CSE 158 Assignment 2

Jimmy Huang, Xinle Yu, Jaehoon Kim, Lando Aldis
jih103@ucsd.edu, xiy033@ucsd.edu, jjk015@ucsd.edu, laldis@ucsc.edu

March 31, 2025

1 Dataset Description

The dataset that we are using is the Twitch dataset. The data was taken in a span of 43 days. Every 10 minutes, information about live streams, streamers, and viewers are compiled into this dataset. Our specific dataset contains interaction data of 100,000 Twitch users and the streams that they have interacted with. This dataset contains a total of 3,051,733 rows, indicating that this was the number of user and streamer interactions contained in this dataset.

Our dataset contains 5 columns: an anonymous ID for each Twitch user (userID), the ID of the specific stream that they were watching (streamID), the name of the streamer (streamerName), the time that they started watching the stream (start) and the time that they stopped watching the stream (stop). The userID column contains integers spanning from 1 - 100,000, each is an anonymous representation of the 100,000 Twitch users. The streamID column has unique identifiers for that specific stream. The streamerName column is a string of the streamer's username on Twitch. The start times and end times are represented as integers, indicating the 10-minute interval during the 43 day sampling period that this data was taken from, i.e. a start time of 2 would indicate a 20 minute difference between the time that this specific interaction was recorded and the start of the sampling period.

The average amount of time that a user would watch a stream is about 31.42 minutes, while the median amount of time a user would watch a stream is approximately 10 minutes. This shows that at least 50 percent of our data has interactions that only lasted about 10 minutes, and the average interaction duration is being pulled upwards by the mean.

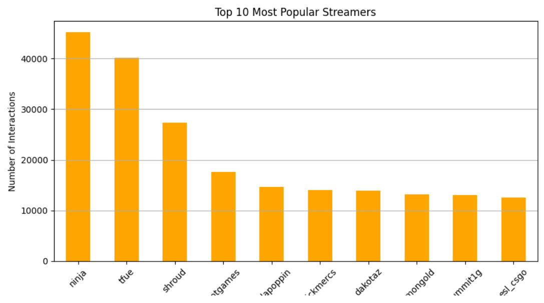

Figure 1: Streamer Popularity

At the time that the data was collected, these were the most interacted with streamers. Both ninja and tfue had over 400,000 interactions in our dataset, however, there are lots more streamers in our dataset. There are 162,625 streamers that users have interacted with. There are 739,991 streams recorded. The max number of users that have interacted with a specific stream is 2,123, which was one of tfue's streams. The mean number of users that have interacted with a specific stream is about 4 users. This shows how most streams tend to have lower viewership, and this gives a sense as to how many interactions are made between a stream and its viewers.

1.1 Most Popular Times to Stream (Hourly)

This graph shows the distribution of interactions throughout the day, aggregated hourly. The most

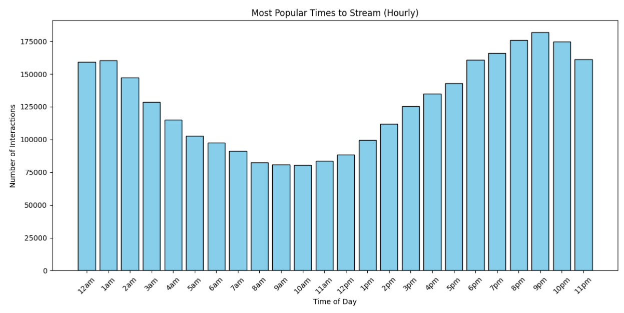

Figure 2: Most Popular Times to Stream (Hourly)

active times for user interactions are between 6 PM and 10 PM, peaking at around 9 PM.

Understanding hourly interaction patterns is important to optimize platform resources and user engagement. For example, streamers should prioritize broadcasting during peak hours to maximize viewership, while the platform can allocate resources (e.g., server capacity) more effectively to accommodate higher traffic during these times.

1.2 Clustering Insights from User Behavior

As part of our EDA, we applied K-Means clustering to segment users based on their behavior patterns. Using features such as average session duration, interaction counts, and revisit percentages, we identified three distinct user groups:

- **Cluster 0: Casual Viewers** - Short session durations, moderate interactions, and low loyalty to streamers.
- **Cluster 1: Loyal Viewers** - Long session durations, fewer interactions, but high loyalty (frequently revisiting the same streamers).
- **Cluster 2: Exploratory Viewers** - Moderate session durations, high interaction counts, but minimal loyalty.

To determine the optimal number of clusters (k), we used the elbow method. This method analyzes the trade-off between the number of clusters and the reduction in inertia (within-cluster sum of squares). $k = 3$ was chosen because it represents the point where the rate of inertia reduction significantly decreases, providing a balance between simplicity and clustering quality.

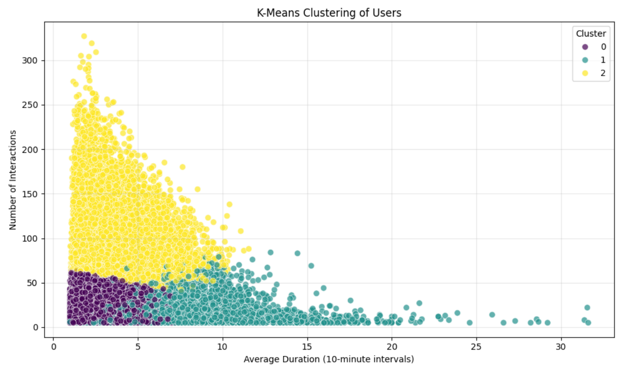

Figure 3: K-Means Clustering of User Behavior

This clustering analysis provides actionable insights for the platform. For instance, loyal viewers (Cluster 1) could benefit from personalized recommendations focusing on their preferred streamers, while exploratory viewers (Cluster 2) may require diverse content suggestions to retain their engagement.

2 Task

We studied two tasks on the Twitch dataset.

2.1 Task 1

The first is to model stream watchers as users and streamers as items, which is the same as the would-read prediction in Assignment 1 and the tasks studied in Rappaz et al. [1]. The baselines are given below and they are run in the repository of the paper: `https://github.com/JRappaz/liverec`. The baselines and model are evaluated using the metric H@1 and H@10.

- REP is a simple baseline that returns the number occurrences of the item(streamer) in the input sequence

- POP is a baseline that returns the popularity of the item in the training set
- MF is a baseline that uses matrix factorization model
- FPMC is an advanced sequential method that models transition from the last item, it is based on Markov Chain

2.2 Task 2

Since the baselines of the first task were borrowed from the paper, we study another task: predicting when a user will watch the next stream. The baselines and model are evaluated by metric: MAE. We made the baselines and model user-ignorant here, only taking the series of timestamps as the input.

We preprocess data by ordering each user's interactions, slicing our data in sliding window of 10, normalizing by the average difference between timestamps in all training sequences.

To get this data, we got each user's interaction history and found the users that have more than 11 entries. This is because we want to fill our vectors that are of length 10. We then sorted their start times and found the differences. The train/validation/test split is 0.7:0.15:0.15.

Here are some baselines and models that we could evaluate our predictive task to:

- GAP is a simple baseline that uses the average gap between a the input's stream watching time
- RNN uses a Recurrent Neural Network to predict the next time the user will watch the stream
- TPP is a model that uses Temporal Point Process. It is usually useful in modeling events like earthquakes and COVID spreads.

3 Model

3.1 Task 1

For the first task, we implemented the Factorization Machine model. It is similar to the MF baseline, while it explores the interactions between multiple features by representing them in a single embedding and calculating the sum of their dot products. We implemented this considering the sparse structure of users/items in the Twitch dataset.

We train the model with a batch size of 100, early stopping epoch of 15, learning rate of 0.0005, l2 of 0.1, embedding size of 64, and Adam optimizer. For the first 10-15 epochs, the validation loss stably decreases, and there is minimal performance gain afterwards. The final epoch for the model is 28.

Initially, we expect the model to perform better than POP, MF, REP as it takes in timestamps as features and utilizes sparse user/item space of the Twitch dataset. However, since there is no other special design, we expect it to be less powerful than FPMC.

3.2 Task 2

For this task, we implemented a simple LSTM model as we believe that it may be an optimal solution without specific data-targeting engineering. We expect it to perform better than GAP and RNN, but we were unsure about its performance against TPP.

The model is LSTM with one fully connected layer with a hidden size of 16 and batch size of 1024. Learning rate was set to a very low value, 1e-4, on 20 epochs. On larger learning rates, the model would sometimes have difficulty converging; there is not much difference in performance for epoch around 10-20. The model is trained using MSE but evaluated using MAE for consistency. Only the last output is used to calculate the loss.

4 Literature Survey

The Twitch dataset is introduced in [1], where it is used to evaluate the proposed model LiveRec in the paper. Compared to other baselines in Task 1, LiveRec achieved state of the art performance with H@1: 0.4122, H@10: 0.7655. More common methods include FPMC used in baseline Rendle et al. [2] and Bert4Rec Sun et al. [4], which uses a transformer-based architecture to capture both the sequential and bidirectional context of user interactions, making it effective for predicting the next item in a sequence.

For Task 2, there are statistical methods like autoregressive integrated moving average (ARIMA), which can take into account regular patterns in the data. It combines dependencies between previous values and the moving average over time steps, which can smooth out short term fluctuations in data that may be outliers. There are also machine learning algorithms, such as support vector machines (SVMs) in the context of a regression problem.

For this experiment we explored several methods. We tested long short-term memory models (LSTM), because it is more robust at handling irregular data than ARIMA Taslim and Murwantara [5] but can also learn long-term dependencies. However, LSTMs perform worse than ARIMA on longer time windows. In a paper comparing the effectiveness of similar deep learning architectures on time series dataShi et al. [3], the findings show that the window size plays a significant role in the models ability. Notably, in a 'fixed look-back window size' similar to the sliding window used in our experiment, larger window sizes drastically reduced performance.

Temporal point process (TPP) is a probabilistic model designed to handle sequences that occur at irregular intervals, which was fitting for the Twitch dataset. Our model was inspired by event recurrent point process (ERPP) Xiao et al. [6]. This approach models events as stochastic processes and learns the intensity function, which represents the rate of event occurrence based on the previous time steps.

5 Results and Discussion

5.1 Task 1

We may see that the performance largely depends on the pattern of the Twitch dataset. We believe that specific data-engineering is required to reach a high performance on Twitch dataset just like the proposed method LiveRec in [1].

REP appears to be very powerful despite being the easiest baseline. The Factorization Machine we implemented appears to have slightly better performance than MF, POP, as we expected.

	h@1	h@10
REP	0.3698	0.5347
POP	0.0317	0.1350
MF	0.0363	0.1537
FPMC	0.0690	0.2515
FM	0.0419	0.1535

Table 1: Performance of Task 1 models on h@1 and h@10 Metrics

5.2 Task 2

	MAE
GAP	0.944
RNN	0.863
LSTM	0.861
TPP	0.964

Table 2: Performance of Task 2 models on MAE/L1 loss

For the second task, TPP's bad performance is unexpected, but the pattern streaming-watching time may not match traditional Point Processes like Hawkes Process or Poisson Process. This shows that complex models, if not engineered to fit the data, often perform worse than simpler models.

RNN and LSTM appear to perform better than GAP, this shows that recurrent neural networks are able to capture some pattern within the user interaction timestamps. There is no significant difference between RNN and LSTM, probably due to the simple and irregular nature of the dataset timestamps.

References

[1] J. Rappaz, J. McAuley, and K. Aberer. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 390–399,

New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3474267. URL `https://doi.org/10.1145/3460231.3474267`.

[2] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 811–820, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605587998. doi: 10.1145/1772690.1772773. URL `https://doi.org/10.1145/1772690.1772773`.

[3] J. Shi, M. Jain, and G. Narasimhan. Time series forecasting (tsf) using various deep learning models, 2022. URL `https://arxiv.org/abs/2204.11115`.

[4] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, 2019. URL `https://arxiv.org/abs/1904.06690`.

[5] D. G. Taslim and I. M. Murwantara. A comparative study of arima and lstm in forecasting time series data. In *2022 9th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pages 231–235, 2022. doi: 10.1109/ICITACEE55701.2022.9924148.

[6] S. Xiao, J. Yan, X. Yang, H. Zha, and S. M. Chu. Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1597–1603. AAAI Press, 2017.