# Transformer-based Screenplay Summarization Using Augmented Learning Representation with Dialogue Information

**Myungji Lee**[1] **Hongseok Kwon**[2] **Jaehun Shin**[1]
**WonKee Lee**[1] **Baikjin Jung**[1] **Jong-Hyeok Lee**[1 2]
[1]Department of Computer Science and Engineering
[2]Graduate School of Artificial Intelligence
POSTECH, Republic of Korea
{mjlee7, hkwon, jaehun.shin}@postech.ac.kr
{wklee, bjjung, jhlee}@postech.ac.kr

## Abstract

Screenplay summarization is the task of extracting informative scenes from a screenplay. The screenplay contains turning point (TP) events that change the story direction and thus define the story structure decisively. Accordingly, this task can be defined as the TP identification task. We suggest using dialogue information, one attribute of screenplays, motivated by previous work that discovered that TPs have a relation with dialogues appearing in screenplays. To teach a model this characteristic, we add a dialogue feature to the input embedding. Moreover, in an attempt to improve the model architecture of previous studies, we replace LSTM with Transformer. We observed that the model can better identify TPs in a screenplay by using dialogue information and that a model adopting Transformer outperforms LSTM-based models.

## 1 Introduction

Text summarization is one major task in NLP that seeks to produce concise texts containing only the essential information in the original texts. Although most researches have been focusing on summarizing news articles (Narayan et al., 2018; See et al., 2017), as various contents with different structures increase these days, there has been growing interests in applying text summarization to various domains, including social media (Sharifi et al., 2010; Kim and Monroy-Hernandez, 2016), dialogue (Goo and Chen, 2018), scientific articles (Cohan and Goharian, 2017; Yasunaga et al., 2019), books (Mihalcea and Ceylan, 2007), screenplays (or scripts) (Gorinski and Lapata, 2015; Papalampidi et al., 2020a). Among them, this paper focuses on screenplay summarization.

A screenplay is a type of literary text, which typically contains around 120 pages and has a strictly structured format (Figure 1). It usually contains various storytelling elements, such as a story, dialogues, characters' actions, and what the camera sees, thereby elaborating a complex story. In a real-life situation, filmmakers and directors hire script readers to select a script that seems to be a popular movie among numerous candidate scripts. They create a coverage per script, a report of about four pages containing a logline (the indicative summary), a synopsis (the informative summary), recommendations, ratings, and comments.

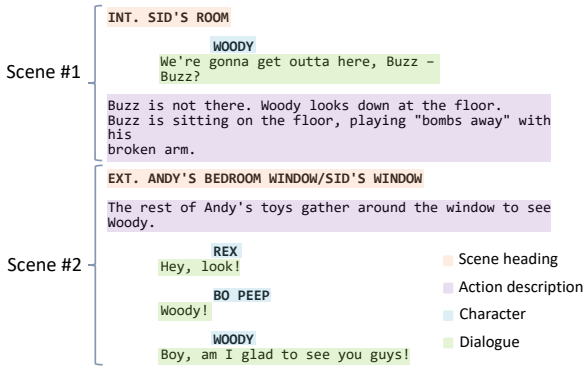


Figure 1: An excerpt from "Toy Story." A screenplay consists of scenes. A scene is an event that takes place at the same time or place. Every scene starts with a scene heading (starts with "INT." or "EXT.") and is followed by action descriptions and dialogues. 'Scene heading' denotes when and where actions take place. 'Action description' explains who and what are in the scene. 'Character' is the speaker. 'Dialogue' is a spoken utterance.

The goal of screenplay summarization is to help speeding up script browsing; to provide an overview of the script's contents and storyline; and to reduce the reading time (Gorinski and Lapata, 2015). As shown in Figure 2, to make this long narrative-text summarization feasible, early work in screenplay summarization (Gorinski and Lapata, 2015; Papalampidi et al., 2020a) defined the task as extracting a sequence of scenes that represents informative summary (i.e., scene-level extractive summarization).

To this end, Papalampidi et al. (2019, 2020b)



*Proceedings of the 3rd Workshop on Narrative Understanding*, pages 56–61
June 11, 2021. 

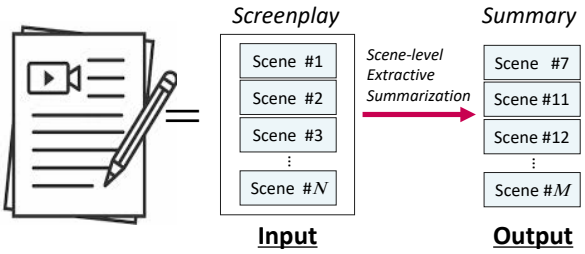


Figure 2: Screenplay summarization is defined as scene-level extractive summarization (Gorinski and Lapata, 2015; Papalampidi et al., 2020a).

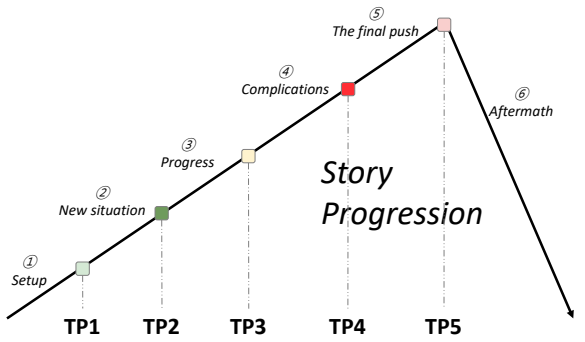


Figure 3: A well-structured story consists of six stages. TPs divide a story into multiple sections and define the screenplay's structure. There are five TPs in a story (Cutting, 2016; Hauge, 2017; Papalampidi et al., 2019).

assumed that such scenes compose a set of events, called turning points (TPs), which change the story's direction and thus determine the progression of the story (Figure 3). The definition of each TP is shown in Table 1.

Following their assumption, we propose two methods to identify TPs better: 1) we suggest using dialogue information included in screenplays (Figure 1) as a training feature, considering one previous study revealed that there is a relation between TPs and the frequency of conversations (Cutting, 2016) in a screenplay; 2) we attempt to use Transformer (Vaswani et al., 2017) instead of LSTM, which have been dominantly used in previous studies (Papalampidi et al., 2019, 2020b), because Transformer has generally shown to be beneficial in capturing long-term dependencies; we can expect that Transformer will summarize long and complex screenplays better.

| |
|---|
| TP1: Opportunity |
| Introductory event that occurs after presentation of setting and background of main characters |
| TP2: Change of Plans |
| Main goal of story is defined; action begins to increase |
| TP3: Point of No Return |
| Event that pushes the main characters to fully commit to their goal |
| TP4: Major Setback |
| Event where everything falls apart, temporarily or permanently |
| TP5: Climax |
| Final event of the main story, moment of resolution and "biggest spoiler" |

Table 1: Definition of TPs (Papalampidi et al., 2019).

## 2 Background and Related Work

### 2.1 Topic-Aware Model

Topic-Aware Model (TAM) (Papalampidi et al., 2019) is one screenplay summarization model that identifies TPs to use them for an informative summary. The key feature of this model is that it takes sentence-level inputs and uses Bi-LSTM to generate their latent representations; it produces scene representations by applying self-attention to the sentence representations belonging to each scene and applying a context-interaction layer to capture the similarity among scenes. At last, TPs are selected among all scene representations. Our proposed model is also inspired by this work, and our work aims to improve this study.

### 2.2 GraphTP

Another TP identification model is GraphTP (Papalampidi et al., 2020b), which uses Bi-LSTM and Graph Convolution Network (GCN) (Duvenaud et al., 2015) to encode direct interactions among scenes, thereby better capturing long-term dependencies. Specifically, they represent a screenplay as a sparse graph, and then the GCN produces scene representations that reflect information of neighboring scenes. It shows comparable performance with TAM. In our experiments, we adopt TAM and GraphTP as baselines.

## 3 Method

### 3.1 Input Augmentation

Recall that screenplay summarization can be defined as identifying TPs, where the story stage's transition occurs. Therefore, we suggest using dialogue information related to the story stage's transition to identify TPs better. The motivation for this method is that a previous study (Cutting, 2016) that analyzed movies found that there is a pattern in which the frequency of conversations changes according to the story stage (Figure 3); there are few conversations until the end of the setup; then the frequency of conversations stay constant for the progress and complication; and finally, it decreases during the beginning of the final push but increases again in the aftermath. This study implies that dialogue information can be a good hint to capture screenplays' story stage transition. However, to our knowledge, there has been no previous work that attempts to utilize such information for screenplay summarization, that is, most previous studies (Papalampidi et al., 2019, 2020b) do not consider employing various elements included in a screenplay.

We expect that adding dialogue information as an additional training feature will help a model predict TP scenes from screenplays better. Therefore, we first extract the binary label $d_i$ from a screenplay by inspecting whether a specific sentence is notated as a dialogue. We then concatenated the sentence embedding ($x_i$) and the binary label ($d_i$) to design a new augmented input $[x_i; d_i]$.

### 3.2 Architecture

It has been generally known that RNN-based architectures, which were used also in aforementioned previous studies (Papalampidi et al., 2019, 2020b), do not capture long-range dependencies well due to the vanishing gradient problem. Also in the case of screenplay summarization, because screenplays are normally long and complex, we speculate that there is a limit to generating a summary by using LSTM. Therefore, we propose a screenplay-summarization model to which Transformer (Vaswani et al., 2017) is applied, which is widely used for various NLP tasks and well known for having less computational complexity and better capturing long-term dependencies.

In detail, we propose a hierarchical screenplay encoder using Transformer (Figure 4). First, it receives a sentence-level input; we use Universal

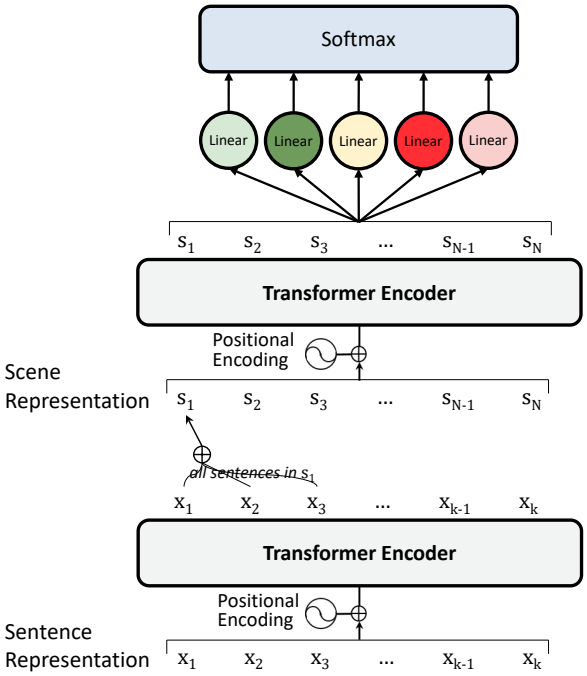


Figure 4: Proposed architecture using Transformer encoders.

Sentence Encoder (USE) (Cer et al., 2018) as in TAM. After the sentence representations become contextualized by the first Transformer encoder, all the sentence representations belonging to the scene are added up to form the scene representation that is fed into the second Transformer encoder. The second Transformer encoder produces the final scene vectors and inputs them into five different linear layers, one classifier per TP, each of which projects the vectors to a scalar value. Lastly, a softmax layer produces five probability distributions over all scenes that indicate how relevant each scene is to the TPs. We then select one scene with the highest probability per TP; each selected scene joins together with its neighbors into three consecutive scenes, which compose the final summary.

## 4 Experiments

### 4.1 Dataset

For both training and evaluating our model, we use TRIPOD (Papalampidi et al., 2019) dataset. This dataset contains screenplays and their TPs; the TPs in the test set are manually annotated by human experts whereas those in the training set are pseudo-TPs. Statistics of the dataset are presented in Table 2.

| TRIPOD | Train | Test |
|---|---|---|
| screenplays | 84 | 15 |
| scenes | 11,320 | 2,083 |
| turning points | 420 | 75 |
| *per screenplay* | | |
| tokens | 23.0k (6.6) | 20.9k (4.5) |
| sentences | 3.0k (0.9) | 2.8k (0.6) |
| scenes | 133.0 (61.1) | 138.9 (50.7) |
| *per scene* | | |
| tokens | 173.0 (235.0) | 150.5 (198.3) |
| sentences | 22.2 (31.5) | 19.9 (26.9) |
| sentence tokens | 7.8 (6.0) | 7.6 (6.4) |

Table 2: Statistics of TRIPOD (Papalampidi et al., 2019).

### 4.2 Experimental Setting

For our experiments, we adapted source codes in two repositories[1] [2] Papalampidi et al. (2020b); Liu and Lapata (2019) to implement our model. We set the training hyperparameters as follows: $L = 1$, $H = 128$, $A = 4$, and $P_{drop} = 0.0$, where $L$ is the number of layers, $H$ is the hidden size, $A$ is the number of heads, and $P_{drop}$ is the dropout rate. We consider two previous methods that receive raw sentence representations as inputs as the baseline systems: TAM (Papalampidi et al., 2019) and GraphTP (Papalampidi et al., 2020b). During training, because TRIPOD does not contain a validation set, we conducted $n$-fold cross-validation with $n = 5$ to extract the validation set from the existing test set. Finally, we averaged out the test results of the five models to obtain the final test results.

### 4.3 Evaluation Metric

To evaluate our model, we used the TP identification evaluation metrics proposed by Papalampidi et al. (2019): Total Agreement ($TA$), Partial Agreement ($PA$), and Distance ($D$). Those Metrics are defined as follows.

$TA$ is the ratio of TP scenes that are correctly identified (Eq. 1). In the equation, $S_i$ is a set of scenes that is predicted as a certain TP in a screenplay, $G_i$ is the ground-truth set of scenes corresponding to that TP event, $T$ is the number of TPs, in our case $T = 5$, and $L$ is the number of screenplays contained in the test set.

$$TA = \frac{1}{T \cdot L} \sum_{i=1}^{T \cdot L} \frac{|S_i \cap G_i|}{|S_i \cup G_i|} \quad (1)$$

$PA$ is the ratio of TP events about which more than one ground-truth TP scenes are identified (Eq. 2).

$$PA = \frac{1}{T \cdot L} \sum_{i=1}^{T \cdot L} [|S_i \cap G_i| \neq \phi] \quad (2)$$

$D$ is the average distance between all pairs of predicted TP scenes ($S_i$) and ground-truth TP scenes ($G_i$) (Eq. 3, 4), where $N$ is the number of scenes in a screenplay.

$$d[S_i, G_i] = \frac{1}{N} \min_{s \in S_i, g \in G_i} |s - g| \quad (3)$$

$$D = \frac{1}{T \cdot L} \sum_{i=1}^{T \cdot L} d[S_i, G_i] \quad (4)$$

$TA$ and $PA$ indicate how correctly a model predicts TPs, and $D$ indicates how well the model has learned TP positions. It can be seen that $TA$ and $PA$ represent the model's prediction bias, and $D$ represents variance, so we can suppose that there is a trade-off between $D$ and $TA$ or $PA$. Also, when the TA and PA scores are similar, it means that the model has a high accuracy.

| Input | Model | TA ↑ | PA ↑ | D ↓ |
|---|---|---|---|---|
| sentence | TAM | 8.15 | 9.33 | 10.59 |
| | GraphTP | 7.41 | **10.67** | 9.24 |
| | Transformer | **10.37** | **10.67** | **9.12** |
| sentence + dialogue | TAM | 7.41 | 9.33 | 9.97 |
| | GraphTP | **13.33** | **14.67** | 11.61 |
| | Transformer | 11.11 | 12.00 | **9.82** |

Table 3: Total Agreement (TA), Partial Agreement (PA), and mean distance (D). The first two rows are the baselines. A **boldface** score is the best score in its column.

| Model | # of parameters | Training time (ratio) |
|---|---|---|
| TAM | 40.1k | 1.12 |
| GraphTP | 41.6k | 1.45 |
| Transformer | 46.3k | 1.00 |

Table 4: The number of parameters and training time of models. Numbers in 'Training time' are ratios to the training time of our proposed model set at 1.

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

[2] https://github.com/nlpyang/PreSumm

### 4.4 Result Analysis

**Input Augmentation** It is revealed that the models trained with augmented inputs outperform those trained only with raw inputs by the TA and PA scores (Table 3). This result supports our assumption that dialogue information will be helpful in finding TPs because the TA and PA scores, which indicate whether TPs are correctly identified, have improved. As aforementioned in Section 4.3, the D score has an inverse relationship with TA in that it represents the variance of model predictions. On the other hand, TAM shows a relatively poor TA score; it seems that dialogue information hardly improves the performance of a model that does not capture long-term dependencies well. One possible reason is that dialogue information provides the model with information that the model already knows even though it does not capture long-term dependencies well. For more accurate explanation, further analyses are required.

**Architecture** In the case of raw sentence inputs, our proposed architecture based on Transformer outperforms the two baseline systems consistently. The result implies that the model that captures long-term dependencies well can improve the performance of summarizing long and complex texts, as we have expected. Because the model's performance has improved over the baseline by all metrics, our proposed architecture can be considered as an adequate model for TP identification, compared to the baselines. Also, even though our model contains a few more parameters than the two baselines, it has faster training speed, especially compared to GraphTP, showing a difference of almost 40% or more (Table 4).

When we fed dialogue-augmented inputs into the model, the TA and PA scores have improved. Although, when we used dialogue-augmented inputs, GraphTP recorded better performance by TA and PA, for D, our model shows much better results. This result means that the model predicts whether a given scene is a TP or not becomes more accurately whereas it does not predict well across all TPs (i.e., TP1 to TP5), but for a given scene, the model predicts certain TPs very well and some other TPs very bad. Therefore, the dialogue feature provides helpful information for TP identification that GraphTP lacks even though it is helpful for some TPs but redundant and even disturbing for some other TPs. This suggests that there is high possibility that not all TPs (i.e., TP1 to TP5) are included in the output summary. In this regard, we can conclude that our proposed model makes more confident predictions.

## 5 Conclusion

In this paper, we suggest using dialogue information as an additional training feature and propose a Transformer-based architecture for TP identification. Our experimental results present that dialogue information has a positive effect on the prediction accuracy on whether the scene is TP or not. However, the opposite was the case for the sequence-based model; further analyses are needed. In addition, the results indicate that using Transformer instead of LSTM significantly improves the overall performance in identifying TP scenes by encoding long-term dependencies among scenes better. We believe that using unique attributes in screenplays, such as dialogues, can help improving the model performance and when summarizing texts that have complex structures including screenplays, Transformer , which handles long histories robustly, is effective. In the future, we plan to go through the human evaluating process to see how dialogue information affects the output summary's informativeness, especially which one is identified better than another, and how the trade-off among automatic evaluation metrics affects the summary output.

## Acknowledgement

We appreciate all of the reviewers giving their invaluable comments on this paper. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)).

## References

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Arman Cohan and Nazli Goharian. 2017. Scientific document summarization via citation contextualization and scientific discourse. *International Journal on Digital Libraries*, 19:287–303.

James E. Cutting. 2016. Narrative theory and the dynamics of popular movies. *Psychonomic Bulletin Review*, 23:1713—-1743.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints.

C. Goo and Y. Chen. 2018. Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 735–742.

Philip John Gorinski and Mirella Lapata. 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.

Michael Hauge. 2017. *Storytelling Made Easy: Persuade and Transform Your Audiences, Buyers, and Clients – Simply, Quickly, and Profitably*. Indie Books International.

Joy Kim and Andres Monroy-Hernandez. 2016. Storia: Summarizing social media content based on narrative theory using crowdsourcing. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work amp; Social Computing*, CSCW '16, page 1018–1027, New York, NY, USA. Association for Computing Machinery.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345.

R. Mihalcea and H. Ceylan. 2007. Explorations in automatic book summarization. In *EMNLP-CoNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *CoRR*, abs/1808.08745.

Pinelopi Papalampidi, Frank Keller, Lea Frermann, and Mirella Lapata. 2020a. Screenplay summarization using latent narrative structure.

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. 2019. Movie plot analysis via turning point identification.

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. 2020b. Movie summarization via sparse graph construction.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

B. Sharifi, M. Hutton, and J. Kalita. 2010. Experiments in microblog summarization. *2010 IEEE Second International Conference on Social Computing*, pages 49–56.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, A. R. Fabbri, Irene Li, D. Friedman, and Dragomir R. Radev. 2019. Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In *AAAI*.