**Data Science team#5 Project Report**

Kim JungHoon, Lee JongMin, Yoon JiSup

1. **Asking a Problem**
   1. **background**

Online broadcast conducted by ‘streamers’ has become increasingly popular worldwide. Twitch, and YouTube is leading the market of online streaming. But not everyone is present to watch their favorite streamers play live, and many of them wants the core essence of the long streams, which tends to be well over several hours per day.

To fill these needs, Twitch, for example, has dedicated archive section that only serves as saving only ‘highlight clips’. That is, whenever interesting moments occurs during live plays, that specific portion of the video is cherry-picked and saved to separate location. People can then revisit those moments whenever they want, and provides the most out of those long live plays for those of who missed the live play.

To pick out these highlighted moments, people called ‘editors’, typically hired by each streamer, has to skim through long live plays to pick out the highlighted clips. This is time consuming, and since done by human manually, may skip very interesting moments, and personal taste of the editor may disturb the generality of the stream.

There have been several tries to automate these ‘cherry-picking’ process, using video analysis since the visual information is main contents for the streaming service. But analyzing video is very computer-intensive, and requires lot of storage spaces, along with good bandwidth to get the whole video.

* 1. **proposed solution**

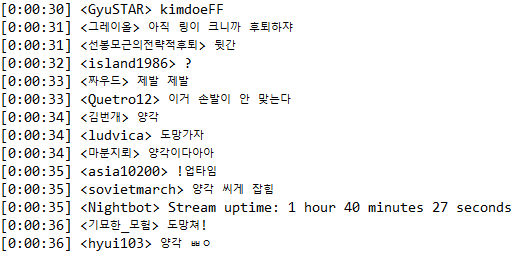
To relieve this problem, we propose video highlight estimation system based on the chat log of the users who watched the play live. We believe analyzing only the chat log prevents most of the problem stated above, and saves costs and time for both streamers and editors.

1. **Collecting Chat Data**

Thankfully, PetterKraabol have open-sourced ‘Twitch Chat Downloader’(tcd) which downloads the chat log of a video uploaded on the Twitch. Using this library, we collected total 5.7million chats from 108 videos from one streamer, consisting of 551 hours of stream through 3 months. To compare our results, we used 3 separate videos from this streamer which has total 79 highlighted clips.

1. **Exploring the Data**
   1. **looking at the data**

Raw data collected from tcd is as follows.

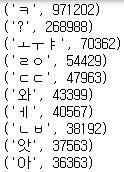


**figure 1:** raw data. Each line consists of time, user id, and corresponding chat.

Raw chat has to be cleaned first. In figure 1, there are 3 cases of data that needs to be eliminated. First one is the first chat, with user id ‘GyuSTAR’. Its chat is displayed as image file provided from the platform, and logged as a corresponding tag. This data can be useful for some cases, but for this experiment, it was excluded. Second case is fifth chat from the bottom, with username ‘asia10200’. Its chat was ‘!업타임’ and exclamation mark at the beginning of the chat is the command for calling the bot residing at the channel. The bot’s response is located at the third bottom row, which is third case to be eliminated. All of these three cases consisted about 5% of the chat.

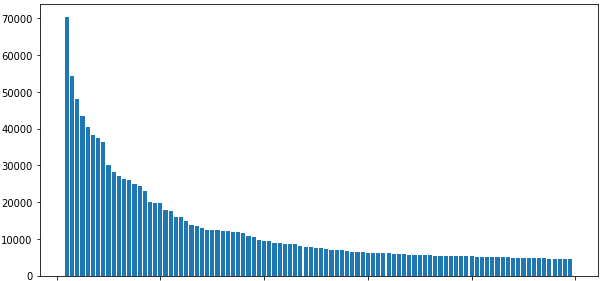
* 1. **common words**

Common words used by viewers may provide some insight to the data. It’s not difficult to list the most used words in descending order, but some cleaning should be done here, too. Monophonic character such as ‘ㅋ’ is used in various length, such as ‘ㅋㅋㅋ’ and ‘ㅋㅋㅋㅋㅋㅋ’. For this case, the length easily went up to over 10 repeated characters. Merging these cases, most common words are as follows.



**figure 2:** Most common words

Two most commonly used words are off the scale relative to the others. To visualize this data, frequency of the most common words are as follows, not including first two dominating words.



**Figure 3:** frequency of the common words. Two most frequent words are excluded.

* 1. **Tendency**

Highlighted clips contain meaningful moments during the stream, which can be interpreted as certain response came out from the viewers. This can be treated as an essence of the streams, and can expect more views from the people who didn’t watch the stream live. For live viewers, this response is expressed as a chat, which are reaction to those moments.

1. **Estimation Model**

Based on the observation that highlighted clips has high ‘chat per second’, we assume, by analyzing the frequency of the chat, we can detect highlighted clips selected by the human editor, and find more highlight candidates that human editors missed. We propose following three method of detecting highlights from the video based on the chat.

* 1. **Type of model**
     1. **Rise**

Rise method records ‘chat per second’ by every second. This record can be done with naive chat data, or **moving average** with certain window. If ‘chat per second’ value is incremented without decreasing for certain **step**, then that point is considered highlight candidate. Hyperparameter with this approach is data method(naïve or moving average), and step.

* + 1. **Burst**

Burst method records the occurrence of **given word** per second. If given word occurs more than the **threshold** during given **duration**, that point is considered to be a highlight candidate. Hyperparameter with this approach is word, threshold, and duration.

* + 1. **Trendy**

Trendy method is more generalized version of Burst Method. This method records occurrence of every word per second, and if any word occurs more than the **threshold** during given **duration**, that point is added to the highlight candidate. Hyperparameter with this approach is threshold, and duration.

* 1. **Fitting model**

If we were to fit the model, the trainable parameters are the hyperparameters of the model. Unfortunately, we couldn’t find the appropriate loss function to train this model. Instead, we tackle this problem with different approach, by exploring every case that might output meaningful result.

For ‘Rise’ method, we tested on naïve, and moving average dataset. For moving average, window size were set as 10, 30, 50, 70, 100. For step size, 1~10 were selected.

For ‘Burst’ method, we chose the word ‘ㅋㅋㅋ’ to be a given word, since it represents the laughter and occurs throughout the dataset. Threshold was given from 10 to 100 with step size of 10, and duration was given from 1 to 10.

For ‘Trendy’ method, threshold was given from 10 to 100 with step size of 10, and duration was given from 1 to 10.

* 1. **Validating model**

Using one of the methods proposed above, our result is the lists of predicted highlights, expressed as a pair of video number the chat is from, and the timestamp of that specific candidate. Then, we compare this list with real highlight timestamps extracted from uploaded videos. We manually recorded the timestamp from the uploaded video, using the chat log shown in the clips. Then we can apply binary classifier for this experiment.

Comparing two lists of highlights, we consider the model appropriately predicted the highlight if the estimated timestamp is within the minute range of the real highlight timestamp. Result can be classified as follows.

True Positive: Model correctly predicted actual highlights

True Negative: Model didn’t pick this time as highlight, and it is not actual highlight.

False Positive: Model predicted as highlight, but actually is not.

False Negative: Model didn’t predict as highlight, but actually is.

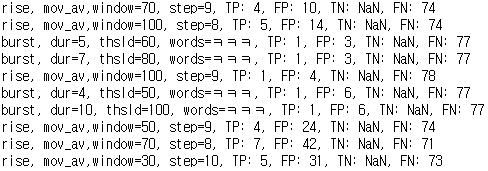
After conducting and saving all the results, then we use following three evaluation method to pick best results. Since NP value is relatively bigger than others, Accuracy measurement was not considered for this experiment.

Precision = TP/(TP+FP)

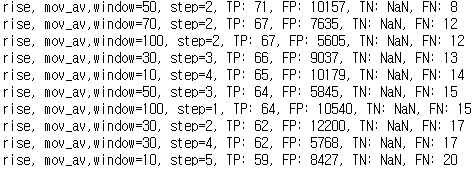
Recall = TP(TP+FN)

F-score =2\*Precision\*Recall/(Precision+Recall)

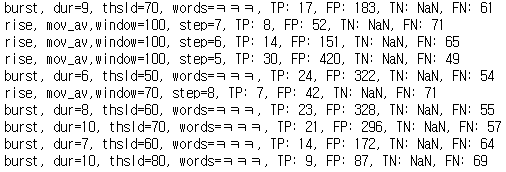
Top 10 of each scoring method is as follows.



**Figure 3.1:** top 10 ‘Precision’ scoring methods.



**Figure 3.2:** top 10 ‘Recall’ scoring methods.



**Figure 3.3:** top 10 ‘F-Score’ scoring methods.

Value of TN was recorded as NaN in the result, because TN value was calculated separately. In actual calculation, appropriate value was used for every experiment.

* 1. **Analyzing Result**
     1. **Precision**

From precision perspective, burst and rise w/ mov\_av method performed best. However, despite the small number of FP, recall rate was too low to be used for the average streamer who wants to have decent amount of highlight clips. Streamers who want their video with high compression can adopt this strategy.

* + 1. **Recall**

Prediction methods that has high recall score was found unusable for this case. To minimize FN value, prediction threshold was set too low so that model predicted over several thousand clips to be highlights for the dataset, which was unlikely. Rise method with low step tended to dominate the highest rankings, since they had very few, and easy constraints to fulfill.

* + 1. **F-score**

Combining both precision and recall score, prediction models with high F-score gave overall best results. Rise and burst method both performed well, and number of FP was relatively low compared to methods from high recall score.

1. **Conclusion**

Overall performance of our model in terms of precision and recall was below our expectation. Even high F-score had more FP than TP, with lot of FN with them. However, after looking through the parts or the original streams that were counted as FPs from the model (that is, our model predicted as a highlight but it wasn’t from the highlight videos editors uploaded) turned out they could be legitimate highlight clips from our perspective. It may be interpreted that the editor who created the highlights had its own taste, or simply had missed many clips and our model discovered it. It would be best to actually make our version of highlight clips from our models’ predictions for comparison, but lack of time prevented us from creating one. We are pleased with our results, and had lot of invaluable experience going through this project.