative structure criterion, no information about sentiment intensity is considered.

ticity and produce proposals (i.e., Random, GRAPHTRAILER), we consider the best trailer out of 10 proposals. The second block of Table 6.4 presents supervised approaches which use trailer labels for training. These include **CCANet** (Wang et al., 2020b), which is the state-of-the-art in trailer moment identification, only considers visual information and computes the cross-attention between movie and trailer shots, **Supervised GRAPHTRAILER without graph** trained for the binary task of identifying whether a shot should be in the trailer without considering graph information, screenplays, sentiment or TPs and, **Supervised GRAPHTRAILER** which is our main network (Section 6.2.2) trained on trailer moment identification with trailer-specific labels.

GRAPHTRAILER performs best among unsupervised methods. Interestingly, TEXTRANK is worse than random, illustrating this task cannot be viewed as a standard summarization problem. GRAPHTRAILER without TPs still performs better than TEXTRANK and random TP selection.[10] With regard to supervised approaches, we find that using all modalities with a standard architecture (Supervised GRAPHTRAILER w/o graph) leads to better performance than sophisticated models using visual similarity (CCANet). By adding graph-related information (Supervised GRAPHTRAILER), we obtain further improvements.

We perform two ablation studies on the development set for GRAPHTRAILER. The first study aims to assess how the different training regimes of the dual network influence downstream performance on trailer moment identification. We observe in Table 6.5 that asynchronous training does not offer any discernible improvement over the base model. However, when we jointly train the two networks (main and auxiliary

[10]Performance on the test set is lower because we only consider trailer labels from the official trailer, while the dev set contains multiple trailers.

	Accuracy ↑
Similarity	24.28
Similarity + TPs	25.79
Similarity + sentiment	23.97
Similarity + TPs + sentiment	**26.95**

Table 6.6: GRAPHTRAILER with different criteria for performing random walks in the movie graph (Algorithm 1, Equation (6.2)).

net) 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.

The second ablation study concerns the criteria used for performing random walks on the graph $\mathcal{G}$ (Equation 6.2). As shown in Table 6.6, 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 (Xu et al., 2015; Wang et al., 2020b) or audiovisual sentiment analysis (Smith et al., 2017), sentiment information on its own is not sufficient and may promote outliers that do not fit well in a trailer. On the other hand, when sentiment information is combined with knowledge about narrative structure (similarity + TPs + sentiment), we observe the highest accuracy. This further validates our hypothesis that the two theories about creating trailers (i.e., based on narrative structure and emotions) 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.

6.4.3 Human Evaluation on 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

	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 (Wang et al., 2020b)	37.7	46.6	14.3	15.2	-0.14
Superv. GRAPHTRAILER	37.7	47.1	21.4	21.4	0.84

Table 6.7: Human evaluation on held-out set. 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.

is the best performing model according to automatic metrics.[11] We generated trailers for all movies in the held-out set by concatenating the identified trailer shots from the movies. 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. We collected assessments from five different judges per movie. We provide more details of the human evaluation setup and the questions in Appendix C.4.

Table 6.7 shows that GRAPHTRAILER with TPs provides on average more informative (Q1) and attractive (Q2) trailers than all other systems (differences are significant). 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 (Louviere et al., 2015), 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.

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

[11] 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.

"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 Section 6.6.

6.5 GRAPHTRAILER as an Interactive Tool

One of the main advantages of our algorithm is that it uses interpretable criteria and can be easily modified to be used interactively with a human in the loop. In the following sections, we describe how our algorithm works via an example, and how it can be used with a human in the loop (Sections 6.5.1 and 6.5.2). We also provide an analysis of the advantages of using a semi-automatic method for trailer creation (Section 6.5.3).

6.5.1 Method

We present in Figures 6.4 and 6.5 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 TP1 (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 excluded a few for 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 6.2.1). 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 that are irrelevant to the main story. Finally, in "Final Step" (Figure 6.5), we assemble the proposal trailer sequence by concatenating all shots in the retrieved path. We also illustrate the path in the graph (i.e., red line; "Final Step" of Figure 6.5).

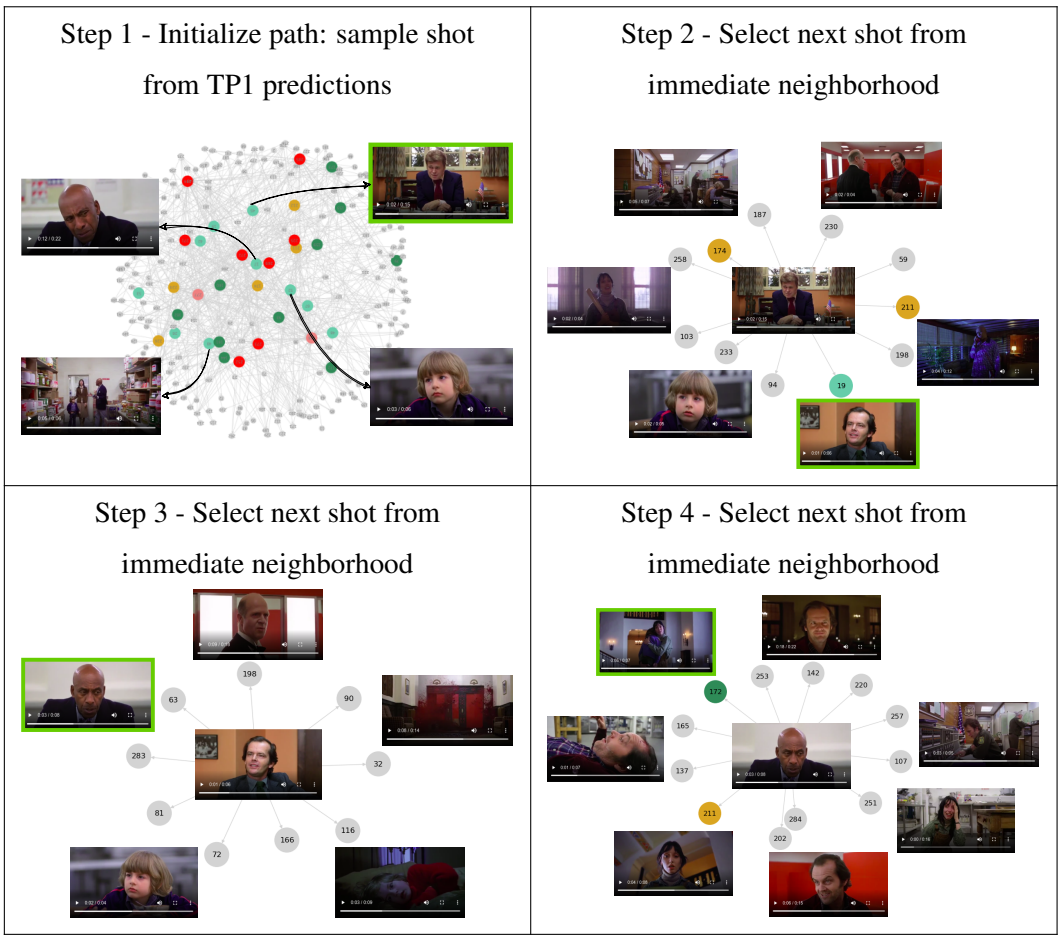


Figure 6.4: Run of GRAPHTRAILER algorithm for the movie "The Shining". Step 1 illustrates the shot-level graph with colored nodes representing the different types of TPs predicted in the movie (i.e., TP1, TP2, TP3, TP4, TP5). Our algorithm starts by sampling a shot identified as TP1 (Step 1). For each 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). Example continues in Figure 6.5.

A similar procedure is followed when GRAPHTRAILER is used as an interactive tool with a human in the loop (see our demo[12] and the example in Figure 6.6). Specifically, given the immediate neighborhood at each step (e.g., Step 2 in Figure 6.4), a human user selects which shot is most appropriate 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

[12]https://movie-trailers-beta.herokuapp.com

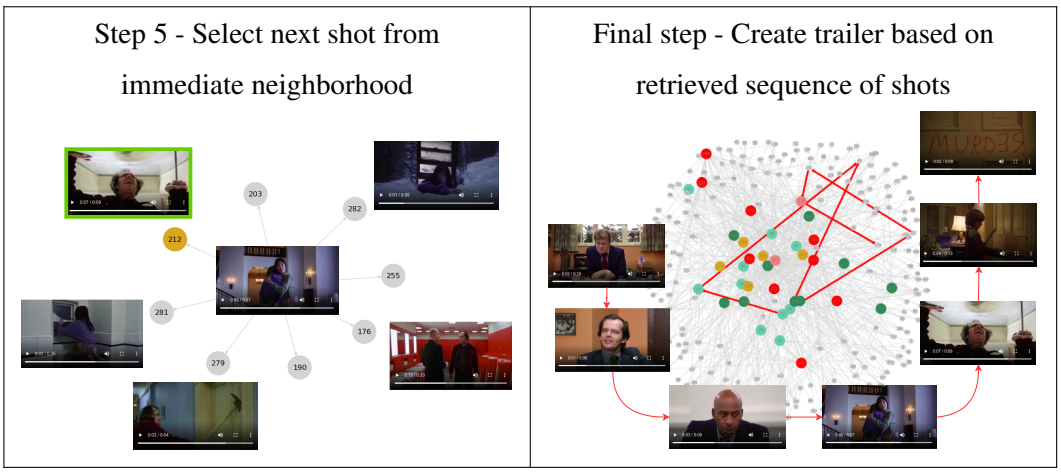


Figure 6.5: Finally, we assemble the proposal trailer (Final step) by concatenating the shots in the path. When our algorithm is used as an interactive tool, it allows users to review candidate shots at each step and manually select the best one while taking into account our criteria. Users create trailers by only reviewing around 10% of the movie.

the trailer and move back and forth depending on the set of options that are presented later in the creation process. Finally, when selecting a shot, the user can trim it to fix possible imperfections that might appear due to the automatic shot segmentation of the movie. We provide more details and examples of the interactive tool in the following section.

Overall, our approach drastically reduces the amount of shots that need to be reviewed to 10% of the movie during trailer creation. Moreover, our criteria allow users to explore different sections of the movie, and create diverse trailers.

6.5.2 Interactive Tool Example

In this section we present in detail how the interactive tool for trailer creation works via an example. We consider the movie “Contagion” and present how a human user can create a trailer in synergy with our algorithm in Figures 6.7 and 6.8.

First, the user is presented with an initial set of options for the first shot of the trailer (Step 1; Figure 6.7). The initial options are the shots that have been identified as TP1, so by definition all shots are of high importance (i.e., high relevance to the storyline). The user can review the shots presented alongside their metadata (i.e., relevance to storyline and sentiment intensity) and decide which the best shot is for inclusion in the trailer. After selecting a shot, they can optionally trim it for correcting imperfections caused by the automatic segmentation of the movie (top right part of Figure 6.7). Next,

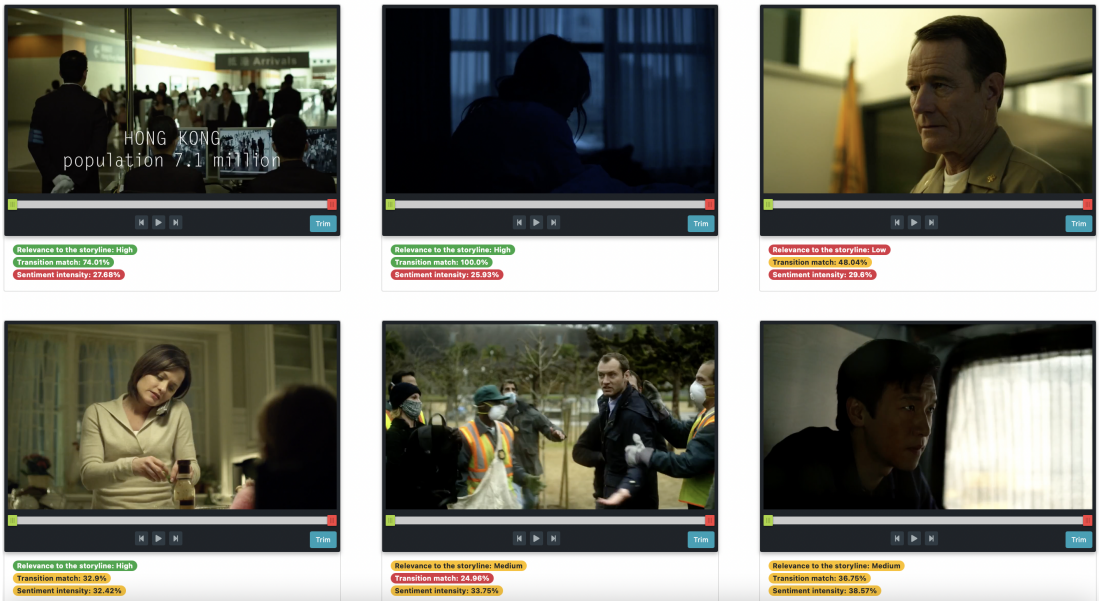


Figure 6.6: A human user can manually review a limited set of shots to be included in the trailer given metadata, such as importance, sentiment intensity, and transition match, select and trim the most appropriate shot and create a movie trailer interactively step by step.

based on the user's selection, a new set of options is displayed. The new set is the immediate neighborhood of the last selected shot in the movie graph (see Figure 6.4 that displays the immediate neighborhood of a shot). The user can again review all new options alongside their metadata (i.e., relevance to storyline, sentiment intensity, and transition match) and decide on the most appropriate trailer shot (bottom left part of Figure 6.7).

The user selects trailer shots iteratively, while monitoring their selection up to the current point (bottom right part of Figure 6.7). Finally, the user can decide to finish the trailer when they are satisfied with the result (left part of Figure 6.8). We should note here that during the creation process, the user can also go back and forth (see button "Go back" in the bottom right part of Figure 6.7 and left part of Figure 6.8) for selecting the best sequence of trailer shots. After finishing the selection process, the user can view the final trailer assembled by concatenating the selected shots and optionally download the video (right part of Figure 6.8).

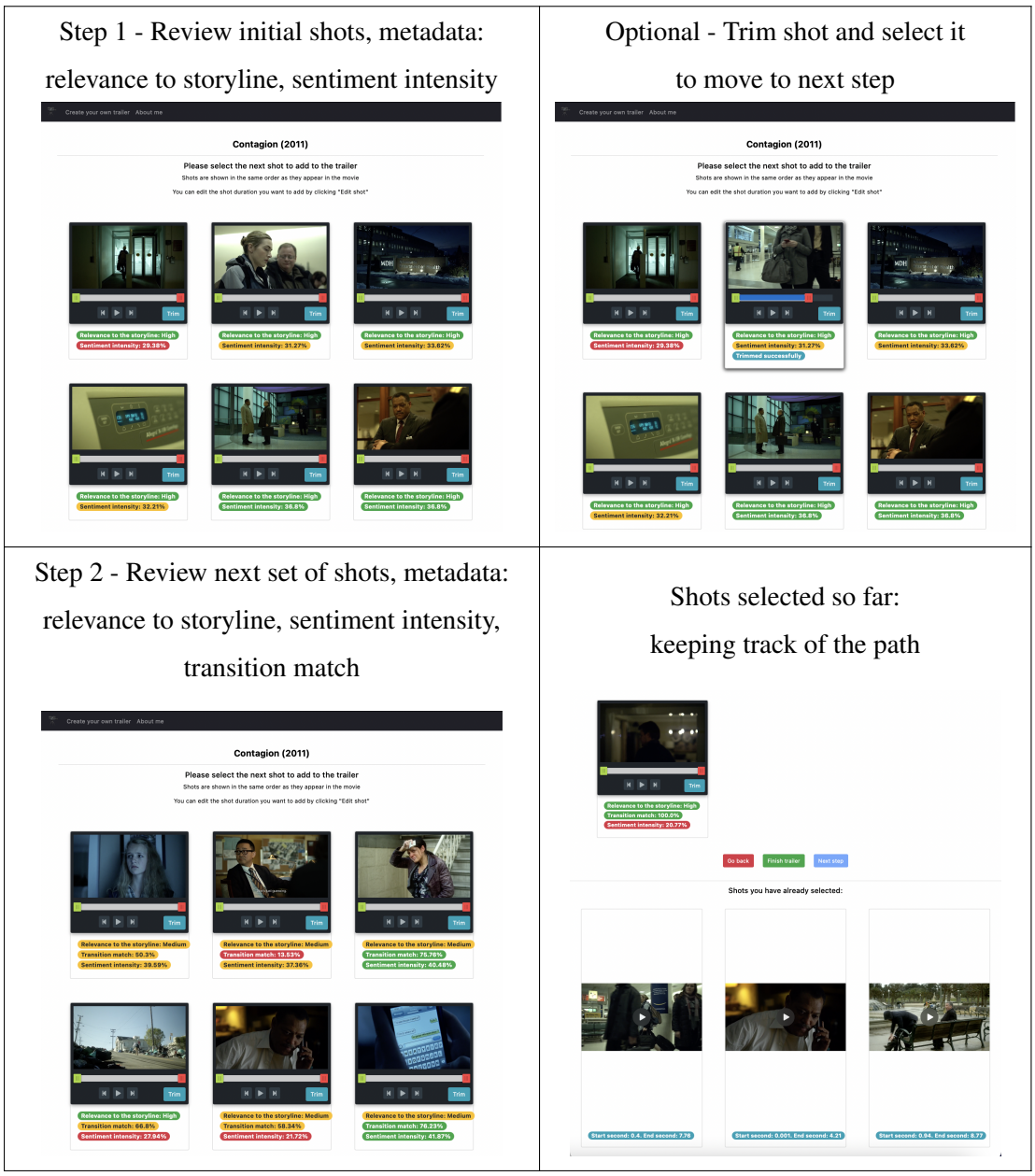


Figure 6.7: Example of use of the interactive tool for trailer creation for the movie “Contagion”. First a set of initial options of shots are presented to the human user. The user can select and trim a shot to correct imperfections caused by the automatic movie segmentation. After selecting a shot, a new set of options is presented in step 2. The user keeps selecting shots to be included in the trailer interactively, while reviewing their selection so far. They can also go back and forth if they are not satisfied with the result so far. The example continues in Figure 6.8.

6.5.3 Semi-automatic Trailer Creation

We evaluate whether semi-automatically selecting trailer shots via our interactive tool can provide good quality outputs while minimizing the time a human creator spends

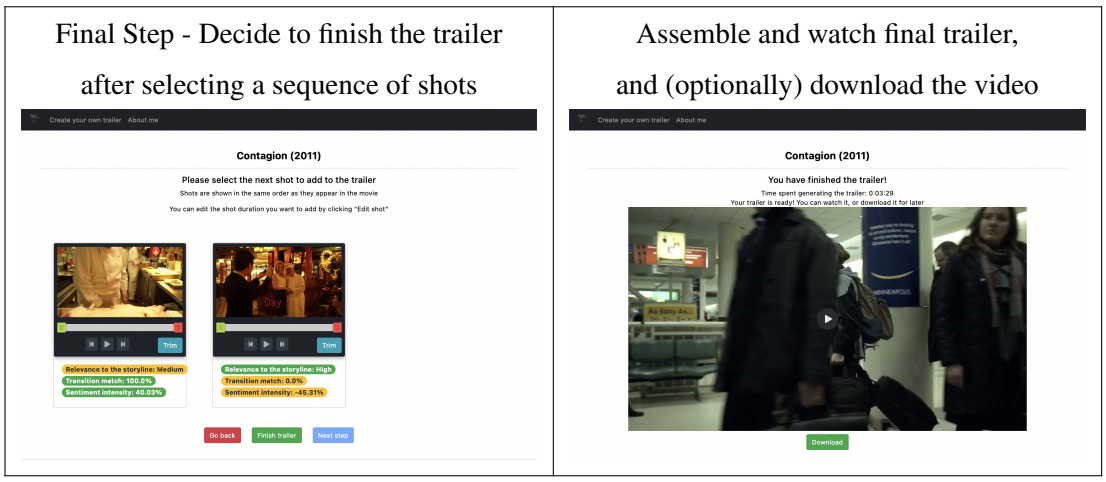


Figure 6.8: Finally, the user can decide when to finish the trailer if they satisfied with their shot selection. Once they finish the trailer, they can review the final result which is assembled by concatenating the selected shots and optionally download the video.

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 trailer shots selected via the interactive tool against ones selected by either a human expert or an automatic system.

For creating semi-automatic trailers, we recruit two non-expert users and ask them to first familiarize themselves with the tool and then select sequences of trailer shots for all movies included in the held-out set of 41 movies. We ask users to *not spend more than 30 minutes per movie* and they can freely go back and forth for selecting trailer shots until they are satisfied with the result. The non-expert users on average spend 22 minutes per movie and perform three back and forths until they select a sequence of trailer shots. We provide examples of the trailers created interactively by the non-expert users[13][14][15][16][17].

After the semi-automatic selection process using the interactive tool, we perform

[13] https://s3.eu-west-2.amazonaws.com/tripodtrailers/demo_generated_trailers/Contagion_2011_26_3_21:44:30:633.mp4

[14] https://s3.eu-west-2.amazonaws.com/tripodtrailers/demo_generated_trailers/Inception_2010_25_4_19:25:29:622.mp4

[15] https://s3.eu-west-2.amazonaws.com/tripodtrailers/demo_generated_trailers/American_Sniper_2014_25_3_12:24:43:110.mp4

[16] https://s3.eu-west-2.amazonaws.com/tripodtrailers/demo_generated_trailers/Anger_Managment_2003_28_4_14:43:36:527.mp4

[17] https://s3.eu-west-2.amazonaws.com/tripodtrailers/demo_generated_trailers/Forrest_Gump_1994_30_4_19:24:34:880.mp4

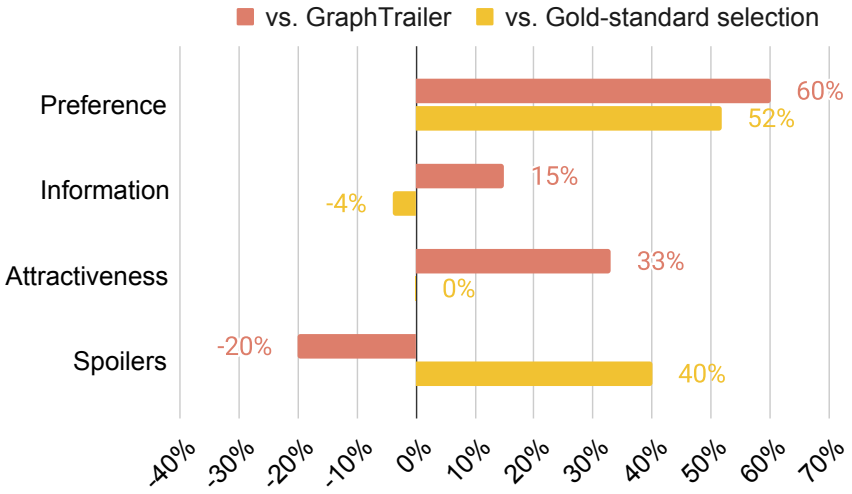


Figure 6.9: Human evaluation on semi-automatic trailers created based on our demo tool with a human in the loop. We compare the semi-automatic trailers against fully-automatic ones by GRAPHTRAILER and a gold standard selection derived from aligning real trailer shots with movie shots.

pairwise comparisons via human evaluation, similarly to Section 6.4.3, between the new trailers and trailers created by GRAPHTRAILER (fully automatic) or assembled given the gold standard trailer labels (gold standard selection; see Section 6.3)[18]. Specifically, we ask AMT crowd workers to watch a pair of trailers for a movie and first answer the same questions as before (Q1: information, Q2: attractiveness) as well as a new question regarding spoilers present in the trailer (Q3: spoilers). After answering all individual questions, the AMT workers also indicate which trailer they prefer. We again collect assessments from five different judges per movie. As pairs of trailers, we consider: (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 Figure 6.9. We present the relative difference between the semi-automatic trailers and either of GRAPHTRAILER or gold standard selection. First, we present the overall preference. We observe that the semi-automatic trailers are preferred by human judges against GRAPHTRAILER 60% of times, indicating that the human in the loop increases the quality of trailer shot selection. Moreover, the semi-automatic trailers are also preferred 52% of times against the gold standard selection, indicating that the quality of outputs for the two

[18]In all cases we first trim the imperfections in the selected shots in order to allow for a fair comparison between different systems.

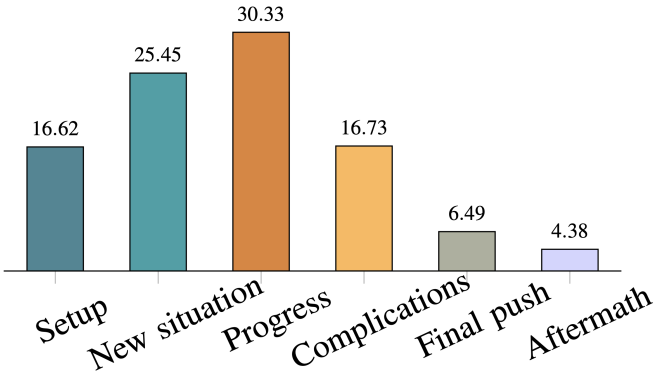


Figure 6.10: Distribution of trailer shots corresponding to different sections of a movie (development set) as determined by TPs. Trailer shots come from all parts of the movie, even from the end, although the majority are from the beginning and middle.

methods is comparable. This suggests that while we minimize the time spent by a human user for selecting trailer shots to under 30 minutes, the quality of the selection is not degraded.

Finally we also present the relative difference between methods for the percentage of Yes-answers to Q1 (Information), Q2 (Attractiveness), and Q3 (Spoilers). We observe that the semi-automatic approach increases both the information and attractiveness of the trailers, while reducing spoilers by 20%. Semi-automatic and gold standard selection are comparable in terms of both information and attractiveness. The semi-automatic trailers however contain 40% more spoilers than the gold standard selection. Although this could be improved in future work, we should note that it does not significantly affect the overall preference of human judges and even with gold standard selection, we still encounter spoilers in 5.85% of the trailers presented.

6.6 Task Decomposition Analysis

How Narrative Structure Connects with Trailers According to screenwriting theory (Hague, 2017), the five TPs segment movies into six thematic units, namely, “Setup”, “New Situation”, “Progress”, “Complications and Higher Stakes”, “Final Push”, and “Aftermath” (see Chapter 2). To examine which parts of the movie are most prevalent in a trailer, we compute the distribution of shots per thematic unit in *gold* trailers (using the extended development set of TRIPOD). As shown in Figure 6.10, trailers on average contain shots from all sections of a movie, even from the last two, which might reveal the ending. Moreover, most trailer shots (30.33%) are selected from the

Opportunity	52.63
Change of plans	55.26
Point of no return	47.37
Major setback	34.21
Climax	34.21

Table 6.8: Percentage (%) of trailers that include at least one shot labeled as a specific type of TP on the development set. The first two TPs (that present an introduction to the story) appear more frequently in trailers, especially in comparison with the last two, which often contain major spoilers.

	Sentiment intensity
First part	11.50
Second part	9.35
Third part	14.75

Table 6.9: Average absolute sentiment intensity per trailer section, when we divide the trailers into three even parts (development set).

middle of the movie (i.e., Progress) as well as from the beginning (i.e., 16.62% and 25.45% for "Setup" and "New Situation", respectively). These empirical observations corroborate industry principles for trailer creation[19].

Next, we find how often the trailers include the different types of key events denoted by TPs. We present the percentage of trailers (on the development set) that include at least one shot per TP in Table 6.8. As can be seen, more than half of the trailers (i.e., 52.63% and 55.26%) include shots related to the first two TPs, whereas only 34.21% of trailers have any information about the two final ones. This is expected, since the first TPs are introductory to the story and hence more important for making trailers, whereas the last two may contain spoilers and are often avoided.

How Sentiment Connects with Trailers Empirical rules for trailer making[20] suggest that a trailer should start with shots of medium intensity to captivate the viewers, then decrease the sentiment intensity in order to deliver key information about the movie, and finally build up the tension until it reaches a climax.

[19]https://archive.nytimes.com/www.nytimes.com/interactive/2013/02/19/movies/awardsseason/oscar-trailers.html

[20]https://www.derek-lieu.com/blog/2017/9/10/the-matrix-is-a-trailer-editors-dream

Here, we analyze the sentiment flow in real trailers from our development set based on predicted sentiment scores (see Sections 6.2.1 and 6.3). Specifically, we compute the absolute sentiment intensity (i.e., regardless of positive/negative polarity) per shot in the (true) trailers. In accordance with our experimental setup, we again map trailer shots to movie shots based on visual similarity and consider the corresponding sentiment scores predicted by our network. We then segment the trailer into three equal sections and compute the average absolute sentiment intensity per section. In Table 6.9 presents the results. As expected, on average, the second part is the least intense, whereas the third has the highest sentiment intensity. Finally, when we again segment each trailer into three equal sections and measure the sentiment flow from one section to the next, we find that 46.67% of the trailers follow a "V" shape, similar to our sentiment condition for selecting sequences of trailer shots with GRAPHTRAILER.

6.7 Summary of Chapter

In this chapter, we proposed an approach for automatic and semi-automatic trailer moment identification, which adopts a graph-based representation of movies and uses interpretable criteria for selecting shots. We overcome the limitations of the previous chapter, by considering the movie video as our main source of information, which is segmented into shots. Although screenplays are no longer required during inference, we still use them as privileged information during training. We distill information from screenplays via contrastive learning and train a video-based model that can be used for TP identification and trailer moment identification. Trailers created by our model were judged favorably in terms of their content and attractiveness. Finally, we also present how our algorithm can be converted into an interactive tool assisting trailer creation with a human in the loop. We find that semi-automatically selecting trailer shots lead to informative and attractive trailers while minimizing the human involvement.

Until this point, we explored ways for identifying important content in movies and TV episodes. We are now able to create informative and attractive video summaries and trailers by selecting salient moments from the full-length narrative. The next question that we will answer in the following chapter is how we can use salient moments that contain multimodal information (i.e., video, text, audio) for producing *abstractive textual* summaries. We will focus on a dataset consisting of TV episodes and explore ways for incorporating multimodal information into a pre-trained textual summarizer.

Chapter 7

Long Video-to-text Summarization of TV Episodes

In the previous chapters we explored ways for identifying important and stylistic content in movies and TV episodes that can be used for creating video summaries and movie trailers. Moreover, we demonstrated that multimodal information and access to the full-length video and audio contributes to successfully retrieving such content. However, there are cases where further processing of salient narrative moments is needed. In particular, another common way of summarizing movies and TV episodes is by producing multiple-sentence *textual* summaries (e.g., IMDb[1], Wikipedia[2]).

In this chapter, we move a step further and explore ways for addressing the task of multimodal abstractive summarization (i.e., *video-to-text*) on TV episodes given salient content from the full-length narrative. We again hypothesize that having access to the video and audio will facilitate the summarization task by encouraging the generation of high-quality and factual textual summaries. Specifically, accessing the video of TV episodes will allow the inference of high-level events involving non-verbal cues, such as actions and emotions.

Focusing on abstractive summarization (Nallapati et al., 2016; See et al., 2017; Liu and Lapata, 2019b), there is an increasing interest in different types of dialogue summarization, e.g., from meeting transcripts (Gliwa et al., 2019; Zhong et al., 2021) or screenplays (Chen et al., 2022a). However, the contribution of modalities other than text remains relatively understudied. This is not entirely surprising given the challenges associated with the multimodal summarization task. Firstly, the input is long, it

[1]https://www.imdb.com

[2]https://en.wikipedia.org/wiki/Wikipedia:Plot-only_description_of_fictional_works

cannot fit into standard sequence-to-sequence architectures, and the different modalities have to be somehow combined; secondly, the output is also long, summaries consist of multiple sentences and rich vocabulary; and thirdly, it involves complex inference over long-range dependencies between events and characters and common sense reasoning. At the same time, creating large-scale multimodal datasets with long videos and aligned textual data is challenging and time consuming, limiting the research conducted in this domain. On the other hand, most prior work on video-to-video summarization, which identifies highlights from YouTube videos, TV shows, or movies (Song et al. 2015; Gygli et al. 2014; De Avila et al. 2011; Chapters 5 and 6), either focuses on short videos or utilizes small datasets with a few hundred examples. Finally, there is very limited work on video-to-text summarization. We are only aware of one large-scale multimodal dataset for this task, namely How2 (Sanabria et al., 2018), which again contains short videos (i.e., 2–3 minutes long) with simple semantics, and short, single-sentence summaries.

In contrast, we focus on video-to-text summarization on TV episodes and investigate how to best utilize multimodal information for condensing long inputs (e.g., an hour-long TV show) into long outputs (e.g., a multi-sentence summary). We create a multimodal variant of SummScreen (Chen et al., 2022a), a recently released dataset comprising of transcripts of TV episodes and their summaries. We collect full-length videos for 4,575 episodes and multiple reference summaries. We build our model on top of a pre-trained sequence-to-sequence architecture (i.e., BART; Lewis et al. 2020) fine-tuned on summarization and capable of generating fluent long text. We convert its textual encoder to a multimodal one by adding and tuning only adapter layers (Rebuffi et al., 2017; Houlsby et al., 2019), which account for 3.8% of model parameters. We also explore strategies for *content selection* inspired by the previous chapters, since the input is too long to fit into standard sequence-to-sequence models. For pre-selecting important moments from the long video and transcript, we experiment with different unsupervised and supervised methods (e.g., traditional retrieval approaches, creating pseudo-oracle labels for training a classifier). During our experimentation on content selection methods, we also consider information about the narrative structure as formulated by turning points (Chapter 3). Similarly to Chapter 4 we consider directly transferring a model trained on TP identification and the TRIPOD dataset to the new SummScreen^{3D} dataset.

Empirical results across various evaluation metrics demonstrate that multimodal information yields superior performance over just text, both in terms of content se-

lection and directly on abstractive summarization. This is the case even when our adapter model is compared to fully fine-tuned approaches and more memory-heavy architectures that can process the entire input (e.g., Longformer; Beltagy et al. 2020). Regarding content selection, we again verify that TP information is transferable across different narrative types and presents competitive performance on abstractive summarization against a supervised content selector trained on the target dataset.

The contributions of this chapter can be summarized as follows:

1. We augment SummScreen (Chen et al., 2022a) with multimodal information, providing videos aligned with transcripts and summaries; to the best of our knowledge, this constitutes the largest available resource for long video multimodal summarization.

2. We propose a *parameter efficient* approach to augment a pre-trained textual summarizer with multimodal information. We show that the multimodal-augmented summarizer can produce textual summaries of higher quality and factuality, even when compared with more memory-heavy and fully fine-tuned textual models.

3. We further validate that multimodal information is important for pre-selecting salient moments in TV episodes while considering different selection methods. During our experimentation, we again verify the transferability of TP-related information, similarly to Chapter 4, which demonstrates competitive performance against a supervised content selector tuned on the target dataset.

7.1 Related Work

Video Summarization Much previous work has focused on text-to-text or video-to-video summarization. We provide a comprehensive categorization of existing datasets according to input/output length and modality in Table 7.1. *Multimodal abstractive summarization* (video-to-text) has attracted less attention, mainly due to the difficulty of collecting large-scale datasets. How2 (Sanabria et al., 2018) is the only publicly available benchmark for this task, it includes short instructional videos with textual transcripts and one-sentence summaries. In contrast, we generate multiple-sentence summaries from long videos and their transcripts. Previous approaches to multimodal summarization have focused on various modality fusion methods with small RNN-based models (Palaskar et al., 2019). We take advantage of large pre-trained LMs

	Modality	Input	Output	Datasets
text-to-text	text	short	short	XSum Narayan et al. (2018), CNN-DailyMail Nallapati et al. (2016), NYT Durrett et al. (2016), Gigaword Napoles et al. (2012)
	text	long	long	SamSum Gliwa et al. (2019), QMSum Zhong et al. (2021), SummScreen Chen et al. (2022a)
video-to-video	vision	short	short	OVP De Avila et al. (2011), YouTube De Avila et al. (2011), SumMe Gygli et al. (2014)
	vision/text	short	short	TVSum Song et al. (2015)
	vision/text(/audio)	long	long/ short	LoL Fu et al. (2017), TRIPOD (Chapter 5), TRIPOD$\oplus$ (Chapter 6)
video-to-text	vision	long	short	TACoS Rohrbach et al. (2014)
	vision/text/audio	short	short	How2 Sanabria et al. (2018)
	vision/text/audio	long	long	SummScreen^{3D}

Table 7.1: Summarization datasets grouped based on the input/output modalities and input/output length.

(Lewis et al., 2020; Raffel et al., 2020; Radford et al., 2019) for generating fluent textual summaries.

Recent years have also witnessed increasing interest in multimodal video captioning, a task related to multimodal summarization, which aims to generate one-sentence descriptions for localized events in short videos (Xu et al., 2016; Rohrbach et al., 2017; Zhou et al., 2018a; Lei et al., 2020b). Existing methods employ strong language-and-vision encoders with massive pre-training (Li et al., 2020; Luo et al., 2020; Xu et al., 2021; Lei et al., 2020a; Li et al., 2021), while the decoder is typically shallow and under-trained. Although good at generating short descriptions, they cannot maintain fluency in long outputs with rich vocabularies.

Realizing the importance of large LMs for generation, recent work has focused on how to efficiently render pre-trained LMs multimodal. Notably, Tsimpoukelli et al.

(2021) convert a pre-trained LM into an image captioning model, by giving images as prompts and training only a vision encoder. Yu et al. (2021) summarize How2 videos by augmenting BART-base with visual information via a new cross-attention block added to every encoder layer. However, their approach cannot easily scale to BART-large and beyond since they add a large number of new parameters, while the dataset sizes are relatively small, leading to over-fitting.

Dialogue Summarization In the context of text-to-text generation, dialogue summarization is challenging due to the difficulty of fitting very long input into pre-trained sequence-to-sequence models. Longformer (Beltagy et al., 2020) alleviates this by employing local self-attention in combination with global tokens for reducing the computational overhead. Despite recent attempts to make self-attention more efficient (Kitaev et al., 2019; Tay et al., 2020; Zaheer et al., 2020), it is still unclear whether it has an advantage over content selection with a full-attention mechanism (Zhang et al., 2021; Shaham et al., 2022) for long dialogue summarization. Zhong et al. (2022) incorporate dialogue-specific objectives for pre-training summarization models, while Zhang et al. (2022) follow a different approach and hierarchically summarize the input chunk-by-chunk.

Parameter-efficient Tuning Fine-tuning is a common approach for transferring pre-trained models to different tasks or domains (Howard and Ruder, 2018). It is customary to fine-tune all the parameters of the pre-trained model which, however, becomes prohibitive as model size and number of tasks grow. Recent work has proposed parameter-efficient transfer learning methods which fine-tune only a small number of *additional* parameters. Two popular approaches include *adapter tuning*, where bottleneck layers are added and tuned at every layer of the model (Rebuffi et al., 2017; Houlsby et al., 2019) and *prompt tuning*, where (soft) prompts are prepended as part of the input (Brown et al., 2020; Li and Liang, 2021). In this work, we utilize the former method for adapting a textual summarizer to our multimodal setting and dialogue input format.

7.2 The SummScreen^{3D} Dataset

SummScreen (Chen et al., 2022a) is a long dialogue summarization dataset containing transcripts from TV episodes and human-written abstractive summaries. We ex-

Episodes	4,575
Input: transcript + video + audio	
Shots	1,048,024
Shots/episode	193.64 (109.09)
Utterances/episode	322.76 (116.52)
Tokens/episode	5720.55 (2223.38)
Output: summaries	
Summaries/episode	1.53 (0.79)
TVMegaSite/#tokens	4,280 395.69 (275.84)
YouTube/#tokens	334 136.22 (45.12)
IMDb/#tokens	946 111.21 (82.18)
tvdb/#tokens	1,454 126.14 (82.14)
Training (unique input-output pairs)	5,199
Validation episodes	296
Testing episodes	296

Table 7.2: SummScreen^{3D} statistics. For summaries, we show their provenance, number of summaries per site (second column), and mean number of tokens per summary; standard deviations are shown in parentheses.

tend this dataset to a multimodal setting by also considering the corresponding full-length videos. SummScreen is divided into two subsets depending on the series genre: SummScreen-FD and SummScreen-TMS. We use the latter subset which mostly covers soap operas from TVMegaSite[3], as it is easier to obtain full-length videos and each series has hundreds of episodes.

For each episode in SummScreen-TMS, we automatically search for the title and release date in Youtube. If there is a match with large duration (indicating that this is a full episode rather than a segment), we download the video and closed captions (CC). Overall, we collected videos for 4,575 episodes from five different shows in SummScreen-TMS. In addition to TVMegaSite summaries (distributed with SummScreen), we further retrieved summaries from YouTube descriptions, IMDb, and tvdb, again using the episode title and release date as search terms. The statistics of our dataset which we call SummScreen^{3D} (3D for language, video, and audio) are in Table 7.2. We also present in Table 7.3 the names of the TV shows and the number of

[3] http://tvmegasite.net

As The World Turns (atwt)	1356
Bold and the Beautiful (bb)	1113
Guiding Light (gl)	836
One Life to Live (oltl)	1118
Port Charles (pc)	501

Table 7.3: Distribution of different TV shows in the augmented dataset.

episodes per show (we made sure to have enough episodes from each TV show).

We split SummScreen^{3D} into training, validation, and test sets with the same distribution over different shows per set. We reserved 296 episodes for validation and the same number for testing, and used the rest for training. Since we have multiple reference summaries for some episodes, we increased the size of the training set by adding m episode-summary pairs, matching the same episode with each of its m references. This resulted in 5,199 unique samples for training.

7.3 Video-to-Text Summarization

Our approach leverages the generation capabilities of large pre-trained sequence-to-sequence models (Lewis et al., 2020; Raffel et al., 2020). As our backbone model, we employ BART-large (Lewis et al., 2020) which has been fine-tuned on CNN-DailyMail (Nallapati et al., 2016; Zhang et al., 2021) and has thus acquired a summarization inductive bias. As TV show transcripts are very long and cannot fit into BART, we select a subset of utterances (i.e., speaker turns) as input via content selection (see details in Section 7.4). We transfer this model to our task and domain (i.e., multimodal dialogue summarization), by adding adapter layers (Rebuffi et al., 2017; Houlsby et al., 2019) in both the encoder and decoder, and tuning them on SummScreen^{3D} while keeping the rest of the network frozen. We briefly discuss below our backbone text-based model and then elaborate on how we incorporate multimodal information.

7.3.1 Backbone Textual Model

Our summarizer follows a standard sequence-to-sequence Transformer architecture (Vaswani et al., 2017). The encoder maps tokens $[x_1, x_2, \ldots, x_N]$ to a sequence of contextualized representations $[h_1, h_2, \ldots, h_N]$ which are then fed to the decoder for generating the summary. The encoder consists of L stacked layers, each of which

has a self-attention block for contextualizing the token representations, followed by a feed-forward network. The decoder has a similar architecture, it additionally contains a *cross-attention* block for identifying relations between the input and currently generated text and makes use of *masked* self-attention to control access to context for each token. The decoder is followed by a linear layer (i.e., Language Model (LM) head) which projects the output representations onto the vocabulary and a final softmax layer. The model is optimized for predicting the next token w_{t+1} in the summary given $[w_0, w_1, \ldots, w_t]$, the context generated so far, and the transcript $[x_1, x_2, \ldots, x_N]$.

7.3.2 Multimodal Augmentation

Our hypothesis is that adding multimodal information to a textual summarizer (i.e., converting the textual encoder to a multimodal one) will increase the quality of its output summaries. We expect that the video/audio will compensate for important non-verbal information typically absent from the transcript (e.g., who is speaking to whom, who is present in the same room, who is crying or yelling). We further expect multimodal information to make up for the loss of context incurred by content selection. We next describe how we compute multimodal representations for an episode and how we augment BART with these representations.

Multimodal Representations We use *utterances* as the unit of representation for multimodal information. We segment episodes into shots (using PySceneDetect[4]) and map these to utterances in the corresponding transcript. Specifically, we align the closed captions in the video which are time-stamped to the utterances in the transcript using Dynamic Time Warping (DTW; Myers and Rabiner 1981; Chapters 5 and 6). We thus create a one-to-many alignment where an utterance corresponds to one or more shots. For each shot, we extract textual, visual, and audio features (see Section 7.5 for details), and compute an utterance-level representation for each modality by average pooling over all aligned shots.

Given textual, visual, and audio representations for utterance i, we learn a multimodal representation as part of our network:

$$\begin{gathered} \text{textual}'_i = f(W_x \text{textual}_i) \quad \text{visual}'_i = f(W_v \text{visual}_i) \quad \text{audio}'_i = f(W_a \text{audio}_i) \\ m_i = f(W_m[\text{textual}'_i; \text{visual}'_i; \text{audio}'_i]) \end{gathered} \tag{7.1}$$

[4] https://github.com/Breakthrough/PySceneDetect

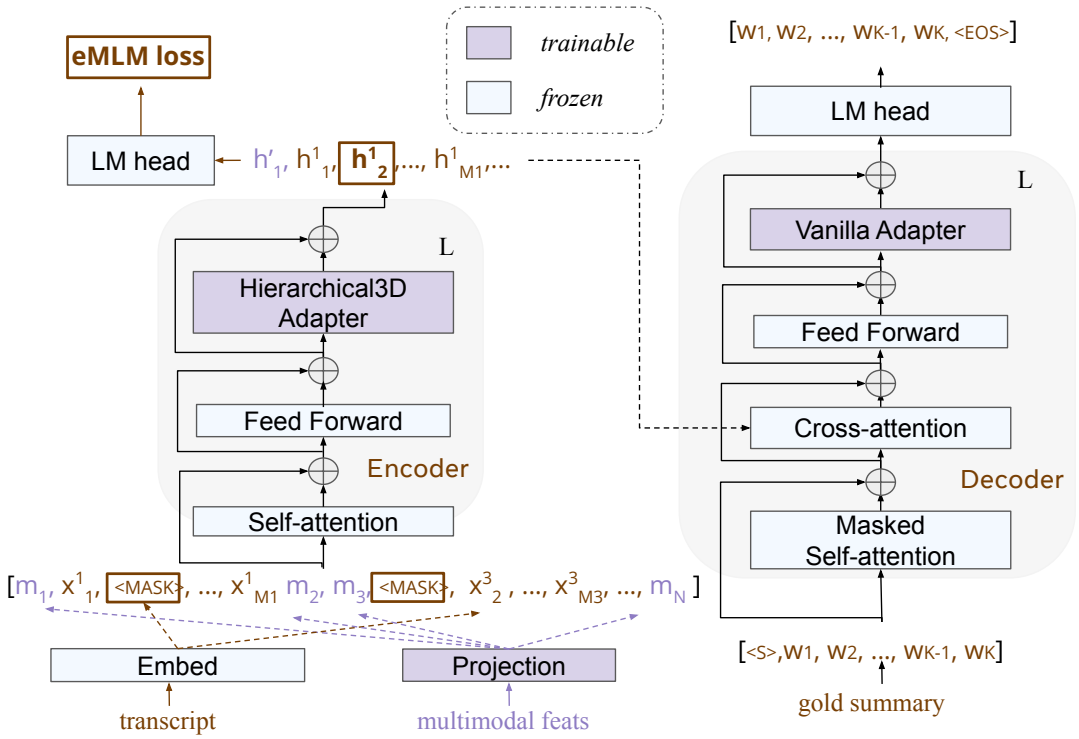


Figure 7.1: Multimodal augmentation of pre-trained BART. We augment the encoder and decoder layers with adapters which we fine-tune on the target dataset, while the remaining network is frozen. As input, we consider textual tokens and coarse-grained multimodal information which we prepend before each utterance. We also corrupt part of the textual input during training and add an auxiliary MLM loss to the encoder for predicting the corrupted tokens.

where $f(\cdot)$ is the ReLU activation function, $[\cdot;\cdot;\cdot]$ denotes concatenation, $W_x \in \mathbb{R}^{d_x x d_i}$, $W_v \in \mathbb{R}^{d_v x d_i}$, $W_a \in \mathbb{R}^{d_a x d_i}$, and $W_m \in \mathbb{R}^{3 d_i x d_m}$ are learnable matrices; d_i and d_m are the input and model dimensions with $d_i << d_m$, and m_i is the final multimodal representation corresponding to the i^{th} utterance in the transcript.

Multimodal Encoder In order to integrate utterance-level multimodal representations with BART, we consider a "global utterance token" inspired by the Longformer architecture (Beltagy et al., 2020). We preprocess the input into utterances and prepend a global token <EOS> per utterance as a placehoder for multimodal representations. The encoder thus receives as input sequence $[\mathbf{m_1}, x^1_1, x^1_2, \ldots, x^1_{M_1}, \ldots, \mathbf{m_N}, x^N_1, x^N_2, \ldots, x^N_{M_N}]$, where M_i is the length of the i^{th} utterance as number of tokens and "global" representations $\mathbf{m}$ constitute a rich multimodal space (i.e., they are not learned solely from text via local self-attention). We illustrate this in Figure 7.1.

7.3.3 Self-supervised Auxiliary Guidance

Our primary loss for training the model described above is the negative log likelihood of predicting the next token in the summary given episode $\mathcal{E}$:

$$\mathcal{L}_{LM} = \frac{1}{K} \sum_{t \in [1,K]} -\log p(w_t | w < t; \mathcal{E}) \tag{7.2}$$

where K is the length of the output summary. We further wish to encourage the model to attend to multimodal information and learn a meaningful projection (Equation (7.1)). To do this, we corrupt part of the textual input by masking tokens (see bottom left part of Figure 7.1) and adding an auxiliary masked language modeling (MLM) loss for the initial training steps only. So as not to disrupt the bias of the decoder, which is already trained on textual summarization, we apply the MLM loss in the outputs of the encoder while the model is trained on the downstream task. Given token-level encoder outputs $[h_1, h_2, \ldots, h_N]$, we copy and re-use the LM head of the decoder in order to project them into the vocabulary (see top left part of Figure 7.1). And compute the negative log likelihood only for the set of masked tokens $\mathcal{M}$:

$$\mathcal{L}_{eMLM} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} -\log p(x | h_{x_i \notin \mathcal{M}}) \tag{7.3}$$

We refer to this loss as encoder-based MLM loss (eMLM; Baziotis et al. 2021). It trains the encoder to reconstruct input text representations while attending to multimodal information. After X initial training steps, we drop the auxiliary loss and stop corrupting the textual input in order for the model to be optimized on summarization. We use a mixture of content word corruption (i.e., masking out named entities, nouns, and verbs excluding auxiliaries) and whole utterance corruption (Zhang et al., 2020; Zhong et al., 2022). We provide more details in Section 7.5.2.

7.3.4 Hierarchical3D Adapters

We specialize BART for our multimodal summarization task by inserting adapter modules (Rebuffi et al., 2017; Houlsby et al., 2019) into each encoder and decoder layer (after the feed-forward block). Each adapter adds only a small number of new parameters, which are randomly initialized and tuned on our end task, while the rest of the network is frozen. A vanilla adapter takes as input hidden representations $[\mathbf{u_1}, h_1^1, h_2^1, \ldots, \mathbf{u_N}, \ldots, h_{M_N}^N]$, where $h_1^1, h_2^1, \ldots, h_{M_N}^N$ are textual token-level hidden representations and $\mathbf{u_1}, \ldots, \mathbf{u_N}$ are multimodal utterance-level hidden representations (in

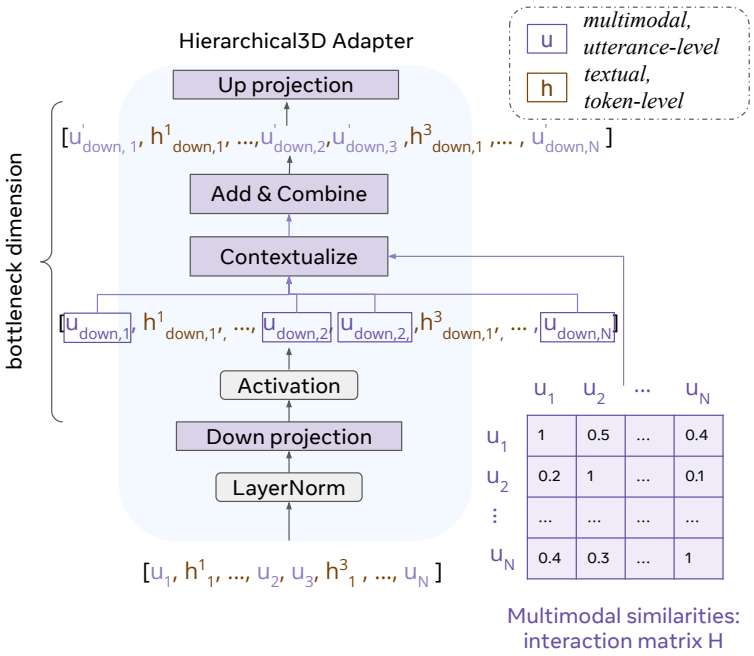


Figure 7.2: We show the hierarchical adapter added to each encoder layer: after down-projecting all representations, we only consider the multimodal ones and further contextualize them via attention. Then, we combine the representations and up-project again to the original model dimension.

accordance to the input format presented in Figure 7.1), and performs the following transformations:

$$h_{down,i} = f(\mathrm{LN}(W_d h_i + b_d)) \tag{7.4}$$

$$h_{up,i} = W_u h_{down,i} + b_u \quad h_i = h_i + h_{up,i} \tag{7.5}$$

where $W_d \in \mathbb{R}^{d_m \mathrm{x} d_{\mathcal{B}}}$, d_m is the model dimension, $d_{\mathcal{B}}$ is the bottleneck dimension of the adapter, $f(\cdot)$ is a non-linearity (here we use the ReLU non-linearity), LN a trainable layer normalization, $W_u \in \mathbb{R}^{d_{\mathcal{B}} \mathrm{x} d_m}$, b_d, and b_u are the corresponding bias vectors, and $h_{down,i}$ and $h_{up,i}$ are down and up projections of h_i.

In this work, we augment the vanilla adapters of the *encoder* with a hierarchical structure (illustrated in Figure 7.2). After computing (low level) self-attention between all input *textual tokens* in an encoder layer, we add a hierarchical adapter to compute *higher-level interactions* between *utterance-level multimodal* representations. By including this interaction block in the adapter, we can better propagate long-range dependencies between utterances and enforce a a more global view of the events in an episode and their associations, while keeping the number of trainable parameters low.

We first compute interaction (aka similarity) matrix H between utterances (see Fig-

ure 7.2) based on their *multimodal representations* $[m_1, m_2, \ldots, m_N]$ using the scaled dot product:

$$e_{ij} = (W_i m_i + b_i)(W_j m_j + b_j)/\sqrt{d_m} \tag{7.6}$$

where W_i, W_j are learnable projection matrices, d_m is the model dimension, and e_{ij} is the degree of similarity between m_i and m_j.

At each adapter layer of the encoder, after down-projecting all vectors to the bottleneck dimension, we further contextualize utterance-level multimodal representations $u_{down,i}$ with respect to each other given the degree of similarity provided by H ("Contextualize" block in Figure 7.2):

$$u'_{down,i} = \sum_{k=1}^{N} \text{softmax}(H_{ik}/\tau) u_{down,k} + u_{down,i} \tag{7.7}$$

where N is the number of utterances, and τ is a low temperature parameter (< 1) for increasing sparsity. After contextualization, we up-project all vectors to the original dimension d_m, as in vanilla adapters (Equation (7.5)).

7.4 Content Selection

As explained earlier, episodes in SummScreen^{3D} are very long (5,720 tokens on average). As a result, BART, which has a maximum token length of 1,024, can approximately encode one fifth of the transcript.[5] We therefore perform content selection, i.e., identify salient utterances and give these as input to BART at inference time. We describe below three approaches inspired by information retrieval, summarization (Gehrmann et al., 2018; Liu and Lapata, 2019a), and computational narrative analysis (Chapters 3, 5 and 6).

Retrieval-based Selection We follow previous approaches (Zhang et al., 2021) in determining salient content with BM25 (Robertson and Zaragoza, 2009). BM25 is a widely known retrieval model similar to tf*idf. It assigns each utterance a "relevance" score (by comparing it against the entire transcript). Utterances with high scores are deemed salient and the K best ones are selected.

[5] We can extend the positional embeddings to 1,536 by applying bilinear interpolation, however, the memory requirements would be prohibitive for longer sequences.

Learning-based Selection Alternatively, we may also model content selection as a binary classification problem. Given a transcript containing N utterances we predict whether each should be selected as input for the downstream summarization task (label 1) or not (label 0). We create noisy labels by matching transcript utterances to (reference) summary sentences. Specifically, we encode sentences and utterances via Sentence-BERT (Reimers and Gurevych, 2019), and assign a positive label to the utterances most similar to the reference sentences. A content selector is then trained on these pseudo-labels to identify salient utterances. We can also incorporate multimodal information in this content selection setting, using the same utterance-level representations fed into BART. We first contextualize them via a shallow transformer encoder, and add a classification head for predicting important utterances. The model is optimized with binary cross-entropy loss. During inference we select the top K predicted utterances.

Turning Point (TP) identification We also perform content selection by identifying the turning points of an episode as salient content. In accordance with the previous chapters, and similarly to Chapter 4, we use the video-based TP identification model of Chapter 6 (see Section 6.2.2), which is pre-trained on TRIPOD, for identifying key events in SummScreen^{3D}. The TP identification model considers the same multimodal information as the content selector above and predicts the utterances that represent each TP. Since TPs are distinguished into five different types depending on their functionality (i.e., Opportunity, Change of Plans, Point of No Return, Major Setback, Climax), we consider the top $K/5$ predicted utterances per turning point as salient utterances.

7.5 Experimental Setup

7.5.1 Dataset Pre-processing

Given full-length video, we extract features for all modalities at the utterance-level as mentioned in Section 7.3.2 and similarly to the previous chapter. For text, we extract sentence-level features using Sentence-BERT (Reimers and Gurevych, 2019). Each utterance in the transcript is thus represented by a fixed-size vector. For the frames, we extract two types of features: frame-level features using the CLIP visual encoder (Radford et al., 2021) and motion-level features from video clips using Slowfast (Feichtenhofer et al., 2019). We then aggregate frame- and motion-level features to

utterance-level given the automatic alignment by mean pooling. Finally, for audio, we use YAMNet pre-trained on the AudioSet-YouTube corpus (Gemmeke et al., 2017) for classifying audio segments into 521 audio classes (e.g., tools, music, explosion); for each audio segment contained in a shot, we extract features from the penultimate layer, and then aggregate representations again to utterance-level via mean pooling (same as in Chapter 6).

7.5.2 Implementation Details

Hyperparameters and Training We corrupt the textual input and use the auxiliary eMLM loss (Section 7.3.3) only for the first X =1,500 training steps, when we train our model for a total of 12,000 steps. During corruption, we mask out all content words (i.e., named entities, verbs, and nouns) and a random 10% of the input utterances. For generating summaries during inference, we use beam search with beam = 5 and 3-gram blocking (Paulus et al., 2018).

We used the Adam algorithm (Kingma and Ba, 2015) for optimizing our networks. We trained all models with a learning rate of $3e{-}5$ for 12k steps using a linear warm-up of 500 steps, followed by inverted squared decay. All BART-based models were trained with batch size of 1 episode on 4 P100 GPUs with 16GB memory and label smoothing (Szegedy et al., 2016) of 0.1. To fine-tune the LED-based models, we used 4 A100 GPUs with 80GB memory.

Training vs. Inference Although we experiment with different content selection methods during inference, we randomly sample input utterances during training. Random sampling acts as data augmentation, since the model sees slightly different input-output pairs during training at different iterations. We experimentally verify in Section 7.6 this is preferable to a fixed selection of utterances, especially considering the small dataset size. We select $K = 60$ utterances to feed into BART models given the input length limit, and order them according to their original position in the transcript.

7.5.3 Evaluation Metrics

We evaluate the generated summaries using ROUGE F1 (Lin, 2004) against reference summaries. However, there is mounting evidence that ROUGE is not always a good indicator of summary quality and does not discriminate between different types of errors, in particular those relating to factuality. We therefore consider additional metrics based

on Question-Answering (QA). We obtain questions based on the gold summaries and then evaluate whether the correct answers exist in the generated summaries. We expect factual summaries to be able to correctly answer a higher percentage of questions.

As in prior work (Maynez et al., 2020; Kryściński et al., 2020; Honovich et al., 2021), we automatically generate QA pairs against reference summaries. We identify named entities and nouns using spaCy (Honnibal and Montani, 2017), and feed them as gold answers alongside the summaries to a question generator. We discriminate between named entities and nouns as answer types for measuring factuality in event-entity associations and other attributes pertaining to nouns. We used T5-base (Raffel et al., 2020) as our question generator and RoBERTa-base (Liu et al., 2019) as the QA system for answering questions given system generated summaries as input passages. Both were fine-tuned on SQuAD2.0 (Rajpurkar et al., 2016).

We measure accuracy as the partial overlap between gold and predicted answers for named entities. For nouns, we resort to textual entailment in order to account for synonyms and paraphrases in the generated summaries. We concatenate the question with gold or generated answer and predict a score for the directional relation between them. If the score is above 0.5, we consider the generated answer correct. We used BART-large (Lewis et al., 2020) fine-tuned on the MultiNLI corpus (Williams et al., 2018) as our entailment model.

We created a test suite of gold QA pairs, by retaining only those that can be answered correctly by the QA model given the reference summaries (Honovich et al., 2021). We overall generated 2,513 questions for named entities and 381 questions for nouns for the 296 episodes in our test set. On average, we have 8.5 questions per episode for named entities and 2.3 questions for nouns.

7.6 Results

Content Selection Table 7.4 compares how different approaches to content selection influence summarization performance according to ROUGE F1. We compare some simple baselines like selecting the Lead, Middle, and Last 60 utterances from the transcript as well as at Random. In addition, we compare a text only summarizer against our Hierarchical3D model. As can be seen, differences amongst content selection methods are generally small. BM25 (i.e., retrieval) performs worse than random whilst a multimodal content selector trained on pseudo-labels performs overall best. As an upper bound, we also report results with oracle labels as input demonstrating that

Selection	R-1	R-2	R-L
Lead	32.91	6.51	30.72
Last	32.65	6.41	30.59
Middle	33.02	6.70	31.03
Random selection	33.07	6.54	30.91
+ Hierarchical3D	34.33	7.24	32.15
Retrieval	32.36	6.30	30.20
+ Hierarchical3D	33.83	6.89	31.42
TP identification	33.31	6.78	31.24
+ Hierarchical3D	34.49	7.36	32.01
Content Selection	33.27	6.74	31.22
+ Hierarchical3D	**34.51**	**7.62**	**32.64**
Pseudo-oracle	*35.09*	*7.96*	*32.85*
+ Hierarchical3D	*35.69*	*8.42*	*33.40*

Table 7.4: Content selection methods for textual and multimodal BART (+Hierarchical3D).

there is still room for improvement. Regardless of how content is selected, we observe that our Hierarchical3D variant significantly improves performance, and interestingly, the performance gap is larger when the selection method is weaker (e.g., random vs. pseudo-oracle). This indicates that to a certain extent multimodal information makes up for suboptimal content selection.

Text vs. Multiple Modalities In Table 7.5 we compare our multimodal model (with the best performing content selector) against textual summarizers developed for processing long input or specifically for dialogue summarization. These include Longformer (LED; Beltagy et al. 2020) with full fine-tuning[6], a variant of LED pre-trained on dialogues (DialogLED; Zhong et al. 2022), and Summ^N (Zhang et al., 2022), a two-stage hierarchical approach for long dialogue summarization. We also present text-only BART variants, with full fine-tuning (FT) and adapter-tuning (AT).

As can be seen in the second block of Table 7.5, tuning only the adapter layers (BART AT) does not hurt performance compared to full fine-tuning (BART FT), pre-

[6]Adding and tuning only adapter layers in LED gave us inferior performance by a large margin, indicating that adapting such a network is not straightforward. Hence, converting it into a multimodal version is also more challenging.

Models	R-1	R-2	R-L
LED FT	33.53	**7.60**	31.77
DialogLED FT	32.66	7.38	31.12
Summ^N FT	24.71	4.42	22.61
BART FT	32.61	6.94	30.83
BART AT	33.27	6.74	31.22
BART AT + Hierarchical3D	**34.51**	**7.62**	**32.64**

Table 7.5: Comparison of our model (BART AT + H-3D) with text-only summarizers for long dialogue summarization. For all BART variants we perform content selection (FT: full fine-tuning, AT: adapter-tuning).

Models	Acc (NEs)		Acc (NNs)	
	text	+H-3D	text	+H-3D
Random selection	20.25	23.64	33.86	38.06
TP identification	21.65	24.07	40.42	**40.68**
Content Selection	20.65	**24.71**	38.58	39.37
Pseudo-oracle	*28.53*	*29.64*	*41.73*	*42.00*
LED FT	20.89	—	37.95	—
DialogLED FT	21.09	—	36.22	—
Summ^N FT	18.03	—	34.91	—

Table 7.6: QA evaluation on named entities and nouns. We denote our Hierarchical3D model with H-3D.

sumably due to the small dataset size. Addition of multimodal information with hierarchical adapters (BART AT + Hierarchical3D) yields substantial ROUGE improvements. Interestingly, our performance is superior to fully fine-tuned, memory-heavy models like LED or DialogLED that process the entire transcript as input. This suggests that representations from multiple modalities are more informative and lead to higher performance compared to efficient self-attention mechanisms. Summ^N performs demonstrably worse than all one-stage methods.

QA Evaluation The results of our automatic QA evaluation are summarized in Table 7.6. The first block focuses on model performance with different content selection variants. We only compare text-only and multimodal (+H-3D) BART variants. Again,

	BoC-p	BoC-r	BoC-f1	BoR-p	BoR-r	BoR-f1
Random selection	82.55	38.71	52.71	29.82	9.39	14.28
+ Hierarchical3D	81.80	47.37	60.00	31.75	13.77	19.21
TP identification	**84.31**	38.93	53.26	**36.79**	10.33	16.13
+ Hierarchical3D	82.20	47.10	59.89	34.82	**14.10**	**20.07**
Content Selection	81.60	36.59	50.52	30.54	8.58	13.40
+ Hierarchical3D	81.90	**48.48**	**60.91**	33.04	**14.37**	**20.03**
Pseudo-oracle	*87.42*	*46.95*	*61.09*	*37.92*	*14.40*	*20.87*
+ Hierarchical3D	*85.53*	*52.37*	*64.96*	*36.67*	*17.51*	*23.70*
LED FT	82.28	33.54	47.65	34.35	10.64	16.25
DialogLED FT	82.93	38.19	52.27	31.71	10.32	15.57
Summ^N FT	82.74	29.14	43.10	34.73	9.39	14.78

Table 7.7: Entity-specific metrics (test set).

we find that augmenting BART with multimodal information regardless of the selection method improves accuracy, especially for named entities (Columns 2 and 4 in Table 7.6 vs 3 and 5). This is true even when content is selected by a pseudo-oracle suggesting that multimodal information provides better associations between events and entities, even when the input contains all salient information. Moreover, we observe that supervised content selection and TP identification offer the best performance. This further validates our hypothesis that identifying turning points can provide important information and facilitate summarization (both in extractive and abstractive settings). The second block compares our approach with state-of-the-art models on dialogue summarization; we find these models perform on par or slightly worse than textual BART (depending on the content selection method) which casts doubts on their ability to efficiently consume longer inputs.

Entity-specific Metrics Chen et al. (2022a) propose a set of entity-specific metrics in order to investigate the role of characters, which are fundamental in TV shows, in the generated summaries. Specifically, they measure several bag of character (BoC) metrics based on character overalp between generated and gold standard summaries. They define precision as the fraction of the correctly mentioned characters with respect to all characters that appear in the generated summary (BoC-p) and recall as the fraction of the correctly mentioned characters with respect to all characters that appear in the gold summary (BoC-r). Given precision and recall, we also measure F1-score

(BoC-f1).

Apart from correctly mentioned characters, Chen et al. (2022a) also compute similar bag of words metrics for the relations between characters in the summaries. Specifically, they consider a pair of characters related if they appear in the same sentence in the summary. They do not account for the direction of the relations and focus only on co-occurrence. For measuring these relations, they again consider precision (BoR-p) and recall (BoR-r) of the intersection of pairs of characters similarly to computing the BoC metrics. We also report F1-score (BoR-f1), given the precision and recall for character relations.

We summarize the entity-specific results in Table 7.7. Overall, especially when considerring the F1 scores for characters and relations, we arrive to similar conclusions as with our automatic QA evaluation (Table 7.6). First, the multimodal information that is incorporated in our Hierarchical3D approach increases most entity-specific metrics in comparison with the text-only variants. Regarding different content selection methods, TP identification and supervised content selection again perform best in comparison with random selection, although differences are not large. Finally, we achieve the best F1 scores in both entity- and relation-specific metrics by using oracle selection, indicating that there is still room for improvement. Interestingly, we again observe a further increase in performance by adding multimodal information in the pseudo-oracle variant, suggesting that video-based information is important even when we consider the most salient parts of an episode.

Finally, we also compare our approach with state-of-the-art, fully finetuned textual summarizers for long dialogues. We again notice that Summ^N is the weakest option regarding the entity-specific metrics. Next, considering efficient architectures for modeling the entire input (i.e., LED, DialogLED) has competitive performance with our text-only variants with content selection. However, Hierarchical3D that considers multimodal information outperforms these memory-heavy models while training only a small fraction of model parameters. This further validates our hypothesis that the video can provide further information that is more important for high-quality summaries than exploring efficient methods to process the entire textual input.

Ablation Studies In Table 7.8 we summarize our findings from ablation studies which aim to isolate which modeling components contribute to better performance. We observe that individual modalities (Text, Audio, Video) perform worse on their own than in combination (Multimodal). The least informative modality is audio, while

Modality	R-1	R-2	R-L
Text	34.74	7.11	32.46
Audio	33.95	6.92	31.90
Video	34.86	7.24	32.73
Multimodal	**34.95**	**7.51**	**33.01**
w/ vanilla adapters	34.25	7.45	32.41
w/o eMLM loss	33.80	6.84	31.88
w/o random augmentation	33.45	6.48	31.81

Table 7.8: Role of multimodal information and hierarchical adapters in summarization performance.

the most informative one is video. While considering multimodal information, we substitute the hierarchical adapters in the encoder with vanilla adapters and observe a small drop in performance. Removal of the auxiliary eMLM loss during training leads to a further performance drop. The auxiliary loss is crucial rendering the textual encoder multimodal and forcing an already tuned summarizer to consider a different type of input. Finally, data augmentation (via random content selection) during training is also important given the small size of the dataset and BART encoder length restrictions.

In Table 7.9, we directly test the performance of different content selectors. We report precision (Pre), recall (Re), and F1 score of model variants based on pseudo-oracle labels. We first consider selectors which have not been trained with pseudo-oracle labels, such as Random, Retrieval (i.e., BM25) and TP identification (we refer to these approaches as unsupervised). We observe that unsupervised baselines have significantly lower F1 score in comparison with a supervised approach. Interestingly, although TP identification "agrees less" with the pseudo-oracle labels in comparison with BM25, TPs still present competitive performance against the supervised content selector on abstractive textual summarization (e.g., Table 7.6). This indicates that although the predicted turning points are not the same as the pseudo-oracle utterances, they still include salient information that facilitates summarization. Finally, comparing the multimodal supervised content selector with equivalent unimodal models, we observe that the highest performance is achieved by combining all modalities. This result again validates our hypothesis from previous chapters that information from the full-length audio and video can contribute to selecting important moments in movies and TV episodes. Investigating the unimodal variants, we also conclude that the most

	Pre (%)	Re (%)	F1 (%)
Unsupervised			
Random	19.55	20.90	20.06
Retrieval	24.63	26.62	25.40
TP identification	20.35	22.10	21.04
Supervised			
Multimodal	**47.57**	**50.68**	**48.57**
Text	45.26	48.54	46.52
Vision	22.97	24.91	23.73
Audio	21.54	23.29	22.23

Table 7.9: Role of multimodal information in content selection. Precision (Pre), Recall (Re) and F1 score for selecting important utterances from the episode. The supervised models are trained given the pseudo-oracle labels.

informative one is the textual modality, while using visual or audio cues alone is not enough to predict salient content.

7.7 Examples of Generated Summaries

In this section we provide examples of generated summaries based on different automatic systems. Moreover, we provide examples of questions and answers used for the automatic QA evaluation described in Section 7.5.3.

Table 7.10 shows examples of the automatically generated question-answer pairs given gold standard summaries. We provide examples of QA pairs for named entities (first 4 rows of the table) and nouns (remaining 6 rows of the table). We observe that most QA pairs are reasonable and correspond to information given in human-written summaries (first column of the table). However, there are cases where the QA pairs do not provide reasonable questions. Such an example is illustrated in the last row of Table 7.10, where the question is generated given the summary segment "Jonathan and Lizzie find out their baby has a medical condition, and make a run for it":

Q: "What do Lizzie and Jonathan do when they learn their baby has a medical condition?"

A: "run"

This QA pair does not correspond to a reasonable fact of the episode. This shows that although it is useful to filter the questions, there are still imperfections with the automatic generation of QA pairs, especially when considering nouns.

Next, we give examples of the generated summaries for the TV show "Port Charles" in Tables 7.11–7.14. In every case, we present the gold or generated summary alongside the QA pairs used for evaluation. First, we compare different content selection methods (i.e., supervised content selection (CS), TP identification (TPs), and pseudo-oracle) for a text-only summarizer based on BART with adapter tuning. We present two examples in Tables 7.11 and 7.13 (we also show gold summaries for each episode). In both cases, we observe that the pseudo-oracle selection provides summaries of better quality, with fewer errors in the questions answered (i.e., errors are illustrated with red). Moreover, when comparing content selection (CS) with TP identification (TPs), we find that these two approaches present competitive performance, as suggested by our main experimental results (Table 7.6). Specifically, in Table 7.11, TP identification seems to provide the most informative summary, whereas in Table 7.13 supervised content selection is the best option.

Secondly, we compare our approach that considers multimodal information (Hierarchical3D) against text-only BART with equivalent content selection, and LED which considers only text and uses an efficient self-attention mechanism for processing the entire input. We present two examples for the same episodes as above in Tables 7.12 and 7.14. We empirically validate that the quality of the generated summaries is improved by adding the multimodal information (both when using supervised content selection and TP identification). Our approach leads to summaries that answer correctly a larger percentage of automatic questions (i.e., correct answers are illustrated with green) outperforming LED, which is fully fine-tuned and memory-heavy. Interestingly, LED summaries cannot answer a large proportion of the given questions, suggesting that such methods may not be suitable for the task and small dataset size.

7.8 Summary of Chapter

In this chapter, we moved from a video-to-video to a video-to-text summarization setting. Given salient content from TV episodes, that can be retrieved based on different methods, we explored ways for producing abstractive textual summaries while considering information from the full-length video and audio. Specifically, we proposed a parameter-efficient way for incorporating multimodal information into a pre-trained

textual summarizer, which has strong generation capabilities and the appropriate inductive bias for the task, by using adapter modules augmented with a hierarchical structure. Our approach adapts the textual summarizer to the multimodal setting while training only 3.8% of model parameters, which is crucial when dealing with smaller datasets, such as SummScreen^{3D}. We overall demonstrate that multimodal information and access to the full-length video and audio is important for generating high-quality and factual abstractive summaries. This reinforces our argument that considering all modalities is crucial for summarization, which is true for both video-to-video and video-to-text settings.

Summary	Question	Answer
Sage goes to live with Jack after she learns Carly is planning to marry Craig. Meg agrees to marry Dusty.	Who does Meg agree to marry?	Dusty
	Who does Sage go to live with?	Jack
Joshua is busy preparing for Allison's arrival, as he unveils Kevin's latest creation; a portrait of Allison and Joshua in their wedding attire. Lucy goes to church to plead for answers. Ian overhears her plea and swears that he will not let her die. Livvie shows Joshua a picture of Allison appearing to be dead and tells him that he was right her fangs are poisoned.	Who goes to church to plead for answers?	Lucy
	Who swears he will not let Lucy die?	Ian
	What does Lucy do at church?	plea
	What part of Allison's body is poisoned?	flangs
Lizzie and Jonathan spend some time with their baby. Jonathan gives in to one of Alan s demands. Gus and Harley find a disk with some interesting information on it. Gus still can t figure out what it is that Blake has on him. Dinah and Mallet argue over who will be the next WSPR star. Tammy is heartbroken after a visit to the hospital. Jonathan and Lizzie find out their baby has a medical condition, and make a run for it. Alan realizes that he may have been outwitted by Jonathan. Gus vows to get to the bottom of his supposed secret.	What does Gus vow to find out about Blake?	secret
	What is Lizzie and Jonathan spending time with?	baby
	What do Gus and Harley find?	disk
	What do Lizzie and Jonathan do when they learn their baby has a medical condition?	run

Table 7.10: Examples of automatically generated QA pairs for evaluation of generated summaries.

Model	Summary
Gold	Caleb is upset when Livvie tells him that Joshua has the ring. Joshua attempts to sway Ian to the dark side, but Ian vows he will continue to fight Joshua and the other vampires. Rafe tells Caleb the only way he can defeat Joshua now is to remain human and Livvie reluctantly agrees. Lucy pleads with Victor to fight Joshua, however, it s too late, as Victor tells her he enjoys the power Joshua has given him. Karen realizes Frank is a vampire.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Frank** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Livvie** • Who realizes Frank is a vampire? -**Karen** • What does Livvie tell Caleb Joshua has? -**the ring** • Who does Karen realize Frank is? - **vampire**
CS (text-only)	Caleb and Rafe discuss how to get close to Joshua and Livvie. Lucy tries to convince Victor that Joshua is an evil vampire who should not be allowed to have his soul. Lucy tells Victor that she can t lose him and wants him to accept her offer to turn him back into a vampire. Joshua tells the people of Port Charles that he will do whatever it takes to breathe new life into this wonderful old place.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Joshua** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Rafe** • Who realizes Frank is a vampire? -**Victor** • What does Livvie tell Caleb Joshua has? -**soul** • Who does Karen realize Frank is? - **vampire**
TPs (text-only)	Caleb and Livvie are shocked to find out that the ring has been taken away from them by Joshua. They are unable to get the ring back, but they are determined to find a way to get it back. Lucy tells Victor that Joshua is a liar and that he should not be allowed to have an important position in Port Charles. Victor tells Lucy that he will not give up on her, but she tells him that she will not go to the hospital because she has to be here for Victor. Lucy and Victor agree that they need to stay at the hospital for the sake of Victor, but Lucy is not willing to give up her job as Victor s guardian angel. Rafe tells Alison that if she turns back, she is going to be under Joshua s control, and he doesn t think she will be too happy about taking orders from someone else. He tells her that he would rather slit her own wrists than have Joshua touch her.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Joshua** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Lucy** • Who realizes Frank is a vampire? -**Lucy** • What does Livvie tell Caleb Joshua has? -**the ring** • Who does Karen realize Frank is? - **vampire**
Pseudo-oracle	Caleb and Livvie find out that the ring that protects them from Joshua is gone, and they have no way of getting it back. Lucy and Ian try to convince Victor to turn Joshua into a vampire, but he tells them that he is not a monster and that he wants to be a good man. Lucy tells Victor that Joshua is evil, but Victor tells her that she is his friend and he will not let her get to him. Victor tells Lucy that he will do whatever it takes to breathe new life into Port Charles. Ricky tells Jamal that he does not want to be his guardian angel anymore, but Jamal tells him to go and hang with the bloodsuckers. Jack tells Jack that the only way he is going to win is to play both sides of the fence. Frank tells Karen that there are benefits to his new role as a vampire and he is willing to do it for the good of the town of Port Charles and his family.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Frank** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Livvie** • Who realizes Frank is a vampire? -**Karen** • What does Livvie tell Caleb Joshua has? -**the ring** • Who does Karen realize Frank is? - **vampire**

Table 7.11: Gold summary with automatically generated QA pairs (top) and model summaries with different content selection methods. Questions which the automatic summary answers correctly are highlighted in green (wrong answers shown in red). All model variants consider the textual modality only (i.e., BART with adapter tuning).

Model	Summary
Gold	Caleb is upset when Livvie tells him that Joshua has the ring. Joshua attempts to sway Ian to the dark side, but Ian vows he will continue to fight Joshua and the other vampires. Rafe tells Caleb the only way he can defeat Joshua now is to remain human and Livvie reluctantly agrees. Lucy pleads with Victor to fight Joshua, however, it s too late, as Victor tells her he enjoys the power Joshua has given him. Karen realizes Frank is a vampire.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Frank** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Livvie** • Who realizes Frank is a vampire? -**Karen** • What does Livvie tell Caleb Joshua has? -**the ring** • Who does Karen realize Frank is? - **vampire**
Text-only (TPs)	Caleb and Livvie are shocked to find out that the ring has been taken away from them by Joshua. They are unable to get the ring back, but they are determined to find a way to get it back. Lucy tells Victor that Joshua is a liar and that he should not be allowed to have an important position in Port Charles. Victor tells Lucy that he will not give up on her, but she tells him that she will not go to the hospital because she has to be here for Victor. Lucy and Victor agree that they need to stay at the hospital for the sake of Victor, but Lucy is not willing to give up her job as Victor s guardian angel. Rafe tells Alison that if she turns back, she is going to be under Joshua s control, and he doesn t think she will be too happy about taking orders from someone else. He tells her that he would rather slit her own wrists than have Joshua touch her.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Joshua** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Lucy** • Who realizes Frank is a vampire? -**Lucy** • What does Livvie tell Caleb Joshua has? -**the ring** • Who does Karen realize Frank is? - **vampire**
Hierarchi-cal3D (TPs)	Caleb and Livvie are shocked when they find out that their protection against Joshua is gone. Victor and Lucy argue about Victor's role in Port Charles. Lucy tells Victor that Joshua is evil and that he should not be allowed to have an important position with the vampires. Victor tells Lucy that he still has so much to contribute and maybe this is his chance to have people listen to him again. Lucy is upset that Victor wants to give Joshua an important role in the town. Lucy and Victor are at the hospital and Lucy tells him that she is going to be there for Victor, but he tells her to stay away from him. Frank tells Karen that he has grown a pair of fangs. Karen is shocked to hear that Frank is a vampire.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Frank** • Who pleads with Victor to fight Joshua? -**Lucy** • Who tells Caleb that Joshua has the ring? -**Lucy** • Who realizes Frank is a vampire? -**Karen** • What does Livvie tell Caleb Joshua has? -**their protection against Joshua** • Who does Karen realize Frank is? - **vampire**
LED	At the end of the night, Caleb and Livvie s wedding ring is revealed to Rafe and Caleb s ring. Caleb tells Rafe that the ring is a vampire slayer. Rafe tells Caleb that he s going to be a vampire again. Lucy tells Victor that Victor is going to take over the town of Port Charles. Victor tells Lucy that he wants to talk to Lucy. Lucy asks Victor to join her in the new venture, but Victor tells her that he is not going to do it. Caleb tells Olivia that he has been drugged by Kevin, and he s been turned into a vampire. Olivia tells him that she wants to be part of the new club, but Caleb tells her to stay away from him. Joshua tells Ian that he will not be able to get Victor away from Victor. Ian tells Joshua that Joshua is not one of the vampire slayers, but he is the one of them.
QA pairs	• Who tells Lucy that he enjoys the power Joshua has given him? -**Victor** • Who does Karen realize is a vampire? -**Caleb** • Who pleads with Victor to fight Joshua? -**Ian** • Who tells Caleb that Joshua has the ring? -**Ian** • Who realizes Frank is a vampire? -**Rafe** • What does Livvie tell Caleb Joshua has? -**wedding ring** • Who does Karen realize Frank is? - **slayer**

Table 7.12: Gold summary with automatically generated QA pairs (top) and model summaries. Questions which the automatic summary answers correctly are highlighted in green (wrong answers shown in red). We compare our approach (i.e., Hierarchical3D) with state-of-the-art textual summarizers (i.e., LED).

Model	Summary
Gold	Joshua tells Elizabeth he wants to turn Allison and demands she help ease Allison into her new life as his wife. Elizabeth tells Joshua she will kill him before she allows him to hurt Allison. Livvie is able to fend off her need to feed while she and Caleb make love. Frank searches for Allison. When Frank attempts to kidnap Allison from Rafe, he discovers that it really is Lucy and I an in disguise. Allison and Rafe reappear in Caleb s cave.
QA pairs	• Who does Frank try to kidnap Allison from? -**Rafe** • Who does Frank try to kidnap? -**Allison** • Who tries to kidnap Allison? -**Frank** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**wife**
CS (text-only)	Rafe tells Alison that he will never let Joshua take her for his bride, but she tells him that she has no choice in the matter. Elizabeth tells Joshua that she will not stand by and allow him to take her daughter. Joshua tells Elizabeth that he is going to eas e Alison into her new lifestyle as his wife. Elizabeth says that she is not going to let her daughter suffer the kind of nightmare that she lived. She will kill Joshua before he is even that close to turning her. Alison tells Rafe that she thinks this is a little extreme, that is all. Rafe says he will not let Joshua get to her. He promises to keep her away from Joshua and all his goons. Caleb tells Livvie that she doesn t need to feed. He tells her that he can t make love to her the way she wants to. She tells him she can t turn him back, but he tells her he can. He says that he loves her and that he wants to make her his bride.
QA pairs	• Who does Frank try to kidnap Allison from? -**Joshua** • Who does Frank try to kidnap? -**Joshua** • Who tries to kidnap Allison? -**Rafe** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**wife**
TPs (text-only)	Livvie tells Caleb that she can t be with him, knowing what his bite might do to him. Joshua tells Elizabeth that he is going to steal Alison s slayer s intended and that will make him more respectable. He tells her that she is not going to do it, and that he knows that both roads lead Alison right back to Joshua. Elizabeth tells Joshua that she has no idea how much she would love to do that, but they both know that they are too busy figuring out which road to take. Rafe and Alison find themselves in Caleb s old hangout. Alison is shocked to see that Rafe is still there. She tells Rafe that she will feel better when all this is over with. She asks Rafe if he has his back and he says that he does, and he tells her he has her back. Lucy and Ian find out that they have succeeded in their mission to kidnap Alison. Ian and Frank tell Lucy that they will have to tell their boss that their mission didn t work. Frank tells Ian that he will pay for this, and they are both going to pay for it. Ian tells Frank that he has to tell his big, powerful boss that his mission failed.
QA pairs	• Who does Frank try to kidnap Allison from? -**Lucy** • Who does Frank try to kidnap? -**Alison** • Who tries to kidnap Allison? -**Lucy** • Who can fend off her need to feed while she and Caleb make love? -**Rafe** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**Caleb**
Pseudo-oracle	Caleb tells Livvie that he loves her and wants to make love to her, but he can t because his fangs are poisoned and he can s not turn her back. He tells her to take the risk, but she is afraid to do it. She tells him that she loves him and that she wants to be with him, but that she can t do it because of the risk of his bite. He says that he will not let her go back to him. She says that she will never stop loving him. He promises her that he won t and they make love again. Joshua tells Elizabeth that he wants to take Alison as his bride. Elizabeth tells Joshua that she is not going to let him take her. She wants to stop him from taking Alison. Rafe tells Alison that he has a vision of a city of tortured souls, a master vampire race, and his vampire bride. He will never let that happen. He can t let Joshua get to her. He is going to keep her away from Joshua and all his goons. He wants her to be his bride and she will be his wife. She will never love him, Rafe says.
QA pairs	• Who does Frank try to kidnap Allison from? -**Joshua** • Who does Frank try to kidnap? -**Allison** • Who tries to kidnap Allison? -**Joshua** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**vampire bride**

Table 7.13: Gold summary with automatically generated QA pairs (top) and model summaries with different content selection methods. Questions which the automatic summary answers correctly are highlighted in green (wrong answers shown in red). All model variants consider the textual modality (i.e., BART with adapter tuning).

Model	Summary
Gold	Joshua tells Elizabeth he wants to turn Allison and demands she help ease Allison into her new life as his wife. Elizabeth tells Joshua she will kill him before she allows him to hurt Allison. Livvie is able to fend off her need to feed while she and Caleb make love. Frank searches for Allison. When Frank attempts to kidnap Allison from Rafe, he discovers that it really is Lucy and I an in disguise. Allison and Rafe reappear in Caleb s cave.
QA pairs	• Who does Frank try to kidnap Allison from? -**Rafe** • Who does Frank try to kidnap? -**Allison** • Who tries to kidnap Allison? -**Frank** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**wife**
Text-only (CS)	Rafe tells Alison that he will never let Joshua take her for his bride, but she tells him that she has no choice in the matter. Elizabeth tells Joshua that she will not stand by and allow him to take her daughter. Joshua tells Elizabeth that he is going to eas e Alison into her new lifestyle as his wife. Elizabeth says that she is not going to let her daughter suffer the kind of nightmare that she lived. She will kill Joshua before he is even that close to turning her. Alison tells Rafe that she thinks this is a little extreme, that is all. Rafe says he will not let Joshua get to her. He promises to keep her away from Joshua and all his goons. Caleb tells Livvie that she doesn t need to feed. He tells her that he can t make love to her the way she wants to. She tells him she can t turn him back, but he tells her he can. He says that he loves her and that he wants to make her his bride.
QA pairs	• Who does Frank try to kidnap Allison from? -**Joshua** • Who does Frank try to kidnap? -**Joshua** • Who tries to kidnap Allison? -**Rafe** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**wife**
Hierarchi-cal3D (CS)	Livvie tries to convince Caleb to let her take the risk of biting him, but she is afraid that she won t be able to do it. Joshua tells Elizabeth that he wants Alison to be his bride. Elizabeth is shocked when she finds out that Joshua wants to take Alison away from Rafe. Elizabeth tells Joshua that she will find a way to stop him from taking Alison. Rafe tells Alison that he has a vision of a city of tortured souls, a master vampire race, and his vampire bride. He tells her that he can make a perfect bride for her. Alison tells Rafe that she doesn t want to leave her family, but Rafe assures her that she is not going to leave them. Frank tells Ian that he is going to have to tell his boss that his mission didn t work.
QA pairs	• Who does Frank try to kidnap Allison from? -**Rafe** • Who does Frank try to kidnap? -**Allison** • Who tries to kidnap Allison? -**Rafe** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**vampire bride**
LED	Caleb and Livvie try to convince Caleb that they can be together again, but Caleb refuses to believe that he can t make love to Livvie. Rafe and Alison agree that they will be together, but Rafe tells Alison that he will never be able to make love with her again. Elizabeth tells Joshua that she wants him to turn Alison into a vampire bride. Joshua tells Elizabeth that he wants her to be his wife, but Elizabeth tells him that she is not going to do that. Joshua says that he is going to make Alison a vampire, and he will be the one true love of his life. Elizabeth says that she will never let Alison suffer the kind of nightmare that she lived, and she will make her life as a vampire. Joshua and Elizabeth argue about how much she wants to be a vampire and how much he wants to help her. Elizabeth asks Joshua if he s going to help Alison, but he says he will not.
QA pairs	• Who does Frank try to kidnap Allison from? -**Caleb** • Who does Frank try to kidnap? -**Caleb** • Who tries to kidnap Allison? -**Rafe** • Who can fend off her need to feed while she and Caleb make love? -**Livvie** • Who tells Joshua she will kill him before she allows him to hurt Allison? -**Elizabeth** • Who tells Elizabeth he wants to turn Allison into his wife? -**Joshua** • What is Allison s new life? -**vampire**

Table 7.14: Gold summary with automatically generated QA pairs (top) and model summaries. Questions which the automatic summary answers correctly are highlighted in green (wrong answers shown in red). We compare our approach (i.e., Hierarchical3D) with state-of-the-art textual summarizers (i.e., LED).

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we addressed the task of narrative summarization. We revise the hypotheses of the thesis as stated in Chapter 1 in Table 8.1. We first hypothesized that knowledge about the narrative structure of movies and TV episodes would facilitate the summarization task (HYPOTHESIS I). We introduced the turning point identification task and dataset (i.e., TRIPOD) in Chapter 3, where we defined narrative structure in movies via "turning points" inspired by screenwriting theory (Hague, 2017) and provided baseline approaches for assessing whether we can automatically identify them.

Next, we validated HYPOTHESIS I in Chapter 4 by using information about the narrative structure for augmenting supervised and unsupervised summarization algorithms. Given another dataset consisting of TV episodes with very well-defined structure and summary-specific labels, we validated that information about narrative structure can boost performance on summarization. Moreover, we verified that the definition of turning points is general enough and can adapt to different narrative types.

In Chapters 3 and 4 we focused on a text-only setting, where we used screenplays from movies and TV episodes for defining our task and validating our initial hypothesis. We then moved to a multimodal setting in Chapter 5, where we also took into account visual and audio information from full-length videos. Following from HYPOTHESIS I, we assumed we can directly assemble video summaries by identifying turning points, which are by definition key events in a narrative, in the absence of more fine-grained summary-specific labels. We further hypothesized that events in movies should be modeled via a graph structure instead of linear sequence of scenes contrary to previous chapters (HYPOTHESIS II; Table 8.1). Hence, in Chapter 5 we modeled

HYPOTHESIS I	Knowledge about the narrative structure of movies and TV shows can facilitate summarizing them. By identifying key events and segmenting the narrative into meaningful thematic units, we can then address summarization and provide key information in visual and textual summaries.
HYPOTHESIS II	On the surface, movies and TV episodes are a *sequence* of scenes or shots. However, they often contain non-linearities in the story, where events may be presented in a non chronological order, important scenes may be interrupted by redundant events called "fillers", and distinct sub-plots may intervene. Given this observation, we assume that modeling movies as *graphs* would better capture the complex relationships between events offering better contextualization and improved performance on summarization.
HYPOTHESIS III	Extending HYPOTHESIS II, we hypothesize that modeling movies as *sparse* graphs will lead to more interpretable approaches. By utilizing sparse graphs, we hypothesize that we can better navigate a movie, analyze the topology of the graphs depending on the narrative type, and develop interactive approaches to summarization.
HYPOTHESIS IV	Finally, we hypothesize that incorporating information from full-length *video* and *audio*, facilitates the inference of high-level events that are difficult to be captured solely based on dialogue (contained in screenplays or transcripts).

Table 8.1: Hypotheses made in the thesis and presented in Chapter 1.

movies as sparse graphs, where nodes are scenes from the screenplay and edges denote strong semantic relationships between them. We also utilized multimodal information from the full-length video and audio for learning these graphs in the latent space. The experimental results of Chapter 5 showed that modeling movies as graphs offers better contextualization and improves summarization performance, validating HYPOTHESIS II. Moreover, multimodal information contributes to creating more meaningful graphs, which present different topology depending on the movie genre.

Nevertheless, the approach in Chapter 5 has certain limitations, since it considers screenplays as the main source of information and operates over *scenes* which may be several minutes long resulting in lengthy output videos. We addressed these limitations in Chapter 6 by focusing on the task of trailer moment identification, operating on shots from the full-lenth video. We proposed an interpretable unsupervised algorithm that

operates over a sparse movie graph and selects a sequence of *shots* from the video to be used in a movie trailer. We furthermore suggested decomposing the task of trailer moment identification, which is complex and subjective, into two simpler, well-defined subtasks: narrative structure identification (see Chapter 3), and sentiment prediction. Detailed evaluation experiments revealed that our approach provides informative *and* attractive trailers. We also extended our algorithm to a semi-automatic approach with a human in the loop and showed that modeling movies as sparse graphs and interactively traversing them can lead to trailers of good quality and comparable to fully manual trailer shot selection, which further highlights the advantage of operating over sparse graphs (HYPOTHESIS III; Table 8.1).

Finally, given salient content from a TV episode, we explored ways to produce textual summaries. Chapter 7 introduced a video-to-text setting for narrative summarization. We first created a new dataset (SummScreen^{3D}) for multimodal abstractive summarization of TV episodes, which to the best of our knowledge is the largest available for long video multimodal summarization. Next, we proposed ways to efficiently augment a pre-trained textual summarizer, which has strong generation capabilities and the correct inductive bias for the task, with multimodal information from the corresponding full-length videos. We experimentally demonstrated that incorporating multimodal information can lead to textual summaries of higher quality and factuality, which validates the importance of accessing all modalities for narrative summarization (HYPOTHESIS IV; Table 8.1). We showed that multimodal information can especially contribute to forming correct entity-event associations in narratives.

Overall, we attempted to collectively address the various challenges presented by automatically analyzing narratives, as explained in Chapter 1. In contrast to prior work, we considered full-length movies and TV episodes *and* all input modalities for producing video and textual summaries. Throughout this thesis, we demonstrated that:

1. **Structure is important for modeling narratives**, such as movies and TV episodes. Assuming a linear order of events in movies and TV episodes is an oversimplification which leads to worse contextualization. In contrast, we demonstrate that learning connections between events via a graph structure offers better contextualization, improves downstream performance, and is interpretable.

2. **Multimodal information is crucial for addressing narrative summarization**. Most previous work has considered movie/TV episode summarization as an instatiation of textual summarization (of screenplays or transcripts). However, tex-

tual information is incomplete; audiovisual cues from corresponding full-length videos are necessary for producing factual summaries.

3. **The work in this thesis can facilitate real-world applications**, such as video-to-video and video-to-text narrative summarization and semi-automatic trailer creation. Indeed, summarizing content automatically or semi-automatically from movies and TV episodes for different purposes (i.e., trailers, recaps, video and textual summaries) is increasingly becoming necessary for platforms such as Netflix, with hundreds of thousands of movies and TV shows.

4. **Models for narrative multimodal tasks need to be developed in low resource settings**. Gathering *parallel* data with *annotations* for hundreds of thousands of movies or TV episodes is infeasible. In this thesis, we propose ways to overcome data scarcity. For example, we annotate key events at the synopsis level, which is faster, more reliable and scalable, and then propose ways to automatically project such annotations into full-length screenplays. Moreover, we show that we can train a network on more unlabeled textual data (i.e., screenplays) via self-supervised objectives and then transfer this knowledge to a multimodal model. Finally, we demonstrate how we can convert a pre-trained textual summarizer into a multimodal one in a parameter efficient way by tuning only 3.8% of model parameters on a small multimodal summarization dataset.

We hope that the approaches and datasets introduced in this thesis will facilitate future research on this topic.

8.2 Future Work

In this section we look into possible extensions of the work presented in this thesis.

Fine-grained structure We hypothesized that identifying the structure of movies and TV episodes can facilitate narrative summarization. We validated this hypothesis in two ways. First, we considered a more high-level structure, where the objective was to identify key events (i.e., turning points) and segment the narrative into broader thematic units (i.e., six units per narrative). We also modelled movies as graphs in order to learn interactions between events. We represented events in terms of scenes from a screenplay and shots from the video. In both cases, we showed that learning interac-

tions between such events can offer better contextualization and improved performance on summarization.

In the future, we would like to investigate how we can encode and use more fine-grained structure for representing movies and TV episodes. As mentioned in Chapter 2, there is prior work on analyzing character-centered structures, which illustrate interactions between characters (Bamman et al., 2013, 2014; Iyyer et al., 2016; Chaturvedi et al., 2017; Gorinski and Lapata, 2015) and fine-grained events, actions, and emotions around these characters (Black and Wilensky, 1979; Chambers and Jurafsky, 2009; Elson and McKeown, 2009). There is also prior work that learns such graphs in short scenes from movies (Vicol et al., 2018). An interesting research direction is how to encode such fine-grained interactions dynamically across a full-length movie (e.g., how relationships between characters change over time and how characters' attributes and actions change). In this case, we should explore a broadly hierarchical organization of movie content, where high-level interactions are themselves graphs which model more fine-grained structure (e.g., relations between characters and objects).

Finally, regarding video-to-text summarization (Chapter 7) it would be interesting to investigate how we can incorporate more fine-grained information from the video and audio into pre-trained textual summarizers. We only considered coarse-grained utterance-level multimodal representations for augmenting textual BART (Lewis et al., 2020) and showed that this information helps producing higher quality and more factual textual summaries. However, we still lose frame-level information regarding characters' expressions, objects and landscapes. In order to incorporate such low-level information while considering the input length limit imposed by current neural architectures, we need more *structured ways* for combining all different information sources. For example, we could learn and adopt a discrete structure for the episode (e.g., via a learned sparse graph as shown in this thesis) and then contextualize each utterance or shot only with respect to semantically closest events. In this way, we could substitute the full self-attention of the transformer encoder that significantly increases computational complexity with an efficient attention mechanism that incorporates an inductive bias for the structure of the narrative.

Incremental summarization As mentioned in Chapter 1, summarizing narratives, such as movies and TV episodes, can have different functionalities and applications depending on the use case. For example, there are previews and trailers, where the goal is to introduce a viewer to the story without revealing too much information, and

there are recaps and complete summaries, where the goal is to present all important events of the story in order to remind viewers what they have already watched. Apart from these two main categories, there are more use cases for creating summaries of movies and TV episodes. One example is to incrementally produce summaries up to a given point (i.e., incremental summarization). For example, let us assume that someone stopped watching a movie half-way through. After a few months, they may want to continue watching the movie, but they do not want to re-watch from the beginning and do not remember all important parts up to the given point. Producing a video or textual summary containing all important events up to this point would be useful in this case.

We hypothesize that for producing such incremental summaries knowledge about narrative structure would be useful. Apart from identifying key events via turning points as we did in this thesis, we assume that the segmentation of the narrative into broader thematic units that are defined by turning points would be helpful for such an application. Going a step further, the segmentation of narratives can also offer suggestions to users for pausing and resuming watching narrative content at appropriate moments. Although turning points can offer a useful signal for such use cases, developing automatic methods for these applications presents challenges related to collecting training data for training and evaluating incremental summaries.

Controllable narrative generation In this thesis we utilized the narrative structure of movies, for example via turning points, in order to understand and summarize them. However, we would like to also explore ways for exploiting the turning points and their definition for generating new stories and plots. Turning points are used in screenwriting by writers for creating their plays by first defining the high-level structure and then filling in the details (Hague, 2017). A similar approach could be followed for automatic or semi-automatic narrative generation.

For narrative generation, the main characters and their attributes (i.e., setting of the story) should be established before moving on to define the key events and the progression of the plot. As mentioned in Chapter 2, there are theories suggesting that characters in stories tend to follow specific archetypes which connect to their actions (Fludernik, 2002; Jung, 2014). The use of characters in automatic narrative generation has been previously studied in literature (Fan et al., 2018; Goldfarb-Tarrant et al., 2020; Papalampidi et al., 2022b).

After defining characters and their attributes, the next step is to develop models for generating new turning points and new continuations for a given story. Generating sto-

ries based on events has been addressed in prior work. However, previous approaches mostly use keywords and key phrases as prompts for generating full stories (Xu et al., 2020; Rashkin et al., 2020). There are also two-step approaches, which first produce a plan and then generate the full story via a sequence-to-sequence model (Fan et al., 2019; Goldfarb-Tarrant et al., 2020; Wang et al., 2021). These approaches generate detailed plans extracted from the original stories via either semantic role labeling or keyword extraction (e.g., RAKE; Rose et al. 2010). Goldfarb-Tarrant et al. (2020) also score the intermediate plans based on coherence between events and anonymized character mentions in order to improve fluency. Problematically, their plans are sentence-level and very detailed, which makes it difficult to expand to long narratives, such as movies and TV episodes.

We hypothesize that we can create high-level sketches and plans for movies and TV episodes via turning point *generation* either fully automatically or with a human in the loop who participates in the generation process (e.g., Akoury et al. 2020; Sun et al. 2021). After generating such plans, we can fill in the details at a later stage given the sketelon of a story (Tambwekar et al., 2018; Ippolito et al., 2019; Wang et al., 2020c; Ammanabrolu et al., 2021).

Appendix A

Annotation Instructions for TRIPOD

A.1 Annotation Scheme Explanation

- Read the whole plot once: First, you have to be able to answer to the following questions :
  - **Who is/are the main character(s)?** We assume that the plot of a movie is character-based. Therefore, the main characters should be identified.
  - **What is his/her overall goal in the movie?** The most important events in the plot of the movie are also correlated with the protagonist's main goal. So first, you should identify the goal of the main story of the film (e.g. he wants to win the girl, he wants to save someone beloved, he wants to stop the bad guys etc.).
  - **What are the main obstacle towards that goal?** Some of the most important events also depict either the main problems that the protagonist faces or his/her reaction to these problems. Therefore, before annotating the plot synopsis, you should have in mind the main obstacle that the protagonist encounters.
- In our annotation scheme, we make the assumption that there are specific **events** (Turning Points (TPs)) that are crucial in determining the storyline of the movie. Based on this assumption, we seek to identify the storyline of the movie by annotating these events. Each turning point can be mapped to a sentence of the respective plot synopsis. Hence, the turning points will be annotated by identifying and annotating the plot sentences.

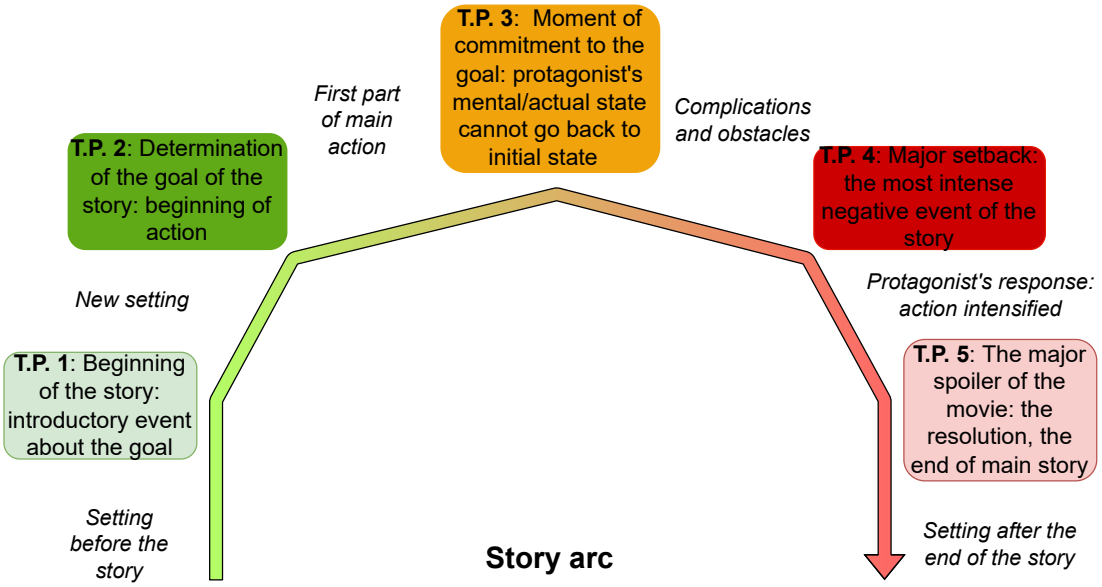


Figure A.1: Story arc for annotation

Now, we present the storyline of every movie while pointing out the definition of each turning point, as depicted in Figure A.1. For better clarification of the storyline and, most importantly, the turning points, we will also use two general examples of movies: a romantic comedy of a boy and a girl falling in love and an adventure movie, where the protagonist struggles to save his/her child from the "bad guy".

STORYLINE

First, you will see that an introduction will be made: Who your character is, what the setting is. Maybe, depending on the movie and genre, a problematic side of the main character's life is going to be presented here, in order to setup audience's emotions towards him/her, as well.

Romantic comedy: The boy and the girl are introduced, their individual lives are presented. For example, the boy is shy and introvert, while the girl is very popular.

Adventure movie: The protagonist and his/her everyday life is introduced, also the role of his/her child in his/her life may be pointed out (e.g. they have a strong relationship or they are estranged).

Turning point 1: The Opportunity (Annotation point): This is an *event* that

is going to alternate or disturb the previous setup. It is not necessarily something big or immediately noticeable when watching the film. However, this is the event that is considered as the **beginning of the story**, after presenting the setting.

Romantic comedy: The boy meets the girl for the first time, even if he does not pay any attention immediately.

Adventure movie: The protagonist starts questioning the safety of his/her child (e.g. he/she cannot reach the child on the phone and does not know exactly where the child is).

Now we can see the effects of the new opportunity, we can observe the alterations in the setting. Even if the protagonist has not yet recognized this opportunity, he somehow reacts to this, his/her life is changed even slightly. However, he/she is not yet actually affected from this opportunity.

Romantic comedy: The boy starts thinking about the girl without realizing it or gets irritated by her without any specific reason. The boy's mental state starts to change at this point.

Adventure movie: The protagonist's anxiety about his/her child is increased and he/she tries to track him/her down without however completely panicking at this point.

Turning point 2: The Change of Plans (Annotation point): This is the specific *event*, where the protagonist as well as the audience can determine the goal of the movie. The opportunity was just the starting point of the story, something more vague. At this point, one can better see the final goal of the movie. So, now the **main action of the movie begins**: from this point on one can see the progress of the story and the events that follow become more intense.

Romantic comedy: The boy admits to himself (and/or friends) that he likes the girl.

Adventure movie: The protagonist is informed that his/her child is indeed in danger, he/she is kidnapped.

After the determination of the goal, we can see the protagonist working towards that goal. At this point, the audience believes that the goal will be achieved. Although obstacles may be presented, the protagonist seems to find solutions. Now the main action of the movie has begun.

Romantic comedy: The boy tries to win the girl and although the girl does not seem interested, the boy's attempts seem to positively alter the situation.

Adventure movie: The protagonist starts forming a plan on how to save the child and searching for the bad guys.

Turning point 3: The Point of No Return (Annotation point): This *event* compels the protagonist to make a decision about his/her future actions: from now on, if the protagonist commits to the goal, returning to his/her initial state (i.e., actual or mental state of the protagonist as it was presented at the beginning of the film) will be difficult. At this point decisions have to be made and the protagonist's options narrow down. So, this is the event that will push the protagonist to **fully commit to his/her goal or return (temporally) on his/her initial state**.

Romantic comedy: The boy and the girl kiss for the first time and become more intimate, so they cannot go back to the relationship that they had in the start of the film.

Adventure movie: The bad guys are informed about the protagonist and his/her attempts to ruin their plans, so they start chasing him/her - the protagonist's life is also in danger now.

Since the protagonist is fully committed to the goal, we now observe the main complications of the story. More and more obstacles appear, while the protagonist responds to them. From now on, more intense scenes appear and emotion alterations, since there are conflicts. However, the audience still believes that the protagonist is able to overcome the obstacles and achieve the goal.

Romantic comedy: The girl avoids the boy due to his social status (not popular). The boy tries to change her mind and get her attention.

Adventure movie: The protagonist tries to find his/her child while avoiding the bad guys' threats and keeping a low profile. He/She keeps overcoming obstacles

and solving complications at this point.

Turning point 4: The Major Setback (Annotation point): This is **the most intensive *negative event*** in the story. Just when the audience believes that everything can be solved, the major obstacle of the movie is presented to the protagonist. At this point, everything may fall apart or the protagonist may encounter the main obstacle of the movie.

Romantic comedy: The boy is informed that the girl is now flirting with someone else and gets discouraged.

Adventure movie: The bad guys find the protagonist and capture him/her in order to kill him alongside with his/her child.

After the major setback, we observe the reaction of the protagonist. From this point on, the sequence of events are mainly the protagonist's actions who tries to reach his/her goal and overcome the major setback. More complications may appear, while the protagonist's efforts reach a peak.

Romantic comedy: The boy may try to forget the girl at first, but driven by his anger, he then confronts the "other guy" and the girl.

Adventure movie: The protagonist tries to escape and find the exact location of his/her child in order to save him/her as well.

Turning point 5: The Climax (Annotation point): This *event* is the highlight of the movie. Here we observe the final response of the protagonist to the obstacles and his/her crucial action that can lead to resolving the previously presented conflicts. If the movie does not have a happy ending, this is the point where everything really falls apart and the protagonist's fate is determined. This event is the end of the main story and most commonly **the "biggest spoiler" in the film**.

Romantic comedy: The boy talks to the girl about his feelings and they kiss again.

Adventure movie: The protagonist escapes and saves the child while injuring the bad guys. (If the film does not have a happy ending: Both the protagonist and

the child are murdered by the bad guys).

After the resolution of the protagonist's objective, the new life/situation is presented. This is the situation, where the protagonist is, after the end of the main story of the movie.

Romantic comedy: The boy and the girl live happily ever after.

Adventure movie: The protagonist and the child are safe again and their relationship has become stronger after this adventure. (Alternatively, for a bad ending, the plans of the bad guys for their next victims are presented.)

A.2 Examples

In this section we provide two examples of how we can apply the aforementioned annotation procedure to the plot synopses of movies.

A.2.1 "Drive"

Genre: Crime, Drama
Release date: 2011
Metadata from: Wikipedia[1]

The unnamed Driver (Ryan Gosling), who lives in an Echo Park, Los Angeles apartment, works as a mechanic and a part-time movie stuntman. Managed in both jobs by auto shop owner Shannon (Bryan Cranston), the duo also provide a getaway driver service. With Shannon organizing jobs, the Driver gives criminals a strict five-minute window to commit crimes and reach his vehicle (lest they be left behind).: Now we see the setup of the story. The main character and some basic information about his life.

Turning point 1 (The Opportunity): *Meeting his new neighbor, Irene (Carey Mulligan), the Driver soon becomes close to her and befriends her young son, Benicio*

[1] https://en.wikipedia.org/wiki/Drive_(2011_film)

(Kaden Leos).: The event of meeting his neighbor is going to change his life even though neither he or the audience know it yet. This is the starting point of our story.

This is undone, however, when Irene's husband, Standard Gabriel (Oscar Isaac), is released from prison. Standard, while initially hostile toward the Driver, soon warms to him. Meanwhile, Shannon persuades Jewish mobsters Bernie Rose (Albert Brooks) and Nino (Ron Perlman) to purchase a stock car chassis and to build it for the Driver to race. Standard, owing protection money from his time in prison, is beat up by Albanian gangster Cook (James Biberi). Threatening both Standard and his family, Cook demands he rob a pawnshop for $40,000 to pay off the debt.: The life of the neighbor is presented here, as well as the progression of the life of the main character and how it is affected by his new friend.

Turning point 2 (The Change of Plans): *The Driver, concerned for the safety of Irene and Benicio, steals a Ford Mustang and offers to act as the getaway driver for the pawnshop job.*: Here it becomes clear that the goal of the protagonist is to keep the neighbor and her child safe, for whom he deeply cares.

While waiting for Standard and Blanche (Christina Hendricks) to complete the heist, the Driver sees a Chrysler pull into the lot. As Blanche returns with a large bag, Standard is shot and killed by the pawnshop owner.: The main action starts here. The plan of keeping the neighbor and her son safe seems to working, while her husband is getting killed.

Turning point 3 (The Point of No Return): *The Driver flees with Blanche and the money, but they are pursued by the Chrysler, which tries to force them off the road; eluding the other vehicle, the Driver hides with Blanche in a motel.*: Now the character is deeply involved in the story. They start to chase him, so he has no longer the option of returning back to his normal life, as he is now exposed.

Learning the money actually totals a million dollars, the Driver interrogates Blanche, who admits she and Cook planned to double-cross him and Standard, and that the Chrysler belongs to Cook. Minutes later, two of Cook's men attack them in the motel room, killing Blanche and injuring the Driver before he manages to kill them both. The Driver confronts Cook in his strip club, breaking his fingers with a hammer and

threatening to kill him; Cook reveals that Nino was behind the heist. The Driver offers to return Nino's money, but Nino declines and instead sends a hitman (Jeff Wolfe) to the Driver's apartment building.: Now the main obstacles are presented. The action of the movie builds on and the protagonist encounters more and more problems.

Turning point 4 (The Major Setback): *Entering the apartment elevator with Irene, the Driver encounters the hitman in the elevator.*: This is the major setback of the movie. While he tries to keep Irene safe from the beginning of the movie, now he encounters a killer in a very contained space while he is with her. So, they are both in great danger now.

Spotting the hitman's pistol, the Driver kisses Irene before violently beating the hitman, killing him while Irene watches in horror. In his pizzeria, Nino explains to Bernie and Cook that the heist money belonged to a crime family and, since anyone tied to the robbery could lead the Mafia to them, they need to kill everyone involved. Nino further explains that it was his plan all along to steal the money from the crime family, and it was his idea to set up the $40,000 dummy robbery. Bernie then proceeds to murder Cook, before killing Shannon when he refuses to divulge the whereabouts of the Driver. The Driver, disguising himself with a mask, follows Nino to the Pacific Coast Highway and T-bones Nino's car onto a beach. With Nino injured and weakened, the Driver drowns him in the Pacific Ocean. The Driver, using Nino's phone, arranges to meet Bernie at a Chinese restaurant. The Driver makes a final phone call to Irene to tell her he is leaving, and says that meeting her and Benicio was the best thing that happened to him. At the restaurant, Bernie promises Irene's and Benicio's safety in exchange for the money, but he warns that he cannot guarantee the safety of the Driver himself. Outside the restaurant, the Driver gives Bernie the money, only for Bernie to stab him in the stomach.: In this point, we observe the efforts of the protagonist to overcome the problems and actually achieve his goal. This is the part where the main action of the protagonist is presented.

Turning point 5 (The Climax): *The Driver retaliates by fatally stabbing Bernie in the neck; he then departs in his car, leaving the money with Bernie's corpse.*: The protagonist reaches his goal in this point, he kills the man who was threatening Irene's safety.

That evening, Irene knocks on the Driver's apartment door to no response; the Driver is then shown driving away into the night.: This is the closing part, it depicts what happened after the resolution.

A.2.2 "It's Complicated"

Genre: Comedy, Drama, Romance
Release date: 2009
Metadata from: Wikipedia[2]

Jane (Meryl Streep), who owns a successful bakery in Santa Barbara, California, and Jake Adler (Alec Baldwin), a successful attorney, divorced ten years ago. They had three children together, two girls and a boy, who are grown. Jake, who was cheating on Jane, married the much younger Agness (Lake Bell).: Here, the main characters alongside with their lives are presented.

Turning point 1 (The Opportunity): *Jane and Jake attend their son Luke's college graduation from St. John's University in New York City.*: Here is the opportunity for the divorced couple to be in the same room, in a family gathering.

After a dinner together, the two begin an affair, which continues in Santa Barbara. Jane is torn about the affair; Jake is not. While Agness has Jake scheduled for regular sessions at a fertility clinic, Jake is secretly taking medication, a side effect of which reduces his sperm count. After one of his sessions he has a lunchtime rendezvous with Jane at a hotel. Jake collapses in the hotel room and a doctor is called. The doctor speculates that the reason for Jake's distress may be the medication and says he should stop taking it. Jake and Jane's children know nothing of the affair, but Harley (John Krasinski), who is engaged to their daughter Lauren, spots the pair and the doctor in the hotel but keeps silent. Adam (Steve Martin) is an architect hired to remodel Jane's home. Still healing from a divorce of his own, he begins to fall in love with Jane. On the night of Luke's graduation party in Santa Barbara, Jane invites Adam to the party. She is stoned when he picks her up because she has smoked a marijuana joint that Jake had given her earlier. Later at the party, Adam also smokes a joint with Jane.: The result of the opportunity is the formation of the new situation, which is the development of

[2] https://en.wikipedia.org/wiki/It%27s_Complicated_(film)

an affair for the divorced couple. In this stage, events derived from the new situation alongside with details about the other characters that are involved and maybe affected by the affair, are presented.

Turning point 2 (The Change of Plans): *Jake becomes jealous observing them, but with some cajoling by Jane, he gets stoned with them as well.*: The affair that was something casual for both of them, seems in this point to actually affect their mental state.

Agness then observes Jake and Jane dancing together and becomes suspicious of their closeness. When they leave the party, Adam asks Jane if they could have something to eat. Jane takes him to her bakery and makes him chocolate croissants. This takes hours, and they enjoy their time together.: In this stage, we observe that the affair indeed starts to affect the other characters as well (Agness). At the same time the progress of the plot is continued.

Turning point 3 (The Point of No Return): *Jake and Agness separate, although it is not clear who leaves whom.*: Now there is an actual consequence of the affair. Jake and Agness separate and things can no longer go back to the initial situation (stopping the affair and proceed with their lives as they were).

Eventually by a webcam in Jane's bedroom, Adam sees Jake naked and realizes that the two have been having an affair. Adam tells Jane he cannot continue seeing her because it will only lead to heartbreak.: Now we observe more complications and problems in the life of the protagonists. Adam also separates from Jane because of Jake.

Turning point 4 (The Major Setback): *Jane's kids also find out, and they are not happy about Mom and Dad getting together again because they are still recovering from the divorce.*: Although the audience thinks that with all these complications in the protagonists' lives, they are going to end up together, here is the big obstacle: their children's objections.

Jane tells them she is not getting back with Jake.: This stage does not contain a lot of action. Practically, it's just Jane who gives in to the obstacle (her children's

objections).

Turning point 5 (The Climax): *Jane and Jake talk and end their affair on amicable terms.*: Now we have the bigger spoiler of the movie: they are not going to end up together. So, this is the event of the final break up.

The film ends with Adam at Jane's house ready to commence the remodeling. Before the credits roll, Jane and Adam are seen laughing while walking into her house.: Here is the aftermath of the movie, the protagonists are not together, but they maintain a good friendly relationship.

A.3 Annotation procedure

- The procedure to be followed for annotating a movie is described below:

  1. Read the whole plot once.
  2. Identify the main characters, goal of the movie and main obstacles presented as explained in Section 1.
  3. if you identify time shifts or multiple stories in the plot, then try to determine the main story alongside with its elements by paying attention to the following points:
     - Inspect the number of plot sentences that correspond to each story: Since we have a synopsis, **most of the references should be about the main story**, so if the story you consider as main is not mentioned in the vast majority of plot sentences then something is wrong: either this is a secondary storyline or a major substory of the main one.
     - **First spot the Climax** of the plot. The *Climax* is about the main story. So, try to identify the *Climax* first, in order to connect it with the main story. In this case, you will consider as *Climax* the point **after which the tension starts to decrease**, even though important events continue to occur.
     - After annotating the Climax and the corresponding main story, **BE CONSISTENT** with the rest of your annotations: **all annotated TPs should correspond to the same storyline**. For example, if there is

both action and romance in a movie, decide which is the main story and select TPs that are relevant to this story.

4. After selecting the Climax, continue by selecting the Opportunity and Major Setback, since these turning points are going to frame the story.
5. Finally jointly identify the remaining turning points (Change of plans and Point of no return): first spot both turning points and annotate them.

- **Determine the genre of the movie**: in order to correctly identify the TPs you should keep in mind the broad genre of the movie (i.e. either comedy, romance, social or adventure, action, thriller). Next, we present technical tips that could help the identification of the TPs in a plot synopsis.
  - **Opportunity**
    - Overall: since opportunity is the start point of the main story, most of the times **all main characters alongside with their background have been introduced** at this point. Also, **do not confuse the presentation of some "problematic" acts or behaviors of the main characters** as the Opportunity event, they might be just part of the introduction of the protagonists.
    - Romance, comedy: In this case, most commonly the Opportunity event is the moment where the main characters meet/chat/are connected. **The problems/obstacles of the story have not yet been introduced**.
    - Action, adventure: In comparison with the previous case, here the Opportunity event is when **the challenge is introduced**. In order to not confuse the description of the setting of the story with the introduction of the challenge, be careful to **choose an introductory sentence, after which the story begins with events relevant to the main storyline (i.e. to the Climax) and not more introductory events about the setting**.
  - **Change of plans**
    - Overall: here, we search for an event that **clearly progresses the plot**. In the plot synopsis, there is, sometimes, a contradiction either between the target sentence and the previous one or within the target sentence. **BE CAREFUL**: after the Opportunity (beginning of the

story), **there will be some other, additional events for determining the direction of the story and then the Change of plans event will occur**. **Do not rush in annotating the first event** that you come across after the Opportunity. Moreover, the "Change of plans" (or the desired contradiction in general) **should be associated with the goals that the protagonist has during a large part of the story, not just during the setting period**.

  * Action, adventure: In the Change of plans event the **main problem (i.e. main story) of the movie has already been introduced**.

- **Point of no return**

  * Overall: at this point **the action has already begun, important events have already started to occur**. Here, there is a significant increase in tension, a larger commitment to the story and the starting point of more important events and significant complications. Empirically, **the Point of no return can be found somewhere in the middle of the story, several plot sentences apart from both Change of plans and Major Setback, since these three turning points frame the majority of the important events of the main story**.
  * Romance, comedy: here, the character actually **commits to his/her goal/story or temporarily returns to his/her previous life**.
  * Action, adventure: This is the event that indicates **the increase of tension and action** in the movie and the **faster switching of events** thereafter.

- **Major setback**

  * Overall: Most of the times there are several setbacks and complications. However, the Major setback is somewhere in the middle of these complications, meaning that typically you do not annotate as **"Major setback" neither the first main complication of the story nor the exact previous sentence from the Climax (which is the ending/resolution)**. In particular, **the "Major setback" can be considered as the peak between the appearance of more and more complications and the beginning of the 100% effort of the main character to resolve them**. Also, bare in mind that the Major setback should be a **significant complication that reasonably can lead to**

the upcoming Climax and therefore should not be far away in the synopsis from the already annotated Climax. In fact, more events (and therefore more plot sentences) typically occur between Point of no return and Major setback than between Major setback and Climax.

* Romance, comedy: here, it is more straightforward to recognize the Major setback: it is the moment that everything falls apart and **you question if there will be a happy ending or not after all**.
* Action, adventure: here, the identification of the Major setback is more difficult due to the faster change of events and the multiple problems of the protagonists. However, you should **pick a single negative event that directly affects the protagonist and not the description of some situation or just the moment that the protagonist realizes that something bad is happening**.

– **Climax**

* Romance, comedy: this is the **final event of the main story** that **determines the "new situation"** that is presented afterwards (after the end of the main story).
* Action, adventure: in this case the main criterion for the selection the turning point is that **before this point there is constant or even increasing tension and action** in the movie, whereas right **after this event the tension begins to drop** even if this drop is not instant and some supplementary events follow.

A.4 Further guidelines

- When you are searching for the Turning Points (TPs), you are searching for specific events, not the description of extended situations that are depicted in the film.

- Our annotation scheme approaches the movie structure in a character-based way. This means that everything is linked with the protagonist(s) of the movie. Therefore, most of the times the TPs are actually events that happen to the main character, are the main character's actions or directly affect their mental, emotional or actual state (etc. affect their goals, the way they consider things, their options).

- In general, the TPs signal a significant change in the main character's (inner or actual) state or a main alternation in the progress of the main story.
- In all cases you should track the five TPs that are described above, even if some TPs are not so straightforward in the plot synopsis.
- You may need to annotate two consecutive plot sentences as two different TPs, since we are examining the summary of the film and not the whole screenplay. Therefore the details occurring between the TPs (as described above) may be missing from the plot synopsis.
- In some cases you may think that one TP corresponds to a part of a plot sentence. Even if this plot sentence includes other information as well, you should annotate the whole plot sentence as the respective TP.
- Even if you think that a TP corresponds to two plot sentences, you have to select a single plot sentence as the most representative for the respective TP.

A.5 How to annotate the plot files

You will use the "annotation_tool" (see corresponding README.md file) and annotate each plot synopsis by selecting the plot sentences that correspond to each turning point:

- *Opportunity*: This sentence represents the first turning point.
- *Change of plans*: This sentence represents the second turning point.
- *Point of no return*: This sentence represents the third turning point.
- *Major setback*: This sentence represents the fourth turning point.
- *Climax*: This sentence represents the fifth turning point.

A.6 Example of Annotation Interface

We present an example of the interface used for annotating TP sentences in plot synopses of movies given the above instructions in Figure A.2. Given all sentences of a plot synopsis for a movie (here we illustrate the movie "Arbitrage"), annotators have to read the whole synopsis and then decide which sentence represents each one of the turning points. For each turning point, they can select *exactly one* sentence.

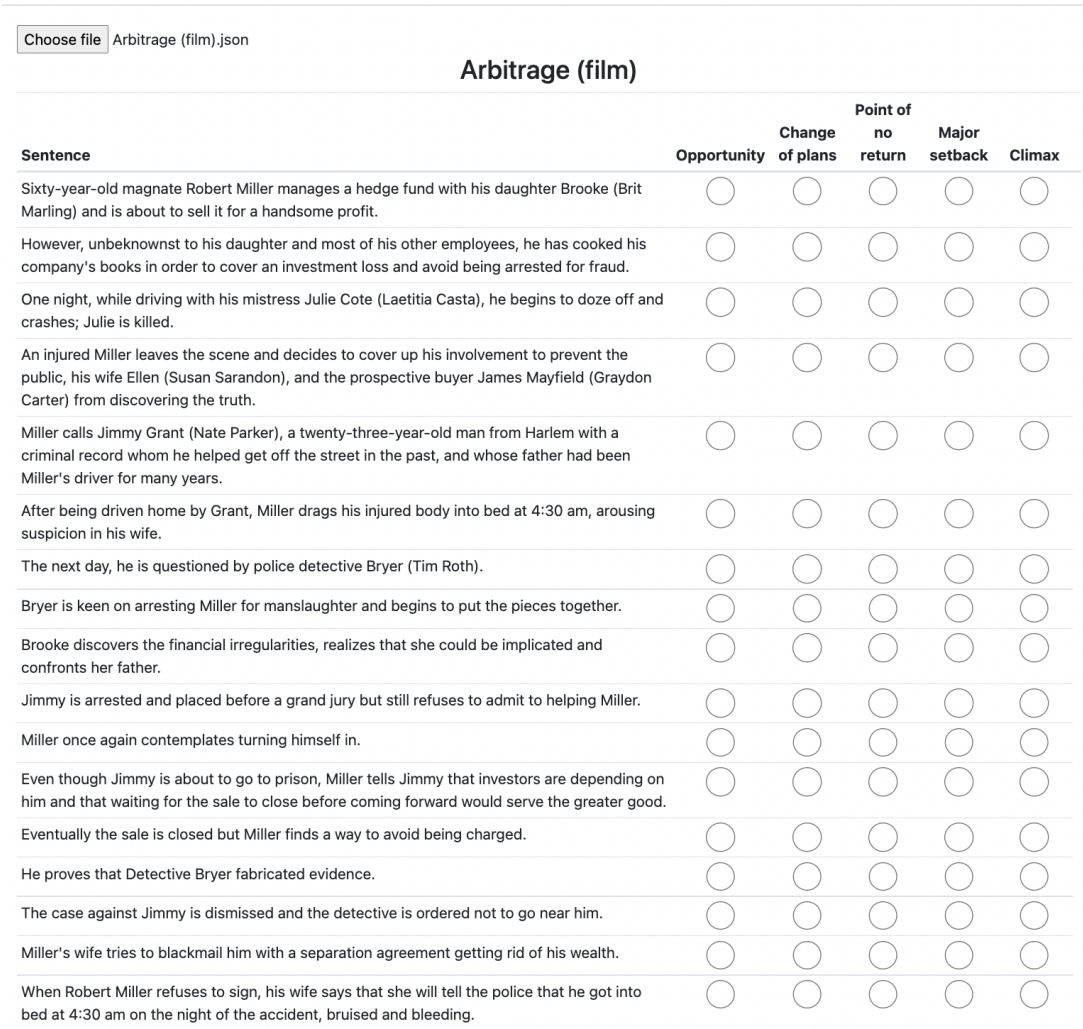


Figure A.2: Example of the annotation interface for the movie “Arbitrage”. Annotators read the plot synopsis of the movie sentence-by-sentence and decide which sentence represents each TP.

Appendix B

TRIPOD Dataset Details

We present the names of all movies included in the train and test sets of the TRIPOD dataset and the held-out set used for evaluating trailer creation (see Chapter 6) in Table B.1. Next, we also present the distribution of movies per set depending on genre (Figures B.1, B.2, and B.3), release year (Figures B.4, B.5, and B.6), and IMDb score (Figures B.7, B.8, and B.9).

Overall, we observe that we include movies characterized by diverse genres in all TRIPOD sets. The most common genre in all sets is "drama", followed by "romance", "comedy", "thriller", "crime", and "action". Moreover, most of the movies included in the TRIPOD sets are released between 1990 and 2020. Although the TRIPOD train and test sets include some movies from as far back as 1950 and 1960, the held-out set used in Chapter 6 for evaluating trailer creation only includes movies from the last three decades. This is because the Moviescope dataset[1] (Cascante-Bonilla et al., 2019) used for creating the held-out set comprises of more recent movies. Finally, given the IMDb scores for the movies, we again observe a large variation, where movies included in all TRIPOD sets receive scores from 5 to 9.

[1] http://www.cs.virginia.edu/ pc9za/research/moviescope.html

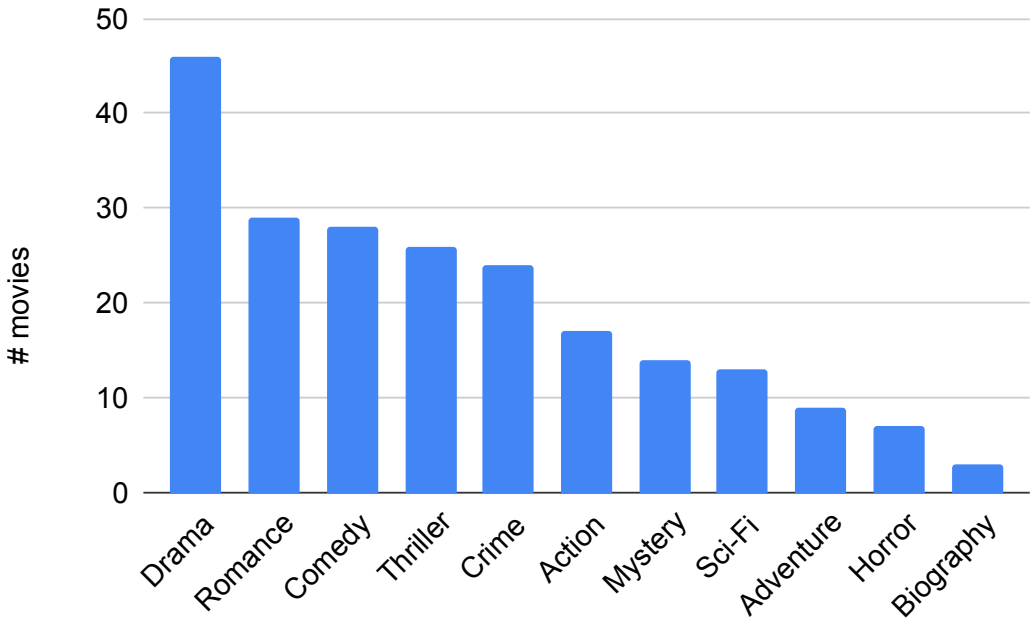


Figure B.1: Distribution of movies included in the TRIPOD training set over different genres.

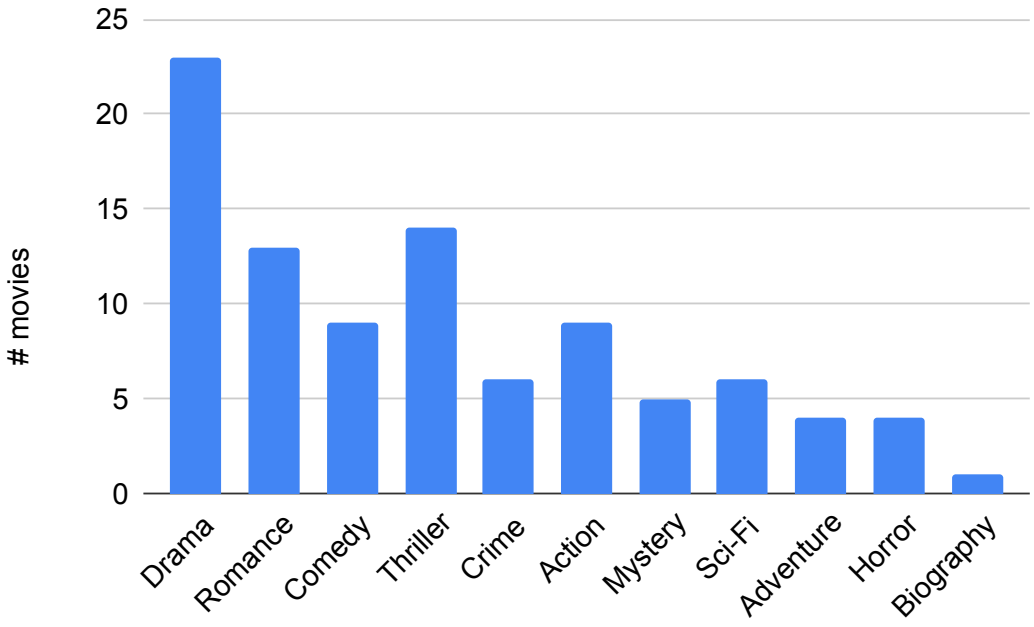


Figure B.2: Distribution of movies included in the TRIPOD test set over different genres.

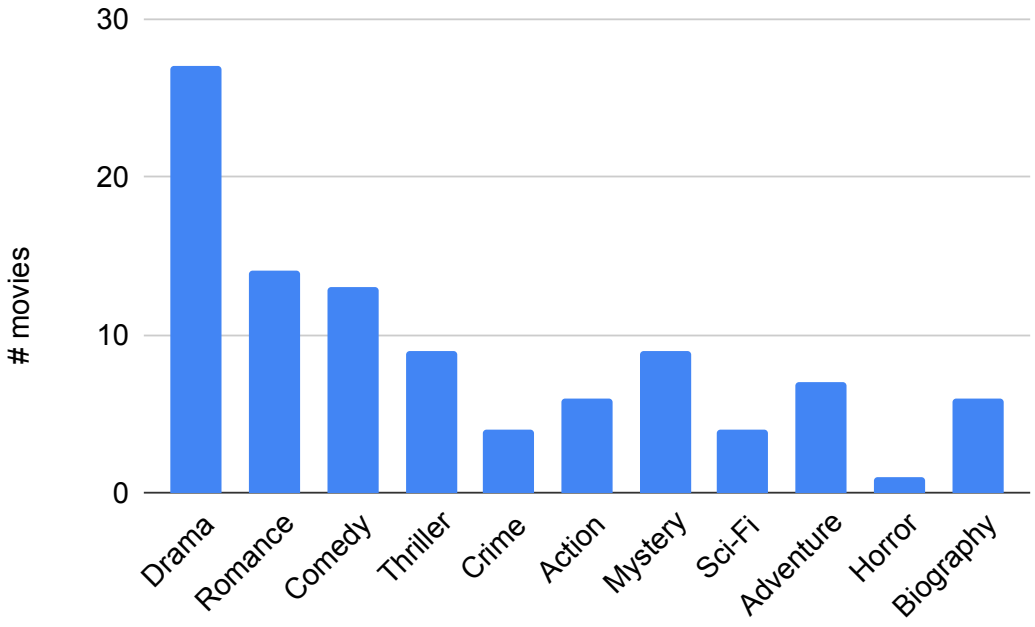


Figure B.3: Distribution of movies included in the TRIPOD held-out set (Chapter 6) over different genres.

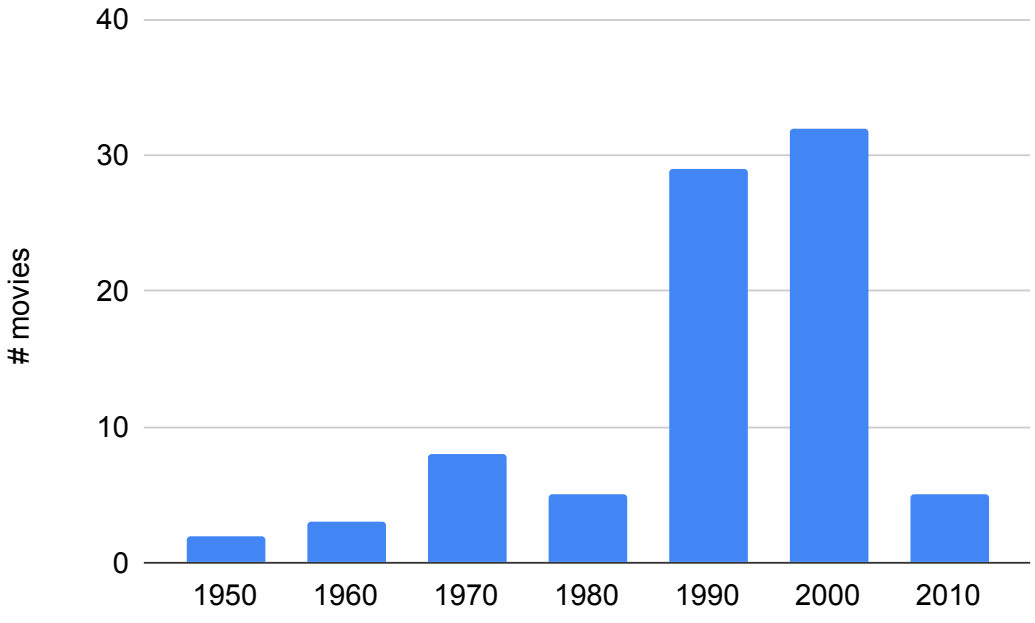


Figure B.4: Distribution of movies included in the TRIPOD training set depending on the release year.

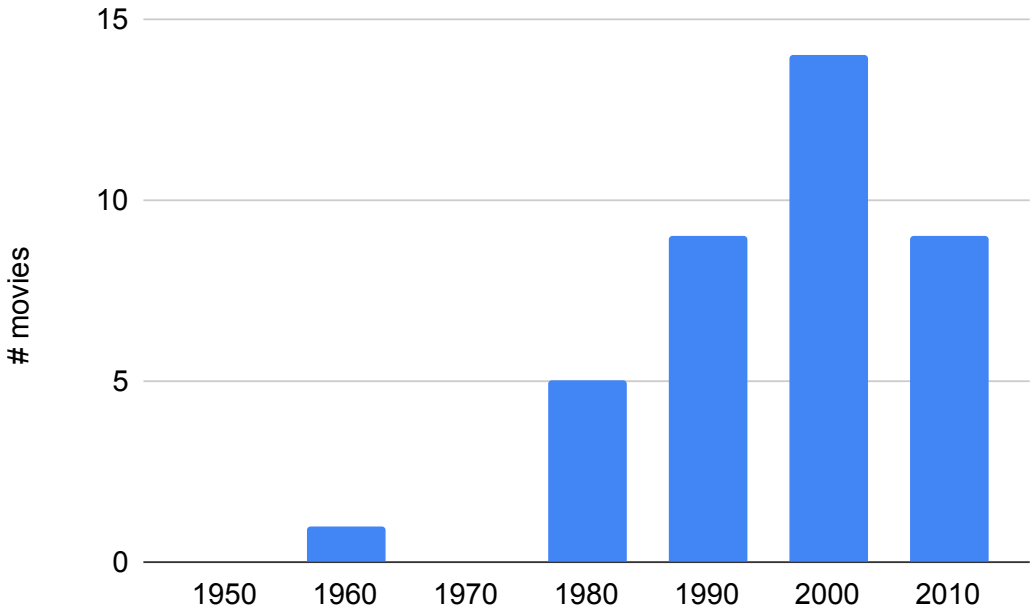


Figure B.5: Distribution of movies included in the TRIPOD test set depending on the release year.

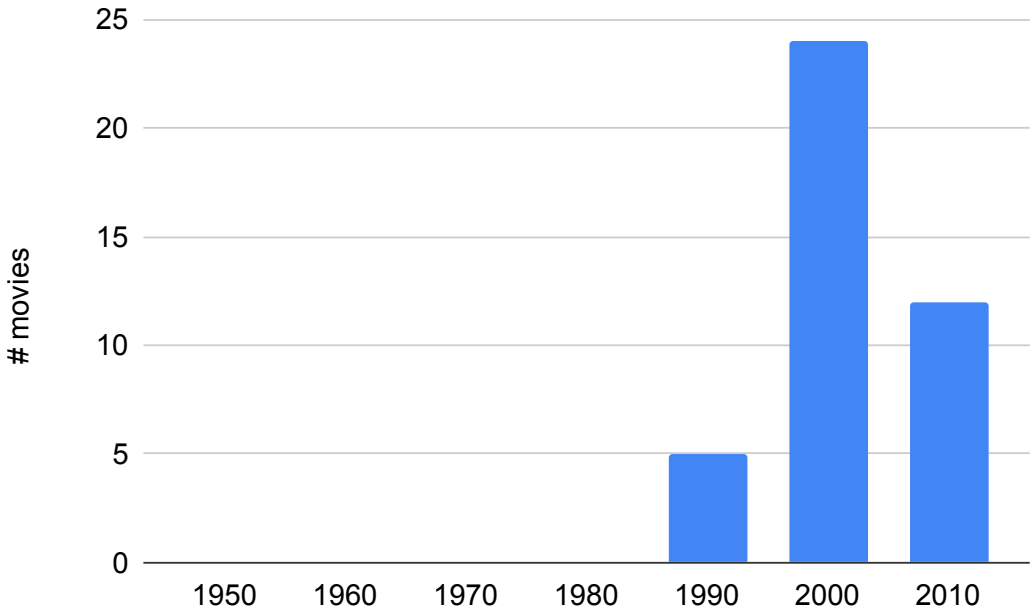


Figure B.6: Distribution of movies included in the TRIPOD held-out set (Chapter 6) depending on the release year.

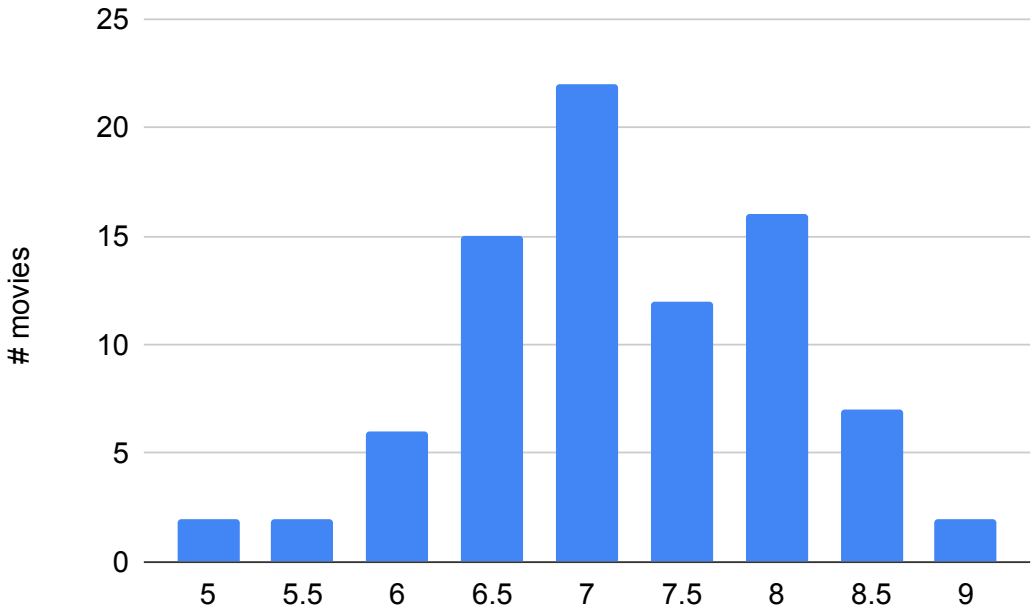


Figure B.7: Distribution of movies included in the TRIPOD training set depending on the IMDb score.

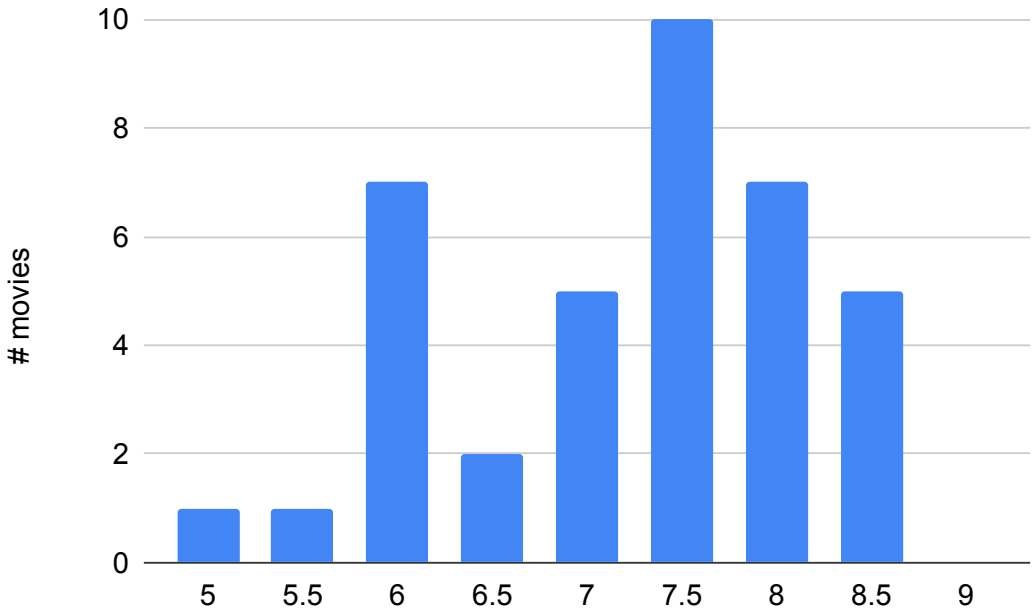


Figure B.8: Distribution of movies included in the TRIPOD test set depending on the IMDb score.

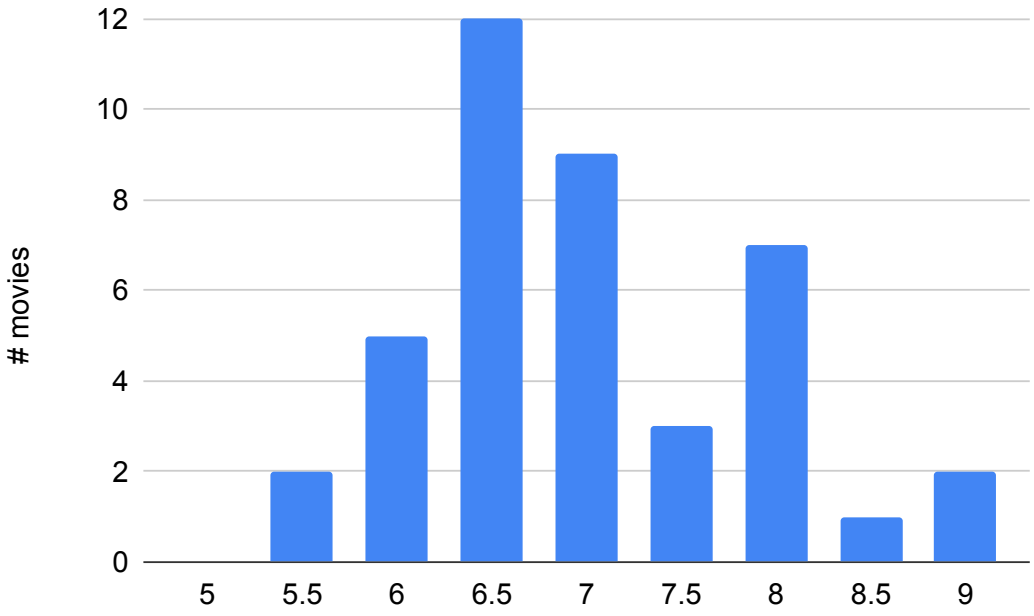


Figure B.9: Distribution of movies included in the TRIPOD held-out set (Chapter 6) depending on the IMDb score.

Set	Movies
Train	'10 Things I Hate About You', '15 Minutes', '17 Again', '25th Hour', '30 Minutes or Less', '48 Hrs.', '8mm', 'Alien', 'Alien Nation', 'Angel Eyes', 'At First Sight', 'Bamboozled', 'Basic Instinct', 'Beloved', 'Blade', 'Bonnie and Clyde', 'Bridesmaids (2011)', 'Drive (2011)', 'Erin Brockovich', 'Funny People', 'Gran Torino', 'I Am Legend', 'I Am Sam', 'Indiana Jones and the Kingdom of the Crystal Skull', 'Invictus', 'Jaws', 'Jurassic Park', 'Kalifornia', 'Meet Joe Black', 'Men in Black', 'Minority Report', 'My Best Friends Wedding', 'My Girl', 'Notting Hill', 'One Flew Over the Cuckoos Nest', 'Pirates of the Caribbean: The Curse of the Black Pearl', 'Pretty Woman', 'Pride & Prejudice', 'Reservoir Dogs', 'Ring', 'Romeo + Juliet', 'Saw', 'Seven', 'Shakespeare in Love', 'Sherlock Holmes', 'Sleepless in Seattle', 'Spanglish', 'Star Wars Episode I: The Phantom Menace', 'Sugar & Spice', Superman (1978), 'Terminator Salvation', 'The Apartment', 'The Dark Knight', 'The Exorcist', 'The Girl with the Dragon Tattoo (2011)', ' The Hangover', 'The Kids Are All Right', 'The Limey', 'The Mask', 'The Mummy (1999)', 'The Proposal', 'The Salton Sea', 'The Searchers', 'The Shipping News', ' The Silence of the Lambs', 'The Sixth Sense', 'The Sting', 'The Taking of Pelham One Two Three (1974)', 'The Time Machine (1960)', 'The Truman Show', ' The Ugly Truth', 'Titanic (1997)', 'Top Gun', 'Twilight (2008)', ' Twins (1988)', 'Vertigo', 'V for Vendetta', 'We Own the Night', 'What Women Want', ' When Harry Met Sally...', 'While She Was Out', 'X-Men', 'Young Frankenstein', 'You've Got Mail'
Dev+Test	'2012', '500 Days of Summer', 'American Beauty', 'American Gangster', 'Arbitrage', 'A Walk to Remember', 'Black Swan', 'Collateral Damage', 'Crazy, Stupid, Love', 'Die Hard', 'Easy A', 'From Russia with Love', 'Gothika', 'Heat (1995)', 'House of 1000 Corpses', 'Jane Eyre (2011)', 'Juno', 'Marley & Me', 'Moon', 'No Strings Attached', 'Oblivion', 'One Eight Seven', 'Panic Room', 'Sleepy Hollow', 'Slumdog Millionaire', 'Soldier', 'The Back-up Plan', 'The Breakfast Club', 'The Crying Game', 'The Last Temptation of Christ', 'The Majestic', 'The Shining', 'The Talented Mr. Ripley', 'The Thing', 'The Tourist (2010)', 'The Wedding Date', 'Total Recall (1990)', 'Unforgiven'
Held-out	'50 First Dates', 'A Beautiful Mind', 'Almost Famous', 'American Sniper', 'Anger Managment', 'As Good as It Gets', 'Australia', 'Bridget Jones Diary', 'Cinderella', 'Cinderella Man', 'Contagion', 'Don't Say A Word', 'Eat Pray Love', 'Flightplan', 'Forrest Gump', 'Gone Girl', 'Hannibal', 'Inception', 'Interstellar', 'Meet The Parents', 'Mona Lisa Smile', ' Mr and Mrs Smith', 'Night at the Museum: Battle of the Smithsonian', 'Now You See Me (2013)', ' Outbreak (1995)', 'Runaway Bride', 'Sex and the City (2008)', 'Signs (2002)', 'Silent Hills (2006)', 'The Curious Case of Benjamin Button', 'The Da Vinci Code', 'The Day After Tomorrow', 'The Great Gatsby (2013)', 'The Insider', 'The Internship', ' The Interpreter', 'The Pursuit of Happyness', 'Two Weeks Notice', 'Valentine's Day (2010)', 'Wall Street Money Never Sleeps', 'Watchmen'

Table B.1: Movie names included in TRIPOD and the held-out set used for evaluating trailer creation in Chapter 6.

Appendix C

Instructions for Human Evaluation

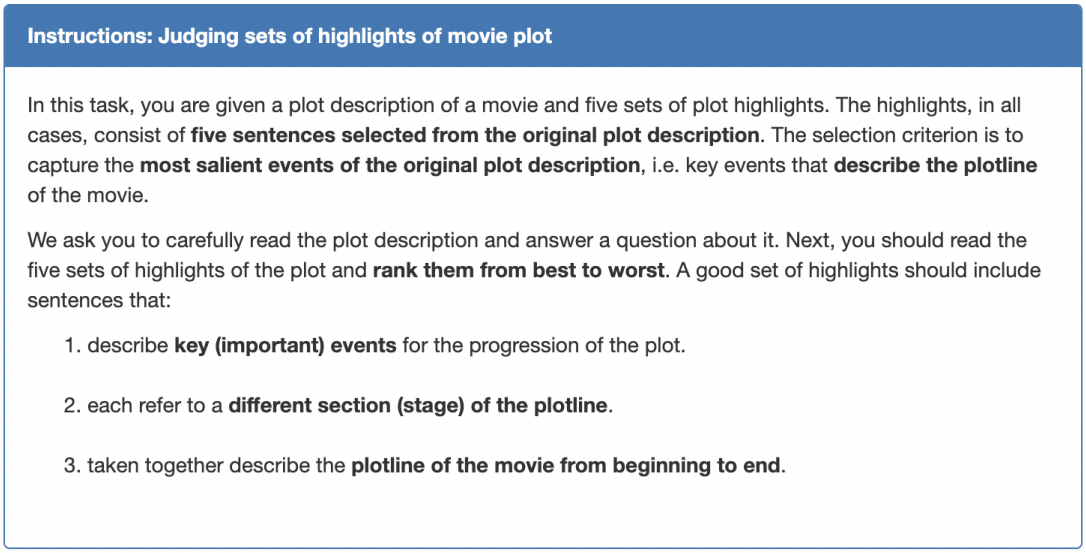


Figure C.1: Instructions

C.1 Human Evaluation on TP Identification in Plot Synopses

In Chapter 3, we conducted a human evaluation experiment on Amazon Mechanical Turk (AMT) in order to compare the performance of our model against gold standard annotations and the distribution baseline. Here, we present the experimental setup for the human evaluation. First, we ask AMT workers to read the instructions as given in Figure C.1. Next, they have to read the full plot synopsis for a movie (Figure C.2) and answer a question related to the plot (bottom part of Figure C.2). We use this question

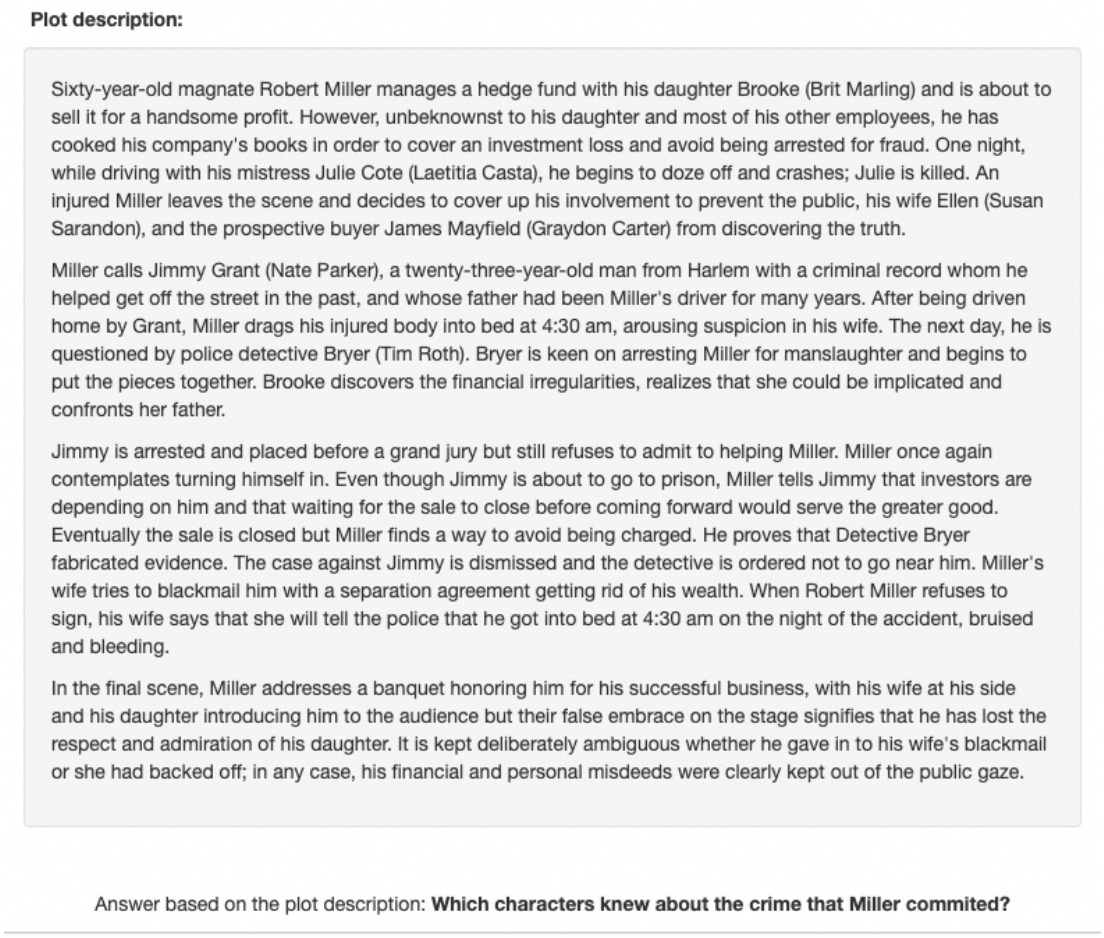


Figure C.2: Plot synopsis

in order to make sure that AMT workers read the synopsis in detail and performed the task correctly. If they do not correctly answer the question, we discard their answers and perform the experiment again. Finally, we ask AMT workers to read a set of highlights that are produced by different models (i.e., ours, gold standard, and distribution baseline) for the corresponding synopsis (Figures C.3 and C.4). After reading all sets of highlights, the AMT workers have to finally rank them from most to least informative (Figure C.5). We then use these rankings for determining the best model.

C.2 Human Evaluation on Screenplay Summarization

In Chapter 4, we conducted a human evaluation experiment on Amazon Mechanical Turk (AMT) for determining the most informative video summaries (ours (SUMMER), gold standard, and SUMMARUNNER*). We first provided AMT workers with in-

Highlights 1:

- Sixty-year-old magnate Robert Miller manages a hedge fund with his daughter Brooke (Brit Marling) and is about to sell it for a handsome profit.
- However, unbeknownst to his daughter and most of his other employees, he has cooked his company's books in order to cover an investment loss and avoid being arrested for fraud.
- One night, while driving with his mistress Julie Cote (Laetitia Casta), he begins to doze off and crashes; Julie is killed.
- An injured Miller leaves the scene and decides to cover up his involvement to prevent the public, his wife Ellen (Susan Sarandon), and the prospective buyer James Mayfield (Graydon Carter) from discovering the truth.
- Miller calls Jimmy Grant (Nate Parker), a twenty-three-year-old man from Harlem with a criminal record whom he helped get off the street in the past, and whose father had been Miller's driver for many years.

Highlights 2:

- One night, while driving with his mistress Julie Cote (Laetitia Casta), he begins to doze off and crashes; Julie is killed.
- Bryer is keen on arresting Miller for manslaughter and begins to put the pieces together.
- Jimmy is arrested and placed before a grand jury but still refuses to admit to helping Miller.
- Miller's wife tries to blackmail him with a separation agreement getting rid of his wealth.
- In the final scene, Miller addresses a banquet honoring him for his successful business, with his wife at his side and his daughter introducing him to the audience but their false embrace on the stage signifies that he has lost the respect and admiration of his daughter.

Figure C.3: Highlights identified by different models (continued in Figure C.4).

structions for the task (Figure C.6). Next, they watched the video summaries and answered six questions per summary related to important information present in the video (e.g., whether the victim or the perpetrator were revealed). We show these in Figure C.7. Finally, AMT workers ranked all summaries from most to least informative given the answers to the individual questions and their overall preference (Figure C.8).

Figure C.4: Highlights identified by different models.

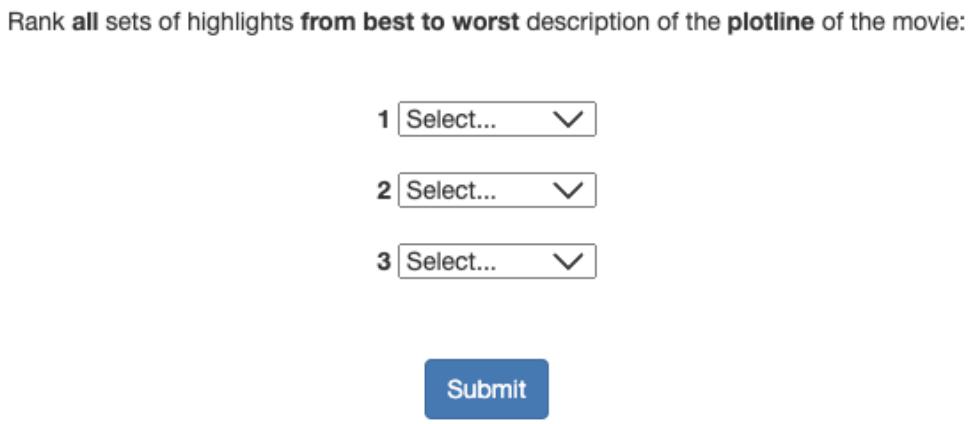


Figure C.5: Final ranking of outputs.

C.3 Human Evaluation on Movie Summarization

In Chapter 5, we conducted a human evaluation experiment on Amazon Mechanical Turk (AMT) in order to compare the video movie summaries produced by our method (GRAPHTP) against gold standard annotations, SCENESUM, and TAM. For this, we again provided AMT workers with instructions for the task as illustrated in Figure C.9. Next, they read a (condensed) textual summary for the movie with the most important events. We colored important sentences in the plot synopsis in order for AMT workers to pay extra attention. After reading the textual summary, AMT workers watched the video summaries for a movie and answered five questions per summary. This setup is

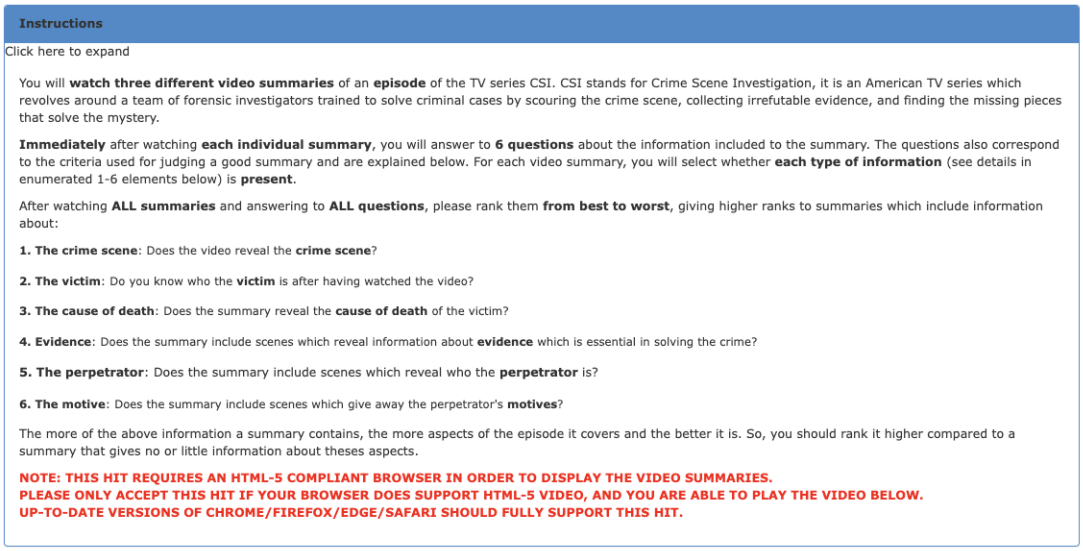


Figure C.6: Instructions

similar to the one described in Section C.2, however the questions now are different, each aiming to examine whether a specific TP is present in the summary. We present an example of our questions from the AMT interface in Figure C.11. Finally, after watching all summaries and answering all questions, AMT workers provided a rating from 1 to 5, where 1 is the least and 5 is the most informative summary (Figure C.12).

C.4 Human Evaluation on Trailer Generation

In Chapter 6, we conducted two human evaluation experiments on Amazon Mechanical Turk (AMT). First, we created and compared the quality of trailers generated by automatic methods in Section 6.4.3. Specifically, we created movie trailers for the following systems: random selection, GRAPHTRAILER (ours) with and without TP information, CCANet, and a supervised version of GRAPHTRAILER. We provided AMT workers with instructions for completing the human evaluation task, as illustrated in Figure C.13. Next, they watched all movie trailers and answered two questions per trailer related to the information provided (Q1) and the attractiveness of the trailers (Q2). We present an example of the questions asked for each trailer in Figure C.14. Finally, the AMT workers selected the best and worst trailers after watching them all and answering all questions (Figure C.16).

We also conducted a second human evaluation experiment in Section 6.5.3, where we wanted to compare the quality of semi-automatic trailers produced by the interac-

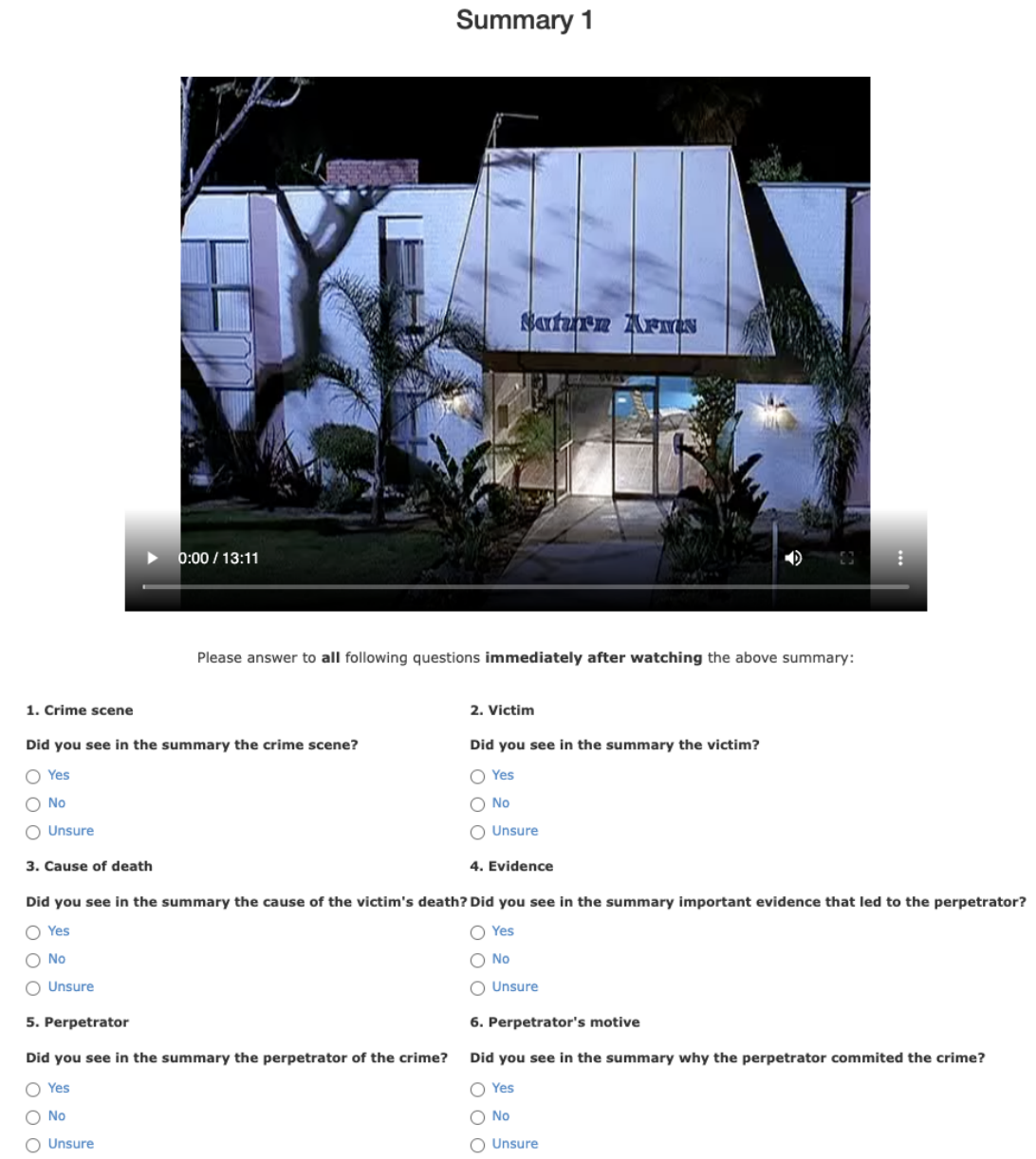


Figure C.7: Questions asked per summary.

tive tool that we developed based on our algorithm against the best performing fully automatic method (i.e., GRAPHTRAILER) and a fully manual selection of trailer shots (based on the trailer labels that we created). The setup of this experiment is very similar to the one presented above. However, we now include one extra question (Q3), relating to spoilers (Figure C.15) and ask them to indicate the best trailer, while performing pairwise comparisons of the semi-automatic trailers against the fully manual or fully automatic ones.

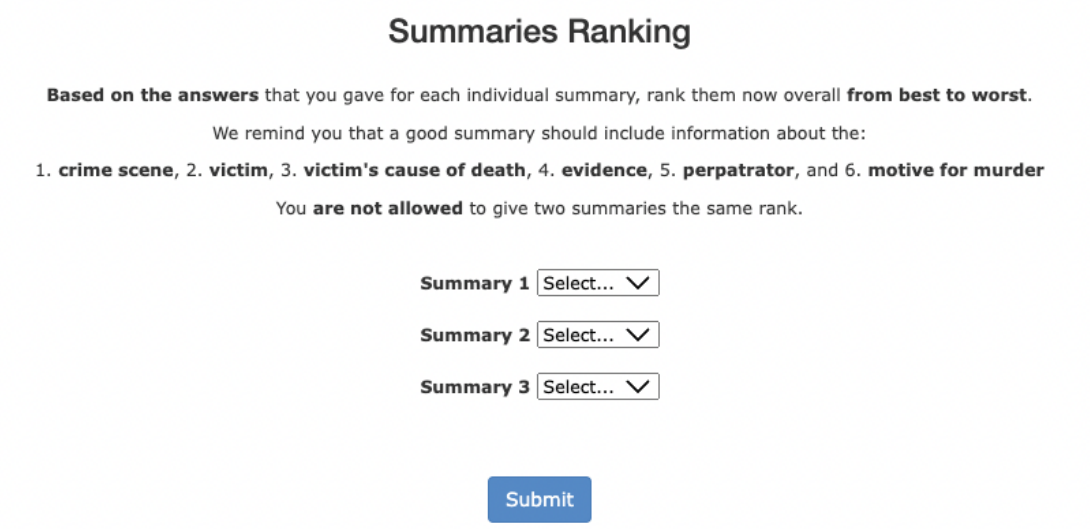


Figure C.8: Final ranking of video summaries.

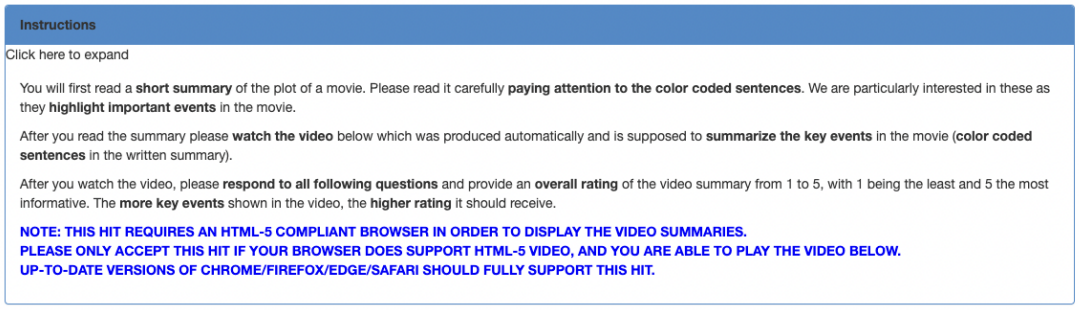


Figure C.9: Instructions

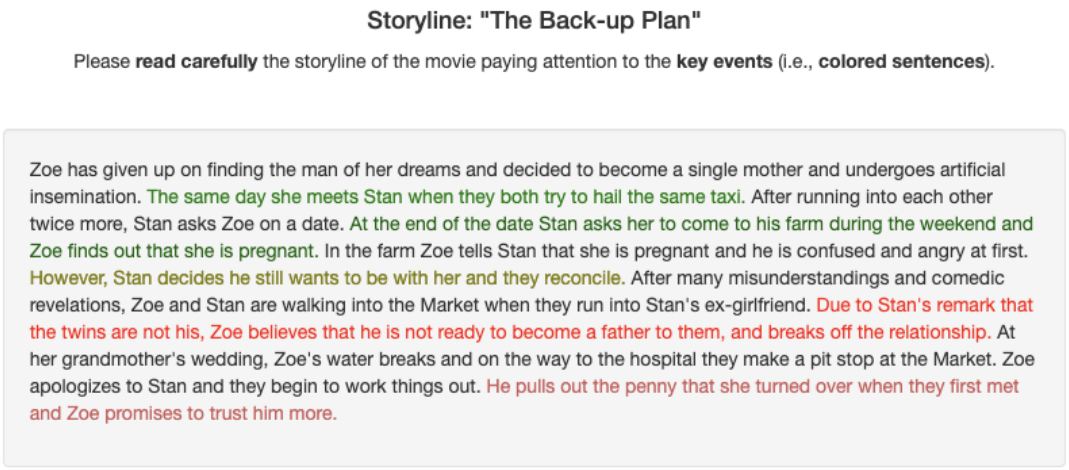


Figure C.10: Plot synopsis

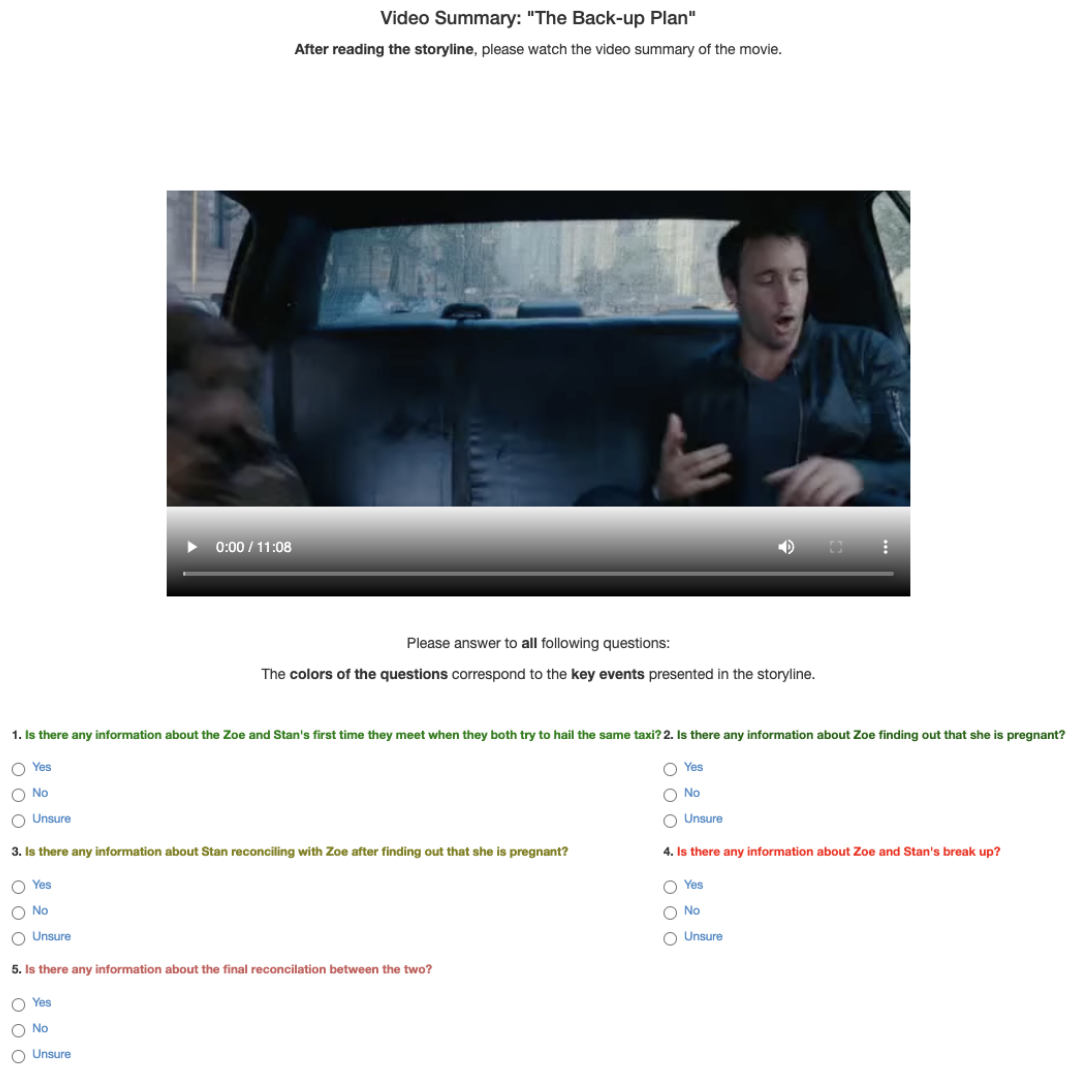


Figure C.11: Questions per video summary.

Figure C.12: Final ranking of outputs.

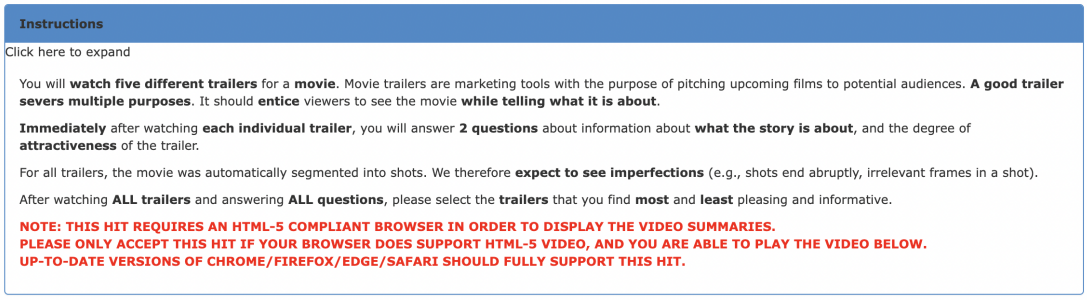


Figure C.13: Instructions

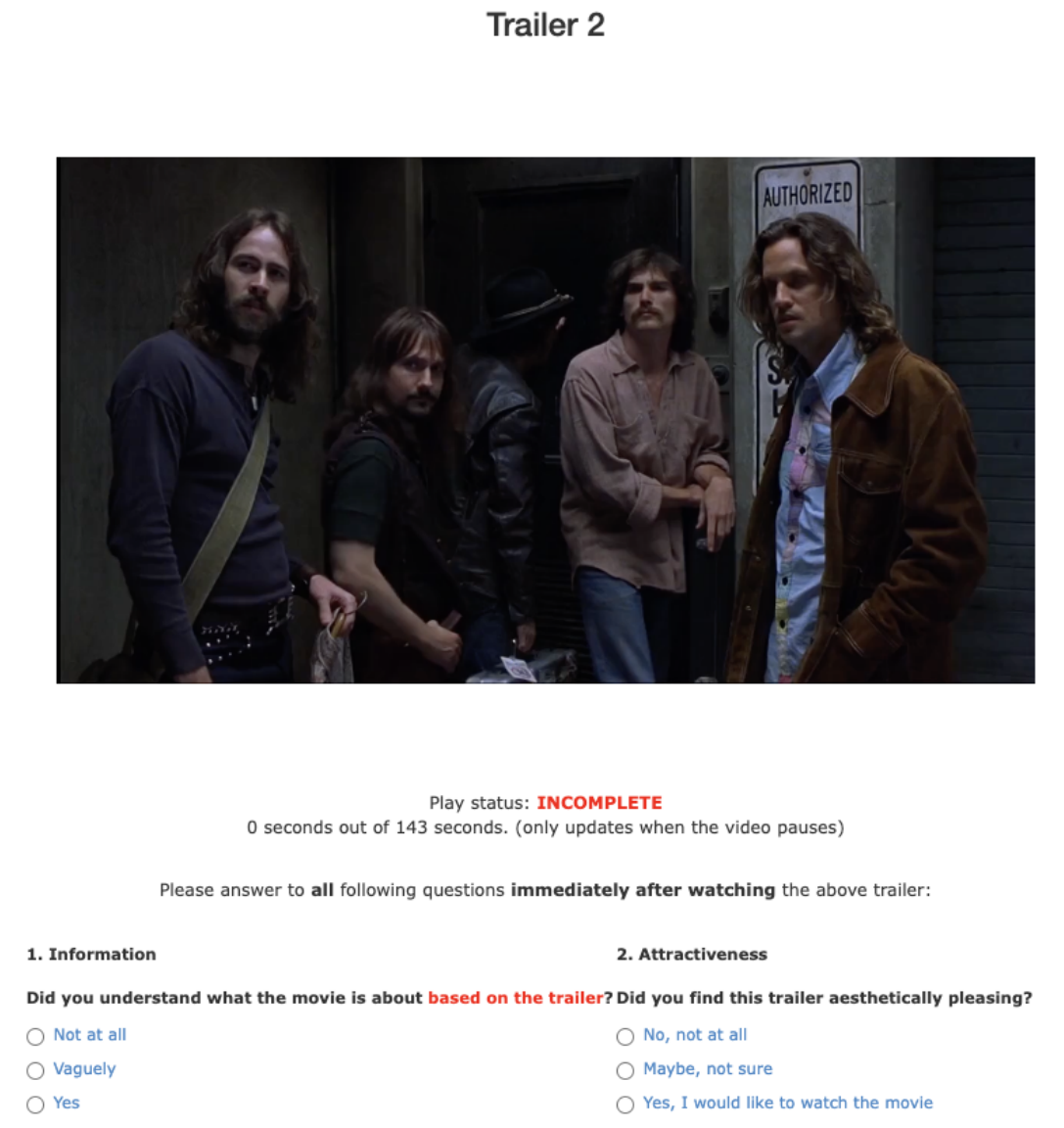


Figure C.14: Questions asked relating to the information provided in the movie (Q1) and the attractiveness of the trailer (Q2).

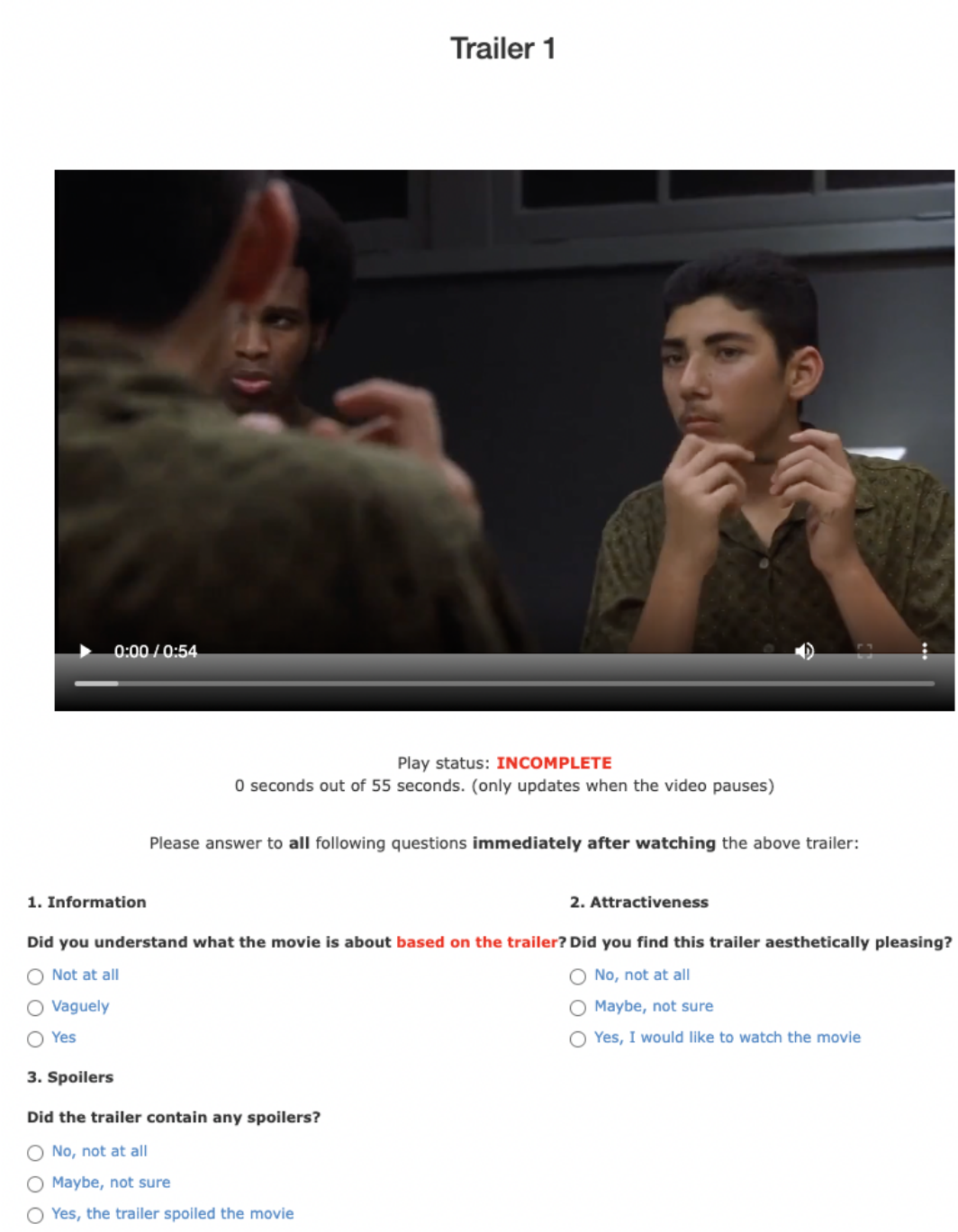


Figure C.15: Questions asked relating to the information provided in the movie (Q1), the attractiveness of the trailer (Q2), and whether the trailer contains spoilers (Q3). This is the setup of the human evaluation for the second part of our study on trailer generation that includes a semi-automatic method.

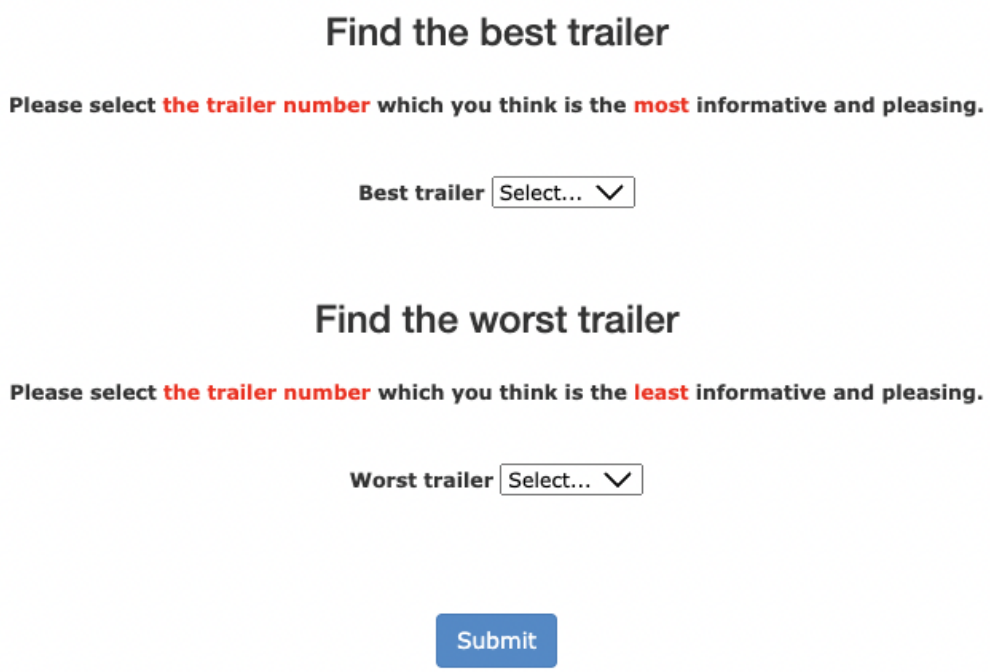


Figure C.16: Selection of the best and worst trailer from a set of five options.

Appendix D

Model Comparison between GRAPHTP and GRAPHTRAILER

As described in Chapter 6, we use an auxiliary text-based network for identifying scenes that represent TP events in screenplays. This network is used for distilling knowledge from screenplays to movie videos (see Sections 6.2.3 and 6.2.4) and has a similar architecture to GRAPHTP described in Chapter 5. The main differences between the two model variants are:

1. The auxiliary text-based network of Chapter 6 has *directed edges* in the learned movie graph. This only allows edges from past to future scenes in the graph. Such a constraint does not exist in the original GRAPHTP model of Chapter 5.

2. In Chapter 6, we substitute the BiLSTMs used for contextualizing sentences with respect to a scene and scenes with respect to the whole screenplay with transformer encoders (see Section 2.2.2 for a detailed analysis of the architecture).

We compare the two variants of GRAPHTP in Table D.1 in order to investigate how the modifications of the network affect model performance on TP identification over screenplays. All models presented in Table D.1 only consider *textual* information from the screenplays for a straight-forward comparison. Moreover, we use the same dev/test set split of gold standard annotated screenplays as in Chapter 6. First, we observe that adding the constraint of only having future connections in the graph (i.e., directed edges) improves the performance of GRAPHTP on TP identification according to all metrics, both for the dev and test sets. Next, we also evaluate model performance when we pre-train the network on Scriptbase (Gorinski and Lapata, 2015) that includes more unlabeled screenplays (see Section 6.2.4 for details). We observe that pre-training the

	Dev set			Test set		
	TA ↑	**PA** ↑	**D** ↓	**TA** ↑	**PA** ↑	**D** ↓
GRAPHTP (Chapter 5)	9.77	12.00	10.52	5.42	6.67	**8.77**
GRAPHTP w/ directed edges	11.49	14.00	10.03	**8.43**	10.00	8.79
+ pre-training on Scriptbase	**12.64**	14.00	10.32	5.42	8.89	9.07
Transformer-based GRAPHTP w/ directed edges	11.49	13.00	9.54	6.02	8.89	8.95
+ pre-training on Scriptbase	10.92	**15.00**	**8.61**	**8.43**	**13.33**	**7.42**

Table D.1: Performance comparison of variants of the GRAPHTP model used on Chapters 5 and 6 for TP identification on screenplays. We use the same dev/test set split as in Chapter 6. Evaluation metrics: Total Agreement (TA), Partial Agreement (PA), and mean distance D (see details in Chapter 3).

network with a self-supervised objective on more data slightly increases performance (only in terms of total agreement) in the dev set, but performance degrades on the test set.

Next, we also evaluate the behavior of the model when we substitute the BiLSTMs with Transformer encoders. In this case, we observe that performance is slightly worse according to most metrics (compare lines 2 and 4 in Table D.1). Presumably, transformer encoders perform slightly worse on the TRIPOD dataset compared to BiLSTMs due to the small dataset size. However, when we further pre-train the transformer-based GRAPHTP on Scriptbase, we observe a significant improvement in performance, especially on the test set. This validates our assumption that transformer encoders improve performance when the dataset size is larger, whereas when training only on TRIPOD, BiLSTMs obtain slightly higher total and partial agreement on TP identification. Given this ablation study, we use transformer encoders for the full model described in Chapter 6.

Bibliography

Agarwal, A., Balasubramanian, S., Zheng, J., and Dash, S. (2014a). Parsing Screenplays for Extracting Social Networks from Movies. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature*, pages 50–58, Gothenburg, Sweden.

Agarwal, A., Dash, S., Balasubramanian, S., and Zheng, J. (2014b). Using Determinantal Point Processes for Clustering with Application to Automatically Generating and Drawing xkcd Movie Narrative Charts. In *Proceedings of the 2nd Academy of Science and Engineering International Conference on Big Data Science and Computing*, Stanford, California.

Akoury, N., Wang, S., Whiting, J., Hood, S., Peng, N., and Iyyer, M. (2020). Storium: A dataset and evaluation platform for machine-in-the-loop story generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484.

Alon, U. and Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.

Ammanabrolu, P., Cheung, W., Broniec, W., and Riedl, M. O. (2021). Automated storytelling via causal, commonsense plot ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5859–5867.

Angelidis, S., Frermann, L., Marcheggiani, D., Blanco, R., and Màrquez, L. (2019). Book qa: Stories of challenges and opportunities. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 78–85.

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Proceedings of the Advances in Neural Information Processing Systems*, pages 2654–2662, Montreal, Quebec, Canada.

Badamdorj, T., Rochan, M., Wang, Y., and Cheng, L. (2021). Joint visual and audio learning for video highlight detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8127–8137.

Bahdanau, D., Cho, K. H., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Bain, M., Nagrani, A., Brown, A., and Zisserman, A. (2020). Condensed movies: Story based retrieval with contextual embeddings. In *Proceedings of the Asian Conference on Computer Vision*.

Bajgar, O., Kadlec, R., and Kleindienst, J. (2016). Embracing data abundance: Booktest dataset for reading comprehension. *arXiv preprint arXiv:1610.00956*.

Bamman, D., Lewke, O., and Mansoor, A. (2020). An annotated dataset of coreference in english literature. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 44–54.

Bamman, D., O'Connor, B., and Smith, N. A. (2013). Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria.

Bamman, D., Popat, S., and Shen, S. (2019). An annotated dataset of literary entities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2138–2144.

Bamman, D., Underwood, T., and Smith, N. A. (2014). A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland.

Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J., and Silva, J. P. (2019). The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*.

Bauml, M., Tapaswi, M., and Stiefelhagen, R. (2013). Semi-supervised learning with constraints for person identification in multimedia data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3602–3609.

Baziotis, C., Titov, I., Birch, A., and Haddow, B. (2021). Exploring unsupervised pretraining objectives for machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2956–2971.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150*.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Black, J. B. and Wilensky, R. (1979). An evaluation of story grammars. *Cognitive science*, 3(3):213–229.

Bojanowski, P., Bach, F., Laptev, I., Ponce, J., Schmid, C., and Sivic, J. (2013). Finding actors and actions in movies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2280–2287.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Busso, C., Bulut, M., Lee, C.-C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J. N., Lee, S., and Narayanan, S. S. (2008). Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE Computer Society.

Cascante-Bonilla, P., Sitaraman, K., Luo, M., and Ordonez, V. (2019). Moviescope: Large-scale analysis of movies using multiple modalities. *arXiv preprint arXiv:1908.03180*.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder for English. In *Proceedings of the 2018 Conference on EMNLP: System Demonstrations*.

Chambers, N. and Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore.

Chaturvedi, S., Iyyer, M., and Daume III, H. (2017). Unsupervised learning of evolving relationships between literary characters. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Chen, M., Chu, Z., Wiseman, S., and Gimpel, K. (2022a). Summscreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615.

Chen, R., Zhou, P., Wang, W., Chen, N., Peng, P., Sun, X., and Wang, W. (2021). Pr-net: Preference reasoning for personalized video highlight detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7980–7989.

Chen, S., Hao, X., Nie, X., and Hamid, R. (2022b). Movies2scenes: Learning scene representations using movie similarities. *arXiv preprint arXiv:2202.10650*.

Cheng, J. and Lapata, M. (2016). Neural summarization by extracting sentences and words. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 484–494. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Clark, J. H., Choi, E., Collins, M., Garrette, D., Kwiatkowski, T., Nikolaev, V., and Palomaki, J. (2020). Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Cutting, J. E. (2016). Narrative theory and the dynamics of popular movies. *Psychonomic bulletin & review*, 23(6):1713–1743.

De Avila, S. E. F., Lopes, A. P. B., da Luz Jr, A., and de Albuquerque Araújo, A. (2011). Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Do, T. T. H., Tran, Q. H. B., and Tran, Q. D. (2018). Movie indexing and summarization using social network techniques. *Vietnam Journal of Computer Science*, 5(2):157–164.

Dong, Y. (2018). A survey on neural network-based summarization methods. *ArXiv*, abs/1804.04589.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. (2019). Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Durrett, G., Berg-Kirkpatrick, T., and Klein, D. (2016). Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany. Association for Computational Linguistics.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems*, 28:2224–2232.

Egri, L. (1972). *The art of dramatic writing: Its basis in the creative interpretation of human motives*. Simon and Schuster.

Elgohary, A., Zhao, C., and Boyd-Graber, J. (2018). A dataset and baselines for sequential open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1077–1083.

Elson, D. and McKeown, K. (2009). Extending and evaluating a platform for story understanding. In *Proceedings of the AAAI 2009 Spring Symposium on Intelligent Narrative Technologies II*, page ??, ??

Elson, D. K., Dames, N., and Mckeown, K. R. (2010). Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147, Uppsala, Sweden.

Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Fan, A., Lewis, M., and Dauphin, Y. (2019). Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Field, S. (2005). *Screenplay: Foundations of Screenwriting*. Dell Publishing Company.

Finlayson, M. A. (2012). *Learning Narrative Structure from Annotated Folktales*. PhD thesis, Massachusetts Institute of Technology.

Fludernik, M. (2002). *Towards a'natural'narratology*. Routledge.

Frermann, L. (2019). Extractive narrativeqa with heuristic pre-training. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 172–182.

Frermann, L., Cohen, S. B., and Lapata, M. (2018). Whodunnit? crime drama as a case for natural language understanding. *Transactions of the Association of Computational Linguistics*, 6:1–15.

Freytag, G. (1896). *Freytag's technique of the drama: an exposition of dramatic composition and art*. Scholarly Press.

Frijda, N. H. et al. (1986). *The emotions*. Cambridge University Press.

Frow, J. (2014). *Character and person*. Oxford University Press.

Fu, C.-Y., Lee, J., Bansal, M., and Berg, A. (2017). Video highlight prediction using audience chat reactions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 972–978.

Gehrmann, S., Deng, Y., and Rush, A. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE.

Ghosal, D., Majumder, N., Gelbukh, A., Mihalcea, R., and Poria, S. (2020). Cosmic: Commonsense knowledge for emotion identification in conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2470–2481.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. (2019). Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79.

Goldberg, A. and Zhu, X. (2006). Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for NLP*.

Goldfarb-Tarrant, S., Chakrabarty, T., Weischedel, R., and Peng, N. (2020). Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4319–4338.

Gorinski, P. J. and Lapata, M. (2015). Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.

Gorinski, P. J. and Lapata, M. (2018). What's this movie about? a joint neural network architecture for movie content analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1770–1781.

Goyal, A., Riloff, E., and Daumé III, H. (2010). Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86, Cambridge, MA.

Gu, C., Sun, C., Ross, D. A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., et al. (2018). Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Gygli, M., Grabner, H., Riemenschneider, H., and Gool, L. V. (2014). Creating summaries from user videos. In *European conference on computer vision*, pages 505–520. Springer.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hague, M. (2017). *Storytelling Made Easy: Persuade and Transform Your Audiences, Buyers, and Clients – Simply, Quickly, and Profitably*. Indie Books International.

Haurilet, M.-L., Tapaswi, M., Al-Halah, Z., and Stiefelhagen, R. (2016). Naming tv characters by watching and analyzing dialogs. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE.

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

Hearst, M. A. (1997). Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Honnibal, M. and Montani, I. (2017). spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1):411–420.

Honovich, O., Choshen, L., Aharoni, R., Neeman, E., Szpektor, I., and Abend, O. (2021). Q2:: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7856–7870.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Huang, Q., Xiong, Y., Rao, A., Wang, J., and Lin, D. (2020). Movienet: A holistic dataset for movie understanding. In *European Conference on Computer Vision*, pages 709–727. Springer.

Ippolito, D., Grangier, D., Callison-Burch, C., and Eck, D. (2019). Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, pages 37–43.

Irie, G., Satou, T., Kojima, A., Yamasaki, T., and Aizawa, K. (2010). Automatic trailer generation. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 839–842.

Iyyer, M., Guha, A., Chaturvedi, S., Boyd-Graber, J., and Daumé III, H. (2016). Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net.

Jannidis, F. (2009). Character. In *Handbook of narratology*, pages 14–29. de Gruyter.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Jung, C. G. (2014). *The archetypes and the collective unconscious*. Routledge.

Kazantseva, A. and Szpakowicz, S. (2010). Summarizing short stories. *Computational Linguistics*, 36(1):71–109.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608.

Kim, H., Tang, Z., and Bansal, M. (2020). Dense-caption matching and frame-selection gating for temporal localization in videoqa. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4812–4822.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Kitaev, N., Kaiser, L., and Levskaya, A. (2019). Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Kline, R. (2005). *Principles and Practice of Structural Equation Modeling*. The Guilford Press, New York.

Kočiskỳ, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2018). The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Kratzwald, B. and Feuerriegel, S. (2018). Adaptive document retrieval for deep question answering. *arXiv preprint arXiv:1808.06528*.

Kryściński, W., McCann, B., Xiong, C., and Socher, R. (2020). Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346.

Kukleva, A., Tapaswi, M., and Laptev, I. (2020). Learning interactions and relationships between movie characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9849–9858.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Lal, Y. K., Chambers, N., Mooney, R., and Balasubramanian, N. (2021). Tellmewhy: A dataset for answering why-questions in narratives. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 596–610.

Lavandier, Y. (2005). *Writing drama: A comprehensive guide for playwrights and scriptwriters*. Le Clown & l'Enfant.

Lehnert, W. G. (1981). Plot units and narrative summarization. *Cognitive Science*, 5(4):293–331.

Lei, J., Wang, L., Shen, Y., Yu, D., Berg, T., and Bansal, M. (2020a). Mart: Memory-augmented recurrent transformer for coherent video paragraph captioning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2603–2614.

Lei, J., Yu, L., Bansal, M., and Berg, T. (2018). Tvqa: Localized, compositional video question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1379.

Lei, J., Yu, L., Berg, T. L., and Bansal, M. (2020b). Tvr: A large-scale dataset for video-subtitle moment retrieval. In *European Conference on Computer Vision*, pages 447–463. Springer.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Li, L., Chen, Y.-C., Cheng, Y., Gan, Z., Yu, L., and Liu, J. (2020). Hero: Hierarchical encoder for video+ language omni-representation pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2046–2065.

Li, L., Lei, J., Gan, Z., Yu, L., Chen, Y.-C., Pillai, R., Cheng, Y., Zhou, L., Wang, X. E., Wang, W. Y., et al. (2021). Value: A multi-task benchmark for video-and-language understanding evaluation. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks*.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., and Niu, S. (2017). Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 986–995.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Liu, J., Chen, W., Cheng, Y., Gan, Z., Yu, L., Yang, Y., and Liu, J. (2020). Violin: A large-scale dataset for video-and-language inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10900–10910.

Liu, Y. and Lapata, M. (2019a). Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Liu, Y. and Lapata, M. (2019b). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Lohnert, W. G., Black, J. B., and Reiser, B. J. (1981). Summarizing narratives. In *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 1*, pages 184–189.

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. (2015). Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*.

Louviere, J., Flynn, T., and Marley, A. A. J. (2015). *Best-worst scaling: Theory, methods and applications*.

Luo, H., Ji, L., Shi, B., Huang, H., Duan, N., Li, T., Li, J., Bharti, T., and Zhou, M. (2020). Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mandler, J. M. and Johnson, N. S. (1977). Remembrance of things parsed: Story structure and recall. *Cognitive psychology*, 9(1):111–151.

Mani, I. (2012). *Computational Modeling of Narative*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Marcheggiani, D. and Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on EMNLP*.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919.

McInnes, L., Healy, J., Saul, N., and Großberger, L. (2018). Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

McIntyre, N. and Lapata, M. (2010). Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572, Uppsala, Sweden.

Miech, A., Alayrac, J.-B., Smaira, L., Laptev, I., Sivic, J., and Zisserman, A. (2020). End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889.

Miech, A., Zhukov, D., Alayrac, J.-B., Tapaswi, M., Laptev, I., and Sivic, J. (2019). Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640.

Mihalcea, R. and Ceylan, H. (2007). Explorations in automatic book summarization. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 380–389.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Mou, X., Yang, C., Yu, M., Yao, B., Guo, X., Potdar, S., and Su, H. (2021). Narrative question answering with cutting-edge open-domain qa techniques: A comprehensive study. *Transactions of the Association for Computational Linguistics*, 9:1032–1046.

Mou, X., Yu, M., Yao, B., Yang, C., Guo, X., Potdar, S., and Su, H. (2020). Frustratingly hard evidence retrieval for qa over books. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, pages 108–113.

Myers, C. S. and Rabiner, L. R. (1981). A comparative study of several dynamic time-warping algorithms for connected-word recognition. *Bell System Technical Journal*, 60(7):1389–1409.

Nallapati, R., Zhai, F., and Zhou, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Napoles, C., Gormley, M., and Van Durme, B. (2012). Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada. Association for Computational Linguistics.

Narayan, S., Cohen, S., and Lapata, M. (2018). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization.

In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics.

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Niu, Z.-Y., Ji, D.-H., and Tan, C. L. (2005). Word sense disambiguation using label propagation based semi-supervised learning. In *Proceedings of the 43rd Annual Meeting of ACL*.

Oono, K. and Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Ozaki, K., Shimbo, M., Komachi, M., and Matsumoto, Y. (2011). Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. In *Proceedings of the fifteenth conference of CoNLL*, pages 154–162.

Palaskar, S., Libovickỳ, J., Gella, S., and Metze, F. (2019). Multimodal abstractive summarization for how2 videos. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6587–6596.

Pan, B., Cai, H., Huang, D.-A., Lee, K.-H., Gaidon, A., Adeli, E., and Niebles, J. C. (2020). Spatio-temporal graph for video captioning with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10870–10879.

Papalampidi, P., Cao, K., and Kocisky, T. (2022a). Towards coherent and consistent use of entities in narrative generation. *arXiv preprint arXiv:2202.01709*.

Papalampidi, P., Cao, K., and Kocisky, T. (2022b). Towards coherent and consistent use of entities in narrative generation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17278–17294. PMLR.

Papalampidi, P., Keller, F., Frermann, L., and Lapata, M. (2020). Screenplay summarization using latent narrative structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1920–1933.

Papalampidi, P., Keller, F., and Lapata, M. (2019). Movie plot analysis via turning point identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1707–1717.

Papalampidi, P., Keller, F., and Lapata, M. (2021a). Film trailer generation via task decomposition. *arXiv preprint arXiv:2111.08774*.

Papalampidi, P., Keller, F., and Lapata, M. (2021b). Movie summarization via sparse graph construction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Papalampidi, P. and Lapata, M. (2022). Hierarchical3d adapters for long video-to-text summarization. *arXiv preprint arXiv:2210.04829*.

Papasarantopoulos, N., Frermann, L., Lapata, M., and Cohen, S. B. (2019). Partners in crime: Multi-view sequential inference for movie understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2057–2067, Hong Kong, China. Association for Computational Linguistics.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Pavis, P. (1998). *Dictionary of the theatre: Terms, concepts, and analysis*. University of Toronto Press.

Poria, S., Hazarika, D., Majumder, N., Naik, G., Cambria, E., and Mihalcea, R. (2019). Meld: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 527–536.

Propp, V. I. (1968). *Morphology of the Folktale*. University of Texas.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Ramanathan, V., Joulin, A., Liang, P., and Fei-Fei, L. (2014). Linking people in videos with "their" names using coreference resolution. In *European conference on computer vision*, pages 95–110. Springer.

Rashkin, H., Celikyilmaz, A., Choi, Y., and Gao, J. (2020). Plotmachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295.

Reagan, A. J., Mitchell, L., Kiley, D., Danforth, C. M., and Dodds, P. S. (2016). The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):1–12.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 506–516.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Richards, W., Finlayson, M. A., and Winston, P. H. (2009). Advancing computational models of narrative. Technical Report 63:2009, MIT Computer Science and Atrificial Intelligence Laboratory.

Robertson, S. and Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Robertson, S. E., Walker, S., Jones, S., et al. (1995). Okapi at trec-3.

Rohrbach, A., Rohrbach, M., Qiu, W., Friedrich, A., Pinkal, M., and Schiele, B. (2014). Coherent multi-sentence video description with variable level of detail. In *German conference on pattern recognition*, pages 184–195. Springer.

Rohrbach, A., Rohrbach, M., Tandon, N., and Schiele, B. (2015). A dataset for movie description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3202–3212.

Rohrbach, A., Torabi, A., Rohrbach, M., Tandon, N., Pal, C., Larochelle, H., Courville, A., and Schiele, B. (2017). Movie description. *International Journal of Computer Vision*, 123(1):94–120.

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Rumelhart, D. E. (1980). On evaluating story grammars. *Cognitive Science*, 4(3):313–316.

Sadhu, A., Gupta, T., Yatskar, M., Nevatia, R., and Kembhavi, A. (2021). Visual semantic role labeling for video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5600.

Sanabria, R., Caglayan, O., Palaskar, S., Elliott, D., Barrault, L., Specia, L., and Metze, F. (2018). How2: A large-scale dataset for multimodal language understanding. In *. In Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL). NIPS.*

Sang, Y., Mou, X., Yu, M., Yao, S., Li, J., and Stanton, J. (2022). Tvshowguess: Character comprehension in stories as speaker guessing. *arXiv preprint arXiv:2204.07721*.

Schank, R. C. and Abelson, R. P. (1975). Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, pages 151–157, Tblisi, USSR.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2017). Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., et al. (2022). Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*.

Sims, M., Park, J. H., and Bamman, D. (2019). Literary event detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3623–3634.

Sivic, J., Everingham, M., and Zisserman, A. (2009). "who are you?"-learning person specific classifiers from video. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1145–1152. IEEE.

Smeaton, A. F., Lehane, B., O'Connor, N. E., Brady, C., and Craig, G. (2006). Automatically selecting shots for action movie trailers. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 231–238. ACM.

Smith, J. R., Joshi, D., Huet, B., Hsu, W., and Cota, J. (2017). Harnessing ai for augmenting creativity: Application to movie trailer creation. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1799–1808. ACM.

Song, Y., Vallmitjana, J., Stent, A., and Jaimes, A. (2015). Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5179–5187.

Srivastava, S., Chaturvedi, S., and Mitchell, T. (2016). Inferring interpersonal relations in narrative summaries. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*, pages 2807–2813, Phoenix, Arizona. AAAI Press.

Sun, S., Gan, Z., Fang, Y., Cheng, Y., Wang, S., and Liu, J. (2020). Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508.

Sun, S., Zhao, W., Manjunatha, V., Jain, R., Morariu, V., Dernoncourt, F., Srinivasan, B. V., and Iyyer, M. (2021). Iga: An intent-guided authoring assistant. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5972–5985.

Sun, Y., Chao, Q., and Li, B. (2022). Synopses of movie narratives: a video-language dataset for story understanding. *arXiv preprint arXiv:2203.05711*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Szummer, M. and Jaakkola, T. S. (2003). Information regularization with partially labeled data. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1049–1056. MIT Press.

Tambwekar, P., Dhuliawala, M., Martin, L. J., Mehta, A., Harrison, B., and Riedl, M. O. (2018). Controllable neural story plot generation via reinforcement learning. *arXiv preprint arXiv:1809.10736*.

Tapaswi, M., Bäuml, M., and Stiefelhagen, R. (2015a). Aligning plot synopses to videos for story-based retrieval. *International Journal of Multimedia Information Retrieval*, 4(1):3–16.

Tapaswi, M., Bauml, M., and Stiefelhagen, R. (2015b). Book2movie: Aligning video scenes with book chapters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1827–1835.

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., and Fidler, S. (2016). Movieqa: Understanding stories in movies through question-answering. In *Pro-*

ceedings of the IEEE conference on computer vision and pattern recognition, pages 4631–4640.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. (2020). Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR.

Tay, Y., Wang, S., Luu, A. T., Fu, J., Phan, M. C., Yuan, X., Rao, J., Hui, S. C., and Zhang, A. (2019). Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4922–4931.

Thai, K., Chang, Y., Krishna, K., and Iyyer, M. (2022). Relic: Retrieving evidence for literary claims. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7500–7518.

Thompson, K. (1999). *Storytelling in the new Hollywood: Understanding classical narrative technique*. Harvard University Press.

Tran, Q. D., Hwang, D., Lee, O.-J., and Jung, J. E. (2017). Exploiting character networks for movie summarization. *Multimedia Tools and Applications*, 76(8):10357–10369.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2016). Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Tsimpoukelli, M., Menick, J., Cabi, S., Eslami, S. A., Vinyals, O., and Hill, F. (2021). Multimodal few-shot learning with frozen language models. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Tsoneva, T., Barbieri, M., and Weda, H. (2007). Automated summarization of narrative video on a semantic level. In *International Conference on Semantic Computing (ICSC 2007)*, pages 169–176. IEEE.

Valls-Vargas, J., Zhu, J., and Ontanon, S. (2014). Toward automatic role identification in unannotated folk tales. In *Proceedings of the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 188–194, Raleigh, North Carolina.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vicol, P., Tapaswi, M., Castrejon, L., and Fidler, S. (2018). Moviegraphs: Towards understanding human-centric situations from videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8581–8590.

Vogler, C. (2007). *Writer's Journey: Mythic Structure for Writers*. Michael Wiese Productions.

Wang, D., Liu, P., Zheng, Y., Qiu, X., and Huang, X. (2020a). Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of ACL*.

Wang, L., Liu, D., Puri, R., and Metaxas, D. N. (2020b). Learning trailer moments in full-length movies with co-contrastive attention. In *European Conference on Computer Vision*, pages 300–316. Springer.

Wang, S., Durrett, G., and Erk, K. (2020c). Narrative interpolation for generating and understanding stories. *arXiv preprint arXiv:2008.07466*.

Wang, W., Li, P., and Zheng, H.-T. (2021). Consistency and coherency enhanced story generation. In *European Conference on Information Retrieval*, pages 694–709. Springer.

Wang, Y., Liu, K., Liu, J., He, W., Lyu, Y., Wu, H., Li, S., and Wang, H. (2018). Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1918–1927.

Welbl, J., Stenetorp, P., and Riedel, S. (2018). Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. `https://github.com/facebookresearch/detectron2`.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Xiao, F., Kundu, K., Tighe, J., and Modolo, D. (2022). Hierarchical self-supervised representation learning for movie understanding. *arXiv preprint arXiv:2204.03101*.

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

Xiong, Y., Huang, Q., Guo, L., Zhou, H., Zhou, B., and Lin, D. (2019). A graph-based framework to bridge movies and synopses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4592–4601.

Xu, H., Ghosh, G., Huang, P.-Y., Arora, P., Aminzadeh, M., Feichtenhofer, C., Metze, F., and Zettlemoyer, L. (2021). Vlm: Task-agnostic video-language model pre-training for video understanding. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4227–4239.

Xu, H., Zhen, Y., and Zha, H. (2015). Trailer generation via a point process-based visual attractiveness model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

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

Xu, P., Patwary, M., Shoeybi, M., Puri, R., Fung, P., Anandkumar, A., and Catanzaro, B. (2020). Controllable story generation with external knowledge using large-scale

language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845.

Xu, Y., Wang, D., Yu, M., Ritchie, D., Yao, B., Wu, T., Zhang, Z., Li, T., Bradford, N., Sun, B., et al. (2022). Fantastic questions and where to find them: Fairytaleqa–an authentic dataset for narrative comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 447–460.

Yamada, I., Asai, A., Shindo, H., Takeda, H., and Takefuji, Y. (2018). Wikipedia2vec: An optimized tool for learning embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280*.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. (2018). Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Ye, P. and Baldwin, T. (2008). Towards Automatic Animated Storyboarding. In *Proceedings of the 23rd Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, pages 578–583, Chicago, Illinois.

Ye, Q., Shen, X., Gao, Y., Wang, Z., Bi, Q., Li, P., and Yang, G. (2021). Temporal cue guided video highlight detection with low-rank audio-visual fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7950–7959.

Yu, T., Dai, W., Liu, Z., and Fung, P. (2021). Vision guided generative pre-trained language models for multimodal abstractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3995–4007.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Zhang, Y., Ni, A., Mao, Z., Wu, C. H., Zhu, C., Deb, B., Awadallah, A., Radev, D., and Zhang, R. (2022). Summ^N: A multi-stage summarization framework for long input dialogues and documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604.

Zhang, Y., Ni, A., Yu, T., Zhang, R., Zhu, C., Deb, B., Celikyilmaz, A., Hassan, A., and Radev, D. (2021). An exploratory study on long dialogue summarization: What works and what's next. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4426–4433.

Zheng, H. and Lapata, M. (2019). Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247.

Zhong, M., Liu, Y., Xu, Y., Zhu, C., and Zeng, M. (2022). Dialoglm: Pre-trained model for long dialogue understanding and summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11765–11773.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Hassan, A., Celikyilmaz, A., Liu, Y., Qiu, X., et al. (2021). Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921.

Zhou, L., Xu, C., and Corso, J. J. (2018a). Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., and Zhao, T. (2018b). Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663.

Zhu, X. (2005). *Semi-supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University. CMU-LTI-05-192.