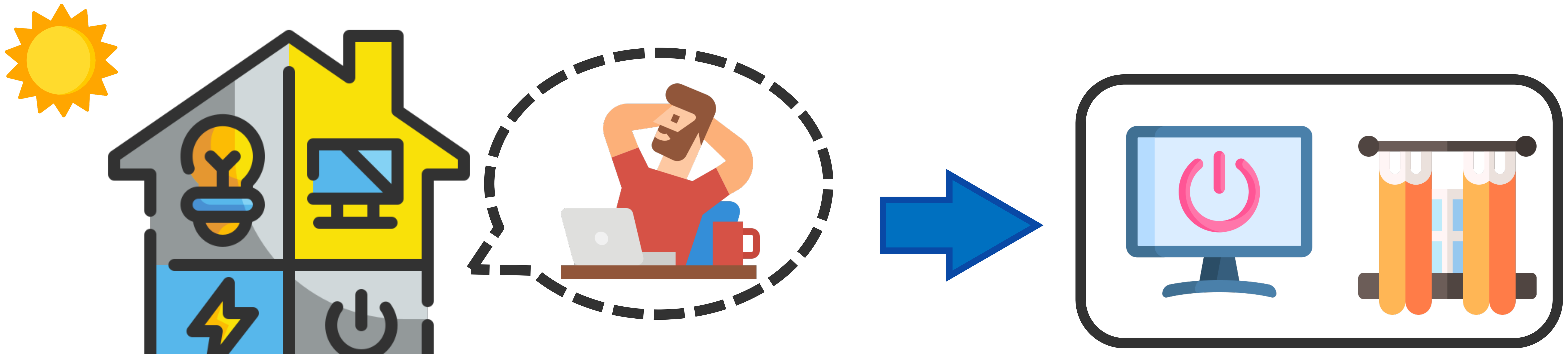
Accurate Action Recommendation for Smart Home via Two-Level Encoders and Commonsense Knowledge

Hyunsik Jeon, *Jongjin Kim*, *Hoyoung Yoon*,
Jaeri Lee, and *U Kang*
Seoul National University

CIKM 2022

Overview

- **Q.** How can we accurately **recommend actions** for users to control their devices at home?
- **A.** ***SmartSense*** accurately recommends device controls to users!



A user in Smart Home

SmartSense *recommends device controls*

Outline

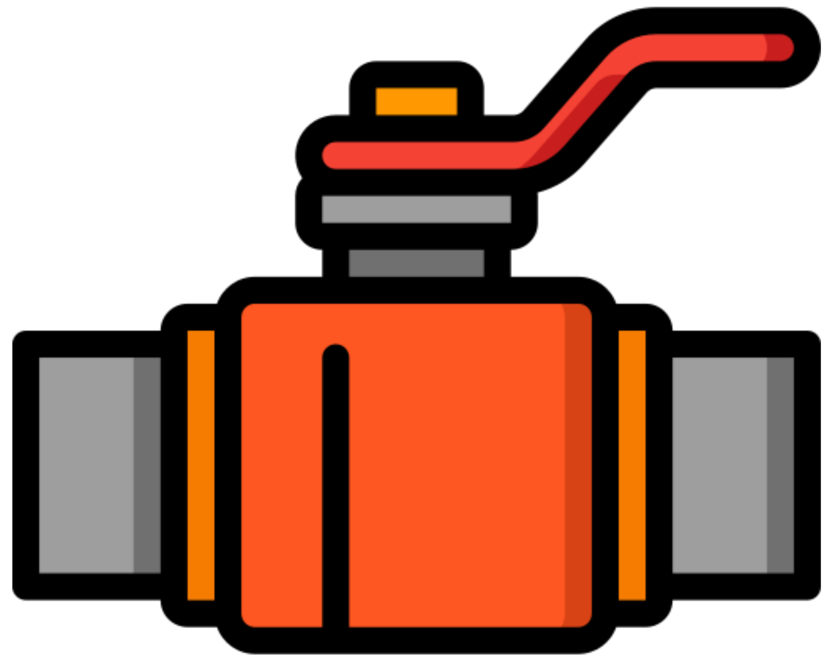
- **Introduction**
- Proposed Method
  - Motivation
  - Main Ideas
- Experiments
- Conclusion

Recommender System

- **Recommender systems** provide personalized items among the plethora of ones for each user
- They are **essential** in various online services
  - They **enhance** users' experience and **increase** sales revenue
- Applications
  - Amazon (e-commerce)
  - YouTube (video)
  - Netflix (movie)
  - Spotify (music)

Action Recommender System

- **Action recommender system** is necessary for **smart home**
  - It keeps users safe when they forget a critical action (e.g., shutting off a gas valve)
  - It reduces the hassles of users when performing a cumbersome action (e.g., arming an alarm)



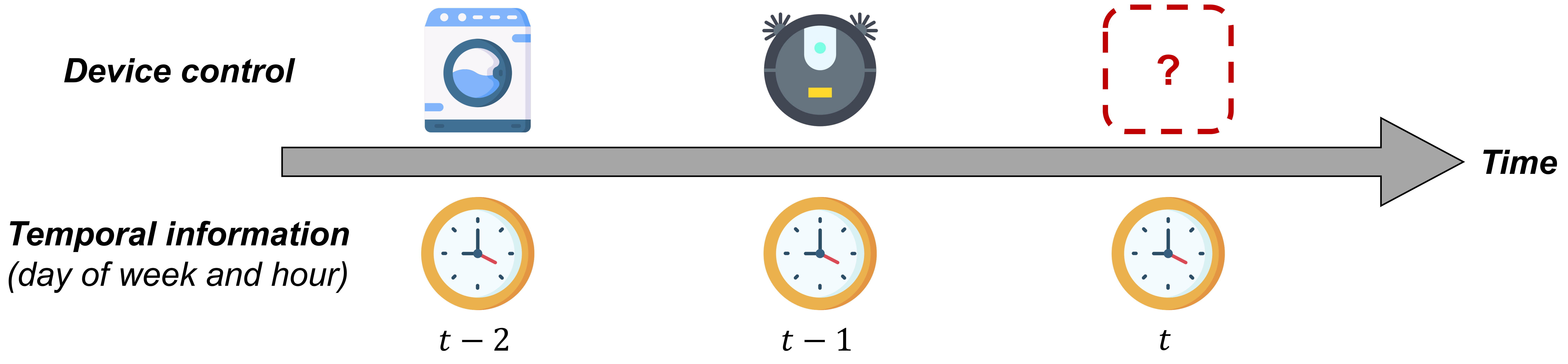
Recommend shutting off a gas valve

Recommend arming an alarm

Problem Definition

Action recommendation for smart home

- **Given** a user's historical actions before time t and temporal information at time t
  - Each action contains a device control and its temporal information
- **Predict** the user's device control at t

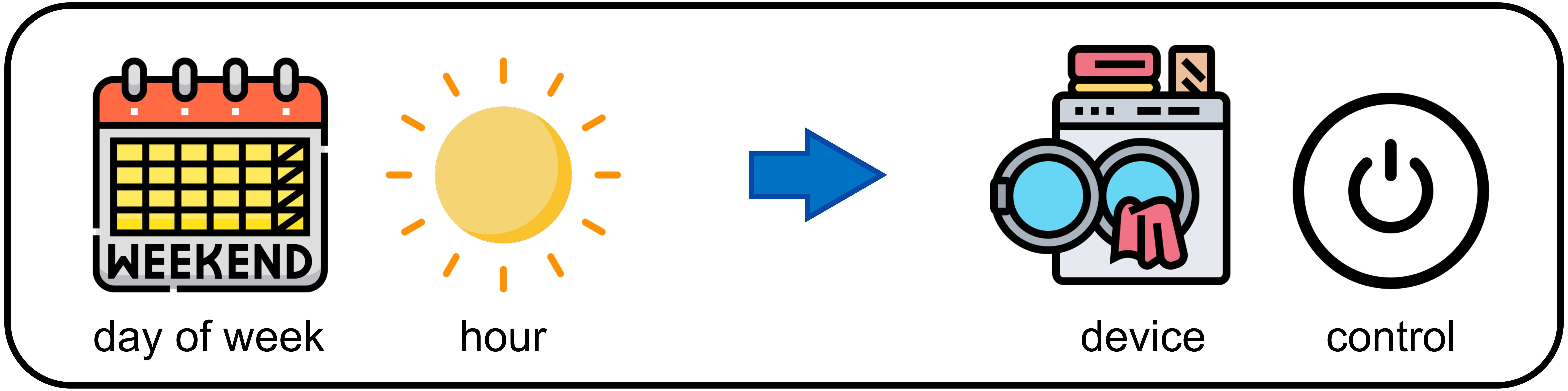


Outline

- Introduction
- **Proposed Method**
  - **<u>Motivation</u>**
  - Main Ideas
- Experiments
- Conclusion

Challenges

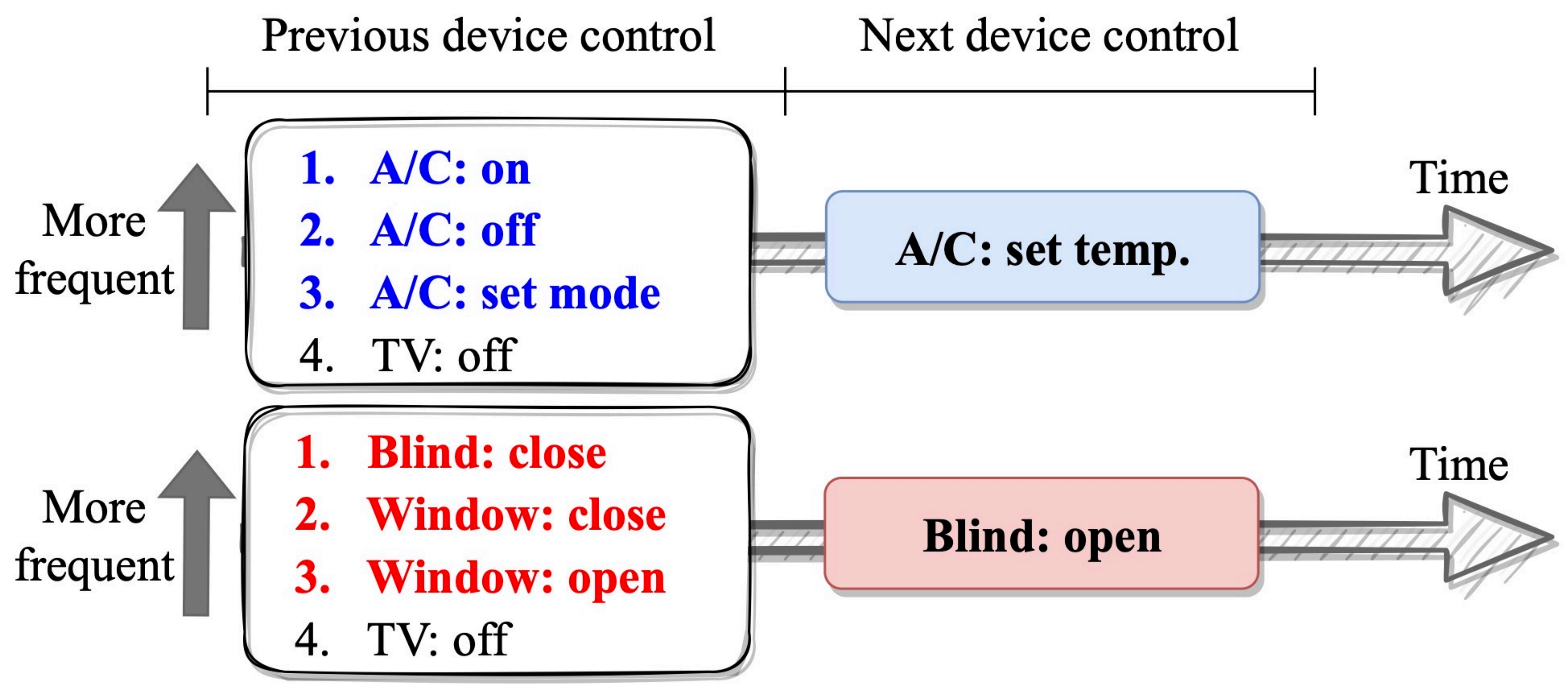
- **Complicated correlation** of an action
  - A user's action is affected by complex temporal information
    - E.g., people usually do laundry during the day on weekends



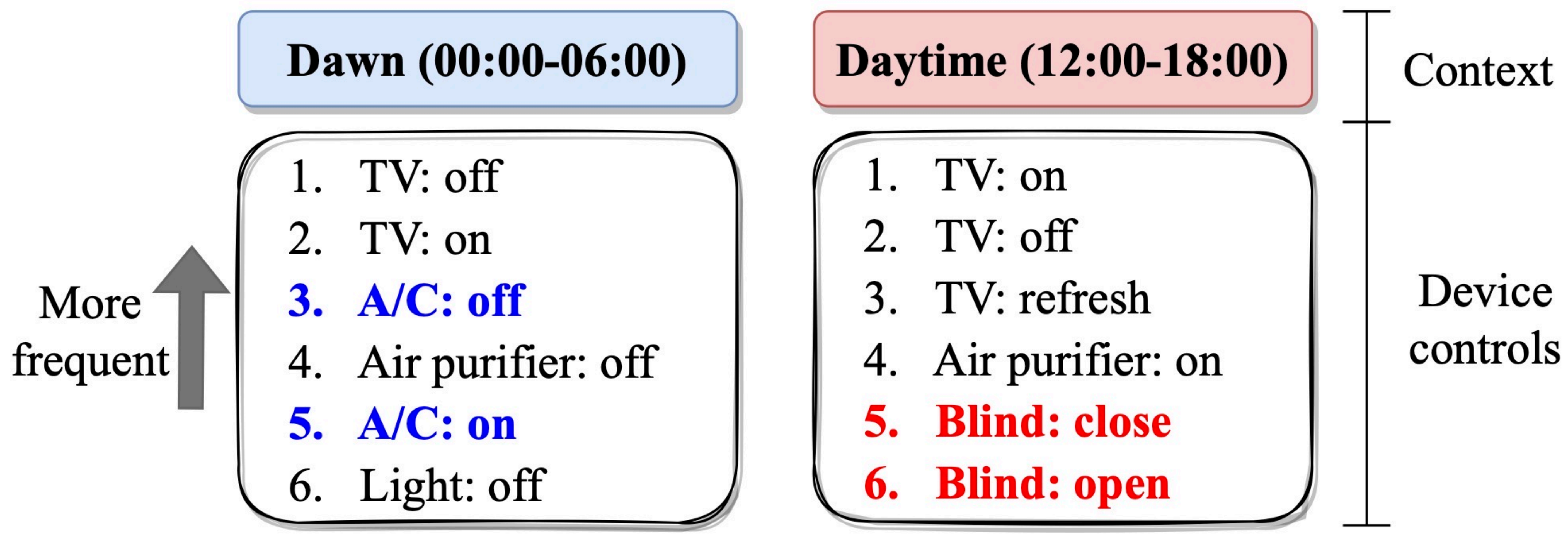
***Complicated correlation** of an action*

Challenges

- **Historical and contextual dependencies** of an action
  - A user action depends both on the history and the current context



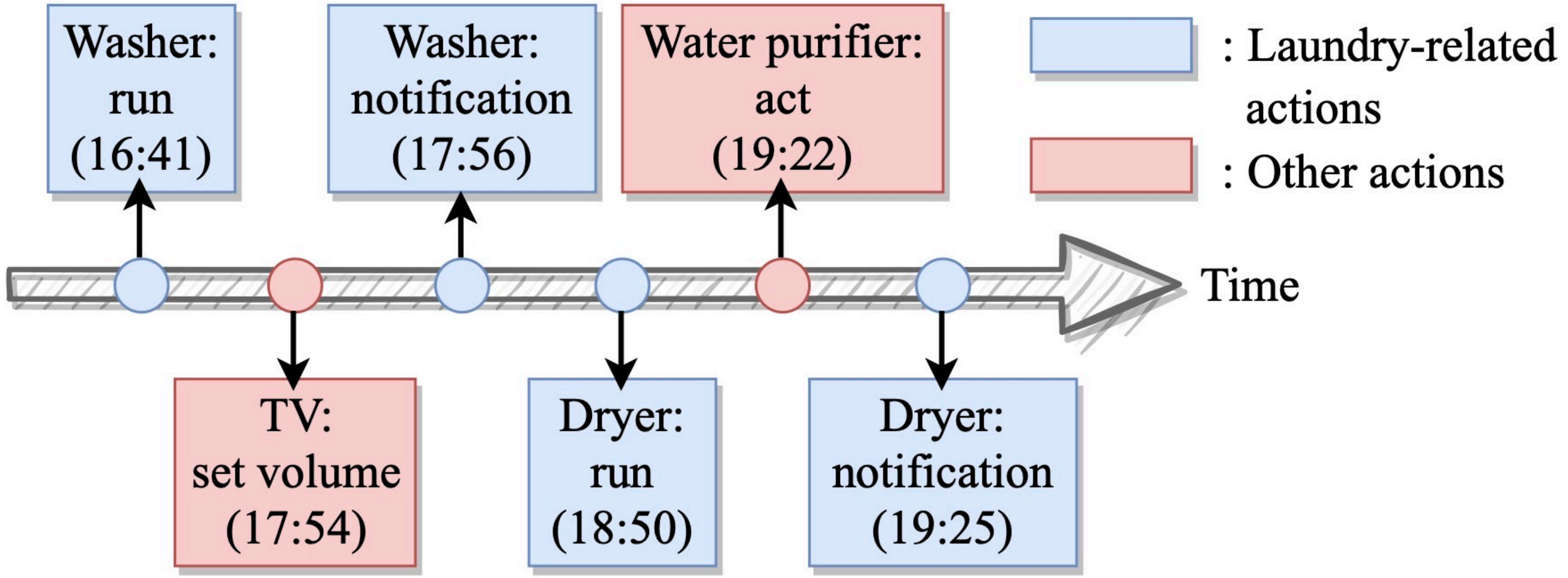
***Historical dependency** of device controls*



***Contextual dependency** of device controls*

Challenges

- **Capricious intention**
  - A user's sequential actions contain capricious intentions
    - People do not always act in sequence with only one intention
    - This easily leads to degraded performance of recommendation



A sequence of actions contains ***capricious intention***

Research Motivation

- **Existing methods for action recommendation**
  - They **miss** addressing one or more of the main challenges

Method \ Challenges	Complicated correlation	Historical and contextual dependencies	Capricious intention
FMC, TransRec, Caser, SASRec, BERT4Rec			
SIAR	✓		
CA-RNN	✓	✓	

How to address the three main challenges?

Outline

- Introduction
- **Proposed Method**
  - Motivation
  - **<u>Main Ideas</u>**
- Experiments
- Conclusion

Proposed Method – Overview (1)

- We propose ***SmartSense***
  - An accurate action recommender system for smart home
- **Idea 1.** Self- and query-attention for an action
  - To capture significant correlations in an action
- **Idea 2.** Self- and context-attention for a sequence
  - To handle historical and contextual dependencies in a sequence
- **Idea 3.** Knowledge transfer from common routines
  - To learn proximity between devices

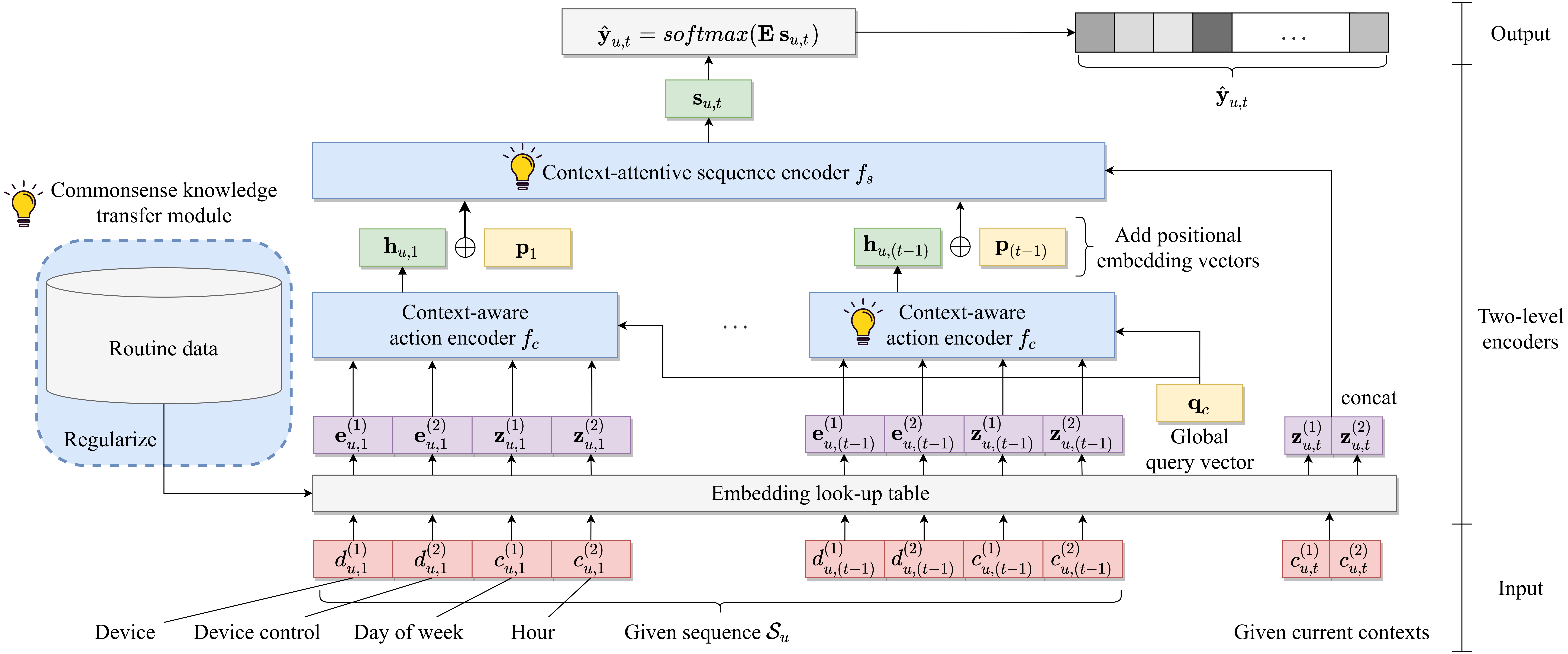
Proposed Method – Overview (2)

- The proposed ***SmartSense*** **wins** on features

Challenges / Method	Complicated correlation in an action	Historical and contextual deps. in a sequence	Capricious intention
FMC, TransRec, Caser, SASRec, BERT4Rec			
SIAR	✔		
CA-RNN	✔	✔	
SmartSense (ours)	✔	✔	✔

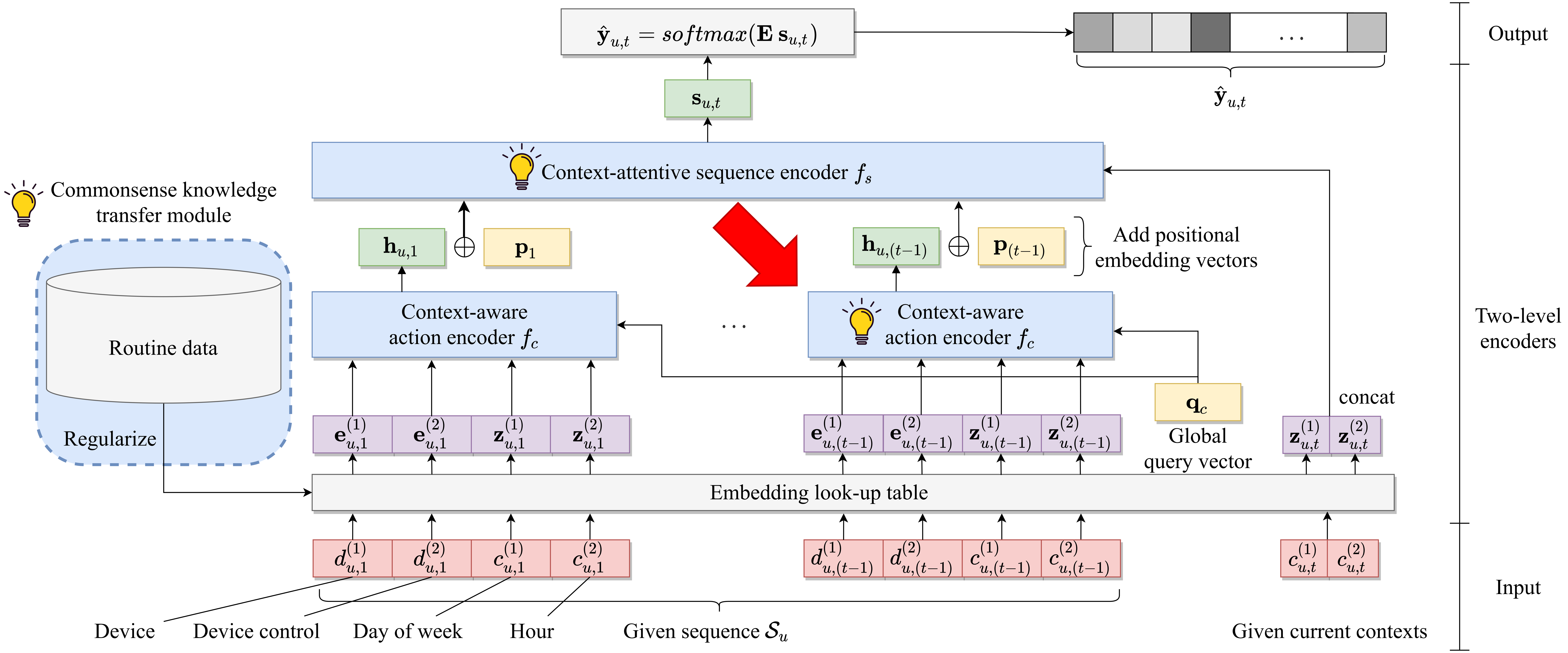
Architecture

- The overall architecture of ***SmartSense***
  - Action encoder, sequence encoder, knowledge transfer module



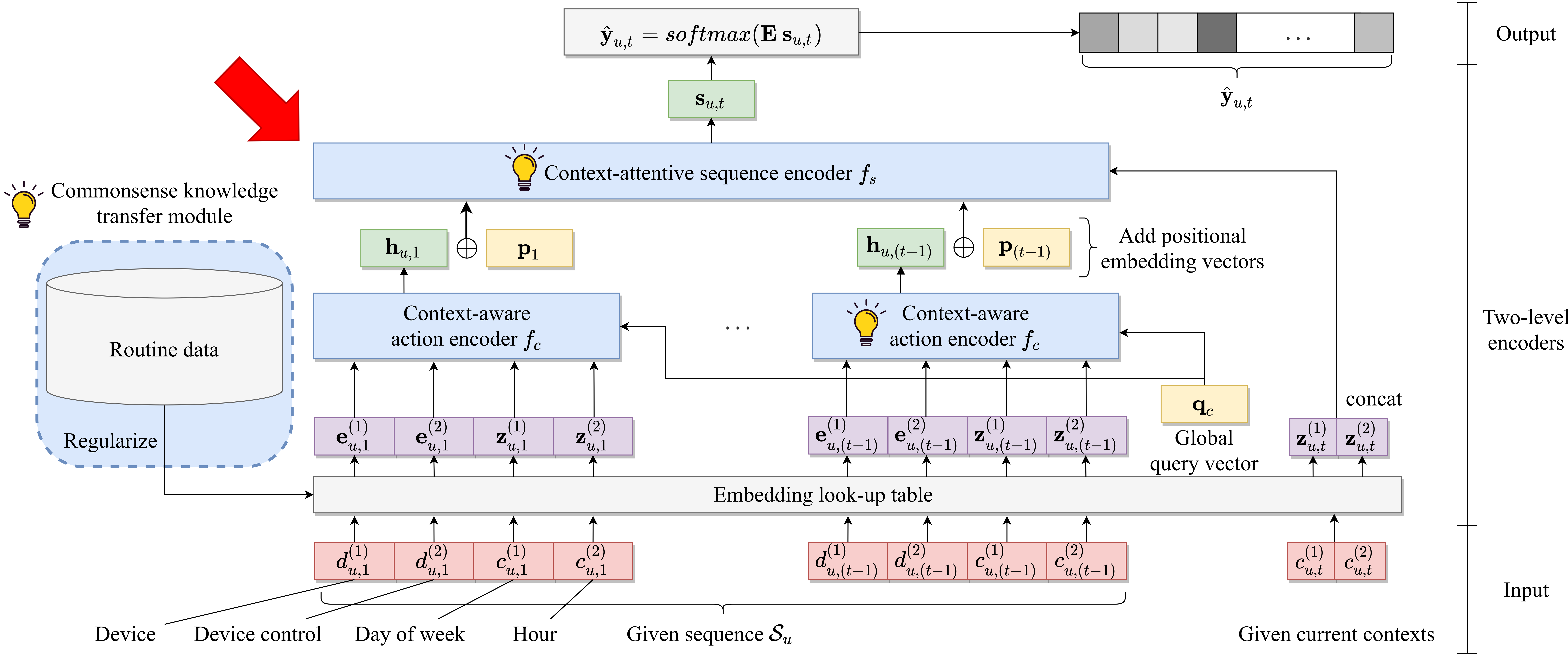
Architecture

- **Action encoder** summarizes an action into a vector, capturing significant correlations in the action



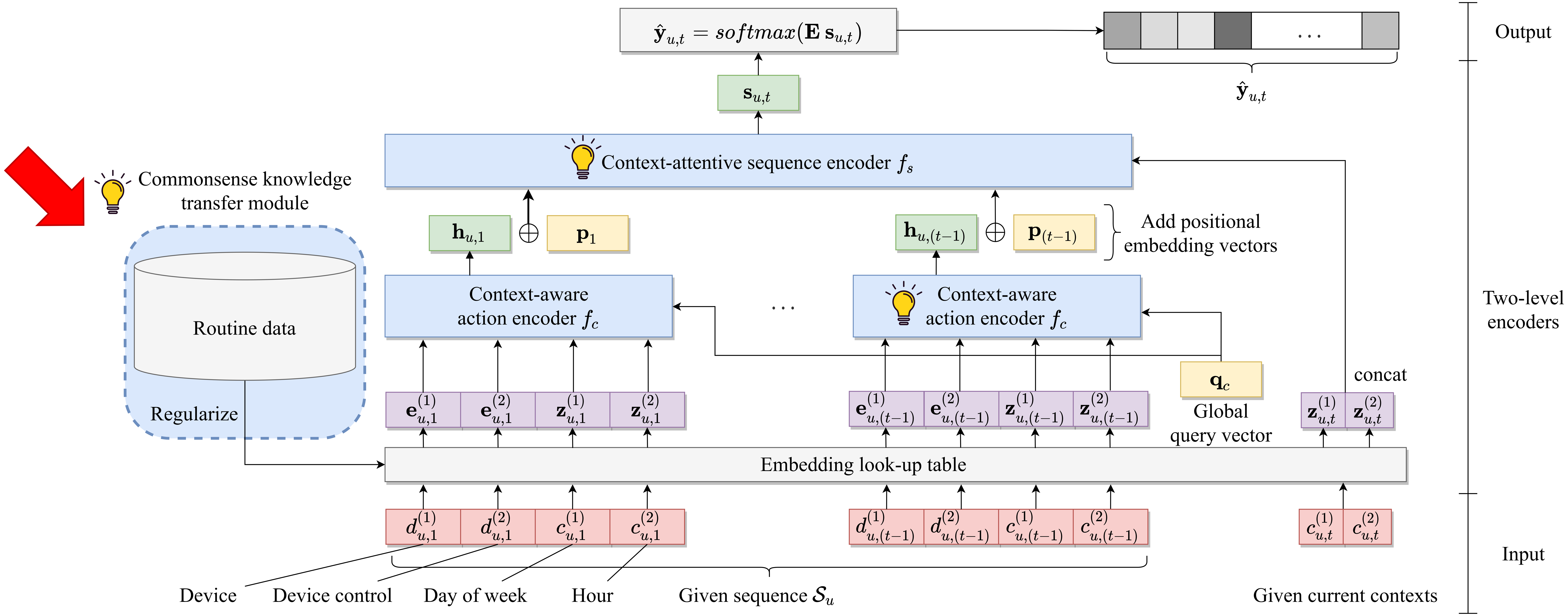
Architecture

- **Sequence encoder** summarizes sequential actions into a vector, handling historical and contextual dependencies



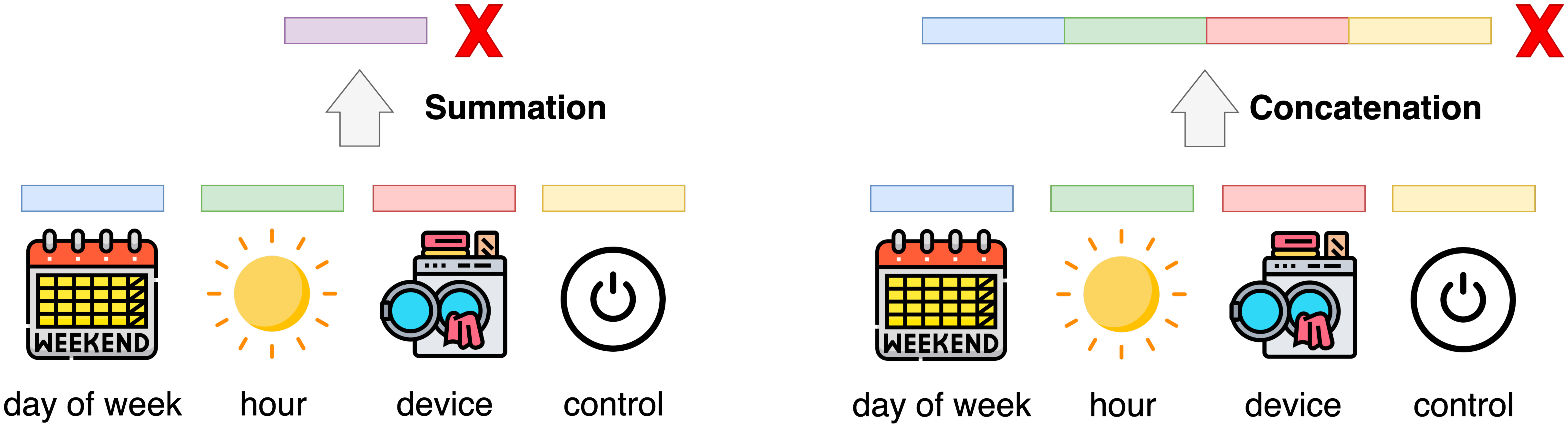
Architecture

- **Knowledge transfer module** regularizes device embeddings utilizing routine data to learn proximity between devices



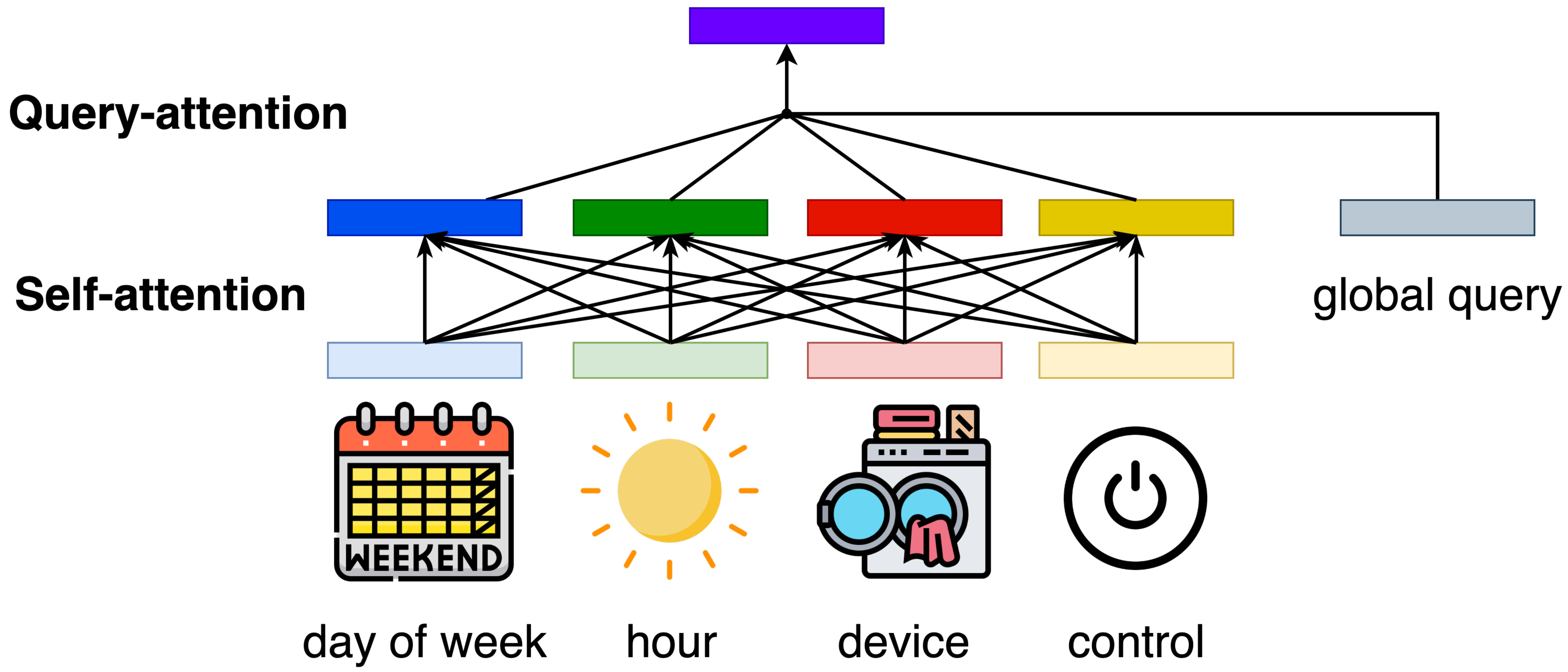
Action Encoder (1)

- **Challenge 1.** How to capture the **significant correlation** among the complicated ones in an action?
  - Simple aggregations (e.g., summation or concatenation) cannot identify the complicated correlations between the variables



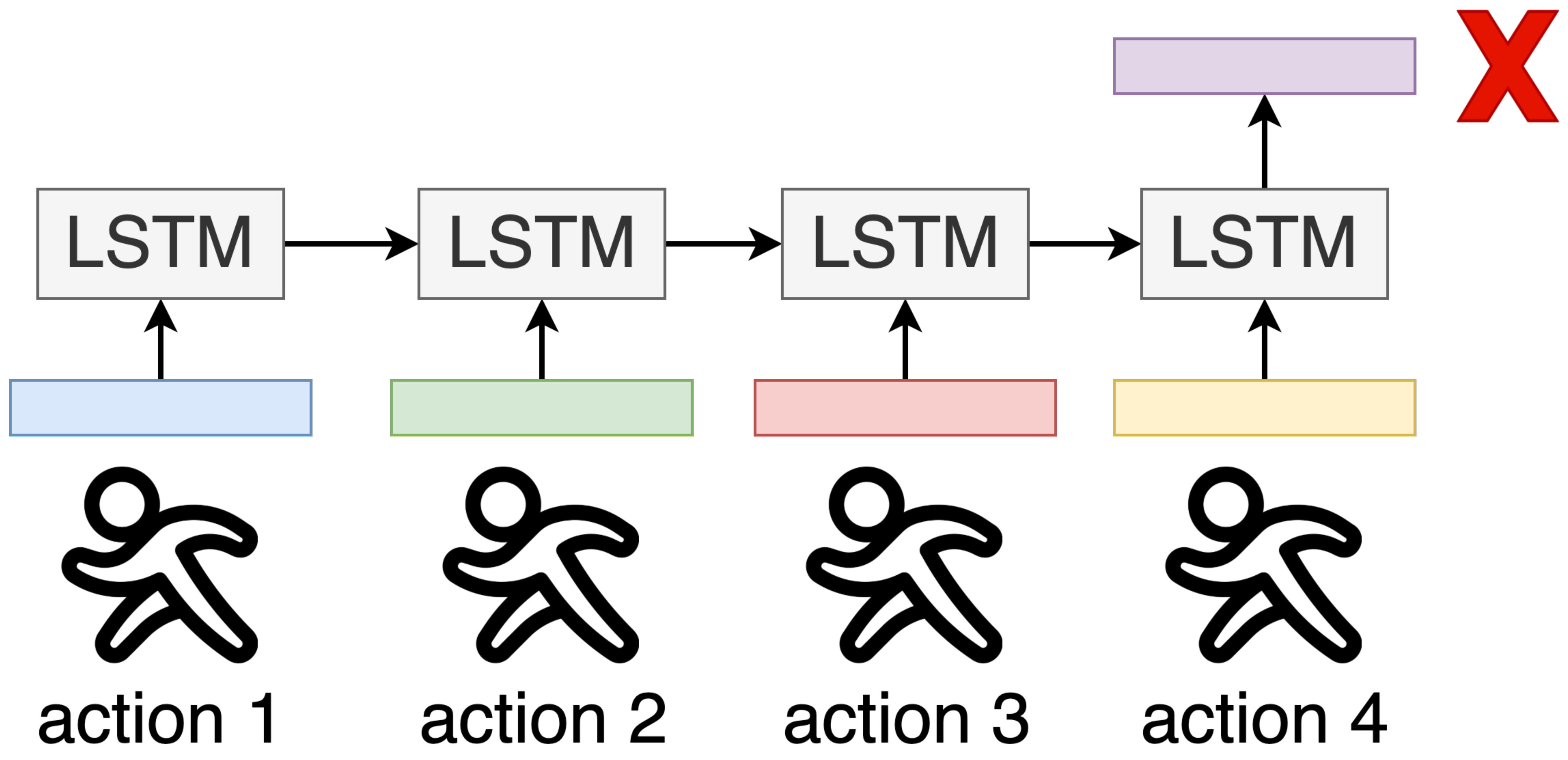
Action Encoder (2)

- **Idea 1-1. Self-attention** for input variables
  - To correlate given information in the action
- **Idea 1-2. Query-attention** for summarization
  - To capture significant correlations in the action



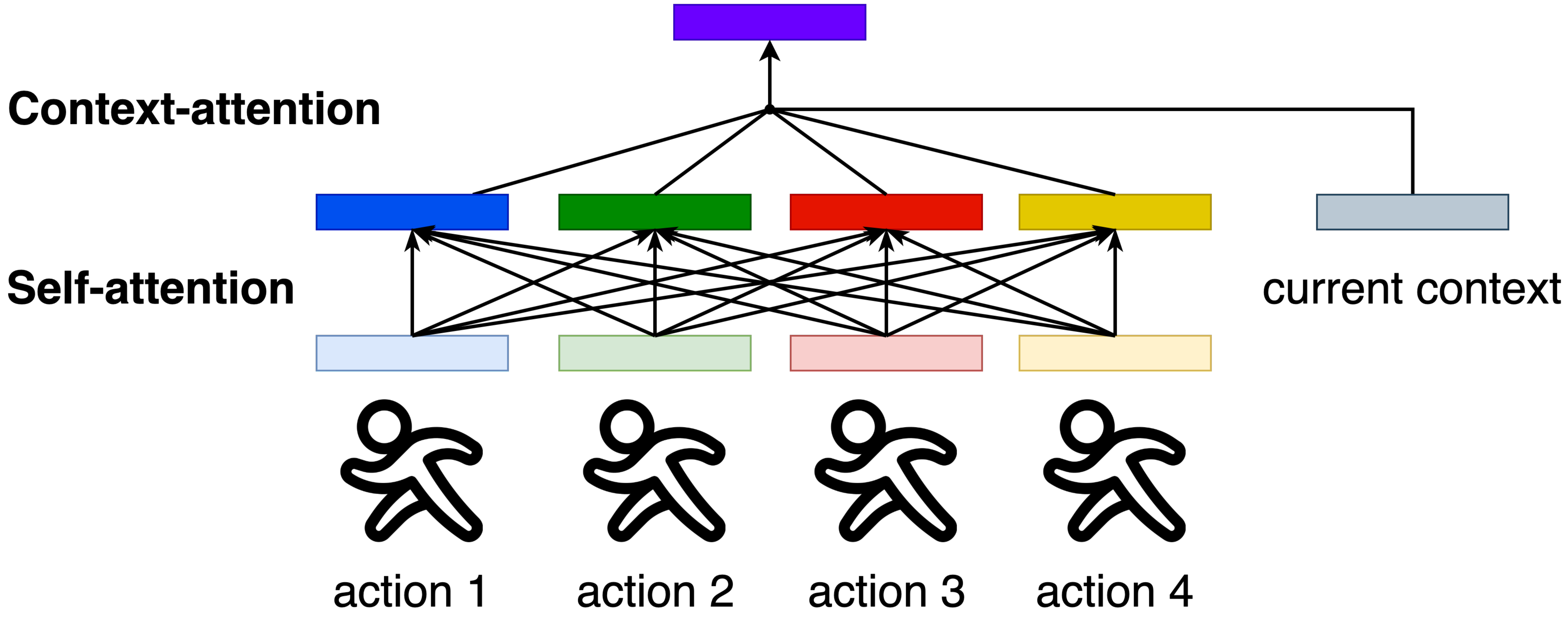
Sequence Encoder (1)

- **Challenge 2.** How to handle the **historical and contextual dependencies** in a sequence?
  - Two types of correlations are important
    - Between actions in a sequence (historical dependency)
    - Between each action and the current context (contextual dependency)
  - A simple RNN-based model is restricted for both dependencies



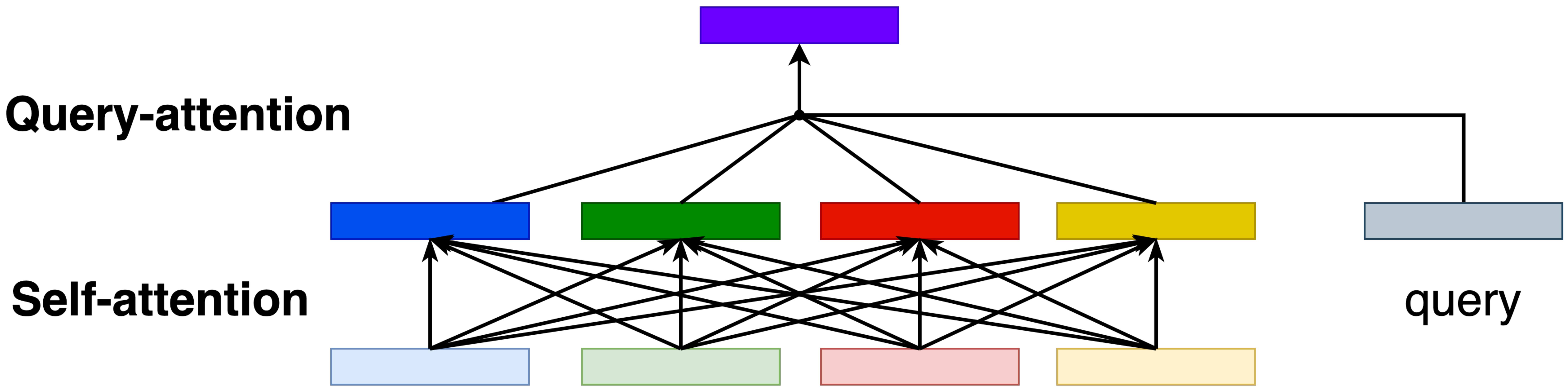
Sequence Encoder (2)

- **Idea 1-1.** **Self-attention** for sequential actions
  - To correlate between actions in the sequence
- **Idea 1-2.** **Context-attention** for summarization
  - To correlate each action and the current context



Queried Transformer Encoder (1)

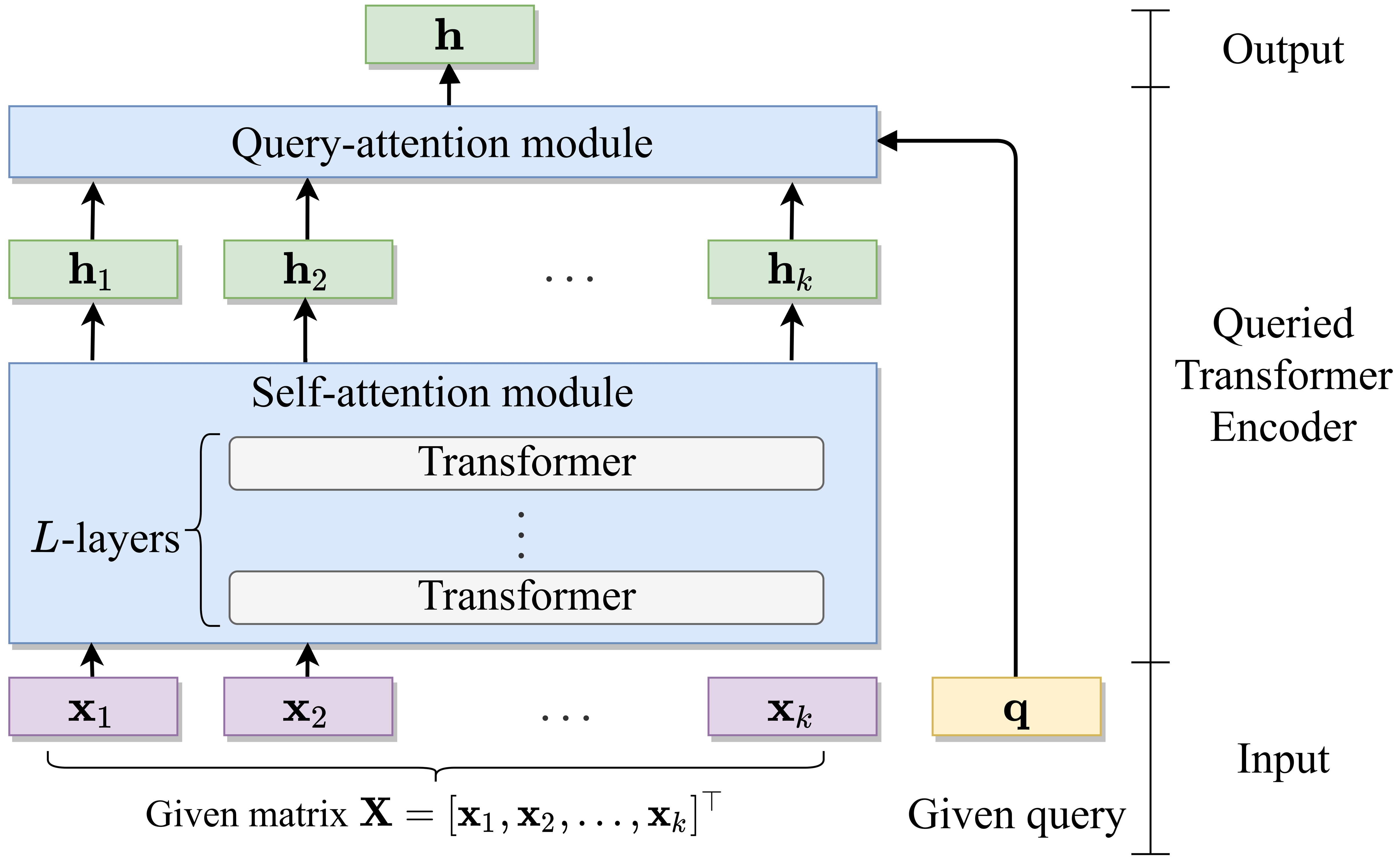
- The action encoder and sequence encoder necessitate the common functionalities: **self- and query-attention**



How to design an architecture to embody the two functionalities?

Queried Transformer Encoder (2)

- We propose **QTE** (Queried Transformer Encoder)
  - **Definition:** $\mathbf{h} = f(\mathbf{X}, \mathbf{q})$
    - f is the QTE function
  - **Given** a set of input vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_k]^\top$, and a query vector $\mathbf{q}$
  - **QTE**
    - It transforms the input matrix X into $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_k]^\top$ through **self-attention module**
    - It summarizes $\mathbf{H}$ into a vector $\mathbf{h}$ with **query-attention module** using the given query vector $\mathbf{q}$



Queried Transformer Encoder (3)

- **Self-attention module** of QTE
  - **The goal** is to correlate the given vectors $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_k]^\top \in \mathbb{R}^{k \times d}$
    - k is the number of input vectors, and d is the size of vector
  - **Make** query, key, and value matrices as follows:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \mathbf{K} = \mathbf{X}\mathbf{W}^K, \mathbf{V} = \mathbf{X}\mathbf{W}^V$$

    - $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are learnable weights
  - **Compute** matrix $\bar{\mathbf{X}}$ as follows:

$$\bar{\mathbf{X}} = \mathbf{A}\mathbf{V} \quad \text{where} \quad \mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)$$

  - **Obtain** transformed matrix $\mathbf{H} \in \mathbb{R}^{k \times d}$ using a network as follows:

$$\mathbf{H} = \text{Trans}(\mathbf{X}) = \mathbf{X} + \bar{\mathbf{X}} + \text{FNN}(\mathbf{X} + \bar{\mathbf{X}})$$

Queried Transformer Encoder (4)

- **Query-attention module** of QTE
  - **The goal** is to summarize the vectors $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_k]^\top \in \mathbb{R}^{k \times d}$ into a vector $\mathbf{h}$ while capturing significant information depending on the query $\mathbf{q}$
  - **Summarize $\mathbf{H}$ into $\mathbf{h}$ using $\mathbf{q}$** as follows:

$$\mathbf{h} = \mathrm{QueryAtt}(\mathbf{H}, \mathbf{q}) = \sum_{i=1}^{k} \alpha_i \mathbf{h}_i, \quad \text{where}$$

$$\alpha_i = \frac{\exp(\beta_i)}{\sum_{j=1}^{k} \exp(\beta_j)}, \quad \beta_i = \mathbf{q}^\top \tanh\left(\mathbf{W}^H \mathbf{h}_i + \mathbf{b}^H\right)$$

    - α_i and β_i are normalized and unnormalized scores of $\mathbf{h}_i$ for $\mathbf{q}$, respectively
    - $\mathbf{W}^H$ and $\mathbf{b}^H$ are learnable weight and bias, respectively

Revisit the Action Encoder

- **Obtain** learnable embedding vectors for each variable in an action: $\mathbf{e}_{u,i}^{(1)}, \mathbf{e}_{u,i}^{(2)}, \mathbf{z}_{u,i}^{(1)}, \mathbf{z}_{u,i}^{(2)} \in \mathbb{R}^d$
  - The embeddings of ith device, device control, day of week, and hour, respectively, for user u
- **Employ** QTE as follows:

$$\mathbf{h}_{u,i} = f_c(\mathbf{X}_{u,i}, \mathbf{q}_\mathbf{c})$$

  - $\mathbf{X}_{u,i} \in \mathbb{R}^{4\times d} = \left[\mathbf{e}_{u,i}^{(1)}, \mathbf{e}_{u,i}^{(2)}, \mathbf{z}_{u,i}^{(1)}, \mathbf{z}_{u,i}^{(2)}\right]^\top$ is the set of input embeddings
  - $\mathbf{q}_c \in \mathbb{R}^d$ is a learnable global query vector
  - $f_c(\cdot)$ is the action encoder with QTE structure

Revisit the Sequence Encoder (1)

- **Obtain** learnable embedding vectors for the current contexts:

$$\mathbf{z}_{u,t}^{(1)}, \mathbf{z}_{u,t}^{(2)} \in \mathbb{R}^d$$

  - The embeddings of ith day of week and hour, respectively, for user u
- **Employ** QTE as follows:

$$\mathbf{s}_{u,t} = f_s(\mathbf{H}_u + \mathbf{P}, \mathrm{concat}(\mathbf{z}_{u,t}^{(1)}, \mathbf{z}_{u,t}^{(2)}))$$

  - $\mathbf{H} \in \mathbb{R}^{(t-1)\times d} = \left[\mathbf{h}_{u,1}, \dots, \mathbf{h}_{u,(t-1)}\right]^{\top}$ is stacked vectors of actions
  - $\mathbf{P} \in \mathbb{R}^{(t-1)\times d}$ is a learnable positional embedding matrix to identify the position of the input vectors
  - $f_s(\cdot)$ is the sequence encoder with QTE structure

Revisit the Sequence Encoder (2)

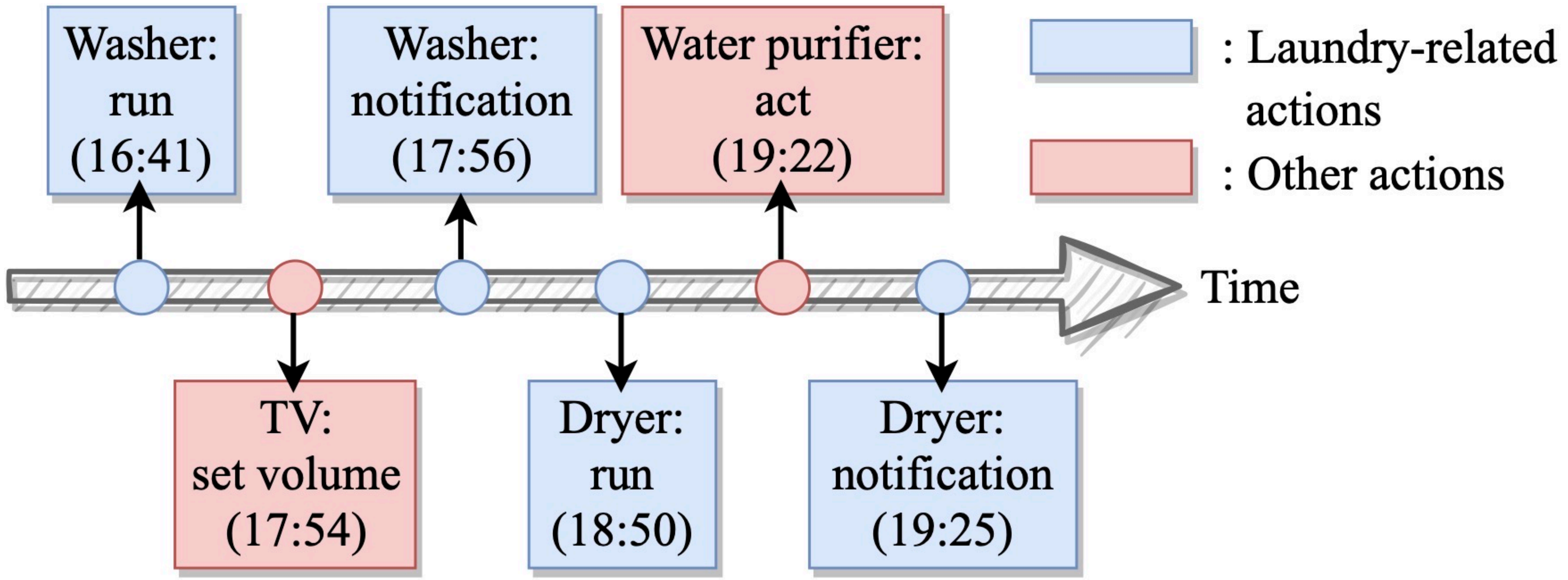
- **Predict** the probabilities of device controls as follows:

$$\hat{\mathbf{y}}_{u,t} = \mathrm{softmax}(\mathbf{E}\,\mathbf{s}_{u,t})$$

  - $\hat{\mathbf{y}}_{u,t} \in \mathbb{R}^{N_d}$ is the predicted probabilities of device controls for user u at time t
  - $\mathbf{E} \in \mathbb{R}^{N_d \times d}$ is the learnable embedding matrix of device controls for the prediction
  - N_d is the number of device controls

Commonsense Knowledge Transfer (1)

- **Challenge 3.** How to learn **proximity between devices**?
  - Capricious intentions in historical actions may mislead the model to learn false proximity between two co-occurred actions



Commonsense Knowledge Transfer (2)

- **Idea 3. Commonsense knowledge transfer** from routine data
  - To effectively learn proximity between devices
- Routine data
  - Collection of frequently used device patterns registered by various users
  - Devices of each routine are probable to share a common intention
    - E.g., sequential actions of laundry, or sequential actions of cooling off the room

Commonsense Knowledge Transfer (3)

- **Regularization term**
  - We regularize the model to learn proximity between devices in the same routines

$$\mathcal{L}_{reg} = -\sum_{i}\sum_{d_j \in \mathcal{R}_i}\left(\log\left(\sigma(\mathbf{e}_j^\top \mathbf{e}_{j+1})\right) + \sum_{d_k \in p(\mathcal{R}_i)} \log\left(\sigma(-\mathbf{e}_j^\top \mathbf{e}_k)\right)\right)$$

    - $\mathcal{R}_i$ is ith routine which consists of sequential devices
    - $p(\mathcal{R}_i)$ is random negative samples of $\mathcal{R}_i$
    - $e_j \in \mathbb{R}^d$ is the embedding vector of device d_j
      - It is **shared** with the device embedding of the model

Objective Function

- We train ***SmartSense*** to minimize the **cross-entropy loss** and the **regularization loss** as follows:

$$\mathcal{L}(\mathcal{X}, \mathbf{Y}) = -\frac{1}{n}\sum_{u}\sum_{i}\mathbf{y}_u(i)\log\hat{\mathbf{y}}_u(i) + \mathcal{L}_{reg}$$

Cross-entropy loss **Regularization loss**

  - $\mathcal{X} \in \mathbb{R}^{n \times l \times 4}$ is an input tensor of n sessions and l time steps
  - $\mathbf{Y} \in \mathbb{R}^{n \times N_d}$ is a matrix of ground-truth labels
  - $\mathbf{y}_u \in \mathbb{R}^{N_d}$ is the one-hot vector of the ground-truth label for session u
  - $\hat{\mathbf{y}}_u \in \mathbb{R}^{N_d}$ is the predicted probabilities for session u
  - $\mathbf{y}_u(i), \hat{\mathbf{y}}_u(i) \in \mathbb{R}$ are ith element in $\mathbf{y}_u$ and $\hat{\mathbf{y}}_u$, respectively

Outline

- Introduction
- Proposed Method
- **<u>Experiments</u>**
- Conclusion

Questions

- We answer the following questions by experiments:
  - **Q1 (Accuracy).** Does *SmartSense* achieve higher accuracy than competitors?
  - **Q2 (Ablation study).** Do the main ideas of *SmartSense* help improve performance?
  - **Q3 (Case study).** How does *SmartSense* recommend device controls according to the current contexts?
  - **Q4 (Embedding analysis).** Does *SmartSense* successfully learn proximity between devices?

Datasets

- We use real-world datasets of Samsung SmartThings users
  - Four log datasets

Name	Region	Time period (Y-M-D)	# Sessions	# Instances	# Devices	# Device controls
KR	Korea	2021-11-20 ~ 2021-12-20	12,992	285,409	38	272
US	USA	2022-02-22 ~ 2022-03-21	4,764	67,882	40	268
SP	Spain	2022-02-28 ~ 2022-03-30	1,506	15,665	34	234
FR	France	2022-02-27 ~ 2022-03-25	388	4,423	33	222

  - Three routine datasets: (AP→KR, NA→US, EU→SP/FR)

Name	Region	# Routines	# Devices
AP	Asia-Pacific	17,773	36
NA	North America	26,241	35
EU	Europe	23,781	28

Baselines

- We compare ***SmartSense*** with 8 competitors
  - **Pop** is a popularity-based recommendation model
  - **FMC**, **TransRec**, **Caser**, **SASRec**, and **BERT4Rec** are sequential recommendation models
  - **SIAR** and **CA-RNN** are context-aware recommendation models

Experimental Settings

- **Evaluation metric**
  - We evaluate the performance with **mean average precision (mAP@k)** which treats higher-ranked items more importantly
- **Experimental process**
  - We create sequential instances with a window of the length of 10
    - 9 input actions / 1 target action
  - We randomly split the instances into trn/vld/test sets by 7:1:2 ratio
  - The hour is one of the 8 time ranges of 3 hours in length
    - 0-3, 3-6, 6-9, 9-12, 12-15, 15-18, 18-21, and 21-24

Q1. Accuracy

- **Q1.** Does *SmartSense* achieve higher accuracy than competitors?
- **A1.** *SmartSense* **outperforms** the competitors

Model	Korea @1	Korea @3	Korea @5	USA @1	USA @3	USA @5	Spain @1	Spain @3	Spain @5	France @1	France @3	France @5
	mAP@k											
POP	0.3416	0.4918	0.5045	0.1886	0.3146	0.3737	0.4973	0.6337	0.6455	0.4949	0.5955	0.6114
FMC [31]	0.5075	0.6391	0.6569	0.4581	0.6082	0.6270	0.4102	0.5953	0.6015	0.4427	0.6330	0.6477
TransRec [7]	0.3854	0.5637	0.5830	0.3351	0.5240	0.5426	0.3819	0.6149	0.6209	0.4255	0.6238	0.6393
Caser [36]	0.5676	0.7064	0.7213	0.5535	0.7051	0.7177	0.7906	0.8548	0.8616	0.7706	0.8249	0.8295
SASRec [16]	0.5763	0.7064	0.7212	0.5657	0.7098	0.7228	0.7929	0.8570	0.8630	0.7740	0.8286	0.8389
BERT4Rec [34]	0.5927	0.7253	0.7393	0.5630	0.7121	0.7254	0.7887	0.8610	0.8662	0.7776	0.8475	0.8507
CA-RNN [25]	0.5703	0.6958	0.7095	0.4860	0.6315	0.6459	0.6748	0.7253	0.7350	0.5141	0.5650	0.5767
SIAR [29]	0.5936	0.7248	0.7381	0.5718	0.7163	0.7288	0.7913	0.8560	0.8628	0.7706	0.8258	0.8311
SmartSense (proposed)	**0.6515**	**0.7650**	**0.7760**	**0.6247**	**0.7541**	**0.7639**	**0.8101**	**0.8707**	**0.8756**	**0.7944**	**0.8544**	**0.8578**

Q2. Ablation Study

- **Q2.** Do the main ideas of *SmartSense* help improve performance?
- **A2.** All three main ideas help **improve** the performance
  - **Act**, **Seq**, and **Reg** refer to action encoder, sequence encoder, and commonsense knowledge transfer module, respectively
    - The encoders are replaced with simple aggregation (e.g., mean of vectors)

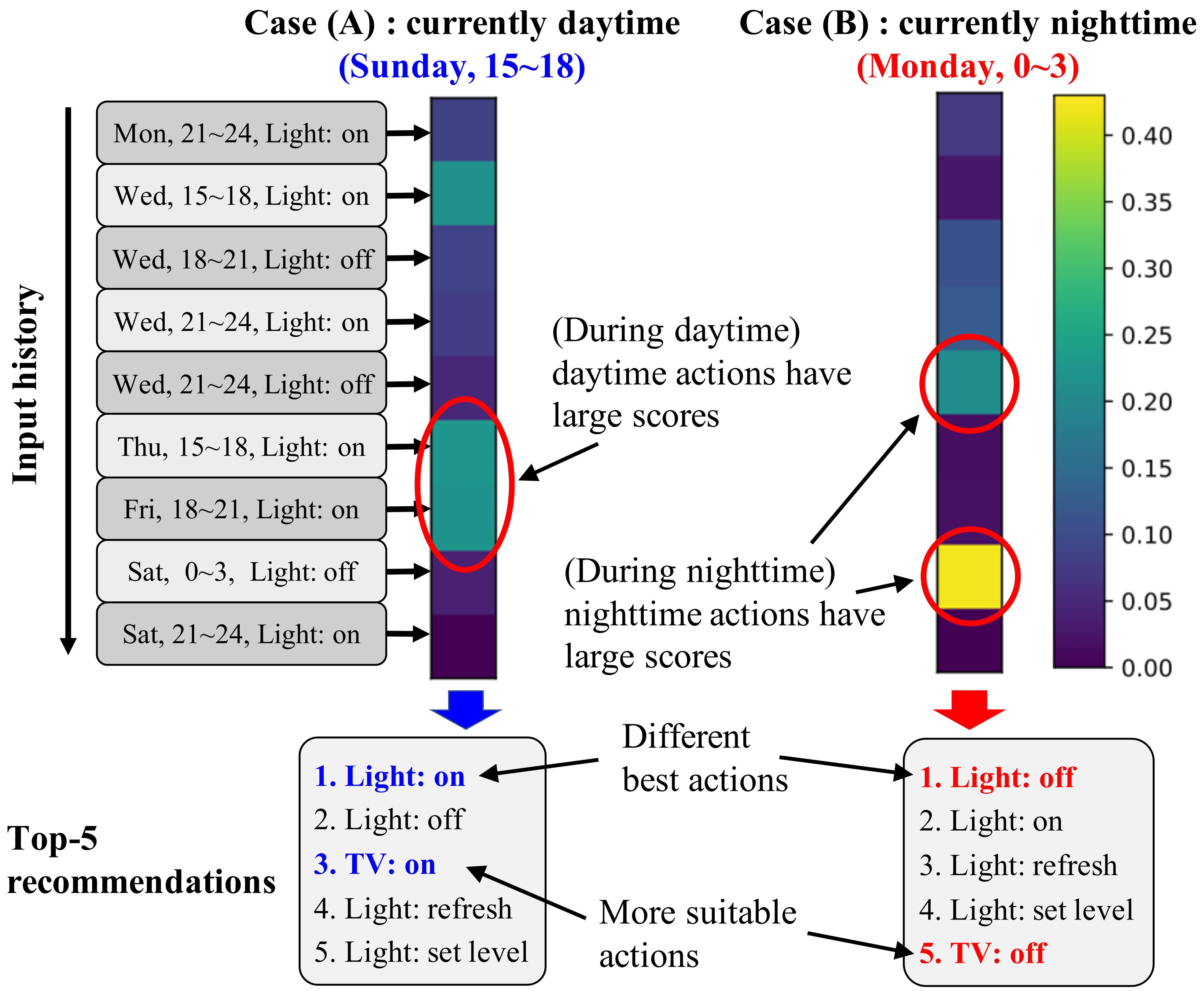
Model	Korea			USA		
	@1	@3	@5	@1	@3	@5
SmartSense-*Act*	0.5925	0.7256	0.7389	0.5802	0.7228	0.7350
SmartSense-*Seq*	0.6484	0.7631	0.7743	0.6194	0.7489	0.7592
SmartSense-*Reg*	0.6461	0.7608	0.7721	0.6189	0.7497	0.7600
SmartSense-*All*	0.5941	0.7265	0.7396	0.5752	0.7198	0.7321
SmartSense	**0.6515**	**0.7650**	**0.7760**	**0.6247**	**0.7541**	**0.7639**

Q3. Case Study

- **Q3.** How does *SmartSense* recommend device controls according to the current contexts?
- **A3.** *SmartSense* **dynamically recommends** device controls reflecting the current context

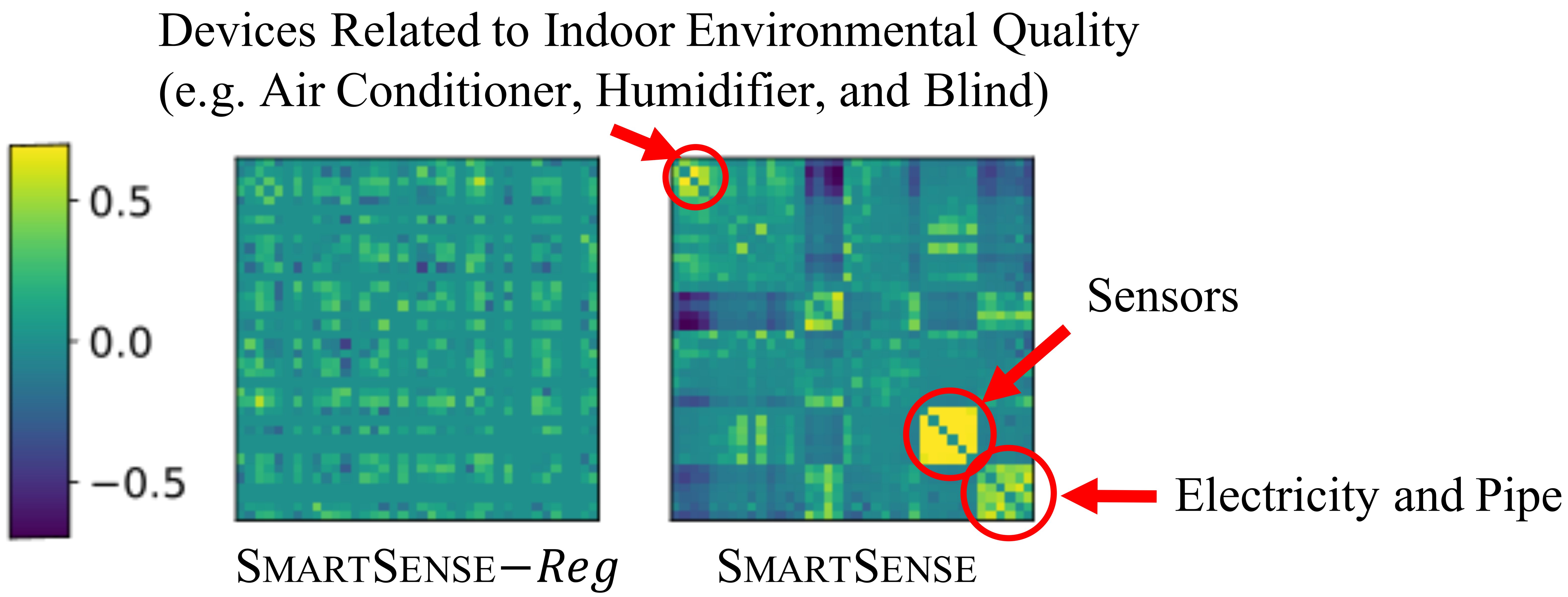
Focuses on past actions relevant to the current context

Recommends device controls relevant to the current contexts



Q4. Embedding Analysis

- **Q4.** Does *SmartSense* successfully learn proximity between devices?
- **A4.** *SmartSense* **successfully learns** the proximity between devices thanks to the commonsense knowledge transfer
  - Cosine similarity between embeddings of related devices is high



Outline

- Introduction
- Proposed Method
- Experiments
- **<u>Conclusion</u>**

Conclusion

- We propose **SmartSense** for action recommendation
- The main ideas are summarized as follows:
    - **Idea 1.** Self- and query-attention for an action
    - **Idea 2.** Self- and context-attention for a sequence
    - **Idea 3.** Knowledge transfer from common routines
- **SmartSense** achieves SOTA performance giving up to **9.8%** higher mAP@1 on real-world datasets

Thank you!

Hyunsik Jeon
Homepage: https://jeon185.github.io
Dataset: https://github.com/snudatalab/SmartSense

Ack: we thank SIGIR for the student travel grant supporting the conference registration