A Hybrid Video-to-Text Summarization Framework and Algorithm on Cascading Advanced Extractive- and Abstractive-based Approaches for Supporting Viewers' Video Navigation and Understanding

Aishwarya Ramakrishnan
Data Science Program
Worcester Polytechnic Institute
Worcester, MA 01609, USA
aramakrishnan3@wpi.edu

Chun-Kit Ngan
Data Science Program
Worcester Polytechnic Institute
Worcester, MA 01609, USA
cngan@wpi.edu

***Abstract*— In this work, we propose the development of a hybrid video-to-text summarization (VTS) framework on cascading the advanced and code-accessible extractive and abstractive (EA) approaches for supporting viewers' video navigation and understanding. More precisely, the contributions of this paper are three-fold. First, we devise an automated and unified hybrid VTS framework that takes an arbitrary video as an input, generates the text transcripts from its human dialogues, and then summarizes the text transcripts into one short video synopsis. Second, we advance the binary merge-sort approach and expand its use to develop an intuitive and heuristic abstractive-based algorithm, with the time complexity $O(T_L \log T_L)$ and the space complexity $O(T_L)$, where T_L is the total number of word tokens on a text, to dynamically and successively split and merge a long piece of text transcripts, which exceeds the input text size limitation of an abstractive model, to generate one final semantic video synopsis. At the end, we test the feasibility of applying this proposed framework and algorithm in conducting the preliminarily experimental evaluations on three different videos, as a pilot study, in genres, contents, and lengths. We show that our approach outperforms and/or levels most of the individual EA methods stated above by 75% in terms of the ROUGE F1-Score measurement.**

Keywords*— *Video-to-Text Summarization, Extractive- and Abstractive-Based Summarization, Binary Merge-Sort Abstractive-Based Algorithm

I. Introduction

YouTube is an American online video sharing and social media platform owned by Google. It is the second largest platform that has hosted large quantities of video contents, including movies, news, sports, commercials, etc. YouTube has over 2.3 billion monthly active users worldwide. In every minute, there are more than 500 hours of new contents being uploaded (i.e., 720,000 hours of new videos per day) [1]. In each day, there are 5 billion videos and 1 billon hours of total videos being watched [2]. With so many video contents available on the Internet every day, it is very challenging for viewers to navigate and understand all of them and then search for a specific and the most informative content that they would like to watch. To ease this navigation process and support viewers' video understanding, an automatic video-to-text summarization (VTS) framework is highly desirable, as a concise video summary can convey the most informative and important theme, idea, and plot to viewers. Some examples of video summary may include trailers or teasers of movies (e.g., Spider-Man: No Way Home) and TV series episodes (e.g., Squid Game), highlights of an event (e.g., sports games, music concerts, or public debates), and video synopses of main activities (e.g., the 24 hours of recordings of a surveillance camera) [3].

More specifically, for a VTS framework to summarize a video content in one short synopsis, it needs to analyze a sequence of key image frames and their corresponding human dialogues over time that are the crucial components for the success of a VTS process. However, a large majority of the current research work [4, 5, 6] mainly focus on image analyses and their pattern learning without considering the importance of human dialogues, as human dialogues can (1) show inter-actions among characters, (2) set the story moods, (3) develop the plot, (4) portray a revelation, and (5) display characters' conflicts [7]. Thus, in this work, we aim to propose the development of an automatic and unified VTS framework and algorithm that mainly emphasize the analyses of human dialogues to generate a short text-based synopsis of a video from its dialogue transcripts. To the best of our knowledge, this kind of work has not been found in the current research literature. Note that for the ease of the writing effectiveness in this paper, we also use "video summary" and "video synopsis" to represent a short text-based summary and synopsis of a video, respectively.

Presently, the state-of-the-art text summarization approaches can be broadly divided into three categories: extractive, abstractive, and hybrid [8]. In extractive summarization methods, for example, Spacy [9], BERT [10, 11], and K-Means [12], they extract the most important sentences from the original input text to form a summary. However, they do not actually paraphrase those sentences close to how a human does to semantically summarize a long piece of text. On the contrary, abstractive summarization methods, for instance, Google T5 [13, 14] and Pegasus [15, 16], paraphrase a long piece of text to generate a short video synopsis like how a human performs, but they occasionally ignore the dialogue specifics in each video scene to produce a concise summary. Apart from that, those methods, Google T5 and Pegasus, have the input text size limitation, which may not generate a summary based upon the entire video dialogue content and its plot. Although some researchers [17] have proposed some specific methods to summarize a long-length video into a text, this method cannot be employed and replicated due to the inaccessibility of the program codes and the shortage of a GPU-cluster infrastructure. Note that our codes in this work will be available to the public after the paper acceptance. In addition, [17] has not conducted and demonstrated a performance evaluation with those existing state-of-the-art approaches stated above. Thus, it cannot be concluded that [17] is the best approach among all the existing methodologies.

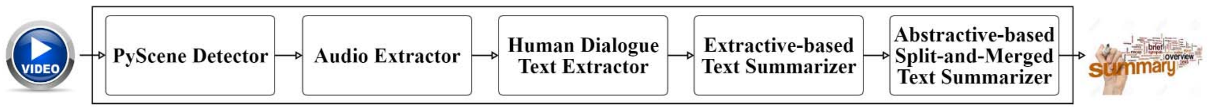
VIDEO
PyScene Detector
Audio Extractor
Human Dialogue Text Extractor
Extractive-based Text Summarizer
Abstractive-based Split-and-Merged Text Summarizer
summary

Fig. 1. Hybrid Video-to-Text Summarization Framework.

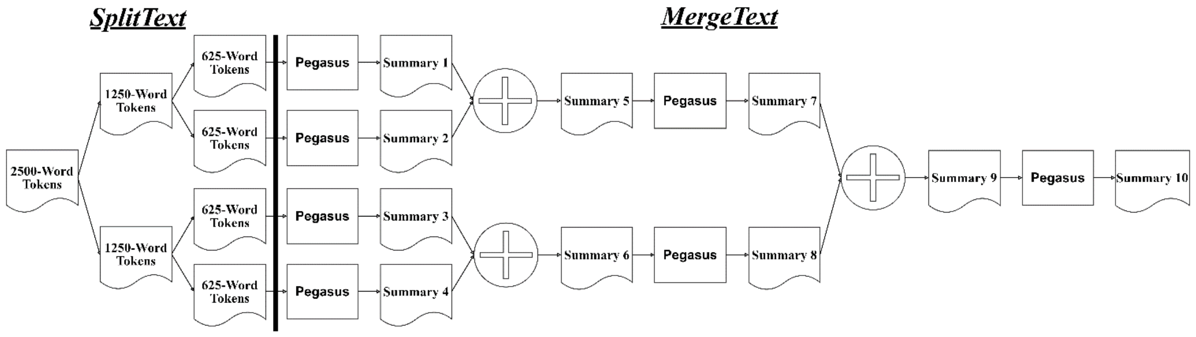

Fig. 2. An Illustrative Example for the ASM Algorithm.

Apparently, both approaches, extractive and abstractive (EA), do not take advantage of each other to generate a concise synopsis for a video. To mitigate the shortcomings of these two existing approaches, various hybrid text summarization techniques [3, 8] have been proposed and developed. These techniques are a combination of different approaches (e.g., supervised, unsupervised, and weakly-supervised) to encounter their mutual shortcomings. In spite of that, these techniques do not consider combining the strengths of both advanced EA approaches together to perform the text summarization. More specifically, to our best knowledge, there is no existing work to merge the above advanced and code-accessible EA methods together in sequence to generate a more concise and semantic video summary. Thus, in this paper, to bridge the above gaps, we propose the development of a hybrid VTS framework on cascading the above advanced and code-accessible EA approaches for supporting viewers' video navigation and understanding. More precisely, the contributions of this paper are three-fold. First, we devise an automated and unified hybrid VTS framework that takes an arbitrary video as an input, generates the text transcripts from its human dialogues, and then summarizes the text transcripts into one short video synopsis. Second, we advance the binary merge-sort approach and expand its use to develop an intuitive and heuristic abstractive-based algorithm, with the time complexity $O(T_L log T_L)$ and the space complexity $O(T_L)$, where T_L is the total number of word tokens on a text, to dynamically and successively split and merge a long piece of text transcripts, which exceeds the input text size limitation of an abstractive model, to generate one final semantic video synopsis. At the end, we test the feasibility of applying this proposed framework and algorithm in conducting the preliminarily experimental evaluations on three different videos, as a pilot study, in genres, contents, and lengths. We show that our approach outperforms and/or levels most of the individual EA methods stated above by 75% in terms of the ROUGE F1-Score measurement.

The rest of the paper is organized as follows. In Section II, we briefly describe our hybrid VTS framework. Section III explains our abstractive-based split-and-merge (ASM) algorithm with an example to show how to tackle the input text size limitation of abstractive models to generate one final short synopsis for a long-duration video. In Section IV, we conduct a preliminarily experimental analysis on our three videos, illustrate the results, and discuss our findings. In Section V, we draw the conclusions and briefly summarize our future work.

II. Hybrid Video-to-Text Summarization Framework

Fig. 1 is our proposed framework that is composed of five main modules, including PyScene Detector (PD), Audio Extractor (AE), Human Dialogue Text Extractor (HDTE), Extractive-based Text Summarizer (ETS), and Abstractive-based Split-and-Merged Text Summarizer (ASMTS). First, a video is passed into the PD that is a Python library [18] to analyze and detect scene changes in the video and then automatically split the video into a number of key content-aware and various-length scene clips, usually in between one second and one minute, which precisely cover the whole video content and plot. After generating those key scene clips, they are passed into the AE in sequence. The AE is the MoviePy library [19] that extracts their audio portions in the waveform audio (WAV) format from those scene clips in the MPEG-4 AVC (MP4) format. The HDTE is the Google Speech-to-Text API [20] that takes the WAV file of each scene clip as the input from the AE and then transcribe the human dialogues into a text transcript. Once the text transcripts are generated from all the scene clips, they are concatenated together in the chronological order to form a complete text transcript based upon the sequential content and plot of all those scene clips. This complete text transcript is then passed into the ETS, e.g., Spacy, BERT, and K-Means, which selects the most important phrases and lines, i.e., the top-K sentences, from it. The ETS then combines all those top-K sentences to create the initial summary. After that, the ASMTS, e.g., Google T5 and Pegasus, takes this initial summary as the input and then uses new phrases and terms, different from the original initial summary but keeping the content and plot the same, to generate one final video synopsis based upon those important sentences, similar to how humans actually perform the text summarization. Note that as each ASMTS has its own input text size limitation, if the size of the initial summary is less than or equal to the threshold limit, e.g., 512-word tokens of Google T5 and 1024-word tokens of Pegasus, this initial summary is passed into one ASMTS. However, if the size is more than the threshold limit, this initial summary is then passed into the ASM algorithm to conduct the split-and-merged process successively to generate one final summary. The ASM algorithm is described and explained in Section III in more details.

III. Abstractive-based Split-and-Merged Algorithm

Table I shows the step-by-step pseudocode of the ASM algorithm in its text summarizer.

TABLE I. ABSTRACTIVE-BASED SPLIT-AND-MERGED ALGORITHM

```
1. Input:
$T_O$: A human-readable and understandable original text constructed by a sequence of words, characters, or sub words, called tokens.
$A$: An abstractive text summarization model.
$N$: A predefined threshold limit of input word tokens of $A$.
2. Output:
$T_S$: A human-readable and understandable text summary constructed by a sequence of word tokens.
3. Initialization:
$T_S$ = "" # Set the initial text summary as an empty string.
C = [] # Set the corpus C as an empty word-token list.
$T_L$ = 0 # Set the number of tokens in C as a zero value.
firstHalfT = secondHalfT = "" # Set the initial first-half and second-half texts as an empty string, respectively.
4. Processing:
STEP 1: Define a function, readTokenize($T_O$), to read and tokenize $T_O$ and then store each word token of $T_O$ in C.
readTokenize($T_O$):
    import pandas as pd
    text = pd.read_csv($T_O$, header=None)
    for row in text.values do tokens = row[0].split(" ");
        for token in tokens do C.append(token);
        endfor
    endfor
    return C;
STEP 2: Define a function, splitText($T_O$), to divide $T_O$ evenly into two half if $T_L > N$ or else pass $T_O$ as the input text to $A$ to generate one text summary, i.e., $T_S = A(T_O)$, if $T_L \leq N$.
splitText($T_O$): # Get the num# of word tokens ($T_L$) in C, i.e., $T_L$ = length(C).
    $T_L$ = length(readTokenize($T_O$));
    if ($T_L > N$) then
        firstHalfT = $\frac{T_O}{2}$; splitText(firstHalfT);
        secondHalfT = $T_O - \frac{T_O}{2}$; splitText(secondHalfT);
        mergeText(firstHalfT, secondHalfT, $T_O$);
    else
        $T_S = A(T_O)$
    endif
STEP 3: Define a function, mergeText(firstHalfT, secondHalfT, $T_O$), to concatenate the first-half and/or second-half text summaries in sequence and then store it in $T_S$.
mergeText(firstHalfT, secondHalfT, $T_O$):
    if (firstHalfT != "" AND secondHalfT != "") then $T_S$ = $A$(firstHalfT) +
                                                        $A$(secondHalfT);
    elseif (firstHalfT != "") then $T_S$ = $A$(firstHalfT);
    elseif (secondHalfT != "") then $T_S$ = $A$(secondHalfT);
    else $T_S$ = "";
    endif
    $T_O = T_S$
STEP 4: Return $T_S$
```

Let us consider an example to illustrate how our ASM algorithm can construct a text summary if its input text size exceeds the threshold limit of an abstractive model. In this example, we use the Pegasus text summarizer (***A***) and its input text size limit (***N***) is 1024-word tokens for an illustration. We also assume that the size of the input text T_O generated from an extractive model is 2500-word tokens (T_L). Using the ASM algorithm step by step, we can generate one final video synopsis. Here is how the process of the algorithm works shown in Fig. 2 on the previous page. In STEP 1, the *readTokenize*(T_O) function generates a corpus C that contains 2500-word tokens. In STEP 2, T_L stores the size of T_O, i.e., 2500. As T_L is greater than ***N***, the input text T_O is divided into two portions, i.e., firstHalfT = 1250 and secondHalfT = 1250, respectively by the *splitText* function. As the size of both text portions are still greater than ***N***, they are further split into two halves (i.e., 625-word tokens) by the *splitText* function. Due to the 625-word tokens not exceeding the threshold limit of ***A*** anymore, each piece of those text transcripts is then passed into ***A*** to generate a partial summary (i.e., Summary 1, 2, 3, and 4) by calling the *mergeText* function in STEP 3. The *mergeText* function then merges those partial summaries (e.g., Summary 1 and 2) together in the chronological order to form a concatenate summary (e.g., Summary 5). Note that the size of this concatenate summary is often much smaller than that of the threshold limit of an abstractive model, so the *splitText* process is not needed. The *mergeText* process is then successively repeated for Summary 3 and 4 to generate Summary 6. After that, Summary 5 and 6 are passed into ***A*** again to generate Summary 7 and 8, as ***A*** can form a new semantic summary from these two separate, different pieces of text synopses. This merging process keeps running until there is no more partial summary left. Finally, the video synopsis (i.e., Summary 10) is outputted in STEP 4. As we advance the binary merge-sort approach in our ASM algorithm, the time complexity and the space complexity are still $O(T_L log T_L)$ and $O(T_L)$, respectively, where T_L is the total number of word tokens on a text.

IV. Experimental Results and Discussion

The datasets for our preliminarily experimental analysis on our three videos include SuperHero (Genre: Fantasy and Duration: 18 min), Nuclear Family (Genre: Drama and Duration: 28 min) and Shooters (Genre: Drama and Duration: 41 min), which can be categorized into short, medium and long videos, collected by NIST [21]. All these three videos are tested in three extractive models (i.e., Spacy, BERT, and K-means), two abstractive models (i.e., Google T5 and Pegasus), and six hybrid-EA models (i.e., Spacy → Google T5, BERT → Google T5, K-means → Google T5, Spacy → Pegasus, BERT → Pegasus, and K-means → Pegasus) respectively. To demonstrate the effectiveness of our framework and algorithm, we humanly create the reference summary (R) for each video and then evaluate the performances among all the models' generated summaries (T_S) based upon the ROUGE F1-Score, which is computed by the ROUGE recall and precision, i.e., $\frac{2*(ROUGE\ recall*ROUGE\ precision)}{ROUGE\ recall+ROUGE\ precision}$, where the ROUGE recall is the ratio of the number of *n*-grams in R that appear also in T_S over the number of *n*-grams in R and the ROUGE precision is the ratio of the number of *n*-grams in T_S that appear also in R over the number of *n*-grams in T_S. Note that as our focus is to evaluate the quality of each generated video summary on the same ground among all its respective models, we output every video synopsis in a consistent length. The 72 performance results are shown in Table II, III, and IV. For SuperHero, our approach has 16 WINs, 4 DRAWs, and 4 LOSSES; for Nuclear Family, our approach has 14 WINs, 2 DRAWs, and 8 LOSSES; and for the Shooters video, we have 12 WINS, 6 DRAWs, and 6 LOSSES. That is, there are 42 WINS (58.33%) and 12 DRAWS (16.67%) using our hybrid-EA approach that is 75% better than and/or the same as just using individual EA models alone. Note that ± means 10% score tolerance. Even if there are still some "LOSS" scenarios, the ROUGH F1-Scores among them are very miniscule.

TABLE II. MODELS' PERFORMANCE METRICS FOR SUPERHERO

MODEL (ROUGH F1-SCORE)	Spacy (±0.56)	BERT (±0.16)	K-Means (±0.24)	Google T5 (±0.26)
Spacy → Google T5 (±0.56)	DRAW	WIN	WIN	WIN
BERT → Google T5 (±0.26)	LOSS	WIN	WIN	DRAW
K-means → Google T5 (±0.26)	LOSS	WIN	WIN	DRAW
Model (ROUGH F1-Score)	**Spacy (±0.08)**	**BERT (±0.07)**	**K-Means (±0.18)**	**Pegasus (±0.21)**
Spacy → Pegasus (±0.24)	WIN	WIN	WIN	WIN
BERT → Pegasus (±0.11)	WIN	WIN	LOSS	LOSS
K-means → Pegasus (±0.21)	WIN	WIN	WIN	DRAW

TABLE III. MODELS' PERFORMANCE METRICS FOR NUCLEAR FAMILY

Model (ROUGH F1-Score)	Spacy (±0.15)	BERT (±0.08)	K-Means (±0.07)	Google T5 (±0.28)
Spacy → Google T5 (±0.16)	WIN	WIN	WIN	LOSS
BERT → Google T5 (±0.17)	WIN	WIN	WIN	LOSS
K-means → Google T5 (±0.12)	LOSS	WIN	WIN	LOSS
Model (ROUGH F1-Score)	**Spacy (±0.18)**	**BERT (±0.00)**	**K-Means (±0.07)**	**Pegasus (±0.03)**
Spacy → Pegasus (±0.40)	WIN	WIN	WIN	WIN
BERT → Pegasus (±0.03)	LOSS	WIN	LOSS	DRAW
K-means → Pegasus (±0.03)	LOSS	WIN	LOSS	DRAW

TABLE IV. MODELS' PERFORMANCE METRICS FOR SHOOTERS

Model (ROUGH F1-Score)	Spacy (±0.33)	BERT (±0.09)	K-Means (±0.12)	Google T5 (±0.25)
Spacy → Google T5 (±0.33)	DRAW	WIN	WIN	WIN
BERT → Google T5 (±0.25)	LOSS	WIN	WIN	DRAW
K-means → Google T5 (±0.15)	LOSS	WIN	WIN	LOSS
Model (ROUGH F1-Score)	**Spacy (±0.17)**	**BERT (±0.04)**	**K-Means (±0.09)**	**Pegasus (±0.04)**
Spacy → Pegasus (±0.17)	DRAW	WIN	WIN	WIN
BERT → Pegasus (±0.04)	LOSS	DRAW	LOSS	DRAW
K-means → Pegasus (±0.09)	LOSS	WIN	DRAW	WIN

V. CONCLUSION AND FUTURE WORK

In this paper, we propose the development of a hybrid VTS framework on cascading the above advanced and code-accessible EA approaches for supporting viewers' video navigation and understanding. More precisely, the contributions of this paper are as follows. First, we devise an automated and unified hybrid VTS framework that takes an arbitrary video as an input, generates the text transcripts from its human dialogues, and then summarizes the text transcripts into one short video synopsis. Second, we advance the binary merge-sort approach and expand its use to develop an intuitive and heuristic abstractive-based algorithm, with the time complexity $O(T_L log T_L)$ and the space complexity $O(T_L)$, where T_L is the total number of word tokens on a text, to dynamically and successively split and merge a long piece of text transcripts, which exceeds the input text size limitation of an abstractive model, to generate one final semantic video synopsis. At the end, we test the feasibility of applying this proposed framework and algorithm in conducting the preliminarily experimental evaluations on three different videos, as a pilot study, in genres, contents, and lengths. We show that our approach outperforms and/or levels most of the individual EA methods stated above by 75% in terms of the ROUGE F1-Score measurement. However, there is still a lack of many important questions, for example, how the video background sound and the key image frames can be integrated into the human dialogues to generate an unified text transcript for summarization.

REFERENCES

[1] Backlinko, 2022. How Many People Use YouTube in 2022? Retrieved from https://backlinko.com/youtube-users.

[2] Global Media Insight, 2022. YouTube User Statistics 2022. Retrieved from https://www.globalmediainsight.com/blog/youtube-users-statistics/.

[3] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I. Metsai, Vasileios Mezaris, and Ioannis Patras, 2021. Video Summarization Using Deep Neural Networks: A Survey. Available: https://arxiv.org/pdf/2101.06072.pdf.

[4] Mrigank Rochan, Linwei Ye, and Yang Wang, 2018. Video Summarization Using Fully Convolutional Sequence Networks. In Proceedings of the European Conference on Computer Vision, pp. 347-363.

[5] Atsushi Kanehira, Luc Van Gool, Yoshitaka Ushiku, and Tatsuya Harada, 2018. Viewpoint-Aware Video Summarization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7435-7444.

[6] Tanveer Hussain, Khan Muhammad, Weiping Ding, Jaime Lloret, Sung Wook Baik, Victor Hugo C. de Albuquerque, 2021. A Comprehensive Survey of Multi-View Video Summarization. Pattern Recognition, Volume 109, January 2021, 107567. DOI: https://doi.org/10.1016/j.patcog.2020.107567.

[7] Joslyn Chase, 2019. Why Is Dialogue Important? 7 Roles Dialogue Plays in a Story. Retrieved from https://thewritepractice.com/why-is-dialogue-important/.

[8] Ishitva Awasthi, Kuntal Gupta, Prabjot Singh Bhogal, Sahejpreet Singh Anand, and Piyush Kumari Soni, 2021. Natural Language Processing (NLP) based Text Summarization – A Survey. In Proceedings of the International Conference on Inventive Computation Technologies, pp. 1310-1317.

[9] Kamal Khumar, 2020. Text Summarization Using spaCy. Retrieved from https://medium.com/analytics-vidhya/text-summarization-using-spacy-ca4867c6b744.

[10] Chris Tran, 2020. Extractive Summarization with BERT. Retrieved from https://chriskhanhtran.github.io/posts/extractive-summarization-with-bert/.

[11] Yang Liu and Mirella Lapata, 2019. Text Summarization with Pretrained Encoders. Available: https://arxiv.org/abs/1908.08345.

[12] Akanksha Gupta, 2020. Understanding Text Summarization Using K-means Clustering. Retrieved from https://medium.com/@akankshagupta371/understanding-text-summarization-using-k-means-clustering-6487d5d37255.

[13] Synced, 2019. Google T5 Explores the Limits of Transfer Learning. Retrieved from https://medium.com/syncedreview/google-t5-explores-the-limits-of-transfer-learning-a87afbf2615b.

[14] Ramsri Goutham, 2020. Simple Abstractive Text Summarization with Pretrained T5 - Text-To-Text Transfer Transformer. Retrieved from https://towardsdatascience.com/simple-abstractive-text-summarization-with-pretrained-t5-text-to-text-transfer-transformer-10f6d602c426.

[15] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu, 2020. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. Available: https://arxiv.org/abs/1912.08777.

[16] Jiahao Weng, 2021. How to Perform Abstractive Summarization with PEGASUS. Retrieved from https://towardsdatascience.com/how-to-perform-abstractive-summarization-with-pegasus-3dd74e48bafb.

[17] Shagan Sah, Sourabh Kulhare, Allison Gray, Subhashini Venugopalan, Emily Prud'Hommeaux and Raymond Ptucha, 2017. Semantic Text Summarization of Long Videos. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 989-997.

[18] Brandon Castellano, 2021. PySceneDetect. Retrieved from https://pyscenedetect.readthedocs.io/en/latest/examples/usage-example/.

[19] Zulko, 2017. MoviePy. Retrieved from https://zulko.github.io/moviepy/.

[20] Laurent Picard, 2021. Using the Speech-to-Text API with Python. Retrieved from https://codelabs.developers.google.com/codelabs/cloud-speech-text-python3#0.

[21] Keith Curtis, George Awad, Shahzad Rajput, and Ian Soboroff, 2020. HLVU: A New Challenge to Test Deep Understanding of Movies the Way Humans do. Available: https://arxiv.org/abs/2005.00463.