Real Time Bot Detection And Heat Analysis

Zirui Tan, Zichen Liu, Haopeng Zhang, Yuting Wang

1 Clear Problem Definition

1.1 Problem Definition

Live video streaming provides a platform for hosts to interact with their audience. Sometimes, hosts may use bots to generate fake audiences and fake barrages to create an atmosphere where the host is popular. Finding out those bots, we could get a more fair judgement on the heat of hosts. Besides, we calculate the real heat depending on our detection. We further find out the increasing edge of the barrages and high frequency words for analysis of hosts operation.

1.2 Word Definition

· Barrage: chat message sent by audience of live video streaming.

· Host: the one who host the room of live video streaming.

· Heat: the popularity of the host.

2 Challenges Faced

· The communication between batches.

· Reasonable algorithms to detect the robot and calculate heat.

· Real-time visualization.

· Find out suitable parameters (scores, batch window size, weight of heat parameters).

Design unit test.

3 Description of The Technical Details

Call the api to get the room information. The information includes barrage, gifts, online audience from open.douyu.com.

3.1 Algorithm

3.1.1 Deal with Barrage:

We first accumulate all the barrage to generate the current barrage dictionary. Then we divide the streaming data into two parts: real bot and suspect bot, depending on the record. As for the real bot, we accumulate the barrage of them to barrage dictionary. As for the suspect bot, we add scores depending on the level of the user, the barrage content, the barrage interval and current barrage. There is a score threshold to determine whether it is a bot or not. Finally, we will record new bot and update bot dictionary and barrage dictionary.

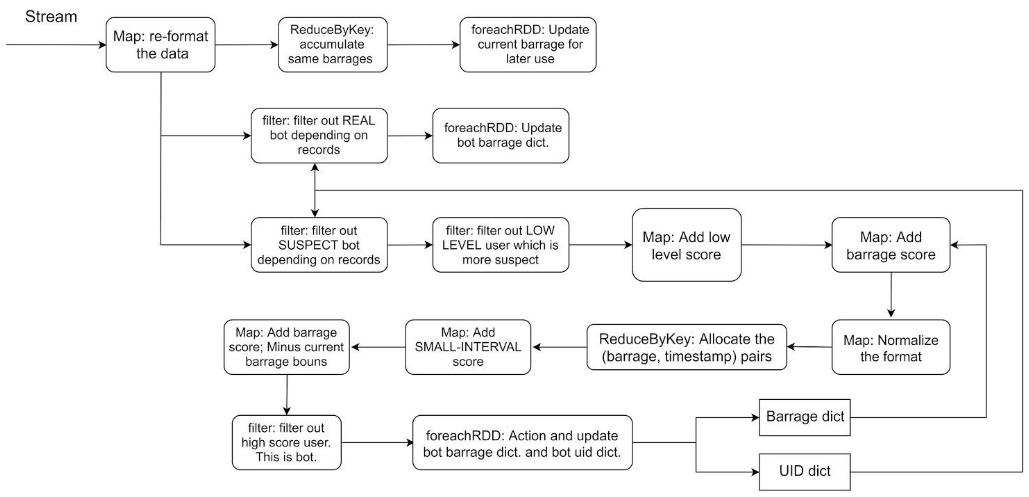

Stream
Map: re-format the data
ReduceByKey: accumulate same barrages
foreachRDD: Update current barrage for later use
filter: filter out REAL bot depending on records
foreachRDD: Update bot barrage dict.
filter: filter out SUSPECT bot depending on records
filter: filter out LOW LEVEL user which is more suspect
Map: Add low level score
Map: Add barrage score
Map: Normalize the format
ReduceByKey: Allocate the (barrage, timestamp) pairs
Map: Add SMALL-INTERVAL score
Map: Add barrage score; Minus current barrage bouns
filter: filter out high score user. This is bot.
foreachRDD: Action and update bot barrage dict. and bot uid dict.
Barrage dict
UID dict

Figure 1. Block diagram of bot detection

3.1.2 Deal with Barrage:

We first accumulate all the barrage to get the quantity of current barrage. Then we split the barrage from the streaming and then split the barrage into words. Accumulate same words to get the number of each word.

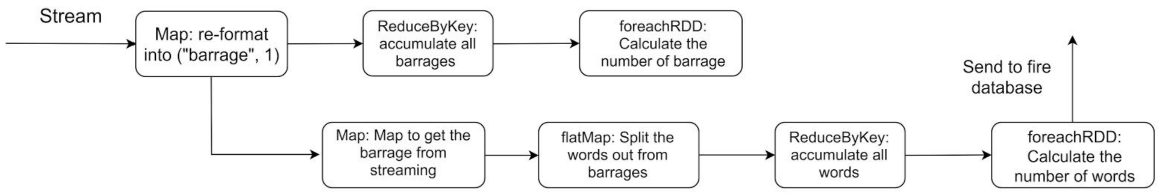

Stream
Map: re-format into ("barrage", 1)
ReduceByKey: accumulate all barrages
foreachRDD: Calculate the number of barrage
Send to fire database
Map: Map to get the barrage from streaming
flatMap: Split the words out from barrages
ReduceByKey: accumulate all words
foreachRDD: Calculate the number of words

Figure 2. Block diagram of Barrage event detector

3.1.3 Deal with gift:

Each type of gift is assigned with an weighted point. Count the number of gifts of each category over a certain period of time, and calculate the weighted point of gifts.

3.1.4 Calculate heat:

Every second, call the api to store the current information.

Every 10 seconds, calculate the average online number of each 1-second time period.

Every 10 seconds, utilize the processed gift and barrage information to calculate the current heat, the And the formula is :

$$\text{online\_number} = (\text{curr\_number} / \text{pre\_number}) * 100 \tag{3-1}$$

$$\text{curr\_heat} = \text{pre\_heat} * \alpha + (\text{online\_number} * 0.6 + \text{gift} * 0.2 + \text{barrage} * 0.2) * (1-\alpha) \tag{3-2}$$

$$\text{pre\_heat} = \text{curr\_heat} \tag{3-3}$$

Update the data in the window size which is equal to 5 and store the curr_heat as pre_heat.

3.2 Real-time Data Visualization

3.2.1 Basic Technic

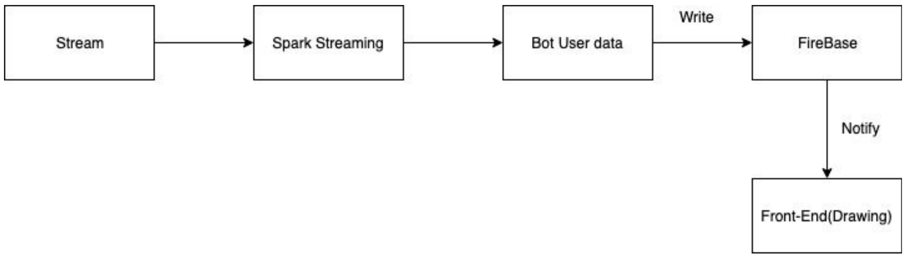
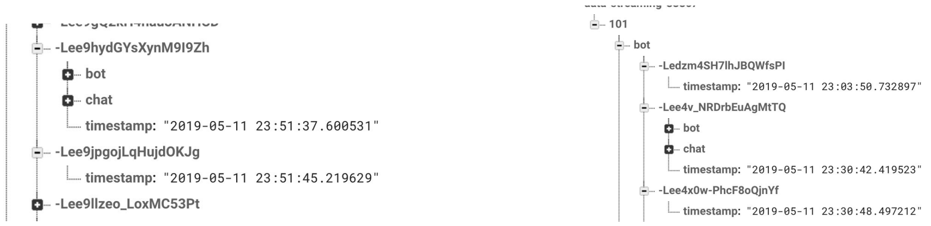


Figure 3. Block diagram of real time visualization

We use firebase to store the bot user data and barrage data, when there is new data uploaded.

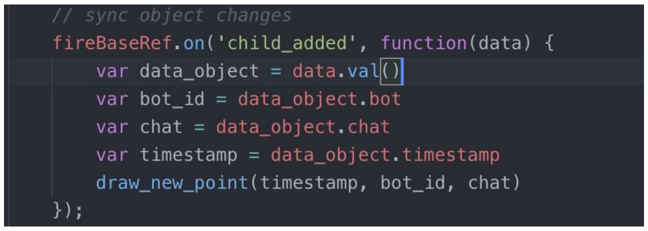


```
-Lee9hydGYsXynM9I9Zh
    bot
    chat
    timestamp: "2019-05-11 23:51:37.600531"
-Lee9jpgojLqHujdOKJg
    timestamp: "2019-05-11 23:51:45.219629"
-Lee9llzeo_LoxMC53Pt
```

```
101
    bot
        -Ledzm4SH7lhJBQWfsPI
            timestamp: "2019-05-11 23:03:50.732897"
        -Lee4v_NRDrbEuAgMtTQ
            bot
            chat
            timestamp: "2019-05-11 23:30:42.419523"
        -Lee4x0w-PhcF8oQjnYf
            timestamp: "2019-05-11 23:30:48.497212"
```

Figure 4. Firebase storage structure detail

To implement the real-time data visualization, we use the on event in firebase, whenever there is a new data in the database, it will notify the front end to draw a new data point.



```
// sync object changes
fireBaseRef.on('child_added', function(data) {
    var data_object = data.val()
    var bot_id = data_object.bot
    var chat = data_object.chat
    var timestamp = data_object.timestamp
    draw_new_point(timestamp, bot_id, chat)
});
```

Figure 5. On event for real time visualization

3.2.2 Result of The Visualization

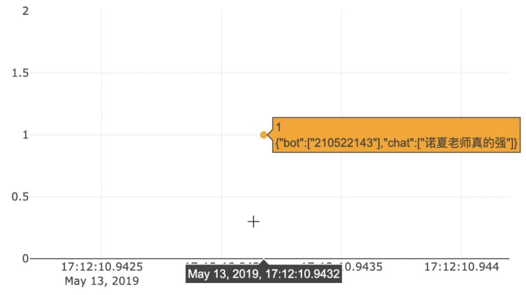

1
{"bot":["210522143"],"chat":["诺夏老师真的强"]}
May 13, 2019, 17:12:10.9432
17:12:10.9425
May 13, 2019
17:12:10.944

Figure 6. Bot detection visualization

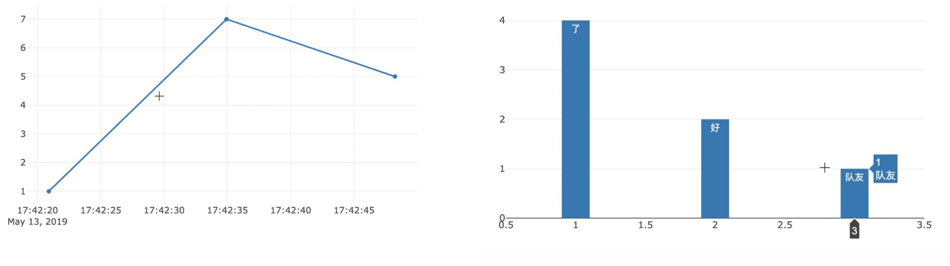


Figure 7. Barrage event detection

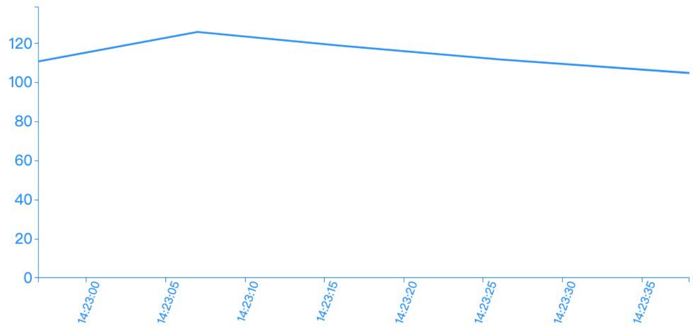


Figure 8. Real Time Heat

4 Streaming/Distributed Processing Used

Not fully exploited now, For barrage data is naturally streaming data, we use the streaming processing in the following way.

We get the barrage data and store them in a directory, the files in the directory create a stream.

We use map, filter, reducebykey for the streaming processing. In our cases,

- We use stream processing in the bot detection process.

- We use stream processing in the counting the barrage.
- We use stream processing to count the number of different chat message.
- We use stream processing to count the gifts.

5 Streaming Algorithm Could Have Been Used

In the cases of counting the number of different chat message and pick the most frequent chat message process, it is actually the problem of finding the majority element of a stream.

In this case, we could apply the Boyer–Moore majority vote algorithm which will only use O(1) space for one counter, and one element.

6 Future Work

6.1 Future Functionality

Applied machine learning to detect the bot in barrage, if there is a data set.

Simultaneous deal with several room information, and pick up the top 5 popular room.

Predict anchor’s potential heat and future hot topic.

More accurate thesaurus.

6.2 Scalability

· Using socket instead of text file to improve the processing speed and communication speed.

· Better devices to deal with large amount of data, for instance a cloud server.

· Better algorithm like Boyer–Moore majority vote algorithm to process streaming with higher efficiency.

References

[1] https://spark.apache.org/docs/latest/

[2] https://firebase.google.com/docs

[3] https://open.douyu.com/