Discover User-App Interactions & Solutions to Reducing the Initial User-CPU Latency

Thy Nguyen Milon Chakkalakal Pranav Thaenraj

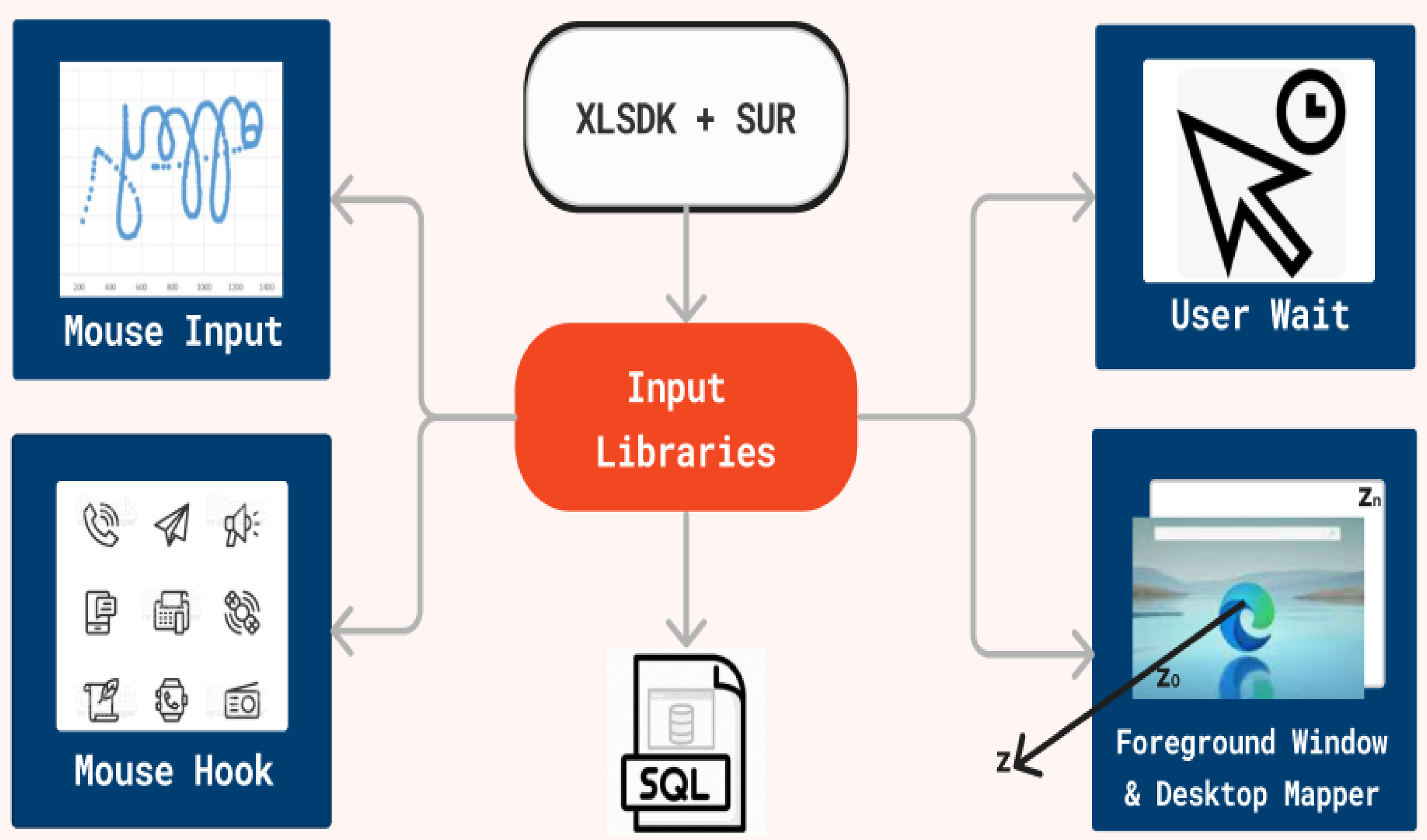
Advisors: Jamel Tayeb, Bijan Arbab, Sruti Sahani, Oumaima Makhlouk, Praveen Polasam, Chansik Im

Abstract

- **Data loading** icons signal an unpleasant user-wait experience and can tear people away from using an app.
- We mitigate the initial latency by collecting system usage data using Intel's Telemetry and analyzing user-app interaction by EDA, HMM, and LSTM/RNN.

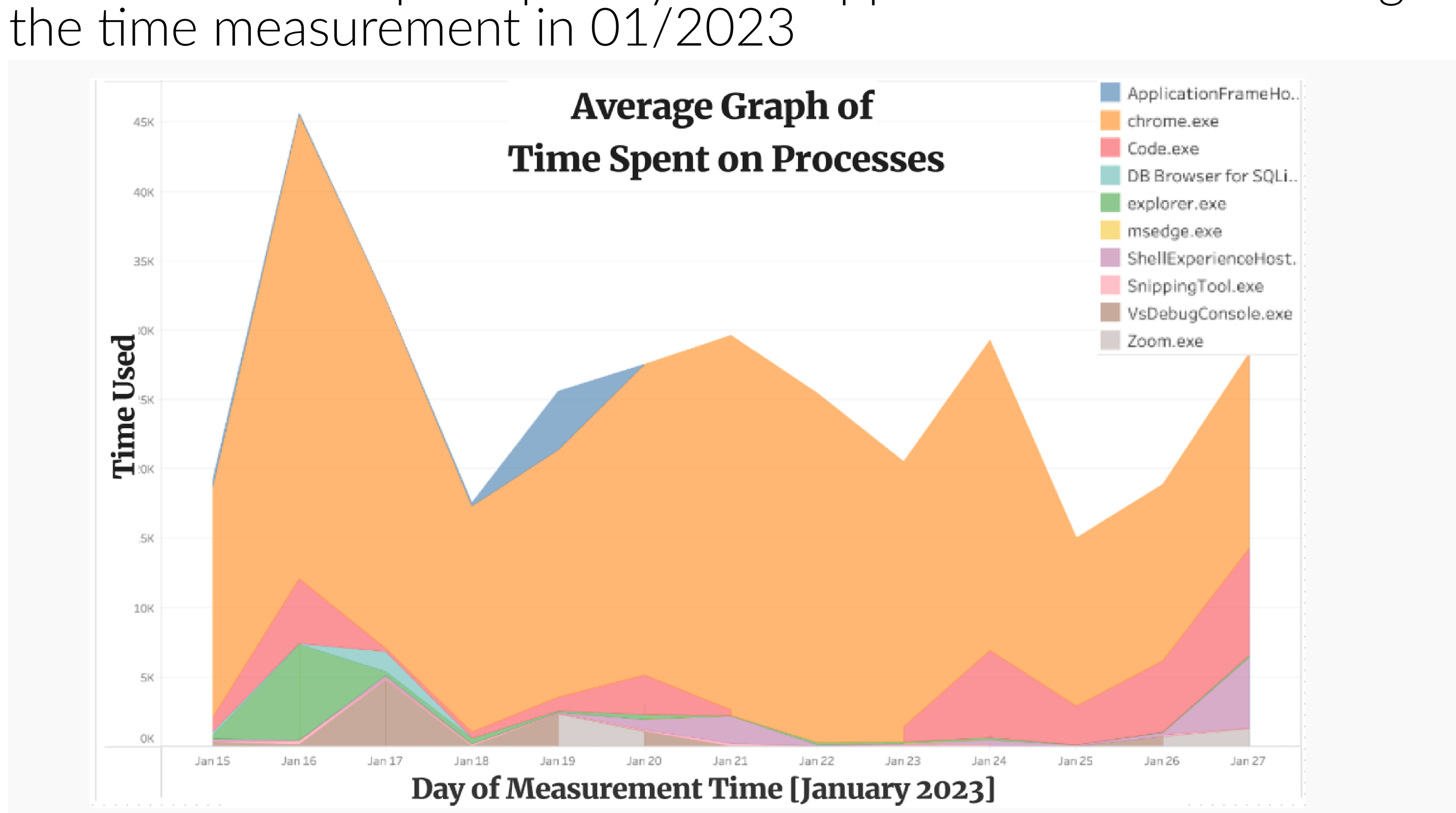
Methodology of Data Collection

- **Tools**: Software Development Kit, Environment Server, Intel® System Usage Report framework
- **Purposes** Anonymously gather and analyze data usage from multiple devices.



Exploratory Data Analysis

Chrome is the top frequently used app of this user according to the time measurement in 01/2023



Methodology of Predictive Tasks

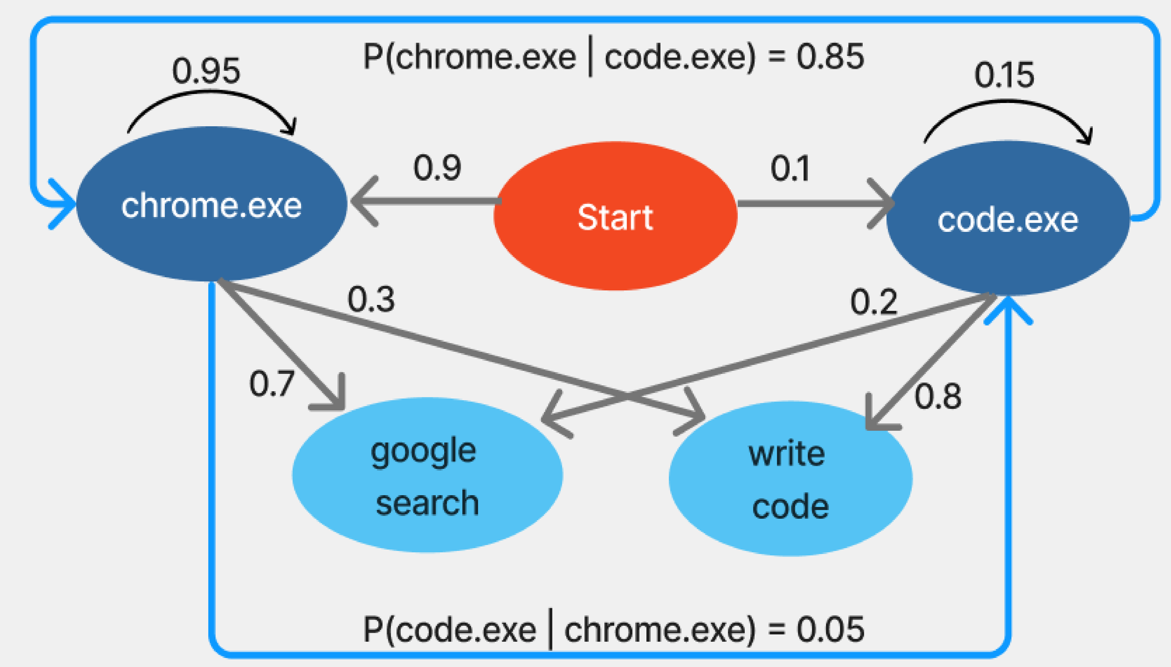
Hidden Markov Model (HMM)

- **Problem Statement**: Predict the likelihood of using an app *given* the former sequence of application usage
- **Idea**: Use conditional probability $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- **A1. Markov Chain**: Only the *current* state q_{i-1} plays the most crucial role in predicting the future in the sequence

$$P(q_i = a|q_1 q_2 ... q_{i-1}) = P(q_i = a|q_{i-1})$$

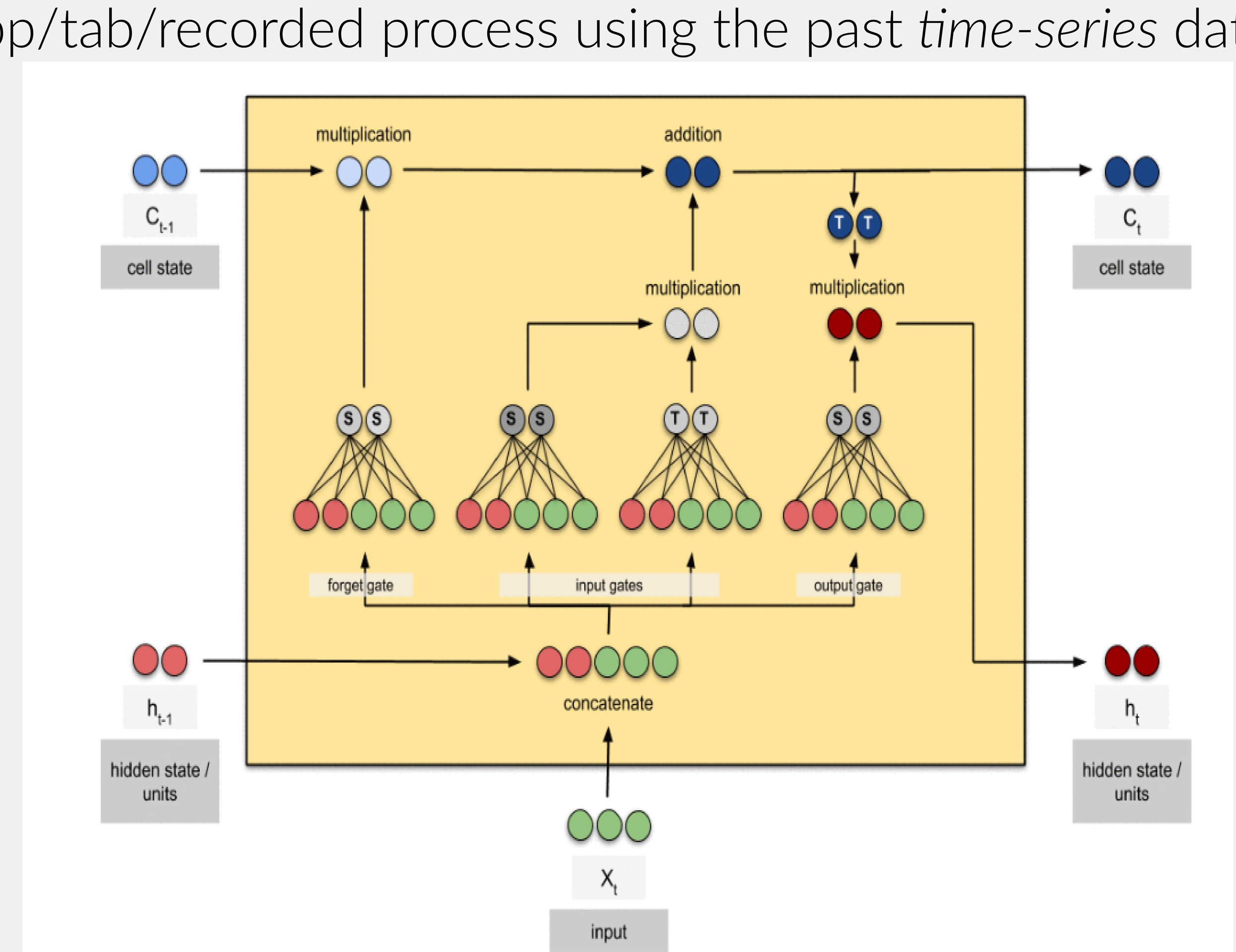
- **A2. Output Independence**: The probability of observing an event o_i only relies on the state q_i that *directly* produced o_i

$$P(o_i|q_1, ...q_i, ..., q_T, o_1, ..., o_i, ..., o_T) = P(o_i|q_i)$$



Recurrent Neural Network (LSTM/RNN)

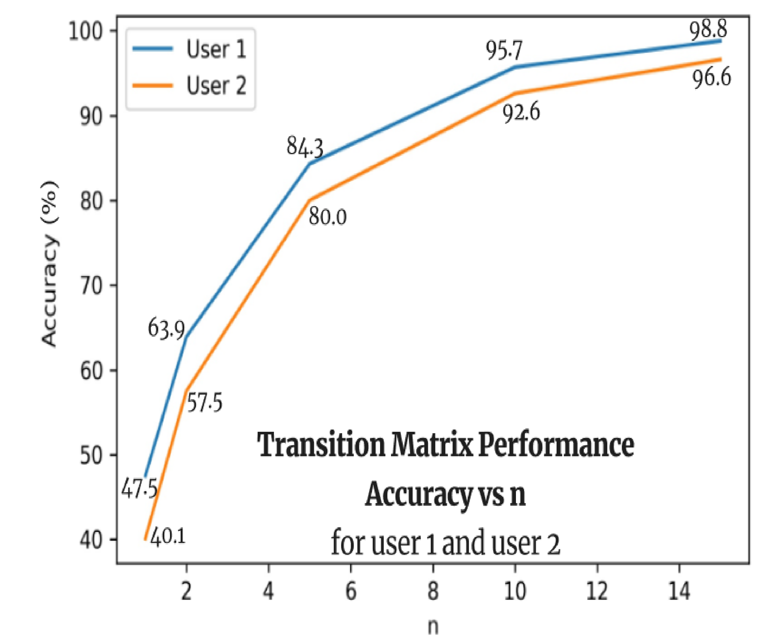
- **Problem Statement**: Predict the (total) time usage of an app/tab/recorded process using the past *time-series* data

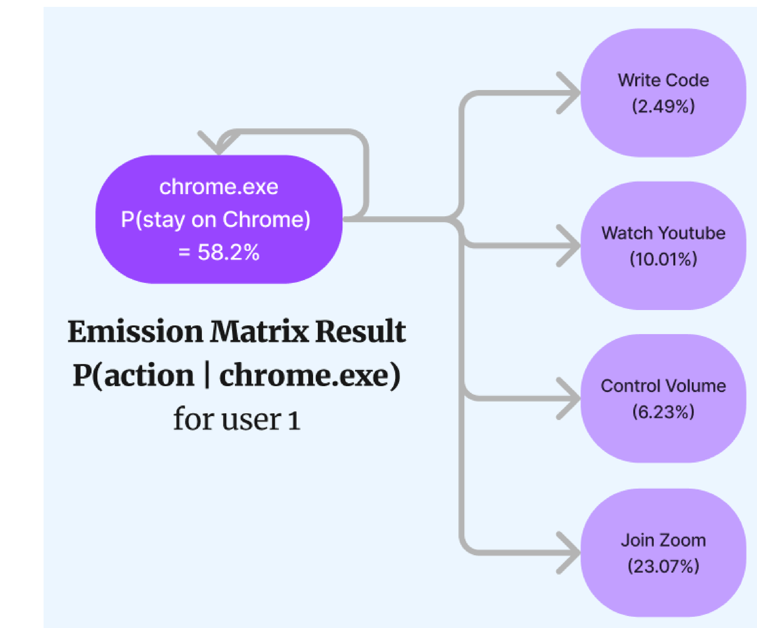


- Feature Engineering:
  1. Hourly split daily usage into 24 cols (labeled 0 - 23)
  2. Lookback 3-5 time steps from the current timestamp
  3. One-hot-encoding; Min-Max scaler
- **Experiments**: Train/Test: 80/20, no shuffle; Keras

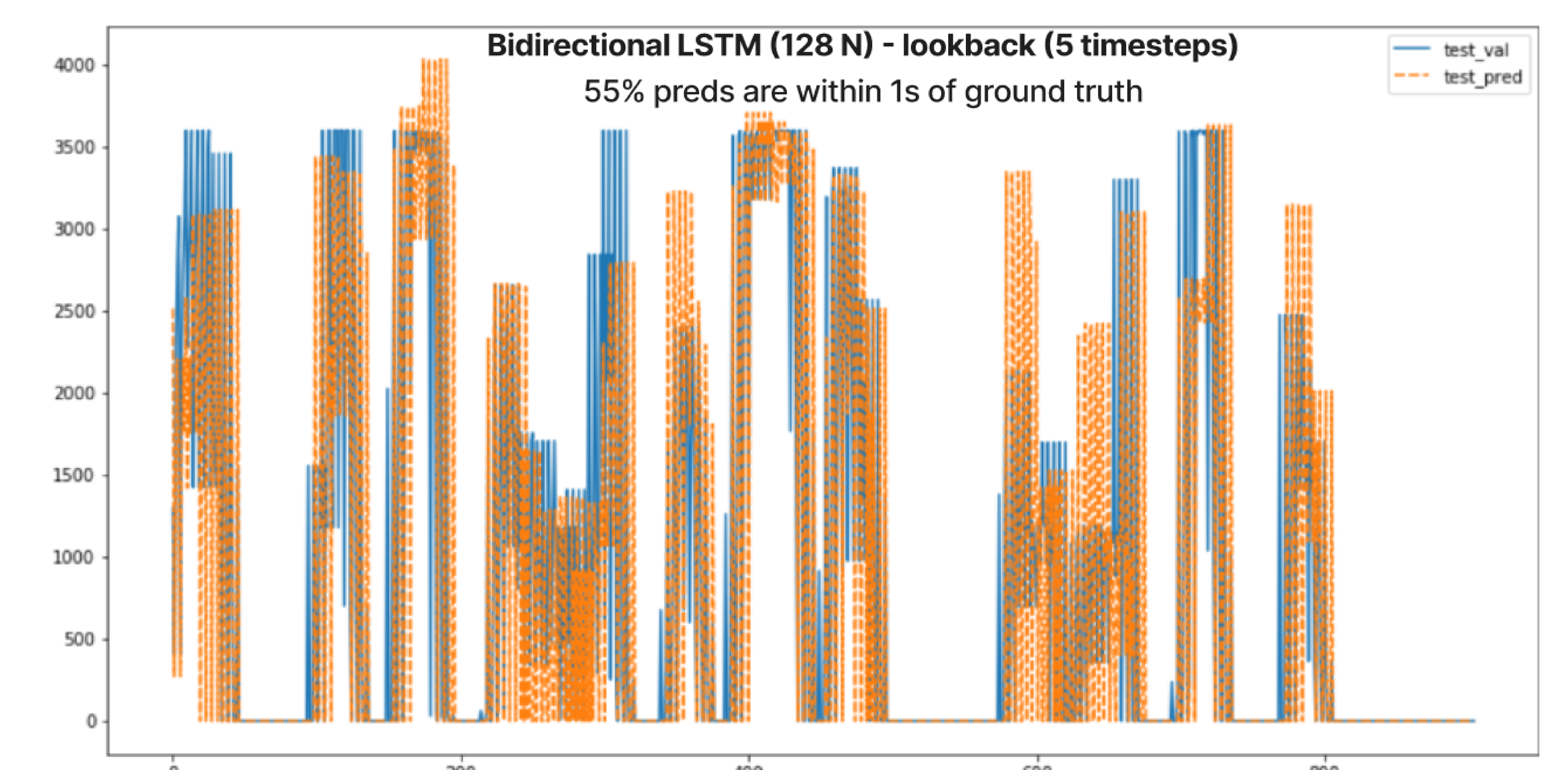
Predictive Results

HMM Metrics: Preds==True if within top n probabilities





LSTM Metrics: RMSE, TP/TN/FP/FN, Preds==True if w/in 1 sec



LSTM/RNN Performance

Model	Design (N=nodes)	Eval Bins/Criteria	Performance
Vanilla LSTM > Split Hourly > OH(Process Name)	Input RNN (64N) Hidden Dense (4N) Output Dense (1N)	[0, 0.01] (0.01, 0.02] (0.02, 0.2] (0.2, max]	TP = 691, TN = 0 FP = 65, FN = 0 ACC = ≈ 91.4%
Stacked LSTM > Split Hourly > OH(Process Name)	Input LSTM (16N) Hidden LSTM (32N) Hidden Dense (64N) Output Dense (1N)	[0, 0.01] (0.01, 0.02] (0.02, 0.2] (0.2, max]	TP = 467, TN = 52 FP = 13, FN = 224 ACC = ≈ 68.65%
Stacked LSTM > Split Hourly > Lookback(5 timesteps)	Input LSTM (50N) Hidden LSTM (50N) Output Dense (1N)	Preds==True if abs(diff) <= 3mins	RMSE = 1134.68 ACC = ≈ 27%

Table 1. LSTM/RNN Performance

Conclusions

- We should collect data *continuously* and *consistently* to achieve high accuracies in detecting patterns of user behaviors
- The results help infer daily app sequence and time usage, so we can develop a script to process background tasks and utilize *Task Manager* to open the next app 2-3 mins beforehand