Data Streaming project - Fraudulent click detector

Laurence TSIZAZA, Amir WORMS

July 8, 2021

Contents

1 Project introduction

Nowadays, online advertising is a strategic way of advertisement. Firms pay websites in order to have their ads displayed to users hoping they would click and find interest in their services.

Naturally, with online advertising come fraudulent manners. For instance, fraudulent clicks consist in irregular clicks made by a user or a malicious software, with the goal of seeming legitimate without adding value to the firm. Fraudulent clicks may benefit several actors, for instance:

- the host of the website is paid according to the number of clicks made on the ad, therefore the more clicks saved, the bigger the paycheck would be;
- competitors of the firm may benefit from this as fraudulent clicks alters the statistics on which the firm might base marketing strategies.

These are the reasons why it is of paramount importance to find a way to detect those invalid clicks and identify the user or the software behind it.

This project has as aim the detection of those fraudulent clicks in a set of events. The first step is to identify patterns that might help identify an invalid user. Then, given a window in which we analyse all the events, we would like to find the patterns identified and extract the fraudulent events and the users that orchestrated these. To do so, for each event, we are given the following attributes :

- The `Topic`, it can either be a ad display or a click on the ad, we consider that only the clicks can be detected as fraudulent;
- The `Timestamp`, it gives the exact moment at which the event occurred;
- The `UserId`, it is a unique identifier the user on the website;
- The `IpAddress`, it is a unique identifier for each device communicating with the Internet;
- The `ImpressionID`, it is a unique numerical identifier for each ad displays.

2 Fraud patterns

After carrying out some research on types of fraudulent clicks, we chose three patterns to detect:

- A click is considered as invalid if we cannot find a display event with the same `ImpressionID` and a lower `Timestamp`;
- A click can also be considered as invalid if it has a `Timestamp` too close to the `Timestamp` of the corresponding display, bots that are set to automatically click on the ad as soon as it appears could match that pattern;
- An `IpAddress` is considered as invalid if it appears too many times in click events, if the ratio between clicks and displays for each address is greater than a fixed threshold, we can consider the address as fraudulent; the same pattern can also be analysed with the `UserId`.

Given these patterns, we are hoping to delete clicks so that we find a click-through-rate (CTR) that circles around 10%. The CTR is the number of clicks divided by the number of displays. A CTR that is beneath 10% may lead us to believe that too many clicks had been flagged as fraudulent, whereas a higher CTR would mean that some fraudulent clicks have been ignored.

3 Practical experiments

We chose to carry out our experiments in Java with the Flink framework, reading the events from a Docker event generator via a Kakfa consumer. Our goal is to write into a text file the list of clicks detected as fraudulent.

Note that `TimeStamps` are given up to the second, therefore to manually create our window of analysis, given a fixed threshold that represents our window size in seconds, each event is going to be analysed if the difference between its `TimeStamp` and the minimal `TimeStamp` of the window is lower than the threshold. As soon as this difference reaches the threshold, then the minimal `TimeStamp` is updated with the current `TimeStamp`.

The class `ClickWithoutDisplayDetector` is dedicated to find clicks that have no corresponding displays within the window. To find such pattern, we keep an updated `Map` of displays first keyed by the `ImpressionId` and the number of occurrences, then when a click is being processed, if no display can be found or a display is found with a counter set to zero, the click is considered as fraudulent. As the detector flagged more than 90% of the clicks as fraudulent, we suspect that we had a misinterpretation of the `ImpressionId`. As a result, we made the decision to take the `UserId` as key instead.

The class `TooQuickClickDetector` is dedicated to find clicks that unfold right after the corresponding display event. To find such pattern, we store into a map each display `ImpressionId` with their last `TimeStamp`, then when a click is being processed, if its `TimeStamp` is too close to its corresponding display, then the click is flagged as suspicious.

The class `HyperactiveIpDetector` is dedicated to finding IP addresses that are too active. To find such pattern, we store, for each IP address, the count of displays and clicks in two separate maps, and we fix two thresholds: the field `clickThreshold` represents the number of clicks after which a device will be considered as potentially suspicious; the field `ctrThreshold` represents the maximal legitimate CTR, above this threshold, the CTR is considered as unusual. The IP addresses that exceed both thresholds will be flagged as fraudulent.

At last, for each class, we create a file (or more as once a file reaches a size of 1GB, it is closed another file is created) of suspicious clicks. Therefore, it is possible to see the clicks that matched each pattern.

Before any processing of the events, the CTR was approximately 34%. After applying the three detectors, we were able to reach a CTR of 4%. We believe the window is partly responsible for this low CTR, as for the first pattern, some click may not find their corresponding display because it is in the previous window.

4 Conclusion

To conclude, we are well aware that our approach was naive, as we based our strategy on known patterns. Hackers and trolls only try to get better once their strategies have been figured out.

However, with more time and more information on the events, we could be able to build more robust detectors. For instance, if we could track the activity of a user after a click and observe whether or not they navigate inside the website, it would be a useful piece of information to detect suspicious clicks of ads.