Personalized TV Watching Behaviour Recommendations for Effective User Fingerprinting

Litan Kumar Mohanta
Data Scientist, Zapr Media Labs
Bengaluru, India

Nikhil Verma
Data Scientist, Zapr Media Labs
Bengaluru, India
Email: Nikhil [AT] zapr.in

Abstract—**ZAPR transcends the data sparsity challenge, leveraging audio fingerprints captured from user's mobile device. Zapr with its partnership with mobile app publishers has a bundled app which boasts a user base of 50 million. We at Zapr, sample 10 million users on a daily basis. Sampling involves observing user's TV watching behavior using mobile devices. This task can be an acutely battery consuming task. To tackle this issue, we need to profile users based on their preferences, and then sample them efficiently; this involves old users as well as new users.**

Keywords-component; user tv profiling; big data; distributed computing; tv-domain; recommender-system

I. Introduction

In a developing country like India, where there is a massive audience of TV watchers [2], Zapr aims to profile users based on their TV watching habits. Zapr with integration with mobile app publishers has an user audience of around 50 million. For fingerprinting, we gather samples in an hour with a particular frequency. If this rate is same for all our user-base, then we encounter the problem of efficiency. With a variety of user trends, using the same frequency for all the users is not productive. Some prefer household dramas which are telecasted in the afternoon, and some prefer late night sports. So there a variance in this field which we need to explore and design a solution, which is efficient and swift. Here we also have the bifurcation of new and old users. We can recommend rates to the early users based on their watching habits, but for new users, we need to devise a methodology for efficiently profiling them without their viewership history. Execution of this method for millions of users every day is another challenge to tackle.

II. User Sampling

A. *Zapr SDK*

Zapr with its integration with multiple app publishers has an audience reach of around 50 million. We get TV samples from these users on a daily basis. Samples are one-way hashes from users, which are matched with our data. After multiple stages of data correction, we get reliable data which we can use for analytics.

B. *User Sampling*

Capturing samples from an user's device is a task that needs to be efficient because it breaks the wake-lock of the device and capture samples, which consumes a fraction of battery. If we capture multiple samples per hour, then we need some pattern for a user's watching behavior, obtaining samples for all the users at the same rate isn't feasible.

Thus having some recommendations over the user's sampling rates for different hours of weekday/weekend would reduce a significant amount of wake-up calls.

For examples, If you are a football TV watcher, you will most likely be spending more time watching TV on evenings or late at night [3][9].

III. Recommendation Task

For serving recommending user sampling rates we have two categories of users, firstly the past users for which we have some viewership data and the new users whom we encounter on a daily basis [4][6].

For the past users, we analyze their viewership history for the past few weeks on weekdays and weekends. Viewership data gives us knowledge about its viewing patterns. We can either extrapolate these numbers and measure if it performs better than default rates, else we use a general trend of rates. Further aggressive testing which can also be done is figuring out best possible rates for that user.

In the case of new users, either we have no viewership history or a few days of history. Using these numbers, we can combine them with the generic user trend and recommend rates to the user, only using the data of few days data does not seem like a good choice as we can't profile user when there is a paucity in data.

Thus the steps as mentioned earlier need to go through rigorous testing, and they need to be updated as per the changes in an user's viewing pattern. This also involves a scalable server with low latency that can serve hundreds of requests per second and an active process with which we can analyze the data every day and recommend rates.

IV. Literature

A. Related Work

Although the usage of the general recommendation algorithms is for a slightly different purpose, there have been some implementations of User-based similarity and Neighborhood-based recommendation. In [1], it discusses the uses of cable VOD TV in recommender systems. It also emphasizes the importance of recommender systems in VOD, as the user base for services like NetFlix and Am-azon is increasing at a substantial rate. According to a research, an average user loses interest in about 150 seconds, so recommender has to extrapolate to this characteristic also. Other researchers have been regarding the period people spend on watching TV, classified by age groups. Thus our problem is a new challenge, to predict when a user will watch TV based on his viewership.

V. Different *Approaches*

The cost function of our problem is defined as the difference in the average mean frequency and the frequency which is calculated for a particular experiment. Thus, in each of the tests, we need to measure the difference b/w the resultant and primary frequency. We will aim to bring the rate close to 4. The resulting solution which we finally use for our purpose will be an ensemble of different approaches as while experimenting we get results which indicate that none of the individual methods yield an acceptable level of results.

Types of Approaches:

1. Users whose patterns can be analyzed and deterministic values can be obtained from their viewership history. For these segment of users, frequencies can be extrapolated from user's preference of timings.
2. Fixed Sampling rates for users who have little to no discernable history. This is again a procedure which involves trials over different groups of people with different frequencies, either from a different region or genre of app, from which we get their samples. We aim to capture approximately the same number of samples with sampling fewer minutes from the device of the user. These experiments will help us in zooming over rates which give consistent patterns of samples.

3. Another approach would be to use an ensemble of the fixed viewership pattern on weekdays/ends naively without any further analysis; Although this helps us in achieving our target rates, this approach doesn't work for the outlier users, which have no fixed timings for TV watching.
4. A better approach would be to use an ensemble of the standard viewing pattern and combine that with the user's history is available, in case of no history; only remaining option would be to fall back on capturing a large number of samples for that our or use the average viewing pattern rates [8].

VI. Methodology

For the first part, we will talk about the past user, who is not necessarily active every day, but we have some information about that user. Invariably there is a pattern in which most of the users watch TV, a graph of the general usage pattern is presented in Fig 1.1

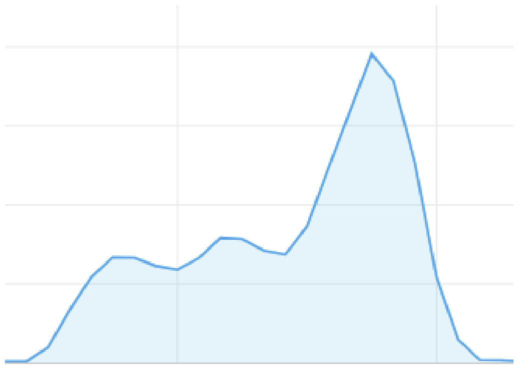
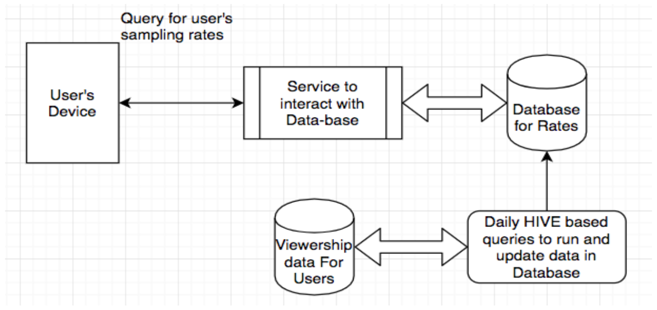


Fig. 1.1


Query for user's sampling rates
User's Device
Service to interact with Data-base
Database for Rates
Viewership data For Users
Daily HIVE based queries to run and update data in Database
Fig 1.2

Users start watching TV in the morning which increases considerably in the afternoon, a slight drop in the evening and then it rises significantly in the night. Thus most of the users will have a generic pattern of watching, although there are outliers which watch late night or usually watch unevenly [7][11].

Thus for initiating our experiments, we start with a generic pattern which resembles the graph shown in Fig 1.1, i.e., collecting samples at low frequency in the night time and at a moderate level in the evening and then at a high rate during the night. This approach may work for most of the users, but eventually we need to do this on a per-user basis as samples sent by every user is essential.

After the above experiment, we get results indicating if we are matching our current approach where we sample at high frequency during all hours, which is inefficient. These results would suggest us about the users for whom we need do a much more in-depth analysis involving a feedback loop to improve over our frequencies [8]. Here, we start with some median frequencies and improve over them by collecting more data about the user.

Now for the new users, our methodology will be similar to the previous approach, although instead of recommending generic rates for every one we get the weighted mean for that user and the general trend. How these weights are devised, is based upon several iterations which help us in coming up with optimal frequencies for users [5].

VII. *Architecture Model*

We need a high-performance database where we store sampling rates for our users, preferably in-memory which offers a low latency during queries. For this use case, we used Aerospike which is an in-memory database with high-performance reading capability and scalability. For our use case, we needed a framework which provides high-performance reads and thus using Aerospike would be beneficial.

Apart from storing our data in a Database, we needed a high-performance server which will act as a middle layer between our database and queries. For this purpose, we chose Finagle which is a framework to build high concurrency servers. Twitter uses the same structure for several of its backend services, we tested the QPS for our purpose, and it outperformed our requirements. We hosted this service on an aws 'm3.xlarge' instance with 16 Gb of memory and four cores; these configurations were sufficient for our use-case.

Our next trying task was to develop a system for processing millions of users every day and analyzing their data for generating sampling frequencies [10]. We thought of two approaches for implementing this system. Either we could use Apache Spark and have a couple of jobs to analyze the data over and then write to our database.

We tried writing jobs in Spark for our purpose, but due to the shuffling of data, it was taking more time than planned and also it would have to integrate in a couple of stages, coordinating jobs in Spark would also be an trying task.

An alternative approach with which we decided to go was with Hive Queries with sufficient resources to run queries over a lot of data in required time.

We configured a workflow which comprised of several Hive Queries over our data, from analyzing to performing data analytics and then a job for inserting data into our database.

The steps for our queries past users will be like:

- Mean view fraction for a user for the past two weeks; Data would be bucketed by Weekday and Weekend
- Scaling the view fractions to numbers
- Applying ceil and floor functions to confine our values within required limits

Steps for new users will be like:

- Mean view fraction for our user-base for the past 2 weeks.
- Calculating the view fractions for the new user, depending on data we have for that user, maybe for 1 or 2 days.
- Weighted mean for both lists, we would have to test out different weighted fraction numbers to achieve best results, for example ([0.7,0.3] or [0.4,0.6])
- Scaling the weighted viewing fractions to numbers
- Applying ceil and floor functions to confine our values within required limits

All of the above steps will be computed on an alternate day basis for the users and then will be inserted, in the database, which will then be available for requests from the device of the user.

Figure 1.2 depicts the architecture of the system

Now that we have conveyed the steps and structure of our approach, we will move forward in the next section to the experiments and results..

VIII. *Experiments*

We conducted a variety of experiments on our data. Our base values for comparison were the successful samples captured in a period for an user, when sampling at a high frequency, this takes a fixed number of samples for every user in every hour at a high rate. Our experiments will compare improved values with these values.

Average rate calculated here uses the formula:

$$\frac{\Sigma\, r_i * ns_i}{\Sigma\, ns_i}$$

r_i- rate of sample
ns_i- samples taken at $r_i$rate

1.

For some subset of users, we sampled with a fixed frequency of 10 samples per hour. In this experiment, we had 9.69 as an average for all the samples taken, i.e., either it matches with any TV or not, and 9.29 for the samples which matched with TV. Similar kind of tests were ran with fixed frequency of 8 and 6. The results were as 9.82, 9.32 for unmatched and matched samples with eight as frequency and 9.94, 9.65 with six as frequency.

Also, we pushed recommended rates to a same sized subset of users, recommended rates are calculated based on user's view-fraction in the hours he likes to watch TV. Here we got 7.8 as an average in samples, 5.8 in matched samples(which is an improvement, as we are sampling at a high rate when users watch TV). Fig 1.3 shows the chart for the above results.

Our base values here are 4; we need to bring our recommended rates closer to 4. But we also need to capture fewer samples than we are catching presently at a rate 4, because at a high frequency we will have a low average rate, but we are not efficient in the number of samples we capture.

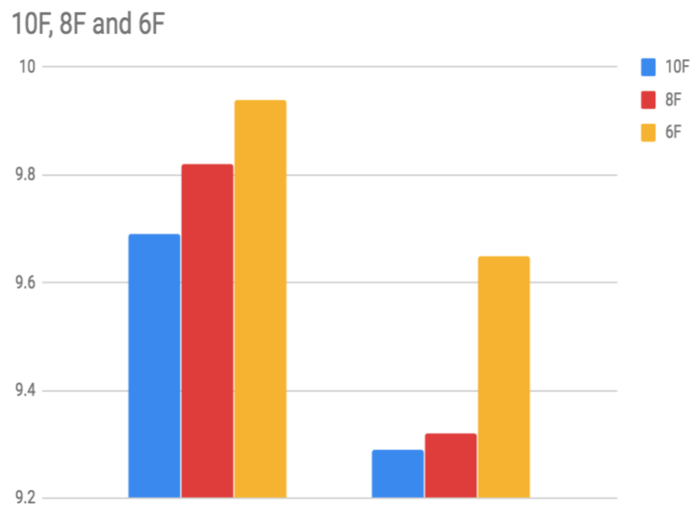


Fig 1.3

2.

We experimented with the overall pattern rates for some subset of users, the general trend in which users watch TV in some region. Rates were of low frequency in late night hours and high in the prime time hours from 7 to 10 pm with a little decline in the afternoon.

Results for these were as follows:

The effective sampling rate for TV (Unmatched samples) when these settings are used is 7.89. In examples where we matched them with TV, content was 6.86.

For a random 10k subset, the rate was 5.04 for unmatched samples and for matched samples it was 4.92.

We can observe that from the above results, it has not improved our old outcomes and we need work on the algorithm of the previous approach. Also, we need to test out the experiments on new users.

Fig 1.4 shows the rates for the above test from 0 - 23 hours.

Rates are high during the night hours and low during the peak.

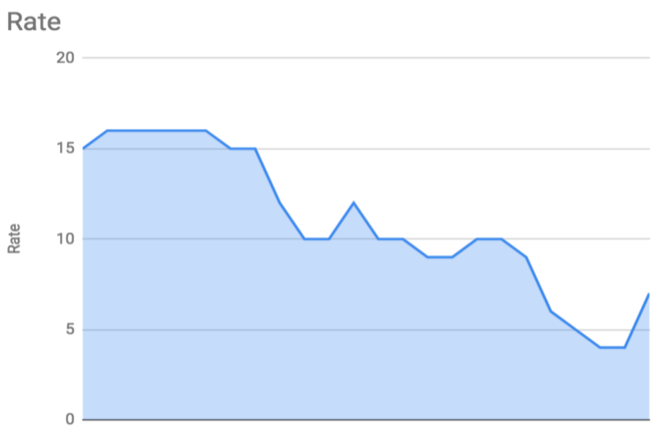


Fig 1.4

3.

We ran a couple of tests by using our algorithm of weighted mean on different subsets of the user by weighted Mean between daily user trend and the new users; the below experiments were run on a distinct subset of users having the same publisher.

Changes were as:

1. 14 days all users trend and three-day-old users in a ratio of 1:1

For the above configuration, we achieved a rate of 9.26 for all of the samples captured and 7.84 for the matched samples.

2. 14 days all users trend and three-day-old users in a ratio of 11:3

For the above configuration, we achieved a rate of 9.03 for all of the samples captured and 7.53 for the matched samples.

3. 14 days all users trend and three-day-old users in the ratio of 6:4

For the above configuration, we achieved a rate of 8.94 for all of the samples captured and 7.4 for the matched samples.

Here also for comparisons, we collected the data for a random 10k users of the same publisher.

For the above configuration, we achieved a rate of 4.02 for all of the samples captured and 4.04 for the matched samples.

After a couple of tests, we noted that for some users the samples captured in some of the hours were empty. These results give us an another perspective of not sampling in some hours at all. Although this has some drawback as the user might not be only one using that device, maybe his/her kids watch TV, and in some rare occasions, we might get samples so for tackling this issue we will samples over those hours in an episodic manner. Based on our feedback loop we will change our settings. This approach will be beneficial for saving a lot of extraneous fingerprinting.

Although we have improved our sampling rates by some substantial measure, it would be better if we would bring it even more down. Although the challenge there is that our base metric is formed by sampling at an equal rate in each of the hours; which is although not an efficient approach but manages to get most of the samples from the user.

Also, there are limitations due to the churn rate of the users. We need user viewing patterns which can then be used to recommend rates. Even young children's might not own smartphones, so they might use their mother's or father's smartphone which has conflicting viewership [3], that is if both watch TV that day, we will have samples for two different times periods or only for one.

IX. CONCLUSION

This paper summarizes the task of predicting the behavior of users from television and using that information to efficiently fingerprint samples from users. The better discussed would be to use an ensemble of a user's history and the general trend, which would be modified on a daily basis. For better implementation model there should be a feedback loop with metrics for monitoring the system and changing thresholds for a better fit. To mitigate the mismatches which occur for users who have an unpredictable pattern, the prevailing trend must be used. This paper provides ways to predict patterns of users based on their tv viewership habits, although it can be used for any form of media or data.

X. REFERENCES

[1] Diogo Gonçalves, Miguel Costa, Francisco M. Couto, "A Flexible Recommendation System for Cable TV", arxiv.org, 2016

[2] Chengang Zhu, Guang Cheng, Kun Wang, "Big Data Analytics for Program Popularity Prediction in Broadcast TV Industries", ieeexplore.ieee.org, 2014

[3] Australian Communication and media authority, "Children's television viewing and multi-screen behaviour", acma.gov.au, 2017

[4] Turrin, R., Condorelli, A, Cremonesi, P, Pagano, R, "Time-based TV programs prediction. In: 1st Workshop on Recommender Systems for Television and Online Video", ACM RecSys, 2014

[5] Billsus, D., Pazzani, M., "User modeling for adaptive news access. User Modeling User-Adap. Interact", Springer, 2000

[6] Mikhail A. Baklanov, Olga E. Baklanova, "Linear TV Recommender Through Big Data", Springer, 2016

[7] Zubritzky, D., Hidasi, B., Petres, Z., Tikk, D., "Personalized recommendation of linear content on interactive TV platforms", UMAP, 2012

[8] Cremonesi, P., Modica, P., Pagano, R., Rabosio, E., Tanca, L, " Personalized and context-aware TV program recommendations based on implicit feedback", E-Commerce. Web Technol, 2016

[9] Pazzani, M., " Learning and revising user profiles: the identification of interesting websites", Mach. Learn, 1997

[10] Mikhail A., Baklanov, "Methods of Machine Learning for Linear TV Recommendations", Springer, 2016

[11] Pazzani, M., "Framework for collaborative, content-based, and demographic filtering", Artif. Intell. Rev, 1999