Predictions on Twitch Stream Watching Instances

Yacun Wang, Huy Trinh

ABSTRACT

Nowadays, live streaming platforms are highly popular tools for people to actively share topic(s) they currently enjoy with a wide variety of users who are interested in similar topics. From customer behaviours or interactions, we have to understand some challenges that there are highly skewed distribution which summons obstacles for our recommendation ideas as well as prediction tasks. For this assignment, the dataset of Twitch [1] streaming and watching interactions are utilized to predict whether the user will watch a particular stream given a user ID and stream ID. We developed models that captured sequential temporal dynamics, implicit feedback, and the basic popularity features.

1 INTRODUCTION

Background

In recent years living streaming platforms have been extensively popular. Such platforms focus more on the experience of content sharing between people who wish to be the source and other people who are receiving the materials from the sharer. Different from platforms like YouTube where recorded videos are made for users to watch, live interactions are emphasized in streams such that live conversations and reactions are made possible. The content of the streams could vary based on the streamers' interest. While most streamers tend to share games, limitations on topics could extend to academia or even only chatting. Under this context, predicting and recommending streams to users becomes a reasonable task to tackle. Specifically, we study interactions between streamers and users from Twitch.

Data

The dataset is adapted from data used in Rappaz, Jérémie et al. [1] (referred to as original paper) and due to machine constraints a subset is taken. According to Rappaz et al., the Twitch data used was collected from Twitch API to represent interactions in a 43-day period for every 10 minute interval. All available streams represented by stream id are taken at each time point, and stream watchers are recorded as users in the dataset. The full dataset contains 15.5M users and more than 120M interactions, which is almost impossible to load in working machines without parallel processing. As a result, we used the benchmark dataset with 100K users for descriptive analysis (over 3M interactions or data points) and the first 500K users (over 12M interactions) from the full dataset to tackle the task described in Section 2. In particular, the input dataset contains the user ID, the stream ID, the streamer username, and times the user started and stopped watching the stream.

We also recognized that the timestamp provided in the dataset means the number of 10-minute intervals after the start time. Since there are 144 10-minute intervals in 1 day, we add two features for each time feature according to the exact timestamp for both start and stop time:

(1) Time In Day: By taking modulo 144, we get the number of intervals into each day, so it gives information regarding the time in a particular day;
(2) Time At Day: By taking integer division 144, we get the number of days after starting time.

These two created columns provide us the tool for our predictive task.

Descriptive Analysis

After examining the metadata and their relationships, we have found that the main property of this dataset is highly right-skewed data. For most features such as total watch time for users, number of streams watched per user, number of streamers watched per user, average watch time, etc. Regardless of the scale we look at the distribution of the variables above, we always observe a maximal peak on the left, and the curve drops down steeply to the right. For example, Figure 1 shows the distribution of the number of streams watched for all users, taken for all numbers > 400. Notice that if we plot the distribution for the same variable for the < 400 part, the same shape will occur; similar shapes occur for the number of users watched per user. Similarly, Figure 2 shows the distribution of average time watched for all users. It could be seen that the skewness is also extreme.

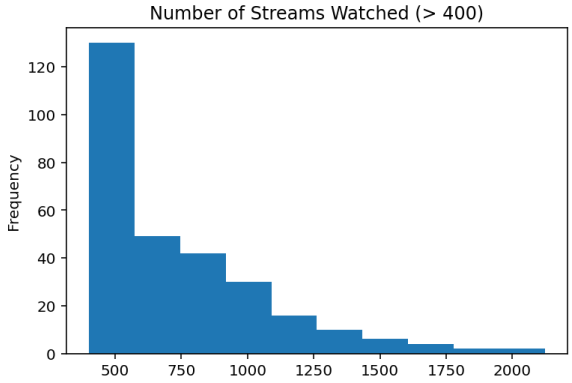

Figure 1: Number of Streams Watched Per User

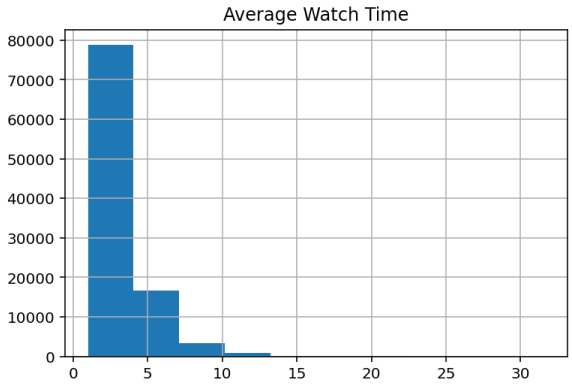

Figure 2: Average Time Watched Per User

Then we recognized that the most popular streamer in the benchmark set is ‘ninja’, who on his own records more than 45,000 interactions in the small sample set. In the process of exploration, we found out that streamer “ninja” went live about once per day and each stream from the same streamer has a different streamID. Also, in randomly selected streams from “ninja”, no user appeared more than once in the same stream. These facts provide high-level descriptions of dataset elements such as the fact that streamIDs in our datasets will not be informative on their own. If availability of the streamer is used as features in our model, then (streamer, time_start) will uniquely identify the stream.

Since availability of streams are essential to the prediction task, we explored the time features and the relationship between times and streamer information. More generally, the timestamp is roughly uniformly distributed across 43 days without a clear trend; from a single day’s perspective, the time_start for all the interactions show a dip around noon. Figure 3 shows that the trend matches our expectation such that streams usually occur at late night to over midnight.

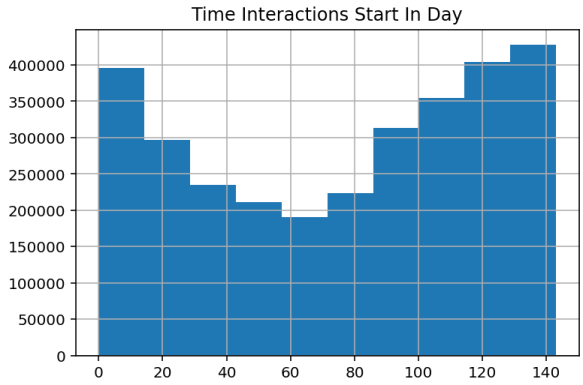

Figure 3: Start Time In a Day

2 TASK

Due to hardware constraints, our predictive task is that from the first 500K users, train on all interactions they made in the first 36 days, and predict if a user will watch a particular stream of a streamer in the last 7 days of the 43-day

period. The training and validation parts are separately given, but both datasets contain all interactions including their userID, streamID, and other features from the original dataset such as timeStart and timeStop. A negative sample is also randomly chosen from unwatched streams, so the size of both datasets are doubled. Note that the negative samples will not have time features associated with them. After the process above, we get 24,701,600 entries in training and 4,893,212 entries in validation, both with balanced class labels.

Since the task is a binary classification task, we also used accuracy to measure the prediction results. Note that since this dataset is closely related to recommendations, we also tend to utilize metrics such as precision and recall in the final stage to determine the model power to recommend will-watch streams accurately. More particularly, precision tends to be more important than recall, since we wish the fraction of items recommended to be more positive in such a recommendation system setting.

Due to the limitations from the input dataset, feature representations are somewhat narrow. During exploration, we discovered that the timestamps could be parsed into days and minutes into each day (See Section 1) and using the 2 different representations as features in baseline models provide reasonable but similar results (See Section 3). In terms of interaction-based data, some models prefer directly using streamer names, while others require mapping these usernames to integer IDs. Details on interaction representations will be discussed with models in Section 3.

To test simple performances, some relevant baselines include using popularity, using some metadata features into a classifier, and some simple latent factor models. These baselines will be introduced along with the final model in Section 3.

3 METHOD

In this section, various models tried during training and exploration are introduced, and the final model is presented at the end.

Baselines

From EDA, we learned that the distribution of almost all important factors is highly right-skewed, which prompts the first baseline. The model sets a threshold between 0 and 1, and predicts 1 if the popularity of the streamer is in the top n_interactions * threshold entries in the training set. This model doesn't suffer from scalability issues and could potentially capture how users decide on streams to watch. In fact, this model provides a strong foundation for testing whether our new implementations perform better than the simplicity of a straightforward method. By taking the threshold to be 0.5, the model is able to attain 0.803 of validation accuracy, which shows the fact that the popularity feature is expectedly compatible with this dataset.

Another baseline that we have is to put time of the day which could show us the behaviours of activities of users amongst different periods. Again from EDA, it appears that since users tend to watch streams at night time, direct time features could be informative. From the exploration, testing accuracy is around 0.67 in both models where time features are touched or untouched.

With interaction data present, one could think about picking a few factors by dealing with the interaction matrix. A simple latent factor model between the user and the streamers in the following form achieves such a purpose:

$$f(u,i) = \alpha + \beta_i + \beta_u + \gamma_i \cdot \gamma_u$$

Equation 1: Latent Factor

This model attempts to capture the essence of interactions using the dot product between 2 low-dimensional vector representations, and the output *f(u, i)* in this dataset could be the number of interactions a user had with a streamer. This model sets the baseline for interactions, but with its limited ability to consider important factors in this dataset, we achieve a validation accuracy of around 0.67. Notice that due to the slow runtime, we could only compute the validation scores. To make predictions, we recognized that the validation set is constructed by taking all the observed interactions and sampling a negative entry for each positive entry. This provides a hint that we could rank the scores based on the latent factor model for each user, and predict 1 for the first half of entries for that user.

Improved Models

To improve the popularity feature, we started from a sorted popularity, assigned percentiles as scores between 0 and 1 to each streamer, and fitted into a simple logistic regression model. For instance, if a streamer is not well known enough then the percentile of him/her should be close to 0. This percentile is an improvement from the previous popularity with threshold one since the score is more continuous and smooth instead of 0s and 1s, while it also gives us a new feature into a better classifier towards this predictive task instead of assigning a single number to for prediction. Note that we are able to assign the same score to streamers who appeared the same number of times in the training set.

Our next improvement in attempts to capture more information from interactions by noticing implicit feedback situations present in this dataset. Since not interacting with a streamer might not be a clear indication that the user dislikes the streamer, Bayesian Personalized Ranking (BPR) focuses on the relative rankings of the interaction entries instead of deterministic scores, and fits interactions in the form:

$$x(u,i,j) = \gamma_u \cdot \gamma_i - \gamma_u \cdot \gamma_j$$

Equation 2: BPR

where *i* is a streamer the user *u* interacted with and *j* is a randomly sampled streamer that the user hasn't interacted with. With the help of the implicit package and some TensorFlow code, we are able to obtain BPR scores and fit those scores using a logistic regression model.

Another improvement to better capture interactions relies on the Factorizing Personalized Markov Chains (FPMC). Compared to BPR, FPMC pays extra attention to the sequence of interactions, and assumes that the next interaction will be solely based on the recent interaction. This model also concerns implicit feedback as a random negative sample is obtained during training. With this idea, FPMC captures more temporally evolving information on the interactions that a user has with a streamer. For example, the user's taste across diverse stream genres could drastically change over time. A user and streamer ID mapping is used during training, similar to a factorization machine. Similar to the latent factor model, FPMC makes slow predictions. So we utilized the same trick that after ranking the scores predicted, the higher half of each user is predicted to be 1. Due to the extra care to temporal dynamics learned by this model, we achieve a validation accuracy of 0.837.

Failures and Limitations

Several failures occur during the process of exploration, and most of them stemmed from a low memory availability or the inability to use parallel processing for a difficult task.

One of our initial baselines is to incorporate maximum Jaccard and Cosine similarity scores with the most similar streamer that the users have watched, aiming to uniquely identify streamers with their time availability and fit it in Logistic Regression. We also attempted to reduce runtime by using set-related utility data structures. However, with the help of the *tqdm* library that shows a progress bar, constructing training data takes around 68-500 hours for our whole dataset. This process is extremely unenergetic which can be processed using Hadoop or Spark that are used for big data. Due to the constraint of knowledge we have, we are unable to test this feature.

Some other models are limited by their slow runtime so that computing training scores is nearly impossible. As discussed above, only obtaining the validation scores would limit the final result, since a simpler prediction pipeline is utilized. Models affected by this limitation include those that require computing more dot products during prediction: BPR and FPMC. Ways to handle this situation are covered in the Final Model section.

Final Model

From the models mentioned above, we investigated the combinations of 5 models: Popularity Percentile, Timestamp Baseline, Latent Factor Baseline, BPR, and FPMC. Due to the limitation that we only obtained validation scores for BPR and FPMC models, we are unable to fit a logistic regression model and let the model decide which submodel appears to be more significant. We then recognize that since for FPMC and BPR we only obtain predictions, we take predictions done on the validation set for all of the submodels, and use the logical OR operator. In particular, we will predict 1 if any of the *k* inputted submodels predicted 1 for that user-streamer combination. We iterated through all possible combinations of the 5 submodels above, and unfortunately none of the combined models outperformed FPMC alone.

With a small modification, the **final model** concerns the combination of Popularity Percentile + BPR + FPMC, and we predict 1 if at least 2 of the submodels predict 1. This combined model gives a validation accuracy of 0.857, validation precision of 0.899, and validation recall of 0.805. See Section 5 for more discussion on this final model.

4 LITERATURE

The Twitch dataset used in this assignment is adapted from Rappaz, Jérémie et al. [1], and the authors of the paper collected the original data for the particular research. According to the original research paper, the full dataset was collected over 6000 rounds of requests from the Twitch API. Yang, Tzu-Wei et al. [2] explored a similar live streaming Twitch dataset in 2013 by focusing on the rapidly evolving user consumption. The paper utilized collaborative filtering, similarity-based recommendations, and sequential temporal layers in a hybrid model. Yang et al. concluded that the extra attention paid to user preference awareness and social dynamics increases improvement on similar data representations.

The availability of streamers are carefully processed in the original dataset. From the models we employed, both BPR and FPMC addresses implicit feedback by sampling unwatched entries from datasets. The original paper emphasized that a negative sample might occur simply because the streamer is offline or the user chooses not to watch the channel. The paper utilizes a more complex sampling method: one negative entry from the full set is sampled, one negative entry from the live stream channels

is sampled, and the model attempts to determine if an available item will be consumed.

The element of repeated consumption is also essential in the dataset. In our model, we simply represented this factor using popularity features as well as using the number of consumptions between a user-streamer pair in our interaction matrix. The original paper stresses this issue by splitting the sequential encoding process into the repeated consumption case as well as a novel consumption case. The temporal dynamics were incorporated into the encoding to learn patterns under the repeats. Wang, Chenyang et al. [3] explored the notion of repeated consumption similar to the setting we have in this dataset. The paper attempted to emphasize the factor that continuously evolving interests of repeat consumption is essential for fitting the patterns that change over time. By setting short-term and long-term consumption patterns using different functions, their model is able to give good predictions on real-time consumption patterns.

The problem of scalability is addressed in various research papers. The original paper proposes the solution that when examining available streams at a time, only pre-ranked top *k* channels are taken into consideration. Other papers addressed the issue when a model includes too many dimensions or has the existence of the curse of dimensionality. A scalability solution mentioned in Gionis, A. et al. [4] talks about locality-sensitive hashing (LSH), a hashing algorithm that place similar items based on their probabilities into the same bucket, compares it to other tree methods to show that the algorithm is optimized in order to increase the performance of high dimensional model as well as how it save disk spaces allocations in it real time systems. This paper enlightens us in a way that if we are to use a lot of model combinations that increases the number of features; the algorithm will save us memory resources as well as time constraint challenges.

In general, we manage to conclude from our relatively simple models that sequential temporal features, implicit feedback, and simple representation of repeated consumption improves our model performance, the research papers delved much deeper into the factors and developed more complicated yet reasonable solutions to these problems.

5 RESULT

Our very first basic model involves only a determined threshold based on popularity. The improved model that we tried is to use popularity and assign percentiles to it then fit it into a Logistic Regression with default parameters or not tuning. For instance, if a streamer is not well known enough then the percentile of him/her belongs to the first percentile. The result of the accuracy of the validation set is 81.9% which increases by more than 1% compared to the model with Popularity with Threshold of 0.5.

We were quite not impressed by the simple-implementation baseline models and started to proceed to another baseline involving Latent Factor from TensorFlow. By modifying rating into interactions between user and streams, we fit them into our Latent Factor then later in FPMC model. Latent model is an advanced baseline that we looked forward to implementing so we could later on compare it with FPMC & BPR. The first result of Latent Factor with 5 latent factors. We began to tune the Latent Factor model to 8 latent factors and increase the epochs to 40,000 instead of 20,000 since our machines do not have enough memories to experiment or play around with different numbers of latent factors or regularization rate. The result came out to be

unexpectedly equal (66.7%) to the previous latent model but with a longer runtime.

Latent Factor model focuses on interactions that a user has with a streamer might be assumed to depend on few implicit factors such as the user's taste across various streamer genres. There, we believe that such hidden patterns are not fully demystified that lead to such a low score. Learning from the Latent Factor model, we were to develop Bayesian Personalized Ranking that captured implicit feedback or unseen unconscious interactions of users and generated hidden rankings for them. Its effectiveness shows that this concealed pattern of ranking classification applied to solve this streamer interaction quite well.

Although the run time of FPMC is sluggish, we are able to achieve an accuracy of 83.7% with its capability to consider temporal dynamics. Notice that the assumption of the Markovian property is not necessarily true in this dataset. According to the repeated consumption property, the next streamer the user chooses to interact with should depend on more previously watched streamers instead of just 1. Due to the inability to consider such cases, it is not too surprising to see a minimal improvement. Similar to the role of mysterious latent factors in the latent factor model, both BPR and FPMC are also finding latent factors from the interaction (or interaction + feature) matrix, only with more information embedded. These parameters are harder to interpret, but from the results, the factors they learned appear to explain more variance in the dataset.

After meticulous consideration and selection from incorporation amongst different methods, we decided to pick the combination of popularity with percentile, BPR and FPMC. This helps us achieve our accuracy to be 85.7% which is a bit higher than other models by themselves. However, with such a large dataset, the small increase here marks a milestone of innovation for our model.

The precision score here is much higher compared to our recall score by 10% which means that our false positives are minimized being our satisfaction in the goal of prediction. Our goal is to see if a user is going to watch a particular video or stream to recommend it to him/her later on. This means we do not care about the unwatched stream; for instance, if we are to run advertisements for those streamers. The money would be invested precisely on those watched streams maximizing the benefits of our customers.

REFERENCES

[1] Jérémie Rappaz, Julian McAuley and Karl Aberer, "Recommendation on Live-Streaming Platforms: Dynamic Availability and Repeat Consumption," RecSys, 2021

[2] T. Yang, W. Shih, J. Huang, W. Ting and P. Liu, "A Hybrid Preference-Aware Recommendation Algorithm for Live Streaming Channels," 2013 Conference on Technologies and Applications of Artificial Intelligence, 2013, pp. 188-193, doi: 10.1109/TAAI.2013.46.

[3] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling Item-Specific Temporal Dynamics of Repeat Consumption for Recommender Systems. The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 1977–1987. https://doi.org/10.1145/3308558.3313594

[4] Gionis, A.; Indyk, P.; Motwani, R. (1999). "Similarity Search in High Dimensions via Hashing". Proceedings of the 25th Very Large Database (VLDB) Conference.

http://delab.csd.auth.gr/~apostol/pubs/pkdd2013_vp.pdf