CS489 Group Report: Comment Clustering

Team 12: Soyoung Yoon, Jinwoo Jeon, Jisu Choi, Jeongyeon Jeon

We open our github link for code: https://github.com/wltn0029/comment-clustering

I. Introduction

Our project was implemented to improve the problems with the current YouTube comment UI. Currently, there are two main problems with YouTube UI. First, it is not easy to see various kinds of opinions at a glance due to the simple enumerated UI. In particular, the ranking of comments is adjusted with its algorithm determined by YouTube. Users cannot control the opinions they want to see. Second, the views of comments tend to be biased to one side. It's easy to get a biased view of the video by just looking at a few comments on top of YouTube. In other words, when opinions are going to be formed on a specific video, only comments that are biased to one side are likely to result in an undesirable view. Likewise, there is a problem in that personal view may be adjusted by malicious manipulation of opinions.

Therefore, we classify YouTube comments into three categories based on their attitude toward the video: Positive, Neutral, and Negative. Through the UI showing each item's comments, the user can see more diverse opinions at a glance than before, and the above-described problems can be improved.

II. Front-end description

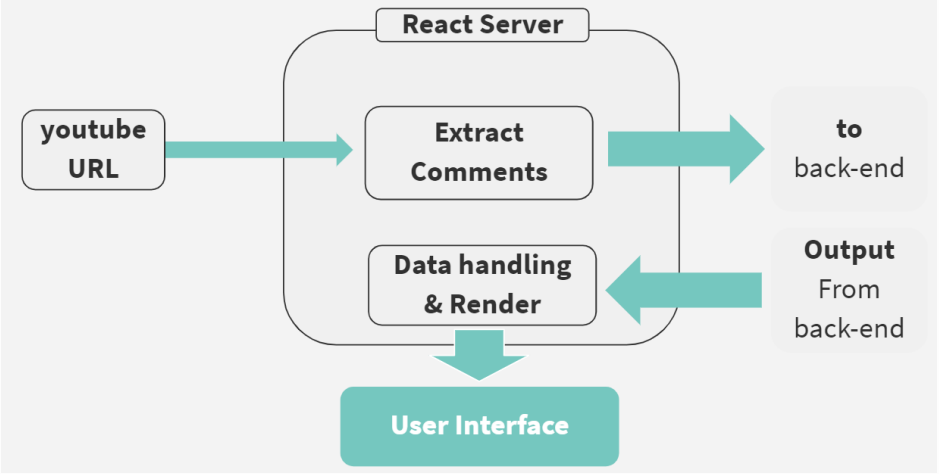

Figure. 1 Logic diagram of back-end

The overall logic flow of the front-end is shown in Fig.1. It extracts the comments from the user's input of the YouTube video URL and sends them to the back-end. After that, the user interface shows receiving the classified output from the back-end (positive, negative, negative).

1) Service Design

In order for users to see the various responses of comments at a glance, a representative comment for each category, positive, negative, and neutral, is shown on the first page. All comments belonging to the selected category are shown on the second page so that users can review the responses in more detail by category. The information provided is kept to a minimum so that people can focus more.

The overall design color was set to light blue, and colors were not divided in order not to prejudice each type of opinion. And it is designed so that only the contents can be grasped at a glance using the clean font.

In the service page, three kinds of information is shown to the user.

- **Video Information**

  Information from youtube url's video is shown on top of the page. Users can check topics covered in the video they select. Information from the youtube video includes video image, channel name, title and descriptions. If the description is too long, we shorten it with the show more button.

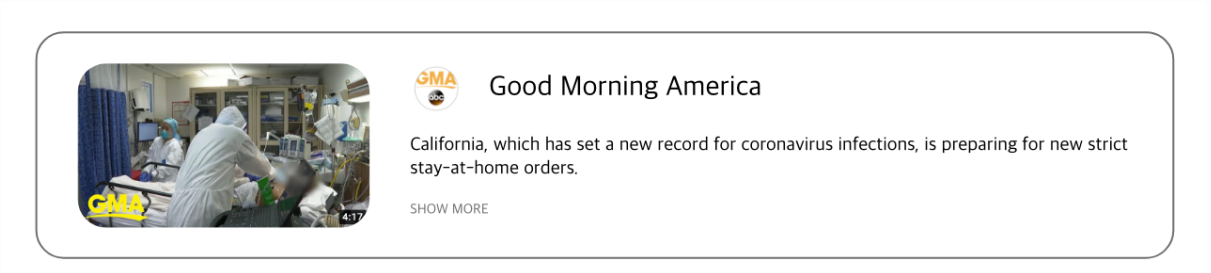

Figure. 2 Video Information Design

- **Categorized Comments**

  We categorized the comments into three categories, positive, negative, and neutral, depending on the content of the comments. When users fill out the YouTube link on the input box and press the *rearrange* button next to it, they can see the representative comments by category. A category's name is written above the comments.

Representative comments are selected randomly among comments in the same category. Users receive an equal amount of comments (one) for each category.

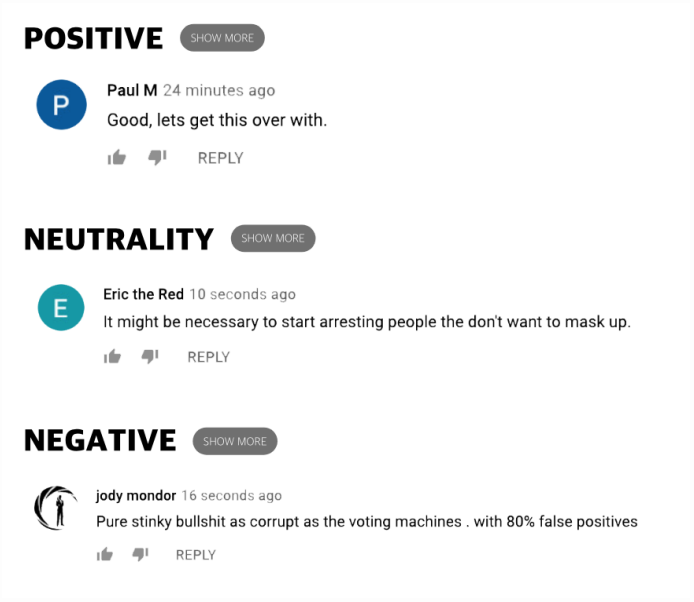

Figure. 3 Comments Design

- **Thread of Comments per Category**

  Users can explore more diverse comments on each category by pressing the *show more* button. The categories and buttons are kept on top so that they can be simplified at any time by pressing the button. When viewing more opinions, only one category of opinions can be viewed. Users have to close the thread to see other categories.

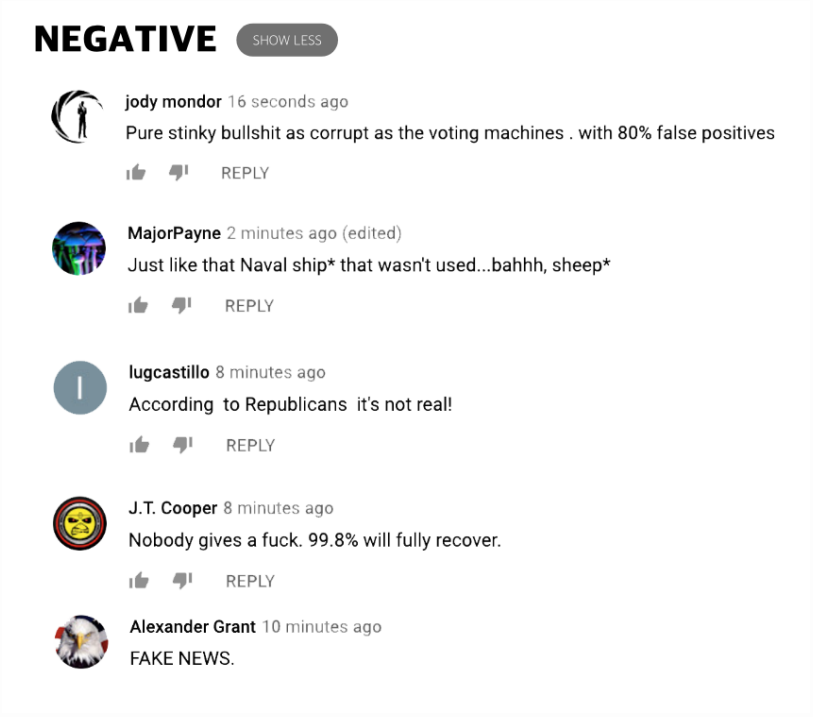

Figure. 4 Thread of Comments Design

2) Data Crawling

All data related to youtube video, information of video and comments, is crawled using Youtube Data Api (https://developers.google.com/youtube/v3). We only use comments in the main thread, excluding those in the thread per comment.

- **Video Information**
  - Command Used

    https://www.googleapis.com/youtube/v3/videos?part=snippet&key=${apiKey}&id=${videoId}
  - We include the title of a video and additional description written by the uploader of a video.
- **Comments**
  - Command Used

    https://www.googleapis.com/youtube/v3/commentThreads?part=snippet&key=${apiKey}&videoId=${videoId}&maxResults=100
  - We limit the number of comments as 100 since we determine more comments would be difficult for users to consider.

3) Deployment

The service is deployed on the web using github deployment server. We decide to provide service on the web for the following reasons :

- Web is accessible from both mobile phone and laptop.
- We aim to improve target users' accessibility to the service who communicate a lot on youtube given most youtube watchers' environment is on the web.

III. Back-end description

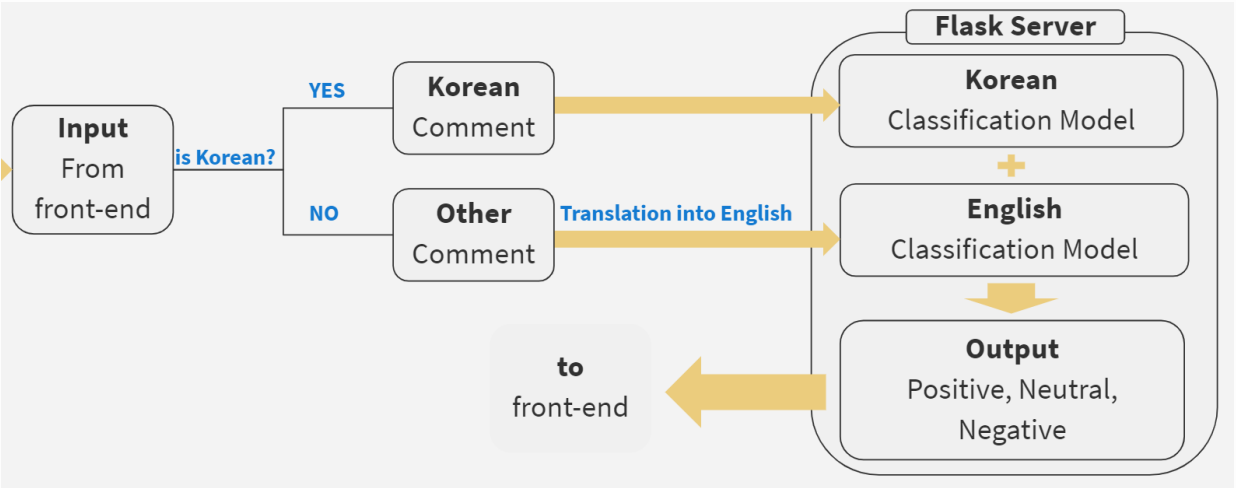

Figure. 5 Logic diagram of back-end

The overall logic flow of the back-end is shown in Fig. 2. The comments list from the front-end is divided into Korean comments and other language comments, and sentimental classification is performed using each model. Finally, each classification result is merged and sent to the front-end.

1) Models, Datasets, Training details.

We train a total of two models. The first one is for english sentiment classification(E-BERT), and the second one is for Korean sentiment classification(K-BERT). For both models, we utilized the pretrained BERT from the huggingface transformers library [1].

General. We use the AdamW optimizer with epsiolon 1e-8, and a linear scheduler with 10% warmup step out of total training steps. We load the pretrained model and finetune it with 3 epochs with each sentiment dataset. We trained the model with a binary sentiment classification task.

K-BERT. Learning rate is 2e-05, and the training seed is set to 42. We use the colab GPU notebook when training. For training the K-BERT, we reference the existing colab notebook that trained BERT with naver movie reviews.(NSMC) [3] We train by 3 epochs and get 87% accuracy on the test set.

E-BERT. We load the pretrained bert-base-uncased model, then fine-tuned with using the sentiment140 dataset. Sentiment140 dataset is extracted using the huggingface datasets library [2]. There are a total of 1600000 train dataset, and 498 test dataset. Learning rate is set to 5e-05. We use the P40 GPU model when training. By manual inspection, we found out that there were only 2 classes(positive, negative) from the training data, and there were 3

classes(positive, neutral, and negative) for the test dataset. Therefore, we thought that it was inappropriate to test the dataset on unseen classes(neutral) when training. As a result, we only conduct training and do not report any accuracy results, evaluated from the test dataset. We upload the fine-tuned K-BERT and E-BERT on google drive. The links are also present in our README of our github repository. The codes used to train K-BERT and E-BERT are also open inside the backend/model directory.

2) Classification Methodology (How we classified neutral text)

When searching for datasets that could be used to fine-tune our sentiment classification model, we find that most of the datasets are composed of only two(binary) labels: positive and negative. However, our objective of this project was to classify 3 classes: positive, negative, and neutral. Therefore, we needed a way to define how we will classify neutral text. The TextBlob python library also conducts sentiment analysis and output sentiment scores ranging from -1 to 1. If the sentence is negative, it outputs -1, and if the sentence is positive, it outputs 1. It outputs near 0 for neutral sentences. Although this library is not based on machine learning but rather rule-based, we thought this could be a good supplementary for revising our binary classification models. So, for E-BERT, before we input the sentence to the model, we first conduct analysis using TextBlob and if the output scores are near 0, (the absolute value is less or same as 0.1) we classify it as neutral. Otherwise, we use the E-BERT to classify the sentences.

However, TextBlob only treats english sentences. Therefore, we needed to think of another way to classify neutral sentences for Korean sentences. When we input a sentence, our trained model outputs numbers with the dimensions to be the number of classes; since we are doing a binary classification, we get two dimensional outputs from the model. There, we take the argmax and select the index with higher value to be the predicted class. Inspired by this scheme, we first made an assumption that the output of the classification model is linear with the sentiment polarity: that is, the higher the model outputs for index related with positiveness, the more positive the sentence is, and the lower score the model outputs for index related with negativeness, the more negative the sentence is. After making this assumption, we can also say that the more difference the model output has in relation to positive & negative scores, the more certain, or more polar, the output is. Followed by this assumption, we made a heuristic rule that if the difference from the value of positive index outcome and the value of negative index outcome is smaller than 1, we classify the sentence as neutral.

3) Model Server building

In our project implementation, the front-end server that interacts with the user and the back-end server that executes the model are separated into separate hosts. Therefore it was necessary to build a model server to communicate with the front-end server.
Since the classification model described above uses GPU parallel computation, sufficient computational resources: GPU resources and RAM resources are required for the back-end server's host computer. Therefore, a host computer with hardware specifications, as shown in the table below, was used as the model server's physical host.

GPU	NVIDIA GeForce GTX 1650
CPU	Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz 8-cores
RAM	16 GB
CUDA ver.	11.1

Table. 1 Hardware specification of back-end server computer

Since the back-end uses a model that has already been tuned through fine-tuning on a pre-trained model, the above hardware specification can be said to have sufficient specifications for the project.

The Python Flask Web framework was used as a software wrapper for building a model server. Server’s physical mapping informations using Flask are as follows:

- internal IP address: “0.0.0.0”, which means broadcast to the network
- internal port number: 8080
- external IP address: 143.248.144.129
- external port number: port forwarding was used

Therefore, by setting the Flask server in the host computer through this physical mapping, it is configured to allow data transmission and reception with the front-end, as shown in the figure below.

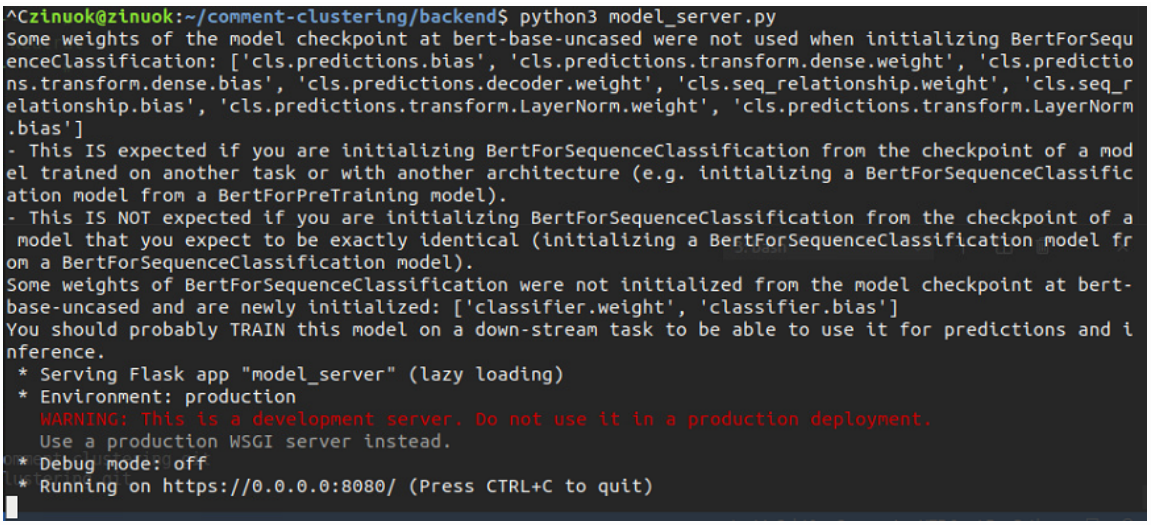

Figure. 6 Execution of model server

Fig. 3 shows the running of the model server using flask on the host computer.

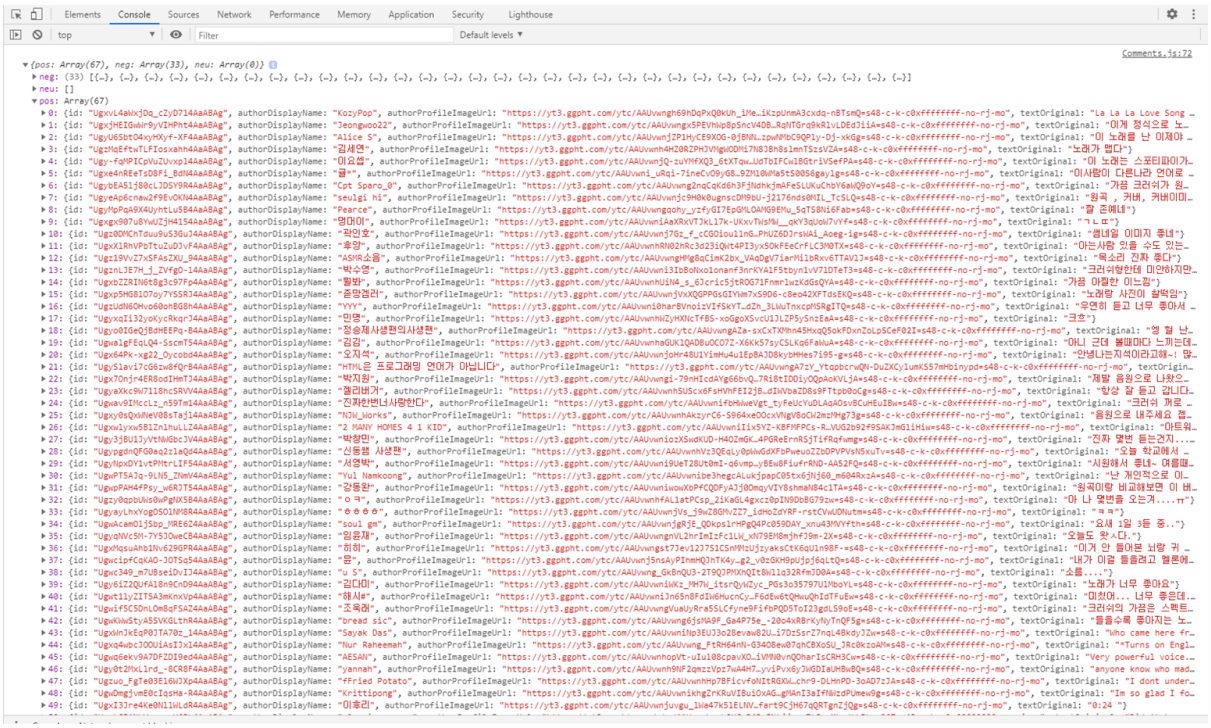

Figure. 7 Response output from back-end server

Fig. 4 shows that the classified results received from the model server were shown in the network console of the Chrome browser by giving an arbitrary youtube URL for the test.

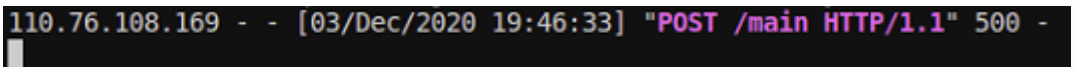

Figure. 8 Request input from front-end server

Fig. 5 shows that the back-end server received the POST request from the front-end server correctly. After receiving the request from the front-end server, the back-end server sends classified output as shown in Fig. 4.

Until now, there are issues related to HTTPS certification, since we had to use HTTPS protocol with private certifications. Therefore, in the currently implemented phase, the model server can receive the request only from a specific client host (for example, Chrome browser on a specific laptop) and send a response.

4) Application function

The following describes the function's logic implemented in the model server for actual classification and data transmission/reception. The overall logic flow is the same as in Fig. 2.

Function: do_analysis( )

STEP 1) garbage collection

collects garbage for caches created by previous analysis work.

STEP 2) get request input from front-end

receives dict-type data sent from the front-end and extracts it in an appropriate format.

STEP 3) separating Korean comments

Since there are two models we use, the Korean and BERT models, Korean comments are separated for use with the Korean model.

STEP 4) For other languages, translate them into English

All other languages, including English, are translated into English to use the BERT model. The translation was done by using the google translator API distributed by Google [4].

STEP 5) extract plain text from comment input

To perform sentimental analysis, plain text is extracted from each comment element.

STEP 6) sentimental analysis

After performing sentimental analysis using each model, the score is calculated as shown in III-2).

STEP 7) classify comments according to sentimental scores

Each comment corresponding to the score is mapped and classified into positive/neutral/negative items.

STEP 8) make output

The output is made in the form of a dict whose keys are pos, neg, and neu.

STEP 9) send the output to the front-end

The maximum number of comments accepted for a YouTube URL input is 100. If this is processed as a simple loop, the time complexity of the do_analysis function will be O(n). That does not fit our project's purpose that guarantees real-time performance. Therefore, to reduce the computing time, STEP 3~7 performed parallel processing using python's multiprocessing library.

The number of CPU cores used for parallel processing is 6, which is 3/4 of the total number of CPU cores described in Table 1.

IV. User Feedback

After implementing the user interface and model, we tested our service for 12 users to get user feedback and contemplate on our service's overall design. Most of them agreed on the necessity of our service, but they pointed out low accuracy of model classification and limited control over standard of category.

1) Participants

12 people who are in their 20's participated in our service's BETA-test. Given the main purpose of service providing a new interface of youtube comments, the test was conducted on the most actively participating age group in Internet communication.

2) Method

Survey is divided into two parts : before using service / after using service. Survey before using the service aims to look into users' thoughts about youtube comments. Survey after the service aims to investigate 1) whether the purpose of service is conveyed properly to users, 2) utility of the service, 3) things to improve. Users are required to use the service for the same YouTube link to maintain the equivalence of their answers. We guide users to participate in the survey through google form.

- Youtube link used in survey
  (korean version) https://www.youtube.com/watch?v=Tm9Wzzr-DUI
  (english version) https://www.youtube.com/watch?v=MXuog-hJfes

In order to get users' response clearly, we choose a video with a topic that could clearly divide the opinions. In this survey, we use google form to guide users and aggregate their responses.

3) Result

- Before using service

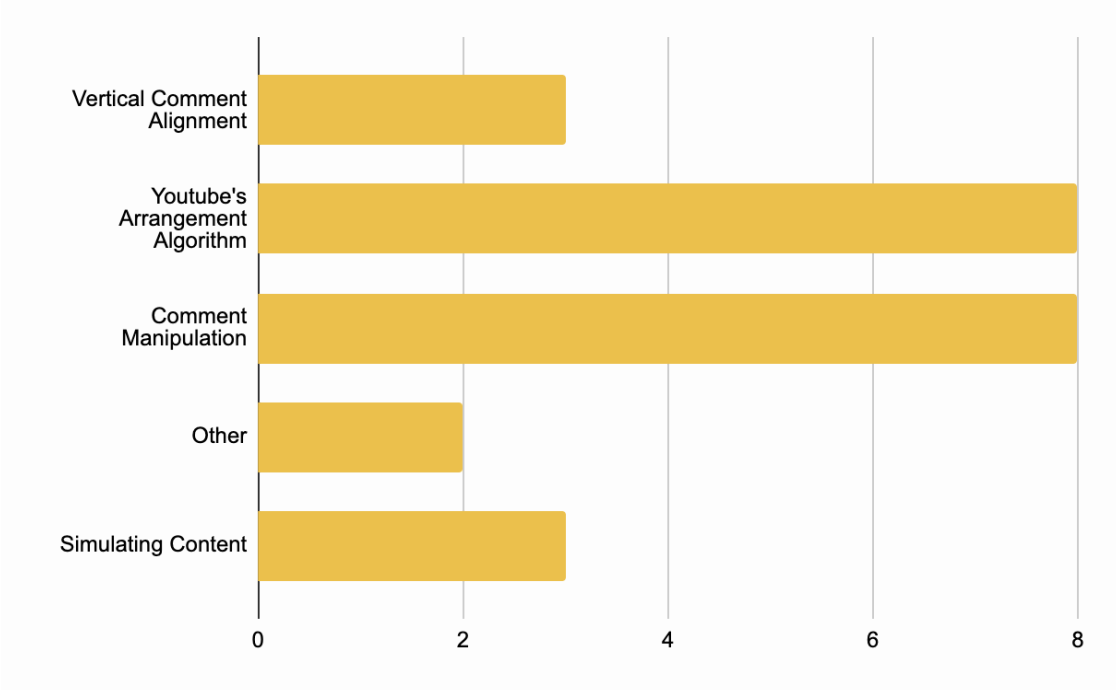

Chart. 1 What do you think youtube comment's problem is?

As seen in chart 1, 33% of users selected youtube's comment arrangement algorithm as the problem of the youtube comment design. Also 33% of users responded they used to realize comment manipulation when using youtube. Those are what exactly we point out as the problem of the youtube comment system which we aim to improve in the service. Some other problems users referred were provocative content, vertical comment alignment, accounting for 12.5% respectively.

- After using service

We investigated the users' responses to the service for the following three points : 1) exposure to diverse opinions, 2) effect to build perspective, 3) necessity of the service.

1) Exposure to diverse opinions

All participants responded they could see more various opinions compared to youtube comments. It implies we implement the user interface properly fitting well with the purpose of service.

2) Effect to build perspective

We compared the view changes on the same video's topic using YouTube and our service. Surprisingly, in both cases, the same percentage of respondents replied they would affect their perspective on the video's topic and vice versa. The result was analyzed as follows :

- Comments were not classified as one of categories, positive, negative, neutral, accurately enough to derive significant differences between youtube comments.
- As we selected representative comments randomly, they did not express the nature of the category explicitly.
- Since users hadn't built their own perspective on the topic we chose, they tended to be affected easily with others' opinion.

For any reason among them, it is clear that we should improve our service design to have a better impact on building opinion than YouTube.

3) Necessity of the service

Almost every participant except one responded they were willing to use our service if available. It implies not only users ordinarily recognize the youtube comments system has a problem, but also they interact a lot through comments as wanting new design of comments. This suggests that more research should be done on the design of comments as the comment is where internet communication occurs actively.

Table 2 shows aggregated responses on closed question survey conducted after using our service.

Question	Yes	No
See more diverse opinions	12	0
perspective change on video's topic using our service	8	4
perspective change on video's topic using youtube	8	4
Willing to use services	11	1

Table. 2 User's responses on each closed question

- **Things to Improve**

  We also investigated things to improve our service. Users responses are divided into two parts : 1) Usability and 2) Service design.

  1) Usability

  - **Video available on the service page**

    It was inconvenient for users to move around between the video page and the comment. Since our service is based on youtube comments, implementing the service as an extension form would lead to supplementary benefit : no need to move pages / explore thread of interested comment. We leave it as future work.

  - **Additional information on the meaning of category**

    Since we only provide the name of the category, it was difficult for some users to understand the meaning of the category. It is very important for users to know the meaning of the category in our service, so we agree on the necessity to supply information on what category means.

  - **Proportion of each category**

    Users responded providing a proportion of each category would help them to build perspective on the topic of video. But we didn't implement this one because we thought being provided with a proportion of category can bias users toward the side with more supporters. Since our service aims to prohibit that, we exclude such information.

  2) Service design

  - **Diversifying category**

    Some users responded they wanted more dynamic interaction in the service page, especially for categories. Since we only provided three categories available, positive, negative, neutral, they felt frustrated that they couldn't categorize the comments as they wanted. As our project goal is exposing diverse comments to users, the standard of category doesn't matter. And we expect to provide an opportunity for users to consider the comments more critically by making them to rearrange comments themselves. We leave it as future work.

  - **Providing information on standard of classification**

    Some participants pointed out that there was no information on the standard of classification and they were worried that our service would follow the problem of

the youtube comments system: Users have no information on how comments are arranged. Given our service aims to solve the problem of YouTube sorting algorithms, which implies that YouTube has all the information exclusively used in sorting comments, such feedback asked a meaningful question on the overall meaning and direction of our service. Therefore, we agreed that how model classification standards can be conveyed to users should be included as future work.

V. Conclusion

We have successfully implemented our comment clustering UI. We trained the english and korean model for sentiment classification, build up our service by React, serve our service through interaction from front-end and back-end, and got user feedback. Lastly, we will discuss ways to improve our proposed product.

- Future works

Although we have successfully completed our comment clustering UI, there are lots of rooms for improvements.

1. We only prepared two sentiment classification models for two languages: English and Korean. For other languages, we relied on the translation api. If we were to improve the accuracy, we may try to add more sentiment classification models with more languages.
2. We applied some heuristics to classify neutral text out of binary classification models. To further improve accuracy, we may try to find a training dataset that already has neutral class texts, and not classify neutral texts by heuristic.
3. We may try to gain higher accuracy for BERT models. Currently Korean BERT achieves 78% accuracy on naver movie review dataset. But, we do not know how **similar** the training data and real youtube comments are. This accuracy and model is only effective and meaningful when we can make concrete assumptions that the training data distributions and the real-time inference data distributions(Youtube comment data) are equal. If there are labeled datasets of youtube comments itself, we may utilize them to gain more correct accuracy. Following this problem, it is actually very vague whether the “polarity” that we want to capture could really be represented as “sentiments”. We think that we need to define the polarity of comments more precise to enable more effective comment clustering UI
4. We may optimize code for faster inference. Currently using the pretrained BERT

model takes a lot of time on inference, or prediction. Since there are a lot of comments for a popular youtube video, we would need to speed up this inference time by distributed processing, and making multiple servers.

5. By solving the HTTPS certificate problem, the model server should be extended to receive requests from various clients and send responses.
6. Currently, the UI seems more focused on the information part of YouTube when viewed by the user, so this needs to be addressed. If we make more than one comment available by category when checking comments through additional functions, and simplify YouTube information after the first search, our main function, the classification of comments, will be more clear.
7. If we change the UI so that comments from other categories can be viewed together when checking comments for each category, it will be more suitable for our purpose of checking comments that are not biased.

VI. References

[1]: https://github.com/huggingface/transformers

[2]: https://github.com/huggingface/datasets

[3]: https://colab.research.google.com/drive/1tIf0Ugdqg4qT7gcxia3tL7und64Rv1dP

[4]: https://github.com/lushan88a/google_trans_new