Towards Extracting Highlights From Recorded Live Videos: An Implicit Crowdsourcing Approach

Ruochen Jiang*§ Changbo Qu*† Jiannan Wang† Chi Wang Yudian Zheng
†Simon Fraser University §Ohio State University Microsoft Twitter
{changboq, jnwang}@sfu.ca jiang.2091@osu.edu wang.chi@microsoft.com yudianz@twitter.com

***Abstract*—Live streaming platforms need to store a lot of recorded live videos on a daily basis. An important problem is how to automatically extract highlights (i.e., attractive short video clips) from these massive, long recorded live videos. However, algorithmic approaches are either domain-specific, which require experts to spend a long time to design, or resource-intensive, which require a lot of training data and/or computing resources. In this paper, we propose LIGHTOR, a novel implicit crowdsourcing approach to overcome these limitations. The key insight is to collect users' natural interactions with a live streaming platform, and then leverage them to detect highlights. We recruit hundreds of users from Amazon Mechanical Turk, and evaluate the performance of LIGHTOR using two popular games in Twitch. The results show that LIGHTOR can achieve high extraction precision with a small set of training data and low computing resources.**

***Index Terms*—Machine learning, Implicit crowdsourcing, Highlight detection.**

I. INTRODUCTION

Live streaming platforms such as Twitch and YouTube Live fulfill the mission of democratizing live video broadcasting. Users interact with broadcasters by leaving chat messages in real time during live streaming. Once a live stream is complete, the time-stamped messages and the recorded video will be archived.

A recorded video is often very long (from half an hour to several hours). Most viewers do not have the patience to watch entirely but only look for a few *highlight clips* to watch, which typically lasts from a few seconds to less than one minute. For example, it could be an exciting battle or a critical knockdown in a Dota game video. In this paper, we study video highlight detection for live streaming platforms, which aims to automatically detect highlights from a recorded live-stream video. It is an essential task in video analysis. It can save users' time in manually finding the highlights.

Deep learning is one of the dominant research directions in research communities. *We argue that implicit crowdsourcing (combined with simple ML models) is superior to deep learning in many aspects, and should gain more attention from researchers and practitioners in the future*. In the following, we will first identify the limitations of deep learning based approaches, and then propose a novel system to leverage implicit crowdsourcing to overcome these limitations.

* Work done at SFU. Both authors contributed equally to this research.

Given the great success of deep learning in image object detection, it is natural to expand the scope by exploring the use of deep learning in video highlight detection. Several recent papers study how to train a deep-learning model (e.g., CNN, RNN, LSTM) from videos and/or chat messages to detect highlights [1], [2]. While better performance is obtained, they have the following limitations.

- *Require large training data.* For each type of video, they need to label hundreds of videos to train a good model [1].
- *Lack of generalization.* The model trained on one type of video (e.g., Dota game) usually does not generalize well to another type of video (e.g., LoL game).

We propose LIGHTOR, a novel *implicit crowdsourcing* system for video highlight detection with low cost. The main idea is to leverage implicit feedback from viewers' activity. Implicit crowdsourcing has achieved great success in many domains, such as reCAPTCHA for digitizing old books and clickthrough data for optimizing ranking in search engine.

Figure 1 depicts the LIGHTOR workflow. The workflow consists of two major components: Highlight Initializer and Highlight Extractor. The former determines which part of the video could be a highlight, and the latter identifies the exact boundary (start and end) of each highlight. We will use a simple example to illustrate how they work and the challenges that they face. Consider a one-hour video $\mathcal{V} = [0, 3600]$, which starts at 0s and ends at 3600s. Suppose the video has a highlight between 1900s and 2005s, denoted by $h = [1990, 2005]$. The goal is to extract h from $\mathcal{V}$.

Highlight Initializer. Highlight Initializer aims to identify an approximate start position of each highlight so that if a user starts watching the video from this position, she can tell that there is a highlight nearby. For example, 2000 is a good position since it is within the highlight range $[1990, 2005]$ but 2100 is a bad one since it is very far away from the highlight. We observe that most live videos allow users to leave chat messages in real time. This can be leveraged as a kind of implicit feedback. However, these chat messages are short and noisy which is challenging to implement an accurate Highlight Initializer. We will discuss details in Section II.

Highlight Extractor. Suppose the above component returns 2000 as a result. We will add a "red dot" at this position on the timeline of the recorded video (see Figure 1). A red dot can be seen as a hint, which informs users that there could

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00176

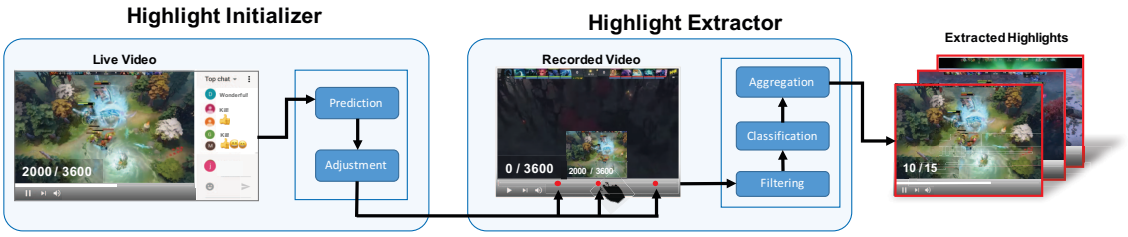


Fig. 1: LIGHTOR: An implicit crowdsourcing workflow for extracting highlights from a recorded live video.

be a highlight nearby. Users can click on the red dot and start watching the video. They may drag the progress bar backward if they find the video clip interesting and want to watch it again, or otherwise drag forward to skip. All these interactions can be leveraged as another kind of implicit feedback to extract highlights. However, user interactions are noisy, which is challenging to implement an accurate Highlight Extractor. We will discuss how to address this challenge in Section III.

The promising evaluation results show that LIGHTOR is able to overcome the limitations of deep learning mentioned above. We believe that implicit crowdsourcing has a big potential in the video analysis field and should gain more attention.

To summarize, our contributions are:

- We study how to leverage implicit crowdsourcing to extract highlights from a recorded live video. We propose LIGHTOR, a novel workflow to achieve this goal.
- We analyze real-world chat messages, and derive a number of interesting observations. Based on these observations, we develop a simple but effective Highlight Initializer.
- We derive a number of interesting observations from real-world user interaction data, and propose a novel Highlight Extractor to identify the exact boundary of each highlight.

The remainder of this paper is organized as follows. We discuss how Highlight Initializer and Highlight Extractor are built in Section II and Section III, respectively. We present experimental results in Section IV and conclusion in Section V.

Please refer to our full report for further details [3].

II. HIGHLIGHT INITIALIZER

This section presents the design of Highlight Initializer. We define the design objective, discuss the design choices, and propose the implementation.

A. Design Objective & Choice

There could be many highlights in a video, but most users are only interested in viewing the top-k ones. Highlight Initializer aims to find an approximate start position for each top-k highlight. Next, we define what is a *good* approximate start position (i.e., a *good* red dot). The goal is to make users see a highlight shortly after they start watching the video from a red dot. Let $h = [s, e]$ denote a highlight and r denote the red dot w.r.t. h. We call r a good red dot if $r \in [s-10, e]$ and $\nexists r'$ s.t. $|r - r'| \leq \delta$, which means r falls within the tolerable range [4] and two red dots should not be too close to each other. δ is a system parameter (120s by default). Further, we define the design objective.

Objective. *Given a recorded live video along with time-stamped messages, and a user-specified threshold k, Highlight Initializer aims to identify a set of k* good *red dots.*

We face different choices when designing the Highlight Initializer. We will explain how the decision is made for each choice (underlined).

Chat vs. Video Data. Comparing with video data, chat data has two advantages. (1) Requiring a small set of training data (e.g., 1 labeled video) to train a good model. But, it is hard to achieve this for video data. (2) Requiring lower computing resources to process chat data.

General vs. Domain-specific Features. We seek to build a Machine Learning (ML) model to identify red dots. General features are independent of video type (e.g., Dota2 game vs. LoL game). In contrast, domain-specific features are highly dependent on the selected domains. For example, *message number* vs "Goal". The former works for any video, whereas the latter might not work for Dota but Soccer. Choosing general features will allow our model to have good generalization.

B. Our Implementation

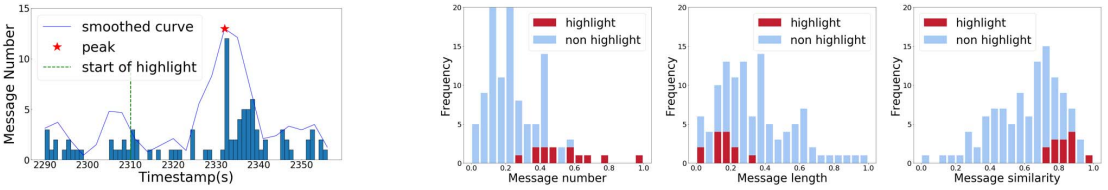
Based on the design choice above, we present a naive implementation and identify its limitations. Then we propose our implementation to overcome these limitations.

A naive implementation is to count which part of the video has the largest message number and put a red dot at that position. Figure 2(a) shows a real-world example. This naive implementation will put a red dot at 2332s. Unfortunately, this implementation does not perform well in practice due to two limitations: (1) The largest message number might be caused by other reasons, such as advertisement chat-bot, instead of highlights. (2) In a live video, people can only chat about a highlight *after* they have seen one. Thus, there is a delay to be aligned between the start position of a highlight and its comments. In Figure 2(a), the delay is around 20s.

Our implementation consists of two stages to address the limitations above respectively.

Prediction. Given a set of chat messages within a short sliding window (e.g., 25s), we build a logistic regression model to determine whether the messages in the sliding window are talking about a highlight or not. We propose three general features of sliding windows for the model.

- *Message Number* is the number of the messages. The naive implementation only considers this feature.
- *Message Length* calculates the average number of words of the messages. When a highlight happens, users tend to leave



(a) The change of message number over time

(b) The comparison of the feature-value distributions between highlights and non-highlights

Fig. 2: Analysis of the Chat Data in a Twitch Video.

short messages. If their messages are long, they typically chat about something else.

- *Message Similarity* measures the similarity of the messages. The higher the similarity, the more likely people will chat on the same topic, instead of random chatting.

We examine the effectiveness of each feature. Figure 2(b) shows the analysis results of a random video with 1860 chat messages. 13 out of 109 non-overlapping sliding windows are highlights. For each feature, we can see that their distributions are quite different.

Adjustment. Given a set of messages in a sliding window which are predicted to be talking about a highlight, we aim to estimate its start position.

Since people can only comment on a highlight after they have seen it, we first detect the *peak* in the sliding window, where a peak represents the time when the message number reaches the top. Then, we train a model to capture the relationship between the peak's position ($\mathsf{time}_{\mathsf{peak}}$) and the highlight's start position ($\mathsf{time}_{\mathsf{start}}$).

The current implementation considers a simple linear relationship, i.e., $\mathsf{time}_{\mathsf{start}} = \mathsf{time}_{\mathsf{peak}} - c$, where c is a constant value. We can learn the optimal value of c from training data. This simple linear relationship leads to good performance as shown in later experiments. We defer the exploration of more complex relationships to future work.

III. Highlight Extractor

This section presents the design of Highlight Extractor. We first define the design objective, then discuss the challenges, and finally propose the detailed implementation.

A. Design Objective

Highlight Extractor aims to identify the boundary (start and end positions) of each highlight using user interaction data.

User Interaction Data. While watching a video, a user may have different kinds of interactions (e.g., Play, Pause and Seek Forward/Backward). We analyze user interaction data, and find that if a certain part of video has a large number of views, then this part is very likely to be a highlight. Based on this observation, we transform user interaction data into *play data*, where each record is in the form of: $\langle user, \mathsf{play}(s, e)\rangle$. For example, $\langle \text{Alice}, \mathsf{play}(100, 120)\rangle$ means that the user Alice starts playing the video at 100s, and stops at 120s. If the context is clear, we will abbreviate it as $\mathsf{play}(s, e)$.

We leverage the play data to extract highlights. Note that if a play is far away from a red dot, it may be associated with another highlight. Thus, we only consider the plays within $[-\Delta, \Delta]$ around a red dot ($\Delta = 60$s by default).

The following defines the objective of Highlight Extractor.

Objective. *Given the play data* $\mathsf{play}(s_1, e_1)\ , \cdots\ , \mathsf{play}(s_n, e_n)$ *w.r.t. a red dot, Highlight Extractor aims to identify the start and end positions of the highlight of the red dot.*

B. Our Implementation

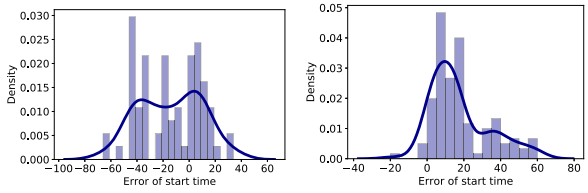
Each play can be considered as a vote for the highlight. For example, $\mathsf{play}(190, 210)$ means that the user votes 190s and 210s as the start and end positions of the highlight. Users may have different opinions about the highlight. We can aggregate their opinions using median which is robust to outliers. Thus, the new start and end positions are computed as $\mathsf{median}(s_1, s_2, \cdots, s_n)$ and $\mathsf{median}(e_1, e_2, \cdots, e_n)$.

Unfortunately, it does not always work well in real world data. We have a very interesting observation: whether this idea works well depends on the relative positions of the red dot and the highlight. There are two possible relative positions:

Type I: the red dot is *after* the end of the highlight;

Type II: the red dot is *before* the end of the highlight.

We observe that for *Type I*, many users may miss the highlight, thus their play data are not reliable indicators of the highlight. For *Type II*, their play data follow a similar pattern (Example shown in Fig 3). This observation poses two new questions: (1) *how to determine whether a red dot belongs to Type I or Type II?* (2) *after being classified, how to aggregate its play data?*



(a) Type I (b) Type II

Fig. 3: Distribution of the difference between each play's start position and the ground-truth start position.

To address the questions, our implementation is as follows:

Classification. Given a red dot, we build a classification model to determine whether it belongs to *Type I* or *Type II*. We find that this (unknown) relative position has a strong correlation with the (known) relative position between the red dot and observed play data, so we identify the following three features:

- *# Plays after red dot* computes the number of plays which start at or after the red dot.
- *# Plays before red dot* computes the number of plays which end before the red dot.
- *# Plays across red dot* computes the number of plays which starts before the red dot and ends after the red dot.

Aggregation. Different aggregation strategies are proposed for *Type I* and *Type II*, respectively.

For *Type II*, as Figure 3(b) shows, the play patterns of most users are similar. The median of the start time offsets is between 5 and 10. This is because that the most exciting part of the highlight usually happens a few seconds after its start point, which causes users to skip the beginning of the highlight. This kind of error is tolerable. Therefore, we can use median to aggregate their play data.

For *Type I*, as seen from Figure 3(a), the distribution of start time offsets is rather random. Therefore, we need to collect more play data. Our main idea is to convert a red point from *Type I* to *Type II*. Given that *Type II* can collect high-quality play data, once a red dot is converted to *Type II*, we can get high-quality play data as well. Specifically, once a red dot is classified as *Type I*, we will move it backward by a constant time (e.g., 20s) and collect new interaction data based on the new red dot location. If the new red dot is classified as *Type II*, we apply the *Type II*'s aggregation approach; otherwise, we move it backward by another 20s.

IV. Experiments

We evaluate LIGHTOR on real live video data. Here we compared our method with the state-of-the-art deep learning method. We provide a reproducibility report in the github repo (https://github.com/sfu-db/Lightor_Exp), including data description, data preprocessing, model parameters and configurations, and a Jupyter notebook to reproduce all experimental figures. Due to space limitation. We leave further comprehensive experiments and analysis in our full technical report [3].

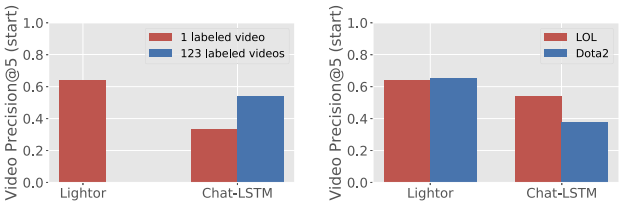
A. Experimental Settings

1) Metric: Video Precision@K (start) is to evaluate the precision of the identified start positions of highlights. Since people typically cannot tolerate more than 10s delay, we say a start position x is correct if there exists a highlight $h = [s, e]$ such that $x \in [s - 10, e]$. Video Precision@K (start) is defined as the percentage of correctly identified start positions out of the k identified start positions.

2) Data: Game videos dominate mainstream live streaming platforms such as Twitch. They were also used to evaluate the state-of-the-art highlight detection approaches [1]. Therefore, we evaluated LIGHTOR using two popular games from Twitch: Dota2 and LoL.

Videos data. (1) Dota2. We crawled 60 live videos in Dota 2. Each video, labelled manually by experienced players, contains 10 highlights on average. (2) LoL. We selected 173 live videos of League of Legends (LoL) from NACLS dataset [1]. Each video contains 14 labeled highlights on average.

Chat and Play Data. LIGHTOR relies on two kinds of user data: chat data and play data. For chat data, live streaming platforms make the data accessible through their open APIs. To our best knowledge, there is no public interaction data available. We simulated the procedure to generate Play data by recruiting game fans from Amazon Mechanical Turk (AMT).



(a) Training Size Comparison. (b) Generalization Comparison.

Fig. 4: Comparison of LIGHTOR and Chat-LSTM (Test Data: 50 LOL Videos).

B. Comparison with Deep Learning

We compared LIGHTOR with the state-of-the-art deep learning approach Chat-LSTM [1]. Both of them use LoL chat messages data to train model. We used Precision@5 (Start) to evaluate the performance.

We first compare LIGHTOR with Chat-LSTM in terms of training data size. Figure 4a shows the result. We can see that LIGHTOR only needed to label a single video in order to achieve a high precision, but Chat-LSTM did not perform well with a single labeled video. This is because that LIGHTOR detects highlights based on a small number of generic features. We increased the training size of Chat-LSTM to 123 labeled videos. Chat-LSTM's performance got improved but still did not perform as good as LIGHTOR. This is because that Chat-LSTM did not consider that there is a delay between chat messages and video content.

We then compare LIGHTOR with Chat-LSTM in terms of model generalization. Figure 4b shows the result. We can see that for LIGHTOR, the model trained on one game type (LoL) can still achieve high precision on another game type (Dota2). LIGHTOR has good generalization because the selected features are very general. In comparison, For Chat-LSTM, there is a big performance gap between LoL and Dota2. That is, if we trained the Chat-LSTM model on one game type (LoL) and then tested it on another game type (Dota2), the model did not generalize well.

V. Conclusion

We presented LIGHTOR, a novel implicit crowdsourcing workflow to extract highlights for recorded live videos. LIGHTOR consists of Highlight Initializer and Highlight Extractor. We explored different design choices and justified our decisions for each component. This paper demonstrates a great potential of the application of implicit crowdsourcing to video highlight detection. One promising future direction is to investigate how to apply this idea to other video analysis tasks (e.g., video querying, video summarization, and video indexing).

References

[1] C. Fu, J. Lee, M. Bansal, and A. C. Berg, "Video highlight prediction using audience chat reactions," in *EMNLP*, 2017, pp. 972–978.

[2] H. Yang, B. Wang, S. Lin, D. P. Wipf, M. Guo, and B. Guo, "Unsupervised extraction of video highlights via robust recurrent auto-encoders," in *IEEE ICCV*, 2015, pp. 4633–4641.

[3] R. Jiang, C. Qu, J. Wang, C. Wang, and Y. Zheng, "Towards extracting highlights from recorded live videos: An implicit crowdsourcing approach," *arXiv preprint arXiv:1910.12201*, 2019.

[4] "Response Times: The 3 Important Limits," https://www.nngroup.com/articles/response-times-3-important-limits, 1993, accessed: 2019-06-01.