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

Hyunsik Jeon  
Seoul National University  
Seoul, South Korea  
jeon185@snu.ac.kr

Jongjin Kim  
Seoul National University  
Seoul, South Korea  
j2kim99@snu.ac.kr

Hoyoung Yoon  
Seoul National University  
Seoul, South Korea  
crazy8597@snu.ac.kr

Jaeri Lee  
Seoul National University  
Seoul, South Korea  
jlunits2@snu.ac.kr

U Kang  
Seoul National University  
Seoul, South Korea  
ukang@snu.ac.kr

ABSTRACT

How can we accurately recommend actions for users to control their devices at home? Action recommendation for smart home has attracted increasing attention due to its potential impact on the markets of Internet of Things (IoT). However, designing an effective action recommender system is challenging because it requires handling context correlations, considering both queried contexts and previous histories of users, and dealing with capricious intentions in history. In this work, we propose SMARTSENSE, an accurate action recommendation method for smart home. For individual action, SMARTSENSE summarizes its device control and temporal contexts in a self-attentive manner, to reflect the importance of the correlation between them. SMARTSENSE then summarizes sequences considering queried contexts in a query-attentive manner to extract the query-related patterns from the sequential actions. SMARTSENSE also transfers the commonsense knowledge from routine data to better handle intentions in action sequences. As a result, SMARTSENSE addresses all three main challenges of action recommendation for smart home, and achieves the state-of-the-art performance giving up to 9.8% higher mAP@1 than the best competitor.

CSC CONCEPTS

• Information systems → Recommender systems.

KEYWORDS

action recommendation, transformer, commonsense knowledge

ACM Reference Format:


Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM ’22, October 17–21, 2022, Atlanta, GA, USA  © 2022 Association for Computing Machinery. ACM ISBN 978-1-4503-9236-5/22/10. $15.00  https://doi.org/10.1145/3511808.3557226

1 INTRODUCTION

How can we accurately recommend actions for users to control their devices at home? Action recommendation for smart home has attracted increasing attention in data mining and machine learning communities due to its potential impact on the markets of virtual assistants [28] and the Internet of Things (IoT) [47]. The
Table 1: **SmartSense** addresses all three challenges of action recommendation for smart home, while competitors miss one or more of the challenges.

<table>
<thead>
<tr>
<th>Method</th>
<th>Challenges</th>
<th>Context correlations</th>
<th>Context-aware personalization</th>
<th>Capricious intentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMC [31], TransRec [7], Caser [36], SASRec [16], BERT4Rec [34]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLAR [29]</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA-RNN [25]</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SmartSense</strong>(proposed)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

The problem is to predict future actions (e.g., device controls) of users and recommend the next action. It is of great value, as an accurate recommendation keeps users safe when they forget a critical action (e.g., shutting off a gas valve), and reduces the hassles of users when performing a cumbersome action (e.g., arming an alarm).

Action recommendation for smart home entails three main challenges. First, the complicated correlations of multiple contexts such as day of the week, hour affect users’ device controls. For instance, laundry-related controls are highly affected by the correlation of day of the week and hour since people usually do laundry on weekend during the day. Second, both the sequential pattern of user actions and queried contextual information affects the future user actions. Figure 1a shows the frequency rankings of previously performed device controls in two different cases from SmartThings users, where SmartThings is a smart home system with 62 million active users worldwide. As shown in the figure, the next actions of people depend on the previous ones. For instance, people often perform various controls on air conditioners before setting the temperature, while they frequently control blinds and windows before opening the blinds. Figure 1b shows the frequency rankings of device controls depending on two different contexts. As shown in the figure, the device controls are affected by the current temporal contexts. For instance, people often control air conditioners before dawn since the proper temperature is an important factor for sleeping, while they frequently control blinds during the day because controlling the amount of sunlight entering the room is important for living. Figure 1 shows that both previous history and the current context are crucial when predicting users’ current device controls. Third, users’ action sequences contain capricious intentions. For example, if a user performs a series of actions: TV off, blind off, and light off, then we assume that the user wants to sleep. However, users do not always do their actions with only one intention. Figure 2 shows an example of an action sequence performed by a SmartThings user. As shown in the figure, a user controls a TV and a water purifier that are not related to laundry while doing laundry-related actions such as controlling a washer and a dryer. Capricious intentions lead to a degraded performance of recommendation models since they are trained to treat the actions in a sequence as highly related actions.

Table 1 compares our proposed **SmartSense** with other methods in various perspectives. **SmartSense** is the only method that handles all the three challenges of action recommendation for smart home: context correlations, context-aware personalization, and capricious intentions.

- **Method.** We propose **SmartSense**, an accurate method for action recommendation for smart home. **SmartSense** correlates a device control and its temporal contexts to capture their correlation. **SmartSense** then summarizes a session’s history while capturing highly related actions to the current contexts. Furthermore, **SmartSense** deals with capricious intentions by transferring commonsense knowledge from routine data.
- **Experiment.** Extensive experiments on real-world datasets show that **SmartSense** provides state-of-the-art performance with up to 9.8% higher mAP@1 in action recommendation for smart home compared to the best competitors (see Table 4).
- **Case study.** We show in case studies that **SmartSense** successfully recommends actions taking into account the context correlations and context-aware personalization (see Figures 5 and 6).
- **Real-world dataset.** We open-source the dataset from **SmartThings** which is a worldwide Internet of Things (IoT) platform with 62 million users. This is the first dataset for studying action recommendation in smart home. We provide sequential device control histories from four countries and device routine data from three territories (see Section 4.1). The datasets are available at [https://github.com/snudatalab/SmartSense](https://github.com/snudatalab/SmartSense).

## 2 RELATED WORKS

### 2.1 Sequential Recommendation

Given a sequence of user behaviors, sequential recommendation aims to recommend items to users by modeling sequential dependencies of the user behaviors [38]; it is different from traditional recommendation systems [14, 21, 27, 33], which model interactions in a static way to capture only the general preference of users and items. On early works, FPMC [31] combines first-order Markov chains (MCs) and Matrix Factorization [21, 33] to model both sequential behaviors and general interests of users. Besides the first-order MCs, higher-order MCs [7, 8] have been studied to consider previous items in a sequence. In recent years, deep neural networks such as
Figure 3: The overall architecture of SmartSense. Given session $u$'th sequence $S_u$ and the current temporal context $(c_{u,t}, t_{u,t})$, SmartSense predicts the probability $\hat{y}_{u,t}$ of session $u$'s device control at time $t$. SmartSense consists of context-aware action encoder (Section 3.4), context-attentive sequence encoder (Section 3.5), and commonsense knowledge transfer module (Section 3.6), where each encoder follows the structure of queried transformer encoder (Section 3.3). Context-aware action encoder interprets an individual action using the global query vector $q_c$. Context-attentive sequence encoder then summarizes the encoded sequence information using the current context. Commonsense knowledge transfer module regularizes device embeddings with routine data. SmartSense estimates the probability of the next device control using the summarized vector.

Recurrent Neural Networks (RNN) [4, 11], Convolutional Neural Networks (CNN) [22], and Transformers [37] have been adopted in sequential recommendation to address the complicated non-linear patterns in user behaviors. For instance, GRU4Rec [10] used Gated Recurrent Unit (GRU) [4] to model sequential patterns for session based recommendation. GRU4Rec+ [9] boosted the performance of GRU4Rec for top-k sequential recommendation by improving the loss function. Caser [36] employed CNN to capture sequential patterns from both time-axis and feature-axis of sequences. SASRec [16] and BERT4Rec [34] utilized unidirectional Transformers and bidirectional Transformers, respectively, to capture sequential patterns in sequences while considering the importance of correlations between behaviors. Advanced techniques such as memory networks [3, 12], translation learning [7], hierarchical attention mechanisms [43], graph neural networks [2, 26, 39], and contrastive learning [45] have been adopted to sequential recommendation. However, such sequential recommender systems fail to achieve high performance in action recommendation for smart home since they do not handle contextual information and capricious intentions in sequences which should not be neglected for an accurate action recommendation.

2.2 Context-aware Recommendation

Context-aware recommendation aims to capture user preferences by considering contextual attributes such as time and temperature as well as interaction information between users and items [1, 23]. Early works on context-aware recommendation [32, 41] adopted Factorization Machines (FMs) [30], which model all correlations between variables, to capture the significant correlations between contexts. Recent works leverage deep neural networks and advanced technologies such as attention mechanisms [40] to model higher-order interactions and generate more meaningful representations of contexts. In recent years, context-aware sequential recommendation, which considers sequential patterns as well as contextual information, has been studied. For instance, CA-RNN [25] employs context-specific transition matrix to represent contextual information, and adopts RNN to capture sequential pattern in a history. Analogously, SIAR [29] utilizes stacked RNN to consider temporal dynamics of both contexts and actions. However, such context-aware recommender systems are still not suitable for action recommendation for smart home since they do not simultaneously consider both queried contexts and histories, or unravel the capricious intentions in histories.

2.3 Transformer

Transformer [37] is a deep learning model adopting self-attention mechanism which differentially weights the significance of each part of the input data to handle sequential tasks. The main advantages of Transformer are that it effectively learns the correlations between the input variables and effectively represents the sequential pattern of data. Transformer has attracted increasing attention in natural language processing [5], computer vision [6], stock prediction [44], and recommender systems [16, 34] due to its superior performance. The self-attention mechanism of Transformer enables us to successfully capture significant correlations between a device control and contexts, and effectively represent sequential patterns resulting in accurate action recommendation for smart home.

3 PROPOSED METHOD

We define the problem of action recommendation for smart home and propose SmartSense, an accurate method to solve the defined problem.
3.1 Action Recommendation for Smart Home

Action recommendation aims to recommend actions to users to help them control their devices at home. We describe the problem definition of action recommendation for smart home as follows.

Problem 1 (Action recommendation for smart home). For each session $u$, we are given a sequence of history $S_u = \{x_{u,1}, \ldots, x_{u,(t-1)}\}$, where $x_{u,i} = (d^{(1)}_{u,i}, q^{(2)}_{u,i}, c^{(2)}_{u,i}, d^{(2)}_{u,i})$ is a quadruplet of indices of $i$th device, device control, day of week, and hour, respectively. Given a previous sequence $S_u$, and the current temporal contexts $c^{(1)}_{u,t}$ and $c^{(2)}_{u,t}$, the goal is to accurately predict the current device $d^{(1)}_{u,t}$ and its control $d^{(2)}_{u,t}$.

Note that we aim to predict only the device control $d^{(2)}_{u,t}$, since $d^{(1)}_{u,t}$ also contains the information of the target device $u_{u,t}$. To address the problem, a method should carefully handle device controls and their temporal contexts in the previous sequence, and effectively reflect the current temporal context in the prediction. Sequential recommendation methods [7, 16, 31, 34, 36] utilize only sequences of device controls without considering any contextual information. Context-aware recommendation methods [25, 29] consider temporal contexts in the previous sequence, or deal with the current temporal context. However, they are still unsatisfactory to action recommendation for smart home since they do not consider capricious intentions in the previous sequence.

3.2 Overview

We address the following challenges to achieve a high performance of smart home recommendation.

C1. Considering the correlations of contexts. The complicated correlations between a device control and various contexts such as day of week and hour affect a user’s future device control. How can we find meaningful correlations in the device control and the various contexts?

C2. Considering both history and the current context. Both a user’s past actions and the current context are crucial for predicting the user’s current action. How can we personalize the prediction while considering the current contexts?

C3. Handling capricious intentions. Capricious intentions make us difficult to learn distant representations for actions with different intentions, which leads to a degraded performance. How can we learn representations of actions such that similar intentions are close to each other and different intentions are distant?

To address the aforementioned challenges, we propose SmartSense with the following main ideas.

I1. Context-aware action encoder (Sections 3.3 and 3.4). We encode a device control and its temporal contexts in a self-attentive manner to capture significant correlations between them.

I2. Context-attentive sequence encoder (Sections 3.3 and 3.5). We encode a sequence of actions in a self-attentive way to capture the correlations between the actions. We then summarize the sequence by a query-attention mechanism to consider the current context in personalization.

I3. Commonsense knowledge transfer (Section 3.6). We transfer commonsense knowledge from routine data. Each routine is intentionally defined by a user, as an explicit sequence of actions. As a result, the knowledge transfer from the routine data enables us to learn more meaningful representations of actions.

Figure 3 shows the overall structure of SmartSense, which consists of context-aware action encoder, context-attentive sequence encoder, and common sense knowledge transfer module. The context-aware action encoder encodes an action of each time step, and then the sequence encoder predicts the current action of the session based on encoded previous actions and the current context. The commonsense knowledge transfer module regularizes device embeddings to capture hidden relationships between them.

3.3 Queried Transformer Encoder

The aim of action recommendation for smart home is to predict a current device control given a previous history and a current context. To achieve the goal, we divide the task into two subtasks: 1) encoding individual action, and 2) encoding the sequential history with the current context. The two subtasks require the two common functionalities as follows. First, we need to consider the correlations between given variables since multiple variables are intricately related. Second, we need to consider the significance of each variable because the importance of each variable is different. Then, how can we design a model to embody the two functionalities? We propose a queried transformer encoder (QTE), which is used for both two subtasks: action encoding and sequence encoding. Our main ideas of QTE are 1) correlating given variables in a self-attentive manner, and 2) capturing significant variables for a query by a query-attentive mechanism. QTE is defined as follows:

$$h = f(X, q),$$

where $h \in \mathbb{R}^d$ is the summarized vector, $f(\cdot)$ is QTE, $X \in \mathbb{R}^{k \times d}$ is the input matrix, $k$ is the number of input vectors, $d$ is the dimensionality of the input vectors, $q \in \mathbb{R}^d$ is the query vector, and $d'$ is the dimensionality of the query vector. $x_k \in \mathbb{R}^d$ denotes the $k$th row of $X$ representing the $k$th input vector. Figure 4 depicts the...
structure of QTE. QTE consists of self-attention module and query-attention module which correspond to functionalities of correlating given variables and capturing significant variables, respectively. We describe the self-attention module and the query-attention module in detail.

**Self-attention module.** The goal of self-attention module is to correlate given variables. We employ transformer encoder [37] for the self-attention module since it represents all pair-wise correlations between the given variables by learning different query, key, and value weight matrices for each variable. Given an input matrix \( X \), we obtain query, key, and value matrices as follows:

\[
Q = XW_Q, \quad K = XW_K, \quad V = XW_V,
\]

where \( Q, K, V \in \mathbb{R}^{k \times d} \) are query, key, and value matrices, respectively; \( W_Q, W_K, W_V \in \mathbb{R}^{d \times d} \) are learnable weight matrices for query, key, and value, respectively. We compute the transformed matrix as follows:

\[
\bar{X} = AV \quad \text{where} \quad A = \text{softmax} \left( \frac{QK^\top}{\sqrt{d}} \right).
\]

\( X \in \mathbb{R}^{k \times d} \) is the transformed matrix, \( A \in \mathbb{R}^{k \times k} \) is an attention score matrix for all pairs between the variables, and \( \text{softmax}(\cdot) \) indicates the row-wise softmax function. We then adopt a position-wise feed forward network (FNN) and residual connections as follows, to impose nonlinearity in the transformation and enable it to learn an identity function if needed:

\[
H = \text{Trans}(X) = X + \bar{X} + \text{FNN}(X + \bar{X}),
\]

where \( H \in \mathbb{R}^{k \times d} \) is the hidden representation matrix of the input variables, \( \text{Trans}(\cdot) \) is the transformer, \( \text{FNN}(\cdot) \) is a 2-layered position-wise feed forward network with the structure \( \mathbb{R}^d \rightarrow \mathbb{R}^{d'} \rightarrow \mathbb{R}^d \). We adopt multi-head attention in \( Q, K, \) and \( V \) because the multi-head attention shows better performance than a single-head attention in various works such as recommender systems [34], natural language processing [5], and computer vision [17]. We also adopt dropout and layer normalization after the attention and FNN to improve the generalization performance. We stack the transformer layer (Equation 4) \( L \) times to represent the complicated relationships between the input variables.

**Query-attention module.** We have the hidden representation matrix \( H \in \mathbb{R}^{k \times d} \) and a query vector \( q \in \mathbb{R}^d \). We denote \( i \)’th row vector in \( H \) as \( h_i \in \mathbb{R}^d \), and each \( h_i \) corresponds to the hidden representation vector of \( x_i \). The goal of query-attention module is to summarize the hidden representation matrix \( H \) into a single vector \( h \in \mathbb{R}^d \) while capturing significant variables depending on the query vector \( q \). In other words, we need to give more weight to information relevant to the query since the input variables do not equally contribute to the result. We propose a query-attention module as follows:

\[
h = \text{QueryAtt}(H, q) = \sum_{i=1}^{k} \alpha_i h_i,
\]

\[
\alpha_i = \frac{\exp(\beta_i)}{\sum_{j=1}^{k} \exp(\beta_j)}, \quad \beta_i = q^\top \tanh(W^H h_i + b^H),
\]

\( h \in \mathbb{R}^d \) is the summarized vector, \( H \in \mathbb{R}^{k \times d} \) is the hidden representation matrix, \( q \in \mathbb{R}^d \) is the query vector, \( h_i \in \mathbb{R}^d \) is \( i \)’th row vector of \( H \), \( \alpha_i, \beta_i \in \mathbb{R} \) are a normalized and an unnormalized scores for \( h_i \), respectively; \( W^H \in \mathbb{R}^{d \times d} \) and \( b^H \in \mathbb{R}^d \) are learnable parameters, and \( \tanh(\cdot) \) is the hyperbolic-tangent function. We first apply a linear projection using the trainable parameters \( W^H \) and \( b^H \) to \( h_i \), to transform it into the space of the query vector \( q \). The hyperbolic tangent function makes the transformed vector give element-wise weights to the query vector \( q \). Finally, we perform the weighted sum of the hidden vectors while considering the importance of each hidden vector \( h_i \). As a result, we obtain the summarized vector \( h \) while considering the correlations between the input vectors of \( X \) using the self-attention module, and capturing the significant input vectors depending on the query vector \( q \) using the query-attention module.

### 3.4 Context-aware Action Encoder

The objective of the context-aware action encoder is to encode the information of an individual action. To effectively encode each action, the encoder requires two functionalities as follows. First, it is necessary to correlate a device control and its temporal contexts. For instance, assume one opens a blind on Monday morning. We need to correlate opening the blind with morning since the correlation is more important than each of them. Second, it is required to give more weight to the significant information in an action. For example, suppose a user turns on an air conditioner on Monday night. In the action, turning on the air conditioner is more important than the other information because the user is likely to turn off the air conditioner in the future. Then, how can we consider the two functionalities when encoding an action?

For \( i \)’th action in session \( u \), we have a quadruplet of indices \((d^{(1)}_{u,i}, d^{(2)}_{u,i}, z^{(1)}_{u,i}, z^{(2)}_{u,i})\), where each of them are indices of device, device control, day of week, and hour, respectively. We firstly gain each embedding vectors \( e^{(1)}_{u,i}, e^{(2)}_{u,i}, z^{(1)}_{u,i}, z^{(2)}_{u,i} \in \mathbb{R}^d \) for the given indices which correspond to the embedding vectors of \( d^{(1)}_{u,i}, d^{(2)}_{u,i}, z^{(1)}_{u,i}, z^{(2)}_{u,i} \), and \( c_{u,i} \) respectively. To embody the aforementioned functionalities, we propose to employ the queried transformer encoder (QTE) for the context-aware action encoder as follows:

\[
h_{u,i} = f_c(X_{u,i}, q_c),
\]

where \( h_{u,i} \in \mathbb{R}^d \) is the hidden representation vector of \( i \)’th action in session \( u \). \( f_c(\cdot) \) is the context-aware action encoder, \( X_{u,i} \in \mathbb{R}^{4 \times d} = \{e^{(1)}_{u,i}, e^{(2)}_{u,i}, z^{(1)}_{u,i}, z^{(2)}_{u,i}\}^\top \) is a matrix of stacked vectors, and \( q_c \in \mathbb{R}^d \) is a trainable global query vector. The context-aware action encoder \( f_c \) uses the structure of QTE. We use the same query vector \( q_c \) for all actions in the all sessions because we aim to learn global representations for the significance of device controls and temporal contexts. As a result, for \( i \)’th action of session \( u \), we obtain the encoded vector \( h_{u,i} \) which considers the correlations between the device control and the temporal contexts, and gives more weight to the significant information.

### 3.5 Context-attentive Sequence Encoder

The objective of the context-attentive sequence encoder is to encode a sequence while considering the sequential patterns and the current context. It is important to consider the correlations
where wants to go out. This example shows that the meaning of sequence variable, which leads us to learn sequential patterns. We define in Figure 2, histories of actions often contain capricious intentions.

3.6 Commonsense Knowledge Transfer

For each session \( u \), we have a set of vectors \( \{h_{u,1}, ..., h_{u,(t-1)}\} \), where \( h_{u,t} \in \mathbb{R}^d \) is the hidden representation vector of \( t \)th action in the session; note that \( h_{u,1}, ..., h_{u,(t-1)} \) are from the context-aware action encoder. To encode the session \( u \), we propose to utilize the queried transformer encoder (QTE) for the context-attentive sequence encoder as follows:

\[
s_{u,t} = f_e(H_u + P, \text{concat}(s_{u,t}^{(1)}, s_{u,t}^{(2)})),
\]

where \( s_{u,t} \in \mathbb{R}^d \) is the encoded vector for the current time \( t \), \( f_e(\cdot) \) is the context-attentive sequence encoder, \( H_u \in \mathbb{R}^{(t-1) \times d} = \{h_{u,1}, ..., h_{u,(t-1)}\} \) is a matrix of stacked vectors, \( P \in \mathbb{R}^{(t-1) \times d} \) is a positional embedding matrix, \( \text{concat}(\cdot) \) is the concatenation, and \( s_{u,t}^{(1)}, s_{u,t}^{(2)} \in \mathbb{R}^d \) correspond to the embedding vectors of the current temporal contexts \( s_{u,t}^{(1)}, s_{u,t}^{(2)} \) where \( c_{u,t}^{(1)} \) and \( c_{u,t}^{(2)} \) are day of week and hour at the current time \( t \), respectively. The context-attentive sequence encoder \( f_e \) uses the structure of QTE. The positional embedding matrix \( P \) allows us to identify the position of the input variable, which leads us to learn sequential patterns. We define the \( P \) as a trainable matrix as in previous works [34, 37] for better generalization performance.

After obtaining \( s_{u,t} \), we compute the probabilities of device controls as follows:

\[
\hat{y}_{u,t} = \text{softmax}(E \cdot s_{u,t}),
\]

where \( \hat{y}_{u,t} \in \mathbb{R}^d \) is the predicted probabilities of device controls at the current time \( t \) for user \( u \), \( \text{softmax}(\cdot) \) is the softmax function, \( E \in \mathbb{R}^{N_d \times d} \) is the matrix of device controls for the prediction, and \( N_d \) is the number of device controls.

### 3.7 Objective Function

We train \textsc{SmartSense} to minimize the cross-entropy loss and the regularization loss as follows:

\[
\mathcal{L}(X, Y) = -\frac{1}{n} \sum_u \sum_i y_{u,t}(i) \log \hat{y}_{u,t}(i) + \mathcal{L}_{reg},
\]

where \( X \in \mathbb{R}^{n \times d} \) is an input tensor, and \( Y \in \mathbb{R}^{n \times N_d} \) is a matrix of the ground-truth label; \( n \) and \( i \) are the numbers of sessions and time steps, respectively, and \( N_d \) is the number of device controls. \( y_{u,t} \in \mathbb{R}^d \) is the one-hot vector of the ground-truth label for session \( u \), and \( y_{u,t}(i) \in \mathbb{R} \) is the \( i \)th element in the vector. Similarly, \( \hat{y}_{u,t} \in \mathbb{R}^d \) is the predicted probability for session \( u \).

### 4 EXPERIMENTS

We perform experiments to answer the following questions:

Q1. \textbf{Accuracy}. Does \textsc{SmartSense} show higher accuracy in action recommendation for smart home than baselines?

Q2. \textbf{Ablation study}. How do main ideas of \textsc{SmartSense} help improving the performance of recommendation?
Q3. Case study. Does SmartSense capture meaningful correlations from each action? How do the recommendation results of SmartSense change according to given contexts?

Q4. Embedding space analysis. Does SmartSense successfully learn useful embedding vectors of contexts and devices?

4.1 Experimental Setup

We introduce our experimental setup: datasets, baselines, evaluation metrics, experimental process, and the hyperparameters.

Datasets. We use real-world SmartThings datasets collected by Samsung from various regions to evaluate the performance. There are four log datasets and three routine datasets. Log datasets contain histories of device controls executed by users of Bixby. These are used to generate sequential instances for general sequential recommendation. As explained in Section 3.6, routine datasets are collections of frequently used device patterns triggered by various contextual backgrounds. Those instances are submitted by users of SmartThings. These datasets are used to gain commonsense knowledge in the corresponding log datasets.

Tables 2 and 3 show the statistics of log datasets and routine datasets, respectively. In the routine datasets, AP and NA correspond to KR and US in log datasets, respectively. EU corresponds to SP and FR in log datasets.

Baselines. We compare SmartSense with existing methods of sequential recommendation and context-aware recommendation.

- POP recommends device controls based on their popularity.
- FMC [31] uses two item embeddings to build an item transition matrix which predicts next device control.
- TransRec [7] represents relationships between consecutive items as a vector operation to perform sequential recommendation.
- SASRec [16] uses time-growing directional transformer encoder to consider sequential patterns of user actions.
- BERT4Rec [34] utilizes BERT architecture [5] into sequential recommendation.
- SIAR [29] stacks RNN layers of contexts and actions to consider temporal dynamics of both contexts and actions.
- CA-RNN [25] uses context-specific transition matrix in RNN cell to consider context-dependent features in a sequential recommendation.

Note that these baselines are not aware of commonsense knowledge of action recommendation for smart home. Thus, baseline methods use only log datasets during evaluation.

Evaluation metrics. We evaluate the performance of competing models with two evaluation metrics: hit ratio (HR@k) and mean average precision (mAP@k). Both metrics compare the recommendation list of the model with the ground truth value. For the k size of recommendation list, HR@k treats every item in them equally important, while mAP@k treats higher-ranked items more importantly. We vary k in {1, 3, 5} for all datasets.

Experimental process. For each session in a log dataset, we create sequential instances with a window of length 10. The first nine events of the window are input of the sequential recommendation model. Each input event is a pair of a temporal context and a device control information. Temporal context of an event is a combination of a day of week and hour based on the event's timestamp. The hour in a context is one of the 8 time ranges of 3 hours length: 0-3, 3-6, 6-9, 9-12, 12-15, 15-18, 18-21, and 21-24. Device control information is composed of a device and its control. The device control of the last event is the ground truth for the window. We randomly split sequential instances into a training, a validation, and a test sets in 7:1:2 ratio. We train the model until the validation accuracy is maximized, and report the test accuracy.

Hyperparameters. All models are trained with Adam optimizer [18] with learning rate 0.001 and l2 regularization coefficient 0.00001. For fair comparison, we set the size of embedding vectors to 50 and the size of mini batch 1024 for all models. For both the context-aware action encoder and the context-attentive sequence encoder, we set the numbers of both transformer layers and the heads as 2. We set the dropout ratio to 0.1, and the size of negative samples m = 5 for the commonsense knowledge transfer module.

4.2 Recommendation Accuracy (Q1)

We measure the accuracy of SmartSense and baselines in four real world datasets. Table 4 shows the result in terms of mAP. Note that SmartSense consistently outperforms baselines in all cases. Table 5 shows the result in terms of HR@k. Note that SmartSense shows the best performance in most cases, for various k. These results show that SmartSense is an accurate method for action recommendation for smart home.

4.3 Ablation Study (Q2)

We verify the effectiveness of our three main ideas, context-aware action encoder, context-attentive sequence encoder, and commonsense knowledge transfer. We compare SmartSense with SmartSense-Act, SmartSense-Seq, SmartSense-Reg, and SmartSense-All. SmartSense-Act is a SmartSense without a context-aware action encoder. This model encodes each pair of a temporal context and a device control as the mean value of corresponding embeddings. Thus, the model is not aware of correlations of temporal context and device control. SmartSense-Reg is a SmartSense without a context-attentive sequence encoder. The sequence encoder of this model summarizes the output of transformer layers as a mean value instead of a weighted sum. This is equivalent to the context-attentive sequence encoder where the given query is a zero vector instead of the current temporal context embedding. In this way, the model is not aware of the current temporal context during prediction. SmartSense-Reg is a SmartSense without the regularization from commonsense knowledge transfer module. This model cannot access information gathered from routine datasets. SmartSense-All is a SmartSense without any of the main ideas.

Table 6 shows that SmartSense is the most accurate model among competitors, while SmartSense-All shows the worst accuracy. In summary, all three main ideas are helpful to improve performance of action recommendation for smart home.

4.4 Case Study (Q3)

We analyze cases to observe how SmartSense deals with context information.

First, we observe the attention scores of the context-aware action encoder to find out how SmartSense reacts to the given pair of
Table 4: MAP@k of SmartSense and competitors for smart-home recommendation on four real-world datasets. SmartSense outperforms all competitors in all cases, demonstrating its superiority of action recommendation for smart home. Bold and underlined values indicate the best and the second-best accuracies, respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Korea @1</th>
<th>Korea @3</th>
<th>Korea @5</th>
<th>USA @1</th>
<th>USA @3</th>
<th>USA @5</th>
<th>Spain @1</th>
<th>Spain @3</th>
<th>Spain @5</th>
<th>France @1</th>
<th>France @3</th>
<th>France @5</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP</td>
<td>0.3416</td>
<td>0.4918</td>
<td>0.5045</td>
<td>0.1886</td>
<td>0.3146</td>
<td>0.3737</td>
<td>0.4973</td>
<td>0.6337</td>
<td>0.6455</td>
<td>0.4949</td>
<td>0.5955</td>
<td>0.6114</td>
</tr>
<tr>
<td>FMC [31]</td>
<td>0.5075</td>
<td>0.6391</td>
<td>0.6569</td>
<td>0.4581</td>
<td>0.6082</td>
<td>0.6270</td>
<td>0.4102</td>
<td>0.5935</td>
<td>0.6015</td>
<td>0.4427</td>
<td>0.6330</td>
<td>0.6477</td>
</tr>
<tr>
<td>TransRec [7]</td>
<td>0.3854</td>
<td>0.5637</td>
<td>0.5830</td>
<td>0.3351</td>
<td>0.5240</td>
<td>0.5426</td>
<td>0.3819</td>
<td>0.6149</td>
<td>0.6209</td>
<td>0.4255</td>
<td>0.6238</td>
<td>0.6393</td>
</tr>
<tr>
<td>Caser [36]</td>
<td>0.5676</td>
<td>0.7064</td>
<td>0.7213</td>
<td>0.5535</td>
<td>0.7051</td>
<td>0.7177</td>
<td>0.7906</td>
<td>0.8548</td>
<td>0.8616</td>
<td>0.7706</td>
<td>0.8249</td>
<td>0.8295</td>
</tr>
<tr>
<td>SASRec [16]</td>
<td>0.5763</td>
<td>0.7064</td>
<td>0.7212</td>
<td>0.5657</td>
<td>0.7098</td>
<td>0.7282</td>
<td>0.7929</td>
<td>0.8570</td>
<td>0.8630</td>
<td>0.7740</td>
<td>0.8286</td>
<td>0.8389</td>
</tr>
<tr>
<td>BERT4Rec [34]</td>
<td>0.5927</td>
<td>0.7253</td>
<td>0.7393</td>
<td>0.5630</td>
<td>0.7121</td>
<td>0.7254</td>
<td>0.7887</td>
<td>0.8610</td>
<td>0.8662</td>
<td>0.7776</td>
<td>0.8475</td>
<td>0.8507</td>
</tr>
<tr>
<td>CA-RNN [25]</td>
<td>0.5703</td>
<td>0.6958</td>
<td>0.7095</td>
<td>0.4860</td>
<td>0.6315</td>
<td>0.6459</td>
<td>0.6748</td>
<td>0.7253</td>
<td>0.7350</td>
<td>0.5141</td>
<td>0.5650</td>
<td>0.5767</td>
</tr>
<tr>
<td>SLAR [29]</td>
<td>0.5936</td>
<td>0.7248</td>
<td>0.7381</td>
<td>0.5718</td>
<td>0.7163</td>
<td>0.7288</td>
<td>0.7913</td>
<td>0.8560</td>
<td>0.8628</td>
<td>0.7706</td>
<td>0.8258</td>
<td>0.8311</td>
</tr>
</tbody>
</table>

**SmartSense (proposed)**: 0.6515 0.7650 0.7760 0.6247 0.7341 0.7639 0.8101 0.8707 0.8756 0.7944 0.8544 0.8578

Table 5: HR@k of SmartSense and competitors for smart-home recommendation on four real-world datasets. SmartSense outperforms all competitors in most cases, demonstrating its superiority of action recommendation for smart home. Bold and underlined values indicate the best and the second-best accuracies, respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Korea @1</th>
<th>Korea @3</th>
<th>Korea @5</th>
<th>USA @1</th>
<th>USA @3</th>
<th>USA @5</th>
<th>Spain @1</th>
<th>Spain @3</th>
<th>Spain @5</th>
<th>France @1</th>
<th>France @3</th>
<th>France @5</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP</td>
<td>0.3416</td>
<td>0.6527</td>
<td>0.7095</td>
<td>0.1886</td>
<td>0.4872</td>
<td>0.7493</td>
<td>0.4973</td>
<td>0.7916</td>
<td>0.8426</td>
<td>0.4949</td>
<td>0.7243</td>
<td>0.7955</td>
</tr>
<tr>
<td>FMC [31]</td>
<td>0.5075</td>
<td>0.7921</td>
<td>0.8683</td>
<td>0.4581</td>
<td>0.7994</td>
<td>0.8811</td>
<td>0.4102</td>
<td>0.7966</td>
<td>0.8226</td>
<td>0.4427</td>
<td>0.8529</td>
<td>0.9161</td>
</tr>
<tr>
<td>TransRec [7]</td>
<td>0.3854</td>
<td>0.7649</td>
<td>0.8478</td>
<td>0.3351</td>
<td>0.7596</td>
<td>0.8405</td>
<td>0.3819</td>
<td>0.8825</td>
<td>0.9085</td>
<td>0.4255</td>
<td>0.8586</td>
<td>0.9247</td>
</tr>
<tr>
<td>Caser [36]</td>
<td>0.5676</td>
<td>0.8711</td>
<td>0.9345</td>
<td>0.5535</td>
<td>0.8886</td>
<td>0.9429</td>
<td>0.7906</td>
<td>0.9295</td>
<td>0.9591</td>
<td>0.7706</td>
<td>0.8859</td>
<td>0.9073</td>
</tr>
<tr>
<td>SASRec [16]</td>
<td>0.5763</td>
<td>0.8603</td>
<td>0.9240</td>
<td>0.5657</td>
<td>0.8862</td>
<td>0.9420</td>
<td>0.7929</td>
<td>0.9320</td>
<td>0.9682</td>
<td>0.7740</td>
<td>0.8938</td>
<td>0.9377</td>
</tr>
<tr>
<td>BERT4Rec [34]</td>
<td>0.5927</td>
<td>0.8825</td>
<td>0.9424</td>
<td>0.5630</td>
<td>0.8932</td>
<td>0.9502</td>
<td>0.7887</td>
<td>0.9461</td>
<td>0.9691</td>
<td>0.7776</td>
<td>0.9303</td>
<td>0.9460</td>
</tr>
<tr>
<td>CA-RNN [25]</td>
<td>0.5703</td>
<td>0.8428</td>
<td>0.9020</td>
<td>0.4860</td>
<td>0.8906</td>
<td>0.8718</td>
<td>0.6748</td>
<td>0.7906</td>
<td>0.8324</td>
<td>0.5141</td>
<td>0.6294</td>
<td>0.6814</td>
</tr>
<tr>
<td>SLAR [29]</td>
<td>0.5936</td>
<td>0.8800</td>
<td>0.9369</td>
<td>0.5718</td>
<td>0.8918</td>
<td>0.9433</td>
<td>0.7913</td>
<td>0.9314</td>
<td>0.9607</td>
<td>0.7706</td>
<td>0.8893</td>
<td>0.9119</td>
</tr>
</tbody>
</table>

**SmartSense (proposed)**: 0.6515 0.8983 0.9454 0.6247 0.9079 0.9495 0.8101 0.9413 0.9623 0.7944 0.9232 0.9379

Table 6: Ablation study of SmartSense. We report the performances by MAP@k. Note that SmartSense without at least one of the three main ideas decreases the performance, while SmartSense with all the ideas shows the best performance.

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

SmartSense is a recommendation model that can be controlled when users are not paying attention to the device, so they can be controlled at any time. In summary, SmartSense successfully captures the important correlation from given inputs with a context-aware action encoder.

Second, we observe how the current temporal context affects recommendation results. Figure 6 shows top-5 recommendation results and attention scores of each action in a history depending on the current temporal context. The attention scores of the sixth and the seventh actions are high in case (A), while attention scores of
SmartSense

Second, we visualize the hour embeddings to see whether
SmartSense is aware of the current temporal context while prediction, so it flexibly generates an accurate recommendation for the given situation.

4.5 Embedding Space Analysis (Q4)

We observe the embeddings of devices and temporal contexts to analyze how they reflect the real world.

First, we observe the device embedding space to verify the impact of commonsense knowledge transfer in the representation learning. Figure 7 shows that the cosine similarities between embedding vectors of devices in SmartSense-Reg (SmartSense without the commonsense knowledge transfer regularization), and SmartSense. The standard deviation of cosine similarities in SmartSense is 0.21 while that of SmartSense-Reg is 0.11, which shows that device embedding vectors in SmartSense-Reg have more complex patterns as the regularizations are applied. Furthermore, embedding vectors of similar devices in SmartSense such as sensors get closer to each other in SmartSense compared to SmartSense-Reg. This shows that routine data are useful to find close relationships between devices due to its task oriented composition.

Second, we visualize the hour embeddings to see whether SmartSense successfully captures the characteristic of temporal contexts. Figure 8 shows that embeddings of close hours are more similar to each other compared to further hours. This shows SmartSense extracts the crucial contextual information from temporal contexts.

5 CONCLUSION

In this paper, we propose SmartSense, an accurate action recommendation method for smart home. To reflect the importance of correlations, SmartSense introduces context-aware action encoder which captures significant correlations between the device control and the temporal context. The context-attentive sequence encoder of SmartSense summarizes users' sequential pattern and queried contextual information to consider both personalization and the current context. Commonsense knowledge transfer in SmartSense enables the model to successfully consider user intentions. As a result, SmartSense shows the state-of-the-art accuracy giving up to 9.8% higher mAP@1 in action recommendation for smart home than the best competitor in extensive experiments. Through case studies, we also show that SmartSense successfully captures the correlations among actions and contexts.

ACKNOWLEDGMENTS

This work was supported by Samsung Electronics. The Institute of Engineering Research at Seoul National University provided research facilities for this work. The ICT at Seoul National University provides research facilities for this study. We thank Inchul Hwang, Sanghee Kim, and Hyunju Seo for their help and insightful discussion. U Kang is the corresponding author.
REFERENCES

[26] Chen Ma, Lihe Ma, Yangce Zhang, Jiaqing Sun, Xue Liu, and Mark Coates. 2020. Memory Augmented Graph Neural Networks for Sequential Recommendation. In AAAI. AAAI Press.
[34] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenzhu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In CIKM. ACM.