CatchLive: Real-time Summarization of Live Streams with Stream Content and Interaction Data

Saelyne Yang*
saelyne@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Jisu Yim
yimjisu99@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Juho Kim
juhokim@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Hijung Valentina Shin
vshin@adobe.com
Adobe Research, USA
Cambridge, MA, USA

Figure 1: Overview of CatchLive: (a) The timeline is segmented into high-level sections. (b) Hovering over a certain section shows brief information about the section. (c) Highlight moments within each section are color-coded in the timeline. Upon clicking, the moment is scrolled into view in the `Highlights` tab. (d) Users can see highlight moments with snapshots and transcripts. (e) Users can also see chat messages near the highlight moment by clicking on the chat icon. (f) The indicator shows where the highlight is relative to the entire stream.

ABSTRACT

Live streams usually last several hours with many viewers joining in the middle. Viewers who join in the middle often want to understand what has happened in the stream. However, catching up with the earlier parts is challenging because it is difficult to know which parts are important in the long, unedited stream while also keeping up with the ongoing stream. We present CatchLive, a system that provides a real-time summary of ongoing live streams by utilizing both the stream content and user interaction data. CatchLive provides viewers with an overview of the stream along with summaries of highlight moments with multiple levels of detail in a readable format. Results from deployments of three streams with 67 viewers show that CatchLive helps viewers grasp the overview of the stream, identify important moments, and stay engaged. Our findings provide insights into designing summarizations of live streams reflecting their characteristics.

*This work was done while the author was an intern at Adobe Research.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3517461

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

live streaming, live summarization, video summarization, user interaction data

ACM Reference Format:

Saelyne Yang, Jisu Yim, Juho Kim, and Hijung Valentina Shin. 2022. CatchLive: Real-time Summarization of Live Streams with Stream Content and Interaction Data. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3491102.3517461

1 INTRODUCTION

Live streams attract viewers with their synchronous and interactive watching experience. The real-time interaction between the streamer and the viewers allows effective communication and creates a sense of connection [10]. Due to such unique benefits, more and more types of content—such as gaming, drawing, programming, and language learning [7, 10, 12, 15, 16, 24, 36, 41, 46]—are being live streamed. Compared to pre-recorded and edited videos, live streams tend to be longer, lasting several hours [7, 10, 15, 36]. The long and real-time nature of live streams means that many viewers join in the middle of the streams.

Viewers who join in the middle of an ongoing stream may get lost [3, 28]. Since they have little context about the previous content, they may fail to understand what the streamer is talking about: for instance, they might not realize which step of the recipe the cooking stream is showing or which mission the gamer is currently playing in a game-playing stream. Viewers may also feel disengaged when they are unable to understand the chat messages from other viewers.

Catching up with the previous parts of the stream is challenging because it is difficult to navigate through long, unedited videos [9, 28]. Watching the previous parts is even more challenging when viewers are trying to keep up with the ongoing stream at the same time. To identify the current practices and challenges of catching up in live streams, we conducted a series of interviews with viewers and observed them joining ongoing live streams of various domains. Although there are differences in needs depending on the live stream genre, we identified three common high-level needs across them: First, viewers want to get a high-level overview of the previously covered content, a guide to what was covered when. Second, viewers want different levels of detail about the previous parts depending on the topics or what is being covered currently. Some viewers want more detail on certain topics, while others want to see an entire history of the live stream in full detail when the current stream gets less intense. Lastly, viewers want to absorb all this information with minimal interruption to the current stream.

To address these challenges, we present CatchLive, a system that provides a real-time summarization of an ongoing live stream (Figure 1). CatchLive provides two types of summaries: (1) an overview of the stream that segments the stream into multiple high-level sections, and (2) a summary of highlights for each section. The highlights are presented in multiple levels of detail, and presented in a familiar chat format that is easy to read with minimal interruption.

We developed two core algorithms for the two types of summaries. First, we propose a real-time, online segmentation algorithm that partitions the stream into meaningful sections as the stream progresses. Second, we identify highlight moments for each of the sections by adopting an existing peak-detection algorithm [1]. These algorithms leverage both the stream content and the user interaction data. From the stream content, we extract visual changes in the screen, transitional cues, keywords, and breaks from the transcript. From user interaction data, we utilize the chat dynamics, keywords from the chat, "likes" on chat messages, and sharing of the stream content through snapshots (similar to Snapstream [51]). These user interactions serve dual purposes: (1) to increase viewer engagement through active participation, and (2) to provide an estimate of viewer engagement as a useful signal for summarizations.

We evaluated our online segmentation algorithm with seven streams from different genres. Results show that our algorithm can produce comparable results to ground-truth segmentations provided by either the streamer or the viewers, but different genres of streams require different weights on each type of stream or interaction data, depending on the characteristics of the stream.

We deployed CatchLive in three different genres of live streams: information sharing, cooking, and gaming streams. We observed that CatchLive helped participants grasp the overview of the stream, identify important moments, and stay engaged. We also learned that participants use summarizations in different ways for different genres of streams. These observations provided insights into how to design live stream summarizations that reflect each stream's characteristics.

Finally, we evaluated CatchLive in more detail with the gaming stream through a comparative study with a baseline interface, and observed that viewers using CatchLive could engage more actively in the stream, compared to viewers who did not use CatchLive.

The primary contributions of this paper are:

- Insights into the unique challenges that viewers face when they join ongoing live streams.
- A system that provides an overview summary and highlights of a live stream in real-time for viewers who join in the middle.
- Results from live deployments that show how viewers use and benefit from live stream summarizations.
- Design implications on live stream summarizations that reflect the stream's characteristics.

2 RELATED WORK

Our paper presents an interactive system that summarizes various types of live streaming. We build upon three areas of research: video summarization and highlight generation, live stream summarization, and live streaming interaction.

2.1 Video Summarization and Highlight Generation Techniques

Video summarization has been a subject of rich prior work. Many approaches use machine learning techniques to create highlights and summaries from videos by utilizing the content itself, such as visual or audio elements of the videos [4, 5, 14, 39], or their metadata, such as the length and the title of the video [43, 50]. For example, Xiong et al. [50] presented a technique that learns to detect highlights in videos by training on shorter, user-generated video segments. Similarly, TVSum [43] finds visually important

moments in videos by selecting shots that are most relevant to the video title.

A different approach is to apply crowdsourcing techniques to generate summaries. Several learnersourcing approaches are proposed to elicit learners' natural motivations and interactions to provide meaningful content of the video. Lecturescape [22] identifies points of importance and confusion in lecture videos from user interaction data. ToolScape [23] suggests a crowdsourcing workflow to produce step-by-step information in how-to videos. Similarly, ConceptScape [26] allows viewers to collaboratively create a concept map of lecture videos. Other systems employ a mixed-initiative approach. For example, VideoDigests [34] combines algorithmic segmentation with crowdsourced section summaries [34], and ElasticPlay [21] allows users to interactively control the length of summarized videos. We use an algorithmic approach that leverages user interaction data to segment the video and extract highlights in real-time as the live stream is happening.

2.2 Live Stream Summarization Techniques

Live streams are unedited displays of ongoing events. A number of techniques have been proposed to summarize live events by analyzing messages posted on microblogging services [19, 30, 38]. In the context of live streaming a video, several methods focused on summarizing live streams after the fact by creating a table of contents by temporally segmenting the video [9], or generating highlights of the video by leveraging user chat messages [11, 18] or audio-visual analysis [32].

A line of research has investigated generating summaries of live streams in *real-time*. Miller et al. [33] proposed a chat design that allows users to see important messages among the flood of chat messages. It controls the messages shown to users by grouping viewers and selecting salient messages through upvoting. Targeting more specific genres of streams, Helpstone [25] allows viewers to check previous game-playing information such as actions done or cards used in live streams of the game Hearthstone. The most similar work to ours is StreamWiki [28], which allows viewers and moderators to generate a summary document of knowledge-sharing live streams in real-time by writing summaries, adding comments, and voting on useful comments.

Similar to previous approaches, our work also leverages voting on chat messages not only to identify important messages but also to identify highlight moments of the stream. In addition to identifying highlights, our system provides an overview of the stream. While previous work focused on specific genres of live streams (e.g., game [25] or knowledge-sharing [28]) to provide an overview, we propose a general-purpose system that can be adapted to different genres of live streams. To do so, we propose an automatic algorithm that leverages viewers' natural interactions to generate the summary in real-time.

2.3 Improving Live Streaming Interaction

A key aspect of live streams is the direct and synchronous interaction between viewers and streamers, which opens up vast possibilities for diverse forms of interaction. In their early work, Isaac et al. [20] studied the challenges of audience interaction in live broadcast presentations. More recently, Pfeil et al. [37] surveyed research articles about live-streaming telepresence and argued that it is important to understand the dynamics of the relationship between all parties involved. Several systems have been proposed to facilitate interaction between the streamer and viewers in various genres of live streams such as visual art [29, 51], video games [12, 24, 25] and online learning [2, 7, 15, 17]. These systems introduced multi-modal tools to improve live stream interaction such as sharing snapshots for visual arts [51] or text, audio, video, image, and stickers for language learning [2]. In the context of live streamed audience participation games (APGs), viewers can decide how scenes should proceed by voting [24] or suggesting hints to streamers by drawing on the video stream [25]. Since understanding the context is important for viewers to participate in APGs, a game tutorial is provided in the audience participation interface [12] or game-playing information such as actions performed or cards used is shown to late-joining viewers, allowing them to get an overview of the current match [25]. Similarly, our work aims to help viewers understand the content of the ongoing stream so that they can actively participate in the stream.

3 FORMATIVE STUDY

We conducted a series of semi-structured interviews to learn about what viewers do and what challenges they encounter when they join a live stream in the middle. We recruited 10 viewers from our academic institution, who watch live streams regularly at least once a week (Table 1). We asked about their previous experiences of joining a stream in the middle and trying to catch up. Additionally, we conducted a think-aloud study with four of the participants (P_1 – P_4), where they watched two different genres of live streams (instructional and entertainment) after joining in the middle. We observed their behaviors and recorded their thought processes. Below we summarize our key findings.

3.1 Current Practices and Challenges of Catching up in Live Streams

People join ongoing live streams for various reasons. For instance, they join an interesting stream that they come across, without knowing exactly when the stream started. P_2 mentioned that she follows a star on Instagram but she does not always get a notification when the live talks start. So, she usually discovers them later and joins in the middle. For scheduled live streams such as online classes or sports events, viewers often miss the earlier parts because of schedule conflicts (P_3, P_4). We observed three main ways that viewers try to catch up with an ongoing live stream:

(1) Reading chat messages: Upon joining the stream, all of the think-aloud study participants immediately scanned through the chat messages. Chat messages can be a helpful source of information for instructional live streams. P_4 said that when she joins an instructional live stream, she usually checks the chat first to see if there are any important announcements or notes she missed. However, chat messages are less helpful in entertainment streams (P_3, P_4). P_4 mentioned that chats are not as informative because there are too many emotional reactions, which make it difficult to identify informational messages relevant to the content [28, 46].

Participants	Interested stream genres	Frequency	Streams for the think-aloud study
P_1	Lecture	Weekly	Tutorial, Entertainment show
P_2	Yoga, Casual talks, Drawing	Daily	Tutorial, Entertainment show
P_3	Game, Sports	Weekly	Tutorial, Game
P_4	Lecture	Weekly	Tutorial, Entertainment show
P_5	Game	3 times / week	
P_6	Game, Casual talks, Sports	Weekly	
P_7	Yoga	Weekly	
P_8	Game, Casual talks	3-4 times / week	
P_9	Game	4 times / week	
P_{10}	Game	2-3 times / week	

Table 1: The types of live stream each participant watches, how frequently they watch live streams, and the streams they joined for the think-aloud study.

Similarly, P_3 mentioned that there are simply too many chat messages and that they are hard to understand, especially when they contain slang words from a particular community.

(2) Asking others in the chat: Some viewers ask in the chat when there is something they do not understand. P_5 mentioned that in such cases, she can grasp the context easily because other viewers summarize what had happened before. However, none of the think-aloud study participants asked in the chat. P_1, P_{3-4} were hesitant to ask for help in the chat because they were afraid to interrupt the current stream, similar to previous findings [28]. Sometimes viewers ask a question, but the question is left unanswered, or it gets buried in other messages.

(3) Rewinding the video: Perhaps surprisingly, only one (P_4) out of four think-aloud study participants rewound the video, which however did not provide any useful information that helped in understanding the ongoing stream. When participants were asked why they did not rewind the video, they said they do not want to miss the current parts and that it is hard to locate the important or interesting parts, especially in long streams (P_1, P_2, P_3). However, P_2 mentioned that in instructional streams, if the goal is to only complete the tutorial content, she would rewind the video and follow the stream asynchronously at the expense of missing out on real-time interaction.

3.2 Needs for Catching up in Live Streams

We asked participants what information they would find useful when joining a live stream in the middle. From the analysis of the interviews and observations, we identified emerging themes of needs across two types of live streams: Instructional and Entertaining live streams. Although there were subtle differences between the two types, the high-level needs were shared.

(1) Overview of the stream: The first thing that viewers look for is an overview of the stream, such as what was covered when. For live streams that deliver instructional content, viewers want to know a step-by-step, sequential overview of the stream. P8 said she would like to get the step information in yoga live streams so that she can know whether the step she wants to see has already been covered. P4 said that she wants to know which topic has been covered in an educational live stream so that she can better understand the current part.

Similarly, viewers of entertainment live streams such as games also want to know the overall flow of the stream (P8, P9, P10). However, they are more concerned about the list of subtopics covered in the stream than the sequence or structure of those content. Unlike instructional streams, entertaining streams are unplanned or deviate from the original plan. P3 said, for this reason, a title or description provided is not sufficient to understand the overall stream. Knowing which subtopics were actually covered in the stream can help viewers better engage with the stream.

(2) Different levels of detail depending on the content and the current context: Once viewers get the overview of the stream, they seek detailed information about parts they are interested in. In instructional streams, viewers look for moments where an important concept or terminology is introduced (P2, P4). The amount of detail viewers want differs depending on the content. P1 mentioned that, in a lecture stream, she wants to watch in detail the beginning section, where the instructor makes announcements, but not the other parts. P2 also said that she prefers to skip the parts that are not important.

For entertaining live streams, viewers like to see highlight moments that other viewers enjoyed, to share the same experience and to be engaged with the community (P2, P5, P6, P8, P9). P5 mentioned that she would like to see highlights of the previous parts not only as soon as she joins but also when the stream gets boring or less intense. This implies that the amount of detail needed differs depending on what is being covered in the current moment as well.

(3) Catching up with minimal interruption to the current stream: As real-time interaction is an important factor in live streams, many viewers value watching the current moment of the stream; catching up with the previous parts could distract them from the current stream. P_3 commented that when watching sports games, he tries to catch up through an online community first before joining the stream so that he can fully enjoy the real-time stream and not miss out on key events. P_4 mentioned that she does not like to rewind the stream unless it is necessary because it makes her miss the current content. Minimizing interruption to the current stream is an important factor when providing summaries in real-time.

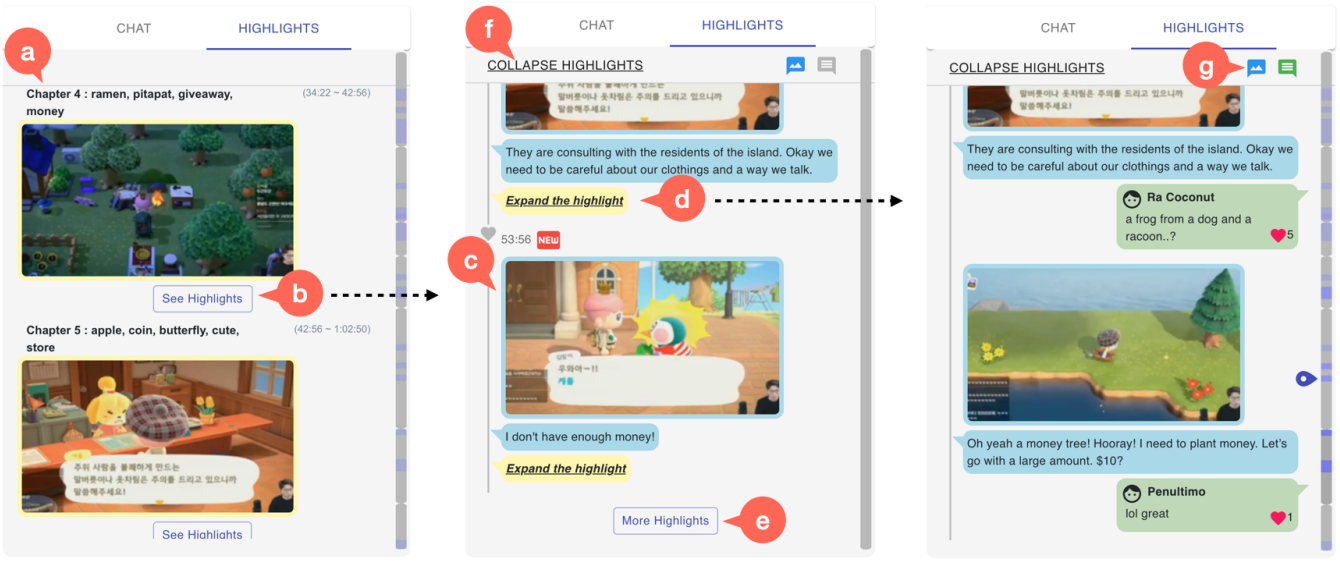


Figure 2: (left) The initial view of the `Highlights` tab. (a) It shows the timeline information in a vertical view, with a representative screenshot and keywords for each section. Once a user is interested in a certain section, (b) they can see highlights of the section by clicking on the `See Highlights` button. (middle) Once a user clicks to see the highlights of a section, (c) the top two highlight moments are revealed with a representative snapshot and a sentence of the transcript. Users can either choose to see (d) more details about a particular highlight or (e) other highlights from that section. (f) `Collapse Highlights` brings the user back to the initial view (left). Once a user clicks to see more details about a particular highlight, (right) the rest of the snapshots and transcripts are shown to users. Users can also see other viewers' chat messages of the moment by (g) clicking on the chat icon.

3.3 Design Goals

We aimed to design a system that supports the high-level needs applicable to general genres of streams. We wanted to first understand how such a system could facilitate the viewing experience for different stream genres, and then further identify detailed needs per genre. Based on the observations, we formed the following three design goals for our system.

- G1: Provide an overall structure of the stream so that viewers can understand the current content with the overall context in mind.
- G2: Provide summaries with multiple levels of detail so that viewers can choose how much detail they want to see depending on their needs.
- G3: Help viewers catch up on previous parts with minimal interruption to the current stream.

4 CATCHLIVE INTERFACE

Based on the three design goals, we designed CatchLive, a system that provides real-time summarization of live streams to help viewers who join in the middle catch up on content they missed (Figure 1). CatchLive presents an overall structure of the stream by segmenting the live stream into high-level sections (G1) (Figure 1a–b). For each section, it also provides a summary of highlight moments (Figure 1c–f). The highlights are accessible on-demand, such that viewers can see more or less detail depending on their needs (G2). We present the highlights in a readable chat format so that users can skim them with minimal interruption to the current stream (G3). This section describes the CatchLive interface in detail, while the next section explains the algorithms behind it.

4.1 Timeline Information of the Stream

When viewers join an ongoing stream in CatchLive, they see a timeline that segments the stream into high-level sections up to the current point. For example, in Figure 1a, the timeline shows seven different sections. Each section represents a coherent segment of the stream such as a single step in a tutorial or a subtopic that was covered in a talk. Hovering over a section in the timeline reveals a representative snapshot and several keywords from the section, with time information (Figure 1b). By skimming over the sections in the timeline, users can quickly grasp the overall structure of the previous content in the stream. Users can see the same section information at once in a vertical view in the `Highlights` tab, which we describe in the next section (Figure 2a).

4.2 Section Highlights with Multiple Levels of Detail

Each section in the timeline contains several highlights, color-coded in light purple (Figure 1c). If a user is interested in finding out more about those moments, they can view more detail in the `Highlights` tab (Figure 1d). Initially, the `Highlights` tab is identical to the horizontal timeline but in the vertical form: it shows a segmented

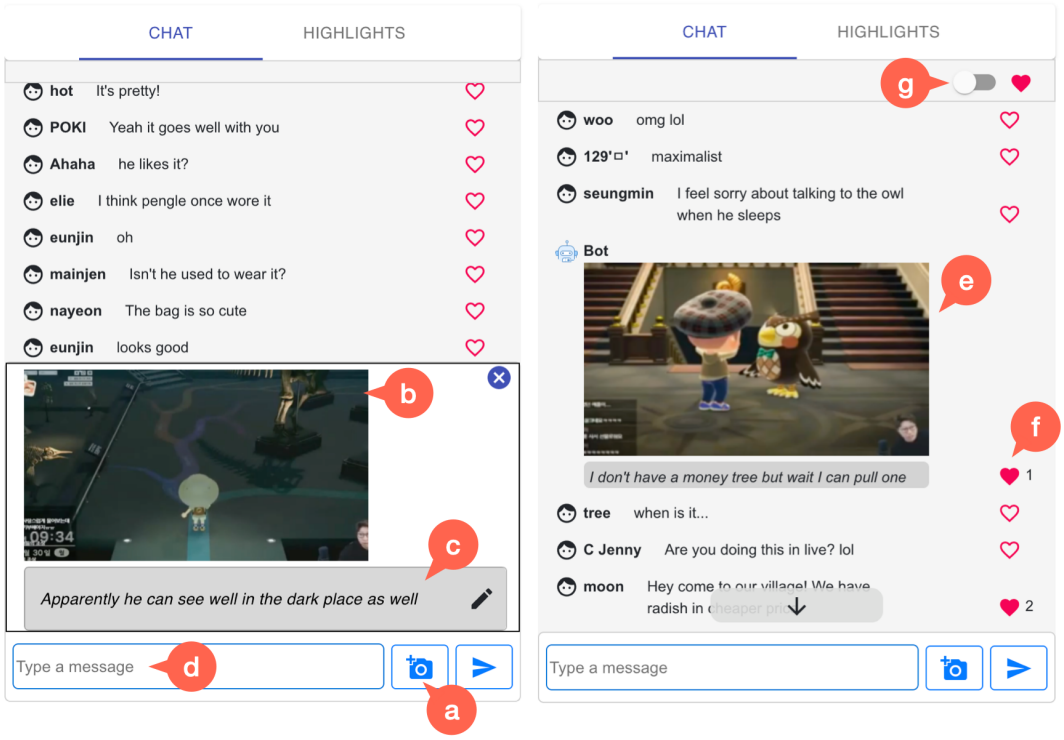


Figure 3: (a) Users can click on the camera button to capture the current content in the stream. Then, (b) a screenshot of the moment and the most recent sentence from the transcript are captured. (c) Users can edit the transcript and/or (d) add their own message, and (e) share it in the chat. (f) Users can "like" others' messages or snapshots and (g) filter the liked items.

timeline with a representative snapshot and keywords for each section (Figure 2a). Clicking on `See Highlights` reveals the top two highlight moments from that section. Each highlight moment is represented by a single snapshot and a sentence of the transcript (Figure 2c). Users can choose to see either more detail about a particular highlight moment (`Expand the highlight`), or other highlights from that section (`More Highlights`) (Figure 2d–e). `Expand the highlight` reveals additional content from the highlight moment, including the rest of the snapshots and transcripts (Figure 2-right). Users can also see other viewers' chat messages of the highlight moments by clicking on the chat icon (Figure 2g). `More Highlights` reveals the next most important highlight moments from the section. We visualized the highlights like chat messages between the streamer (snapshots and transcripts) and viewers (chats) based on our observation that people rely on chat messages as a quick way to review previous material with minimal disruption.

4.3 Annotating and Sharing Moments from the Stream

In addition to conventional chat messaging, CatchLive enables two more ways for users to annotate and share interesting content in the stream. First, similar to Snapstream [51], users can take a snapshot of the stream and share it in the chat. When the user takes a snapshot by clicking on the `camera` button, a screenshot of the moment and the most recent sentence from the transcript is captured (Figure 3-left). Users can edit the transcript or add their own message. Second, users can "like" other viewers' or streamers' messages (Figure 3-right).

We expect that while such interactions can help users better engage with the stream [51], they also provide meaningful signals for summarization. For example, a burst of shared snapshots or transcript sentences may indicate an important or interesting moment in the stream. A chat message with multiple likes may imply that the message accurately describes the current stream or effectively captures viewer sentiment at that moment. Section 5 describes in detail how we leverage these user interaction data in the segmentation and highlight detection algorithms.

4.4 Implementation

We implemented CatchLive using React.js, HTML, and CSS for the front-end web interface, and Node.js and Firebase for the back-end server. CatchLive embeds the streams of YouTube Live videos through its API [53]. To enable sharing the stream content in the chat, we retrieve the video manifest URL using the youtube-dl library [54] and take a screenshot using FFmpeg [8]. We use the MDN Web Speech API [31] for real-time transcription of streams. The client and the server communicate with each other using socket.io [42] to transmit data such as timeline and highlights information.

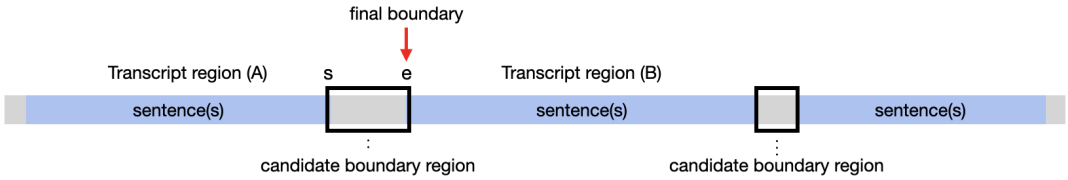


Figure 4: We consider an interval between two adjacent transcript regions as a candidate boundary region. Once selected, we take the endpoint of the region as the final boundary.

5 ALGORITHMS

We develop two core algorithms for CatchLive: (1) a real-time, online segmentation algorithm that partitions the stream into meaningful sections as the stream progresses, and (2) a highlights detection algorithm that extracts important moments from each section.

5.1 Online Segmentation Algorithm

We propose an online segmentation algorithm that segments a stream into sections that cover similar topics in real-time. Our algorithm first identifies all section boundary candidates within a time frame, scores how probable each candidate is to be a section boundary, and selects the highest score boundary. The process iterates once the current portion (i.e., from the last identified boundary to the current point) exceeds a certain length.

We set the minimum and maximum length of one segment as 5 and 20 minutes because segments that are either too short or too long can be less useful. The length of a segment also reflects the distribution of segment lengths of real live streams used in the preliminary evaluation (Table 7, average: 14.3 min) and creative streams (average: 10.5 min) [9]. Then, we consider the portion of the stream within the time frame of {last identified boundary + minimum length of a segment} to {last identified boundary + maximum length of a segment}. Within this range, we identify **transcript regions**, which are composed of one or more sentences that Google Speech-to-Text API [13] identifies as one unit of speech. Then, each interval between the end of a transcript region and the start of the next transcript region is identified as a **candidate boundary region** (Figure 4). We only consider breaks between transcript regions since cutting in between the streamer's sentence would be unnatural. We compute a score for each candidate boundary region, and the endpoint of the interval with the highest score is selected as the final boundary. The above process iterates once the maximum length of a segment (i.e., 20 minutes) has passed after the last identified boundary.

Our method is similar to Fraser et al.'s method, which segments archives of creative live streams [9]. While Fraser et al. segments an entire live stream after-the-fact, taking boundary candidates from the entire stream, our algorithm works in real-time as the stream is happening by considering boundary candidates from a certain time frame and scoring only the breaks between transcript regions or two adjacent transcript regions of a break. Also, while Fraser et al. take into account software application-specific factors in the scoring function, we incorporate generic factors including user interaction data.

For each candidate boundary region, we calculate a score with the following five factors:

(1) Visual difference of scenes: A dramatic visual change might indicate that a new topic was introduced: for example, an ending scene after a round of game ends, or a topic slide in slide-based videos. We convert the snapshots taken from two adjacent transcript regions (if any) to grayscale and compute the Structural Similarity Index (SSIM) [49] between the two frames. Then, we give higher scores if there are more visual differences between two adjacent transcript regions. The visual difference score between two adjacent transcript regions A and B is calculated as

$$S_{\text{Visual}} = 1 - \text{average}_{\forall a \in \mathbf{A}, \forall b \in \mathbf{B}} \text{SSIM}(a, b) \tag{1}$$

(2) Keywords from the transcript and the chat: When the topic changes, the streamer might talk about different content, which will result in different keyword distributions in transcripts. Viewers' reactions in the chat might have shifted as well. To extract the keywords from the transcript and the chat messages, we first exclude stopwords and then take out the 10 most frequent words. We extract the keywords for two adjacent transcript regions. We give higher scores if there are more differences in the keyword distribution. For keyword sets P and Q, the keyword score is calculated as

$$S_{\text{Keyword}} = 1 - n(\mathbf{P} \cap \mathbf{Q})/n(\mathbf{P} \cup \mathbf{Q}) \tag{2}$$

(3) Transitional cues: Transitional cues spoken by a streamer often indicate the start of a new topic. We favor ending cues such as "that's all", "done", and "therefore" to be close to the end of a sentence and starting cues such as "start" and "next" to be close to the start of a sentence. We use eight cue words in total, derived from previous work that leverages cue words to identify transitions [9, 34]. Let two adjacent transcript regions be A and B (i.e., prior-transcript region to the candidate boundary region: A, post-transcript region to the candidate boundary region: B). We give more scores if ending cues are close to the end of a sentence in A and if starting cues are close to the start of a sentence in B. The transition score of the candidate is calculated as

$$\begin{aligned} S_{\text{Transition}} = & \sum_{w \in \text{ending cues}} \text{indexOf}(w, \mathbf{A})/|\mathbf{A}| \\ & + \sum_{w \in \text{starting cues}} 1 - \text{indexOf}(w, \mathbf{B})/|\mathbf{B}| \end{aligned} \tag{3}$$

(4) Chat frequency: We avoid segmenting in the middle of active chat sessions as it could indicate that a topic is still being discussed. A boundary interval has a higher score if there are fewer

chat messages in it. For a set of chats C in a candidate boundary region $[s, e]$ where s is the starting time and e is the ending time of the boundary, the chat frequency score is calculated as

$$S_{\text{Chat}} = (e - s)/n(\mathbf{C}) \tag{4}$$

(5) Duration of a break: A short break between transcript regions might indicate a shift to the next sentence, while a long break might indicate a shift to other topics. Let e and s be the ending time and starting time of a candidate boundary region and d_i be a duration of a candidate boundary region i. The duration score is calculated as

$$S_{\text{Duration}} = (e - s)/\max(d_1, d_2, ...d_n) \tag{5}$$

The final score of a candidate boundary region is defined as the weighted sum of each score:

$$S_{\text{Final}} = C_{\text{Visual}} \times S_{\text{Visual}} + C_{\text{Keyword}} \times S_{\text{Keyword}} + C_{\text{Transition}} \times S_{\text{Transition}} + C_{\text{Chat}} \times S_{\text{Chat}} + C_{\text{Duration}} \times S_{\text{Duration}} \tag{6}$$

We report the optimal weights of each factor analyzed by stream types in Section 6.2.

The endpoint of a boundary with the highest score among all candidate boundary regions is identified as the next boundary of a section. We identify a snapshot with the most likes (or the first one to break ties) as a representative snapshot, and six keywords extracted from the transcripts and chat messages (three each) as representative keywords of a section.

5.2 Highlights Detection Algorithm

The highlights detection algorithm detects peak moments within each section using viewers' interaction data: chat messaging, likes, and sharing of stream content through snapshots. We assume that such interaction data is a good indicator of highlight moments. For example, chat messages can rapidly increase when something interesting happens in the stream. Viewers can also express interest in a particular moment by sharing snapshots or liking others' messages. To account for cases where user interaction is sparse, we implemented a bot that automatically shares snapshots when none were shared for a certain period (a minute in our implementation). Since the snapshots shared by the bot may not always be important (e.g., there may not be any highlights happening for a long period of time), our algorithm only considers snapshots that were liked by at least one human user (Section 5.2). To detect highlights, we divide a section into one-minute intervals and calculate a score for each interval. We chose one minute to account for possible lags in live streams and to ensure enough time for users to identify a moment while keeping it short to only contain important moments in a 5 to 20-minute-long section. The score is calculated as follows:

(1) We give higher scores when there are more chat messages, including snapshots.
(2) Snapshots are weighed three times as much as a plain chat message, as they directly indicate interesting moments of the stream. For snapshots that the bot shared, only the ones that are liked by at least one human user are considered.
(3) The number of likes gives weights to the message, as it represents the viewers' level of interest.
(4) The number of viewers in live streams may fluctuate during the stream. To ensure highlights are detected robustly even when there is a low number of viewers, we divide the overall score with the log of the number of viewers at each interval. This is to make sure that highlights are distributed more evenly.

The score for each chat message is computed as

$$(a + N/M)/\log M \tag{7}$$

where a is the weight of the message (a=1 if the message is a plain message and a=3 if it is a snapshot; the values are empirically determined.), N is the number of likes on the message, and M is the number of viewers. The total score of an interval is the sum of the chat scores within the interval.

Once the scores for each interval are computed, we determine the peak using the peak detection algorithm [1]. If there are consecutive peak intervals, we merge them into one highlight moment. The top 10 peaks are computed as highlights in every section. For the last section where the end time is not yet defined, highlights are updated every five minutes.

To create an expandable summary (G2), we first show two top highlight moments for each section. We show a representative snapshot, transcript, and chat message for each highlight moment, selected by the number of likes. If a user requests to see more highlights, we show the next top highlight moment. If a user requests to see all details of a certain highlight, we show at most five snapshots, transcripts, and chat messages, respectively, within the moment in chronological order.

We did not evaluate the performance of the highlights detection algorithm against ground truth because typical live streams only contain chat messages and no additional interactions that our algorithm takes into account. Instead, we report some of the outcomes of the algorithm from the user studies in Section 7.2.1.

6 PRELIMINARY EVALUATION OF THE ONLINE SEGMENTATION ALGORITHM

We evaluated the accuracy of our online segmentation through comparison with the ground truth segmentation data and identified optimal weights of the five data factors (section 5.1) for different genres of streams. Table 2 lists the seven live streams that we used for the test.

6.1 Method

We collected data from seven publicly available live streams, which had ground truth segment information that was either posted by the streamer (in the video description section) or by a viewer (in the comment section). We downloaded the video using Streamlink [44] and transcribed it using the Google Speech-to-Text API [13]. We also crawled the chat data using the Pytchat library [45] for videos from YouTube and the Twitch API [47] for videos from Twitch. To simulate the default system behavior where snapshots are only taken by the bot, we took a snapshot at every 1-minute interval using FFmpeg [8]. The snapshots taken were used in the algorithm for visual comparisons (Section 5.1).

Although we used the data from the entire stream, the data is processed sequentially online, such that the output of the algorithm

Name	Genre	Visual change	Audio	Length	Viewer #	Chat #	Chat #/Viewer #
Archade[1]	Game playing	High	86.5%	1:37:55	3119	31730	10.17
Suka[2]	Information sharing	Middle	99%	2:45:32	2597	40773	15.7
Minu[3]	Talking	Low	83%	2:39:59	396	7726	19.51
Google[4]	Tutorial (Web development)	High	93.6%	2:47:42	619	2619	4.29
Photoshop[5]	Tutorial (Photoshop)	High	79.2%	2:01:48	45	405	9
Amongus[6]	Game playing	High	70.5%	3:47:01	6756	46683	6.91
Drawing[7]	Drawing	High	66.7%	23:03	23	87	3.78

[1] youtu.be/49_7wGYE0pM [2] youtu.be/UPb9YbKyHuc [3] youtu.be/eXDDrNdLZAc [4] youtu.be/H89hKw06iWs
[5] youtu.be/TEEvVDfJw8A [6] youtu.be/XczTiRf6_XU [7] youtu.be/1OK9GWRHL7w

Table 2: Characteristics of the seven live streams used in the preliminary evaluation of the online segmentation algorithm. The audio percentage is computed by summing up the length of transcript regions divided by the total length of a video.

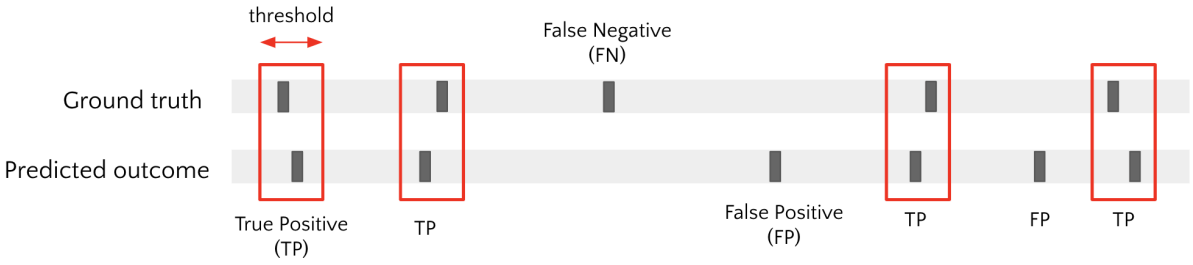


Figure 5: Visualization of the method used to calculate the F1 scores. With the video time as the x-axis, we considered a predicted boundary as a true positive if it is within the threshold distance from a ground truth boundary.

is the same as when applied in real-time with partial data. We evaluated the accuracy of the algorithm's output by calculating the F1 score with the ground truth segment information. Since even ground truth segment boundaries can reflect approximate times, and small time differences are less important for the overall segmentation quality, we used a threshold to determine whether a predicted boundary is correct. A predicted boundary was considered as a true positive if it was within a threshold distance from a ground truth boundary. We tested with two threshold values (details below). Also, since some segments may lie outside the 5- and 20-minute boundary range, we tested the algorithm with the actual minimum and maximum lengths for each stream in addition to the 5- and 20-minute boundaries. Lastly, we measured the accuracy with both uniform weights and optimal weights for the five factors the algorithm uses. The optimal weight is the weight with the highest accuracy. This was computed with an exhaustive search with each weight ranging from 1 to 5, excluding overlapping ratios.

To summarize, we report the accuracy of the algorithm for the following conditions:

(1) Threshold
  (a) minimum between 3 minutes and 3% of the length of the entire stream (`low-threshold`)
  (b) minimum between 5 minutes and 5% of the length of the entire stream (`high-threshold`)
(2) Minimum and maximum lengths of a segment
  (a) 5 min and 20 min (`standard`)
  (b) minimum and maximum length of the ground truth segments (`minmax`)
(3) Weight of the five factors
  (a) uniform weight (`base-coeff`)
  (b) optimal weight with the highest accuracy (`optimal-coeff`)

6.2 Results

With the optimal weight distribution, the average accuracy reached 67.6% (`low-threshold`) and 75.9% (`high-threshold`) with 5 and 20 minutes as the minimum and maximum lengths of a segment (`standard`). With the minimum and maximum length set as that of ground truths (`minmax`), the average accuracy increased to 78.6% (`low-threshold`) and 87.2% (`high-threshold`). In particular, the accuracy for the Minu video doubled (50% → 100%) since the average length of its segments is about 40 minutes. We report the F1 scores of the results in 8 (=2*2*2) different conditions in Appendix A.2.

The algorithm with optimal weights (`optimal-coeff`) yielded much higher accuracy than the baseline (`base-coeff`), by increasing 25 percentage points on average. For example, the accuracy on the Google video increased from 52.2% to 81.8% in `standard` & `low-threshold`, and the Archade video reached 92.9% from 66.7% in `standard` & `high-threshold`. This implies that assigning proper weights to each of the five factors by considering the type and characteristics of the stream is crucial. We report the optimal weight distribution for each stream in Appendix A.3.

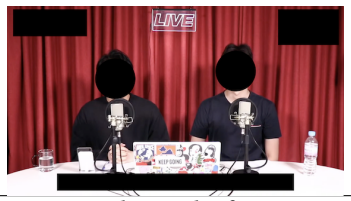
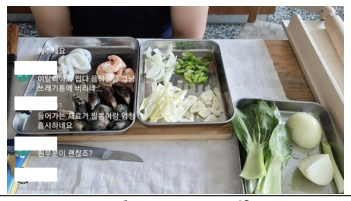
	Stock	Cooking	Game
			
Stream URL	youtu.be/ZuUbMfTB8ic	youtu.be/U3ryFMffB6Q	youtu.be/ngwyO57DJMI
Length	1:02:12	1:36:22	02:24:14
Original viewer #	246	106	965
Original chat message #	732	531	2551
Content type	Unstructured	Procedural	–
Content format	Audial	–	Visual
Stream pace	–	Slow	Fast

Table 3: Stream information used in our study. Characteristics spanning both sides are indicated as '–'.

	Stock		Cooking		Game			
	CatchLive (N=16)		CatchLive (N=18)		CatchLive (N=16)		Baseline (N=17)	
joined at	0:20	0:40	0:30	1:00	0:50	1:20	0:50	1:20
Participant #	5	11	9	9	9	7	8	9

Table 4: The number of viewers that participated in the study. The time is in minutes (e.g., 0:20 means 20 minutes). We conducted an additional comparative study for the Game stream (Section 8).

7 USER EVALUATION 1 - CATCHLIVE FOR DIFFERENT GENRES OF STREAMS

Since CatchLive's main features are designed for live stream viewers, we evaluated our system with viewers only. We conducted two separate user studies. The first study examined how participants use CatchLive to watch different genres of live streams. The second study compared CatchLive with a baseline interface for participants watching a gaming stream. This section describes the first study, and the following section describes the comparative study.

7.1 Methodology

Stream Information: We used three live streams: **(1) Stock:** podcast-style information sharing about the stock market, **(2) Cooking:** sharing the cooking process of a pasta dish, and **(3) Game:** playing a game called Animal Crossing (Table 3) for user studies. The Stock stream is about two stocks experts talking and sharing news about stocks. The information is delivered verbally with minimal visual changes to the scene. The Cooking stream shows a professional chef teaching how to cook a pasta dish. Information is delivered both verbally and visually with substantial changes in the scene at each step. The Game stream is about a streamer playing Animal Crossing. It does not cover informational content but is more for entertainment, with frequent visual changes.

We chose these three streams to see how our approach can be generalized against streams with different characteristics. The streams differ in the following major attributes: content type, content format, and stream pace. **(1) Content type** is procedural if the stream contains procedural knowledge, and unstructured if the stream covers relatively less coherent sub-topics. **(2) Content format** is visual if the information is mostly delivered visually, and audial if there are fewer visual changes and the information is mostly delivered verbally. **(3) Stream pace** is slow if the stream has many idle moments, and fast if the information is being delivered at a fast pace (either visually or verbally). We chose streams that have different characteristics for each attribute and cover topics that people might be interested in. While the three streams are not meant to be representative of all genres of streams, we believe they capture diverse combinations of the three major attributes introduced above (Table 3). We obtained the streamers' permission to use the streams in the study. To recreate the live stream experience, we re-streamed the recorded video and loaded the existing chat data in real-time. We configured optimal weights for the online segmentation algorithm according to stream type by referring to Table 10. All the sections and highlights information were computed in real-time during the replay.

Participants: We recruited participants with a prior interest in the stream topic through postings on online community websites of universities. We recruited 16 (11 male, 5 female, mean age 29.7), 18 (8 male, 10 female, mean age 24), and 16 (9 male, 7 female, mean age 23.8) participants for the Stock, Cooking, and Game streams, respectively. None of the participants had watched the particular stream before. We divided the participants of each stream into two groups. One group joined the stream after one-third of the stream had played, and the other after two-thirds of the stream. We distributed the number of participants equally for each group except for the Stock stream (Table 4). We put more participants in the second group for the Stock stream because the first 20 minutes of

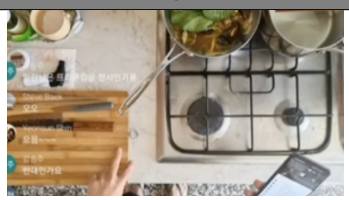
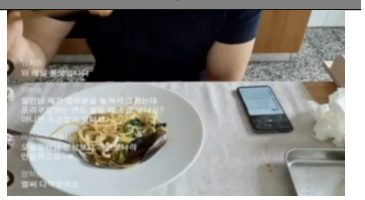
Section	1	2	3
Snapshot			
Keywords	Mic, Hello, Today, Shanghai	Ingredients, Shanghai Pasta, Easy, Chili oil	Noodle, Pasta, Ingredients, Look
Time	01:04-17:23	17:23-24:08	24:08-36:07
*Description	Greeting	Ingredients introduction	Ingredients preparation
Section	**4**	**5**	**6**
Snapshot			
Keywords	Wine, Water, Put, Later	Pre-cooking, In advance, Pasta, Made it	Curious, Looks good, Made it, Pasta
Time	36:07-46:30	46:30-1:05:30	1:05:30-1:13:45
*Description	Cooking on the pan	Cooking while talking about the concept of pre-cooking	Tasting

Table 5: Example results of the online segmentation algorithm on the Cooking stream. The algorithm segmented a 75 min-long stream into six sections. We can see that it identified meaningful sections such as greeting, ingredient introduction, cooking, and tasting. Although it does not provide labels for each section, the representative snapshots and keywords allowed participants to grasp each topic and get an overview of the stream. *Description was not provided to participants.

the stream covered relatively little content. We refer to participants as Vn-{stream name} (e.g., V16-stock).

Procedure: We first gave a tutorial of CatchLive to the participants via video conferencing. Then, viewers were asked to join the stream in the middle using CatchLive. They were asked to watch the stream for 20 (Stock) or 25 (Cooking, Game) minutes. This meant that there was no overlap between the two groups of participants. Participants were informed that they were watching a recorded live stream, such that they may not get responses from recorded chat messages or the streamer. After the viewers watched the stream, we conducted an online survey about their experience. The survey was composed of 5-point Likert scale questions and open-ended questions, regarding the understanding and engagement with the stream and the usefulness of the summary features. Viewers were compensated $10 for their participation in a 50-minute-long study.

7.2 Findings

We first discuss how participants used CatchLive's summary features in general. Then, we analyze how catching up behavior was different for each stream's characteristics.

7.2.1 *How participants use CatchLive's summary features.*

Overall, the participants could understand the stream well with the help of the timeline and highlights. They were also engaged with the stream. Below, we describe how each feature in CatchLive helped participants and how participants used them.

The timeline helped viewers grasp the overview of the stream: Table 5 shows example timeline information shown to the users. The timeline segments a whole stream into several sections and provides a representative snapshot, keywords, and time information of each section. When asked about in what way the timeline was helpful (if it were), 24 out of 50 participants said that they were able to get brief information on previous parts quickly before they started watching the stream. V14-stock said *"I was able to know what specific topics in stocks the stream is covering by looking at the keywords provided in the timeline, such as 'healthcare' and stock names."* V14-cooking said *"I was able to understand how the dish was being cooked, in what order. "* Knowing the overview of the stream helped participants during the stream as well. Seven participants mentioned that the timeline information helped them decide which parts to watch to get the information they wanted while watching the stream. V7-game said *"I could jump back to certain sections by looking at snapshots on the timeline when there was something I couldn't understand."* V12-cooking said *"It was easier to go back through the timeline when I could understand the story and empathize with others only if I knew the story covered earlier."* For a 5-point Likert scale question of how much they could understand the overview of the stream, the average point was 3.84 (SD=0.98) (Figure 6a).

Highlights helped viewers identify important moments, understand more about the stream, and fill the void in live streams: Each highlight moment is provided with representative

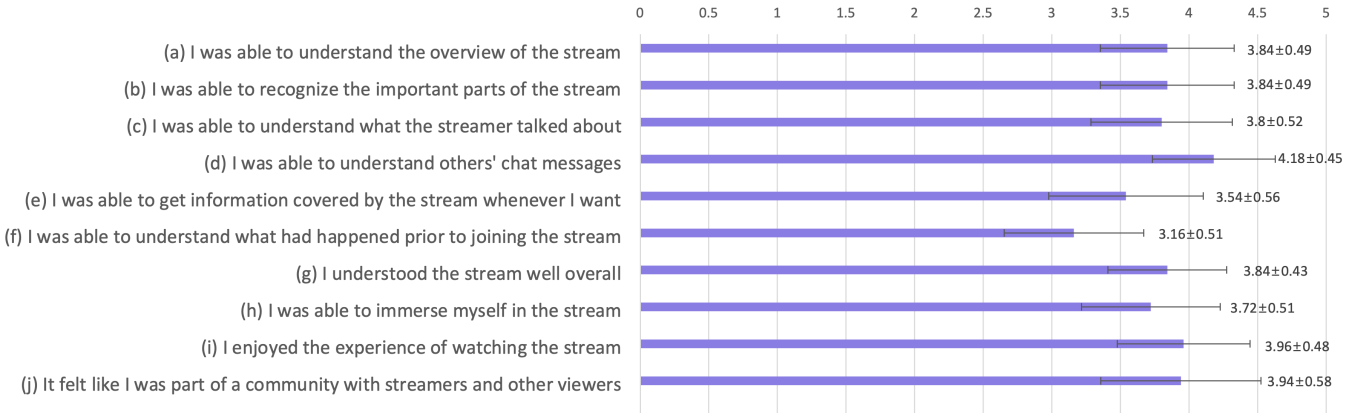


Figure 6: Survey responses regarding (a)–(g) understanding of the stream, and (h)–(j) engagement in the stream with CatchLive (N=50 (16, 18, 16 for the Stock, Cooking, and Game, respectively)).

snapshots, transcripts, and chat messages. When asked about in what way the highlights were helpful (if it were), 17 out of 50 participants said they could identify the important moments in the long stream, and 14 participants said they were able to understand more about what had happened. V16-game said *"I was able to enjoy the stream by looking at the parts that people were mainly interested in."* V15-cooking said *"It was nice to see useful chat messages being organized without unrelated ones and keep them like memos."* Highlights further helped participants understand more about the unseen parts. V12-game said *"I could understand the flow of the game well because it made the order of previous events clear for me."* Moreover, five participants mentioned that Highlights filled the void of live streams. V16-cooking said *"Typical live streams are unedited, so it's easy to get bored when it gets loose. However, the summary feature made me enjoy the stream especially when the streamer was not speaking or something that I'm not interested in was being played."* For a 5-point scale question of how much they could recognize the important parts of the stream, the average point was 3.84 (SD=0.98) (Figure 6b). We present example highlight moments shown to the participants in Appendix B.

The timeline and Highlights allowed viewers to catch up with less interruption compared to rewinding: Participants reported that they could catch up with the previous parts with less interruption using the timeline and highlights when compared to rewinding the video. For a 5-point scale question asking about how less distracting it was to the current stream, the average point was 2.58 (SD=1.5) for rewinding while it was 3.58 (SD=1.28) and 3.54 (SD=1.29) for timeline and highlights, respectively (Mann-Whitney Test, $p<0.01$, $z=2.7$ for the timeline and $p<0.01$, $z=2.81$ for highlights).

More information is needed to fully understand the previous context: However, when participants were asked how much they could understand what had happened prior to joining the stream, the average score was 3.16/5 (SD=1.02), which shows that the information shown to users might have been not enough to fully understand the previous parts (Figure 6f). Although the timeline and highlights provide useful information as described above, seven participants said that they wish to further rewind the video to get the full details. V18-cooking said *"Based on snapshots and keywords, I was able to roughly understand when ingredients were introduced and when and what ingredients were added. However, it was difficult to grasp the exact contents with the highlight feature alone, such as how it was cooked."* Moreover, three participants pointed out that incomplete keywords or transcripts hindered them from grasping the main points. V15-stock said *"The system let me know roughly what happened before, but some keywords were too general so it was hard to know what happened exactly."* V10-stock said *"The transcript was awkwardly transcribed so it was hard to understand the content."* These findings suggest how live stream summarizations can be improved, which we elaborate in Section 9.1.

7.2.2 *How catching up behavior differs across the stream characteristics.*

With three different stream genres, we could identify that the perceived usefulness of the system differs across stream genres. We describe how each characteristic of streams affects users' experiences below.

Participants used the timeline in different ways: The timeline was used in different ways depending on the stream genre. When asked how the timeline was helpful (if it were), 10 out of 16 participants with the Stock stream answered that it was used to understand what topics were covered in the previous parts. On the other hand, 4 out of 18 participants with the Cooking stream specifically mentioned how understanding previous parts led to understanding the current topic. V3-cooking said *"I could understand how previous parts were connected to the current topic. I could see that the ingredient preparation part was already done and thus cooking was in progress."* This might be because the Cooking stream contains procedural knowledge where each step is connected to adjacent steps. Understanding previous steps and how they are connected would help viewers understand what the current step covers in a big landscape.

The highlights were most useful in stream with visual content: The highlights were most useful in the Game stream (3.88/5) while they were least useful in the Stock stream (2.56/5) (Figure 7). In the Game stream, 9 out of 16 participants mentioned that the highlights helped them understand the overall storyline of the game and three mentioned that it helped them enjoy the stream

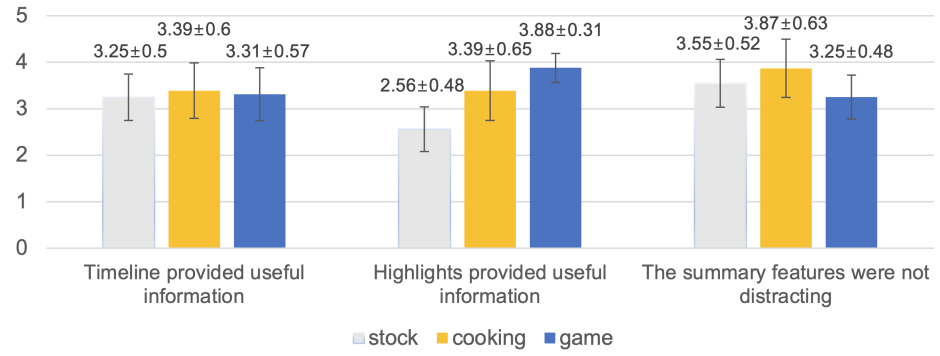


Figure 7: Perceived usefulness of CatchLive's features for each stream.

by watching previous interesting moments. However, in the Stock stream, 4 out of 16 viewers mentioned that the highlights did not provide much useful information. V13-stock said *"the screenshots in highlights were almost the same so they didn't provide much useful information."* V10-stock said *"I needed to rely on the transcripts but it wasn't perfect."* Since snapshots in the highlights play an important role and allow users to quickly grasp what had happened before, streams with visual content would benefit more from the highlights in CatchLive.

The summary features were least distracting in streams with slow pace: While viewers felt less interruption to the current stream with our summary features compared to rewinding the stream for all three types of streams, there was a difference in the degree of reported interruption across stream types. Among the three groups, viewers of the Cooking stream reported the least interruption (3.87/5 for not distracting) while viewers of the Game stream felt the most interruption (3.25/5) (Figure 7). This could be explained by the pace of the streams. The percentage of voice in the audio for each stream was 92.1%, 61.7%, 75.3% for Stock, Cooking, and Game, respectively. Viewers might feel least distracted in the Cooking stream because there was less verbal information shared by the streamer, and the cooking process was shared at a relatively slow pace, as it involved various preparation steps and wait times. On the other hand, viewers might feel most distracted in the Game stream because the visuals kept changing at a relatively fast pace.

8 USER EVALUATION 2 - COMPARATIVE STUDY USING THE GAME STREAM

We conducted a comparative study to evaluate the summary features of the system in more depth, with the Game stream as our target domain. We chose the Game stream as our target domain for the comparative study because the summary features (i.e., the timeline and the highlights) were most useful in the Game stream when averaging the two scores (Figure 7). We hypothesized that viewers with CatchLive will better understand and engage with the stream. Specifically:

- H1: Viewers with CatchLive will have a better understanding of the stream than viewers without it.
- H2: Viewers with CatchLive will be more engaged with the stream than viewers without it.

8.1 Methodology

Stream Information: We used the same gaming stream from the first user study.

Participants: We recruited additional 17 participants as the baseline group (9 male, 8 female, mean age 23.1). The CatchLive group was from the previous user study with 16 participants (9 male, 7 female, mean age 23.8). Note that we used the same results from Section 7 for the CatchLive group and we only added the baseline group. We refer to participants of the baseline group as Vn-game-base.

Procedure: Our study used a between-subject design (baseline and CatchLive). The CatchLive group used the full-featured version of CatchLive, while the baseline group used CatchLive without the summary features (i.e., only the chat, "like" and snapshot interactions described in Section 4.3 were enabled). While snapshot capturing and sharing or liking others' chat messages is not currently supported by most existing platforms, we allowed the baseline group to use such interactions to clearly evaluate the effects of summary features. After giving a tutorial on how to use the system, participants joined the stream with the assigned interface. Viewers were asked to watch the stream for about 25 minutes using the assigned interface. After the viewers watched the stream, we conducted an online survey with them. The survey was composed of 5-point Likert scale questions and open-ended questions, regarding the understanding and engagement with the stream and the overall experience. For the understanding and engagement-related questions, the same questions used in the first study were used. We asked seven and three 5-point Likert scale questions regarding understanding and engagement respectively, to evaluate the self-reported responses thoroughly from diverse aspects. The survey also included factual recall questions asking about the content of the stream to measure the understanding more accurately. We used the number of chat messages as a measure of engagement in addition to the survey responses. Viewers were compensated with $10 for their participation in a 50-minute-long study.

8.2 Findings

We first asked about the participant's prior knowledge about the topic covered in the stream since it could affect their overall understanding of the stream. For a 5-point Likert-scale question, the

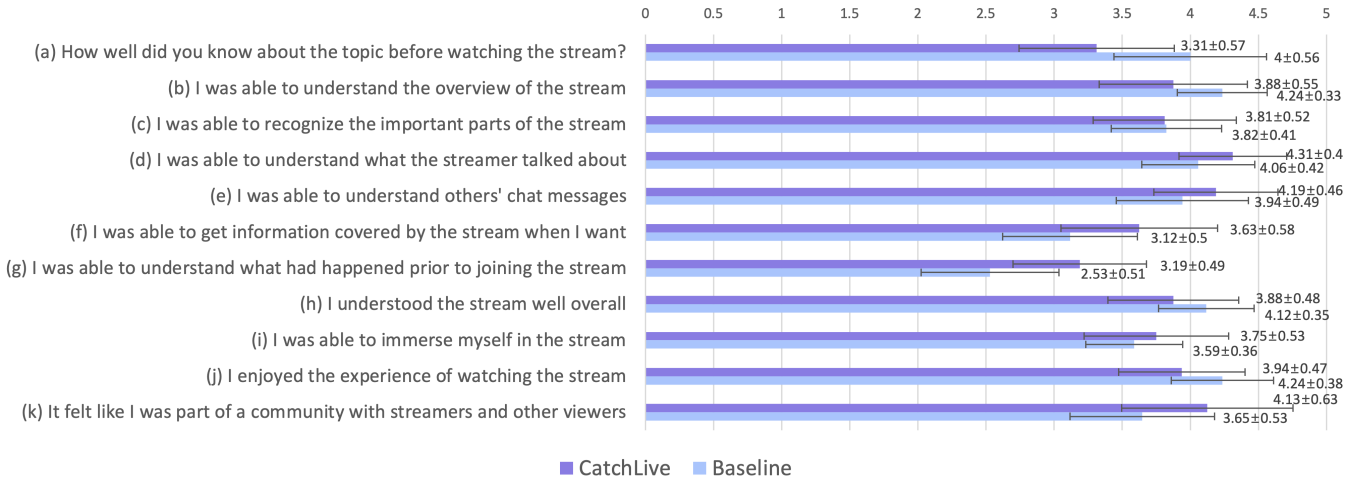


Figure 8: Survey responses regarding (a) the prior knowledge about the topic, (b)–(h) understanding of the stream, and (i)–(k) engagement in the stream of CatchLive (N=16) and the baseline group (N=17).

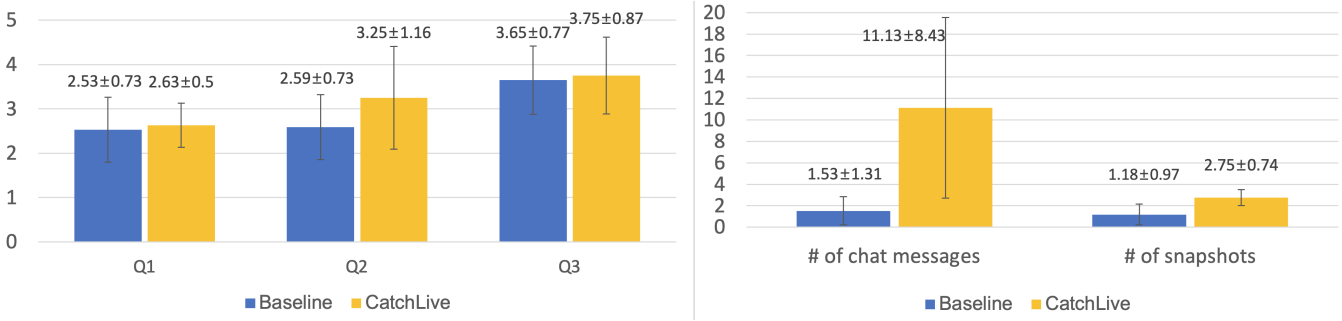


Figure 9: (left) The number of correct answers to questions regarding understanding in the game stream. Q1: List the animals that the streamer met. Q2: List the tools and what the streamer did with the tool. Q3: List the keywords that were covered in the stream other than animals and tools. There was no significant difference between the groups. (right) The number of chat messages and snapshots shared in the chat, as a means of engagement. The difference was statistically significant (p<0.01).

mean self-reported knowledge levels were higher in the baseline group (Figure 8a, 3.31 (SD=1.14) and 4.00 (SD=1.12) for CatchLive and the baseline group, respectively.) Although the difference was not significant (Mann-Whitney test, p=0.101, z=1.639), it should be noted that this might affect the understanding and the overall experience of the stream when comparing two groups.

8.2.1 H1: Viewers with CatchLive will have a better understanding of the stream.

The average understanding score was higher in the CatchLive group but there was no significant difference between the two groups. We asked seven 5-point Likert-scale questions (1-strongly disagree, 5-strongly agree) regarding the understanding of the stream (Figure 8b–h). The average score of the questions was higher in the CatchLive group (3.84, SD=0.99) than the baseline (3.69, SD=0.85), but there were no significant differences for all questions (Mann-Whitney Test, p=0.12, z=1.16). Among the questions, the question asking how much they could understand what had happened prior to joining (Figure 8g) showed the biggest difference between the two groups. The CatchLive group indicated they could understand the previous parts better (3.19, SD=0.49) than the baseline (2.53, SD=0.51) group, although the difference was not statistically significant (Mann-Whitney Test, p=0.09, z=1.68). We also asked factual questions regarding the stream content, to accurately estimate viewers' level of understanding. We asked specific questions about the stream content: (Q1) List the animals that the streamer met. (Q2) List the tools and what the streamer did with the tool. (Q3) List the keywords that were covered in the stream other than animals and tools. While the score was higher in the CatchLive group for all the questions, the average number of answers to these questions showed no significant difference (Mann-Whitney Test, p=0.44, z=0.16) (Figure 9-left).

To summarize, there was no significant difference between CatchLive and the baseline group regarding the level of understanding, for both self-reported measures and recall questions. We believe there are three possible reasons: 1) As people do not try to *remember*

things from games, in the survey which was done *after* the stream, the result might not have shown a big difference. 2) The higher prior knowledge in the baseline group could have also affected the result. 3) The baseline also contained additional features such as snapshots and bookmarking messages, which provided them with the opportunity to review the previous content (V2, 14, 15, 17-game-base). V15-game-base said *"It was easy to remember the important moments by sharing snapshots, and filtering liked items allowed me to review the previous content."*

8.2.2 H2: Viewers with CatchLive will be more engaged with the stream.

CatchLive group was more active in the stream than the baseline group: We compared the number of chat messages and snapshots the viewers made, assuming that more engaged viewers would leave more messages in the stream. The CatchLive group wrote significantly more messages (11.13, SD=8.43) than the baseline group (1.53, SD=1.31) (Mann-Whitney Test, $p<0.01$, $z=3.20$). Also, the CatchLive group shared more snapshots (2.75, SD=0.74) than the baseline group (1.18, SD=0.97) (Mann-Whitney Test, $p<0.01$, $z=2.72$) (Figure 9).

We also asked three 5-point Likert-scale questions (1-strongly disagree, 5-strongly agree) regarding the engagement of the stream (Figure 8i–k). The average score of the questions was higher in the CatchLive group (3.94, SD=1.08) than the baseline (3.82, SD=0.84), but there were no significant differences for all questions (Mann-Whitney Test, $p=0.34$, $z=0.96$).

9 DISCUSSION

In this paper, we introduce CatchLive, a system that summarizes ongoing live streams with stream content and user interaction data. With two user studies, we evaluate CatchLive's effectiveness in terms of understanding and engagement. From the evaluation study with three different genres of streams (Section 7), we could see that CatchLive helped participants understand the stream and stay engaged with the stream. From the comparative study with the Game stream (Section 8), we could see that participants with CatchLive were more engaged with the stream than the baseline group, but there was no significant difference in their level of understanding. We discussed that the nature of gaming streams could be one of the possible reasons to this. Extending the analysis of the results, we first discuss how the characteristics of streams should be reflected in designing real-time summarizations of live streams. Then, we discuss how we can leverage user interaction in live streams. Finally, we discuss the unique challenges and opportunities in designing real-time summarizations.

9.1 Reflecting Stream Characteristics in Summarization

9.1.1 Content type. From our studies, we could see that participants perceived the usefulness of the timeline differently depending on the content type. For procedural content such as the Cooking stream, participants found the timeline useful in identifying step information, and understanding the current step in relation to previous steps. When designing such a timeline, accurate labels of each step beyond the keywords might be helpful in order to present the goal of each step more clearly. Also, all steps should be fully covered with proper granularity. V11-cooking said, *"It was good to see what was happening overall in the stream, but I felt some information I needed was missing."* The step information will be meaningful when all the steps are presented. On the other hand, streams with unstructured content such as the Stock stream might have less clear boundaries when dividing the timeline into multiple sections. Although participants still found the keywords presented in the timeline useful for understanding which concepts had been covered, concrete keywords that distinguish a section from others would be helpful.

9.1.2 Content format. Among the three types of sources through which the highlights are represented (i.e., snapshots, transcripts, and chat messages), snapshots provide information that can be recognized at a glance. Streams with a high portion of visual elements benefit the most from snapshots, allowing users to get a quick overview of highlights. Thus, capturing meaningful snapshots that represent certain highlights would be important. Providing short clips of highlights could add more details to a highlight moment. On the other hand, viewers of streams with little visual change throughout the stream had no choice but to rely heavily on textual information, which requires a lot of user attention. Moreover, ill-transcribed transcripts may exacerbate the problem. Thus, when summarizing less visual or text-heavy streams, the textual information should be provided more accurately and summarized effectively. Predefined keywords, text shortening techniques [35], or keywords visualizations could be used.

9.1.3 Stream Pace. From our studies, we could see that the pace of the stream affects the participants' perceived degree of interruption from CatchLive's summarization features. Viewers of slow-paced streams such as the Cooking stream felt that they were less distracting, suggesting that more levels of detail could be further provided to fill the idle moments. CatchLive supports multiple levels of detail but it could be extended to the fullest degree (i.e., showing all the transcripts and chat messages) or even providing short clips of the stream. On the other hand, streams with a fast pace such as the Game stream need better ways of providing summarizations in a less distracting way. For this, shorter highlight moments combining multiple sources (visuals, transcripts, and viewers' reactions) into one could be designed.

9.2 Leveraging User Interaction Data

Our approach showed the feasibility of leveraging user interaction data for generating summaries of live stream content. User interaction data enhanced the stream by indicating a noticeable moment, which the stream content by itself alone would not have been able to capture. They could also capture the stream content that was not explicitly spoken about by the streamer from what users say about the stream in the chat. To leverage the user interaction data to a degree that is helpful for generating a summary, securing large enough and good quality user interaction data is necessary. CatchLive tried to motivate users by introducing interactions they could enjoy such as snapshots and likes. With such features, users could naturally interact with the streamer and other viewers, while providing explicit data that points to salient moments of the stream. Motivating user interaction led users to contribute to generating a summary, which

Characteristics		Findings	Design implications
Content type	Procedural	The timeline gave step information.	• Accurate label of each step • A full coverage of all steps
	Unstructured	The keywords in the timeline were helpful in checking previous topics.	• Concrete keywords that distinguish a section from others
Content format	Visual	Snapshots gave a quick overview of highlights.	• Representative snapshots • Short clips of a highlight
	Audial	Relying on transcripts to catch up required much time and effort.	• Accurate transcription • Text summarization techniques
Stream pace	Slow	The summary features were not much of a distraction.	• More levels of detail of highlights • Short clips of a highlight
	Fast	The summary features could be distracting.	• Shorter highlight moments • Combining multiple sources into one

Table 6: Summary of the findings and implications on designing live stream summarizations with respect to stream characteristics. The findings are from Section 7.2.2.

again led to additional user interactions with higher engagement. CatchLive seeks to establish this virtuous feedback cycle in live streams.

However, user interaction data might contain noise. There could be a case where the chat contains off-topic comments. Since our approach relies not only on chat messages but also on explicit interactions (e.g., snapshots, likes) and stream content (e.g., keywords in the transcript), we expect undesirable effects could be mitigated. Quality control could be introduced such as removing certain words from the algorithmic computation to ensure high quality.

9.3 Making Use of Active Inputs

While CatchLive leveraged users' natural interactions, live stream summarizations could benefit from users' active input as well. For example, the streamer can designate important terminologies or concepts that will be covered prior to starting a stream, which the system can then use to detect moments when the keywords are introduced and treat them as highlights. For certain genres with a clear structure of what the streamer is going to do such as cooking or lectures, they can set up the timeline information prior to starting a stream, which can be shown in the interface in real-time when the streamer denotes it. Integrating such pre-planned information can make the summaries more reliable. Viewers can also give more active inputs. They can identify useful tips by voting on chat messages, or label questions and answers that future viewers might find useful.

9.4 Generating Summaries in Real-time

In providing a summary of live streams in real-time, we faced several constraints when designing the algorithms. For example, the algorithm has to work fast, work with data being collected in real-time while not altering the previous results to maintain consistency. This led us to design the online segmentation algorithm that considers only the two adjacent regions of a break or the break itself, rather than entire blocks of transcripts as in Fraser et al.'s work [9]. There might be a trade-off between the accuracy and the speed of summary generation. Moreover, there is a delay in live streams when viewers watch the content from the stream source [7, 55]. The delay should be taken into account when considering stream content and user interaction together, to secure synchronized data. Although we did not implement further adjustments as our system processed all data sources from a viewer's point of view, it should be carefully considered when designing live stream systems.

When the time and resources allow it, the live setting has a unique benefit where it can provide real-time feedback to the algorithm and keep improving the quality of the summary. For example, an adaptive segmentation algorithm could be proposed where the algorithm adjusts the weight of the five factors (i.e., Visual differences, Keywords, Transitional cues, Chat frequency, Duration of a break) by reflecting the streamers' corrections. Future techniques could further improve the performance by understanding the challenges and benefits that the live setting brings.

9.5 Beyond Summarizing Live Streams

CatchLive aims to generate real-time summaries of live streams so that any viewer who joins in the middle of the stream could benefit from the summaries. Although our work focused on generating summaries in real-time, it can be applied to recorded videos of the streams by using the video content and interaction data. Streamers can also actively edit the outcomes of the algorithms to make the summaries more accurate. Not only the recorded live streams but also general videos can benefit from a similar approach. Timestamped comments can provide additional signals of where the viewers' interests are at [52] and can be used for generating summaries. The chat-based representation of the summaries can allow users to *read* the video without having to watch it.

10 LIMITATIONS AND FUTURE WORK

Our study mainly focused on exploring how users would use and get help from real-time summarizations in live streams. The study had several limitations. First, we recreated the experiences of live streams by streaming the recorded video and loading the existing chat data in real-time. Participants might have felt the absence of the streamers and other viewers, especially when they did not get any feedback on their reactions. Although there was a certain number of real-time users, it might not have been enough to encourage

the participants to actively engage in chat messaging. Future work can validate the proposed techniques in a real live stream context. While CatchLive supports YouTube Live streams, it can be extended to support live streams of other platforms as long as we can retrieve the manifest files that contain the video content. For example, for Twitch videos, we can use twitch-m3u8 [6] to retrieve the video content. The proposed solution can be also integrated into current live stream platforms by creating extensions such as Twitch extensions [48] or web browser extensions (e.g., 'Better YouTube Live [27]', 'YouTube Live Chat Enhancement [40]').

Second, the summaries in the study were generated only with a group of more than 100 users. Since our system largely relies on user interaction data, securing enough user interaction data would be critical, and it would be worthwhile to investigate how it would work with a smaller number of viewers. Although CatchLive has a bot taking snapshots to secure a substantial amount of interaction data, the quality of the summary could deteriorate in streams with a smaller number of viewers. The dynamics of viewer groups could also affect the summary quality. If the viewer community has developed over multiple sessions and thus evolved to be tightly connected, a new viewer would find it difficult to catch up through summaries since there might be community-specific jargon or content. Here, a series of summaries of previous streams generated by CatchLive could be helpful. New catch-up features such as keyword search could be introduced in future systems, to help users understand when and how such a term was introduced. Analyzing how CatchLive works in different numbers and dynamics of viewers could lead to more useful insights.

11 CONCLUSION

This paper explores providing a real-time summary of live streams, to help viewers who join in the middle of the stream catch up on the previous content. We built CatchLive, a tool that leverages both the stream content and user interaction data to provide a summary of live streams with the timeline and highlights. Our user study demonstrates that CatchLive helps viewers grasp the overview of the stream, identify important moments, and stay engaged. We identify different behaviors of using the summary across stream characteristics, providing insights into designing real-time summarizations of live streams. We hope that our work opens up new opportunities for producing and consuming live media content in an engaging and effective way.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2020R1C1C1007587) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-01347, Video Interaction Technologies Using Object-Oriented Video Modeling).

REFERENCES

[1] J.P.G. van Brakel. 2020. Robust peak detection algorithm using z-scores *(Stack Overflow)*. https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/22640362#22640362

[2] Di (Laura) Chen, Dustin Freeman, and Ravin Balakrishnan. 2019. *Integrating Multimedia Tools to Enrich Interactions in Live Streaming for Language Learning.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300668

[3] Xinyue Chen, Si Chen, Xu Wang, and Yun Huang. 2021. "I Was Afraid, but Now I Enjoy Being a Streamer!": Understanding the Challenges and Prospects of Using Live Streaming for Online Education. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW3, Article 237 (jan 2021), 32 pages. https://doi.org/10.1145/3432936

[4] Wen-Sheng Chu, Yale Song, and Alejandro Jaimes. 2015. Video Co-summarization: Video Summarization by Visual Co-occurrence. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3584–3592. https://doi.org/10.1109/CVPR.2015.7298981

[5] Kenny Davila and Richard Zanibbi. 2017. Whiteboard Video Summarization via Spatio-Temporal Conflict Minimization. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 1. IEEE, 355–362.

[6] Samuel Dudik. 2022 (accessed Jan 8, 2022). *twitch-m3u8*. https://www.npmjs.com/package/twitch-m3u8/

[7] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.Tv. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW, Article 50 (nov 2018), 18 pages. https://doi.org/10.1145/3274319

[8] ffmpeg. 2022 (accessed Jan 8, 2022). *ffmpeg*. https://www.ffmpeg.org/

[9] C. Ailie Fraser, Joy O. Kim, Hijung Valentina Shin, Joel Brandt, and Mira Dontcheva. 2020. Temporal Segmentation of Creative Live Streams. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376437

[10] C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva. 2019. Sharing the Studio: How Creative Livestreaming Can Inspire, Educate, and Engage. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 144–155. https://doi.org/10.1145/3325480.3325485

[11] Cheng-Yang Fu, Joon Lee, Mohit Bansal, and Alex C. Berg. 2017. Video Highlight Prediction Using Audience Chat Reactions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 972–978.

[12] Seth Glickman, Nathan McKenzie, Joseph Seering, Rachel Moeller, and Jessica Hammer. 2018. Design Challenges for Livestreamed Audience Participation Games. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play* (Melbourne, VIC, Australia) *(CHI PLAY '18)*. Association for Computing Machinery, New York, NY, USA, 187–199. https://doi.org/10.1145/3242671.3242708

[13] Google. 2022 (accessed Jan 8, 2022). *Google Cloud Speech-to-Text*. https://cloud.google.com/speech-to-text

[14] Pawara Gunawardena, Oshada Amila, Heshan Sudarshana, Rashmika Nawaratne, Ashish Luhach, Damminda Alahakoon, Amal Perera, Charith Chitraranjan, Naveen Chilamkurti, and Daswin Silva. 2021. Real-time automated video highlight generation with dual-stream hierarchical growing self-organizing maps. *Journal of Real-Time Image Processing* 18 (10 2021). https://doi.org/10.1007/s11554-020-00957-0

[15] Lassi Haaranen. 2017. Programming as a Performance: Live-Streaming and Its Implications for Computer Science Education. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy) *(ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 353–358. https://doi.org/10.1145/3059009.3059035

[16] William A. Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on Twitch: Fostering Participatory Communities of Play within Live Mixed Media *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1315–1324. https://doi.org/10.1145/2556288.2557048

[17] William A. Hamilton, Nic Lupfer, Nicolas Botello, Tyler Tesch, Alex Stacy, Jeremy Merrill, Blake Williford, Frank R. Bentley, and Andruid Kerne. 2018. Collaborative Live Media Curation: Shared Context for Participation in Online Learning. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3174129

[18] Hung-Kuang Han, Yu-Chen Huang, and Chien Chin Chen. 2019. A Deep Learning Model for Extracting Live Streaming Video Highlights Using Audience Messages. In *Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference* (Kobe, Japan) *(AICCC 2019)*. Association for Computing Machinery, New York, NY, USA, 75–81. https://doi.org/10.1145/3375959.3375965

[19] Liang-Chi Hsieh, Ching-Wei Lee, Tzu-Hsuan Chiu, and Winston Hsu. 2012. Live Semantic Sport Highlight Detection Based on Analyzing Tweets of Twitter. In *2012 IEEE International Conference on Multimedia and Expo*. IEEE, 949–954.

[20] Ellen A. Isaacs, Trevor Morris, and Thomas K. Rodriguez. 1994. A Forum for Supporting Interactive Presentations to Distributed Audiences. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (Chapel Hill, North Carolina, USA) *(CSCW '94)*. Association for Computing Machinery, New York, NY, USA, 405–416. https://doi.org/10.1145/192844.193060

[21] Haojian Jin, Yale Song, and Koji Yatani. 2017. ElasticPlay: Interactive Video Summarization with Dynamic Time Budgets. In *Proceedings of the 25th ACM International Conference on Multimedia* (Mountain View, California, USA) *(MM '17)*. Association for Computing Machinery, New York, NY, USA, 1164–1172. https://doi.org/10.1145/3123266.3123393

[22] Juho Kim, Philip J. Guo, Carrie J. Cai, Shang-Wen (Daniel) Li, Krzysztof Z. Gajos, and Robert C. Miller. 2014. Data-Driven Interaction Techniques for Improving Navigation of Educational Videos. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 563–572. https://doi.org/10.1145/2642918.2647389

[23] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. Crowdsourcing Step-by-Step Information Extraction to Enhance Existing How-to Videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 4017–4026. https://doi.org/10.1145/2556288.2556986

[24] Pascal Lessel, Michael Mauderer, Christian Wolff, and Antonio Krüger. 2017. Let's Play My Way: Investigating Audience Influence in User-Generated Gaming Live-Streams. In *Proceedings of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video* (Hilversum, The Netherlands) *(TVX '17)*. Association for Computing Machinery, New York, NY, USA, 51–63. https://doi.org/10.1145/3077548.3077556

[25] Pascal Lessel, Alexander Vielhauer, and Antonio Krüger. 2017. Expanding Video Game Live-Streams with Enhanced Communication Channels: A Case Study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1571–1576. https://doi.org/10.1145/3025453.3025708

[26] Ching Liu, Juho Kim, and Hao-Chuan Wang. 2018. ConceptScape: Collaborative Concept Mapping for Video Learning. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173961

[27] Better YouTube Live. 2022 (accessed Jan 8, 2022). *Better YouTube Live*. https://www.betteryoutube.live/

[28] Zhicong Lu, Seongkook Heo, and Daniel J. Wigdor. 2018. StreamWiki: Enabling Viewers of Knowledge Sharing Live Streams to Collaboratively Generate Archival Documentation for Effective In-Stream and Post Hoc Learning. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW, Article 112, 26 pages. https://doi.org/10.1145/3274381

[29] Zhicong Lu, Rubaiat Habib Kazi, Li-yi Wei, Mira Dontcheva, and Karrie Karahalios. 2021. StreamSketch: Exploring Multi-Modal Interactions in Creative Live Streams. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1, Article 58 (apr 2021), 26 pages. https://doi.org/10.1145/3449132

[30] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. 2011. Twitinfo: Aggregating and Visualizing Microblogs for Event Exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 227–236. https://doi.org/10.1145/1978942.1978975

[31] MDN. 2022 (accessed Jan 8, 2022). *Web Speech API*. https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition

[32] Michele Merler, Khoi-Nguyen C. Mac, Dhiraj Joshi, Quoc-Bao Nguyen, Stephen Hammer, John Kent, Jinjun Xiong, Minh N. Do, John R. Smith, and Rogerio Schmidt Feris. 2019. Automatic Curation of Sports Highlights Using Multimodal Excitement Features. *IEEE Transactions on Multimedia*, 1147–1160.

[33] Matthew K. Miller, John C. Tang, Gina Venolia, Gerard Wilkinson, and Kori Inkpen. 2017. Conversational Chat Circles: Being All Here Without Having to Hear It All. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 2394–2404. https://doi.org/10.1145/3025453.3025621

[34] Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. Video Digests: A Browsable, Skimmable Format for Informational Lecture Videos. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 573–582. https://doi.org/10.1145/2642918.2647400

[35] Amy Pavel, Gabriel Reyes, and Jeffrey P. Bigham. 2020. Rescribe: Authoring and Automatically Editing Audio Descriptions. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 747–759. https://doi.org/10.1145/3379337.3415864

[36] Anthony J. Pellicone and June Ahn. 2017. The Game of Performing Play: Understanding Streaming as Cultural Production. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4863–4874. https://doi.org/10.1145/3025453.3025854

[37] Kevin P. Pfeil, Neeraj Chatlani, Joseph J. LaViola, and Pamela Wisniewski. 2021. Bridging the Socio-Technical Gaps in Body-Worn Interpersonal Live-Streaming Telepresence through a Critical Review of the Literature. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1, Article 120 (apr 2021), 39 pages. https://doi.org/10.1145/3449194

[38] David Shamma, Lyndon Kennedy, and Elizabeth Churchill. 2010. Tweetgeist: Can the Twitter Timeline Reveal the Structure of Broadcast Events? *CSCW Horizons* 26 (2010).

[39] Pushkar Shukla, Hemant Sadana, Apaar Bansal, Deepak Verma, Carlos Elmadjian, Balasubramanian Raman, and Matthew Turk. 2018. Automatic Cricket Highlight Generation Using Event-Driven and Excitement-Based Features. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1881–18818.

[40] sirvival. 2022 (accessed Jan 8, 2022). *YouTube Live Chat Enhancement*. https://chrome.google.com/webstore/detail/youtube-live-chat-enhance/akikhgecfhfbdmiccfbgdmcjjbaiohbi

[41] Max Sjöblom and Juho Hamari. 2017. Why do people watch others play video games? An empirical study on the motivations of Twitch users. *Computers in Human Behavior* 75 (2017), 985–996. https://doi.org/10.1016/j.chb.2016.10.019

[42] Socket.io. 2022 (accessed Jan 8, 2022). *Socket.io*. https://socket.io/

[43] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. 2015. TVSum: Summarizing web videos using titles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5179–5187. https://doi.org/10.1109/CVPR.2015.7299154

[44] Streamlink. 2022 (accessed Jan 8, 2022). *Streamlink*. https://streamlink.github.io/

[45] Taizan-hokuto. 2022 (accessed Jan 8, 2022). *Pytchat*. https://pypi.org/project/pytchat/

[46] John C. Tang, Gina Venolia, and Kori M. Inkpen. 2016. Meerkat and Periscope: I Stream, You Stream, Apps Stream for Live Streams. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4770–4780. https://doi.org/10.1145/2858036.2858374

[47] Twitch. 2022 (accessed Jan 8, 2022). *Twitch API*. https://dev.twitch.tv/docs/api

[48] Twitch. 2022 (accessed Jan 8, 2022). *Twitch Extensions*. https://dev.twitch.tv/extensions

[49] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[50] Bo Xiong, Yannis Kalantidis, Deepti Ghadiyaram, and Kristen Grauman. 2019. Less Is More: Learning Highlight Detection From Video Duration. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1258–1267.

[51] Saelyne Yang, Changyoon Lee, Hijung Valentina Shin, and Juho Kim. 2020. Snapstream: Snapshot-Based Interaction in Live Streaming for Visual Art. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376390

[52] Matin Yarmand, Dongwook Yoon, Samuel Dodson, Ido Roll, and Sidney S. Fels. 2019. "Can You Believe [1:21]?!": Content and Time-Based Reference Patterns in Video Comments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 489, 12 pages. https://doi.org/10.1145/3290605.3300719

[53] YouTube. 2022 (accessed Jan 8, 2022). *YouTube Data API*. https://developers.google.com/youtube/v3/docs/videos

[54] youtube dl. 2022 (accessed Jan 8, 2022). *youtube-dl*. https://www.npmjs.com/package/youtube-dl

[55] Cong Zhang and Jiangchuan Liu. 2015. On Crowdsourced Interactive Live Streaming: A Twitch.Tv-Based Measurement Study. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Portland, Oregon) *(NOSSDAV '15)*. Association for Computing Machinery, New York, NY, USA, 55–60. https://doi.org/10.1145/2736084.2736091

A PRELIMINARY EVALUATION OF THE ONLINE SEGMENTATION ALGORITHM

A.1 Segmentation Information of the Streams Used

Name	Segment #	Average segment length (sec)	Segment examples
Archade	14	35	7:55 World, 11:30 Magyechon
Suka	6	1144	47:43 Video call app, 1:19:41 Coupang, 1:39:10 Bitcoin
Minu	4	2562	00:00 Opening, 01:32:53 How did I start YouTube
Google	10	1075	00:30 Day Opener, 27:58 What's New in Speed Tooling
Photoshop	5	1384	00:00 Setting and chat, 05:56 solution to the 1st problem
Amongus	21	651	11:22 Crewmate, 46:15 Impostor with Koji
Drawing	3	433	6:00 Draw by Swimming (Warm-up), 29:20 Shadow Shapes

Table 7: Ground truth segmentation information of the seven live streams.

A.2 Results of the Online Segmentation Algorithm

Condition	Archade	Suka	Minu	Google	Photoshop	Amongus	Drawing	Avg.
`standard` & `low-threshold`	28.6	58.8	11.8	52.2	42.9	55.6	80	47.1
`standard` & `high-threshold`	66.7	58.8	23.5	69.6	42.9	66.7	80	58.3
`minmax` & `low-threshold`	41.7	25	50	37.5	44.4	46.2	66.7	44.5
`minmax` & `high-threshold`	66.7	25	75	37.5	44.4	66.7	100	59.3

Table 8: F1 score of the results with a uniform weight of the five factors (`base-coeff`).

Condition	Archade	Suka	Minu	Google	Photoshop	Amongus	Drawing	Avg.
`standard` & `low-threshold`	64.3	60	50	81.8	62.5	75	80	67.6
`standard` & `high-threshold`	92.9	70.6	50	81.8	71.4	85	80	75.9
`minmax` & `low-threshold`	78.6	66.7	100	70.6	80	87.8	66.7	78.6
`minmax` & `high-threshold`	80	71.4	100	70.6	100	88.4	100	87.2

Table 9: F1 score of the results with the optimal weight of the five factors (`optimal-coeff`).

A.3 Optimal Weight Distribution for Each Stream

	Weight with highest accuracy (weight ∈ 1, 2, 3, 4, 5)					Genre
	Visual change	Keywords	Transition	Chat frequency	Duration	
Archade	4	0	1	4	4	Game playing
Suka	5	5	0	0	0	Information sharing
Minu	4	4	1	3	0	Talking
Google	2	2	3	1	4	Tutorial (Web development)
Photoshop	2	4	1	3	2	Tutorial (Photoshop)
Amongus	4	0	0	0	1	Game playing
Drawing	2	2	3	2	3	Drawing

Table 10: Optimal weights of the five factors for each stream.

We could see that the optimal weight distributions are highly related to the characteristics of the stream. In the Suka video which shares information with visuals and verbal explanations, the optimal weights of 'visual change' and 'keywords' were equally high. For the Google video which shares a step-by-step tutorial, the optimal 'duration' and 'transition' weights were high, implying that there were substantial

amounts of transitional keywords in the tutorial as well as breaks between steps. For both of the game-playing streams Archade and Amongus, the optimal 'keywords' weight was zero, which implies that keywords carry less meaning in game-playing streams. For Amongus, the optimal weight of 'visual change' was higher than other factors due to the screen change after each round of the game (each round was considered as a segment for this video).

B EXAMPLE HIGHLIGHT MOMENTS FROM THE USER STUDY

Timestamp		Transcript
Stock	(a) 03:57	The interest in stocks is decreasing but there also seems to be some sectors that will rise.
	(b) 16:35	How about the gaming area? It's good. Games are also seen as contents now.
	(c) 26:16	I met one customer yesterday who's a millionaire.
	(d) 38:27	Then the price goes over the budget. Then since there was no demand for the changing of fume hoods, the B2B construction company just provides them.
Cooking	(e) 07:04	What we are going to make today is Shanghai Pasta, which is something that I used to make almost 10 years ago when I worked in Korea. It's a typical Korean-style Italian pasta.
	(f) 22:23	You'll see why. It's because the cooking method is very different.
	(g) 42:07	Flambé should not be done using wine.
	(h) 1:10:30	It's a genuine Korean-Italian food. This cannot taste bad. Even if a total beginner cooked it, it would taste good.
Game	(i) 07:29	I already have a bad feeling about it. No, there is a possibility. I feel it.
	(j) 30:29	That butterfly. That four dollar butterfly. I've got to catch that.
	(k) 36:05	Let's go outside and rewind time just a little bit. What time, no I mean, where should we go to find out?
	(l) 49:05	Oh it's raining a bit right now. A T-shirt, some outdoor trousers, or maybe a red pair of shorts would help. Ok, let's buy this red one.

Table 11: Example highlight moments from user studies. In the Stock stream, we could see that the highlight detection algorithm captures moments that might attract users' attention such as "*I met one customer yesterday who's a millionaire*" (c). They also capture some key topics covered in the stream such as game and construction industry (b, d). In the Cooking stream, the algorithm captures the moment the dish was introduced and finished (e, h), and interesting parts such as where the cooking method is unusual (f) or talking about the concept of Flambé (g). In the Game stream, the algorithm captured suspenseful or urgent moments (i, j, l), or unusual behavior (k).