Session-aware Linear Item-Item Models for Session-based Recommendation

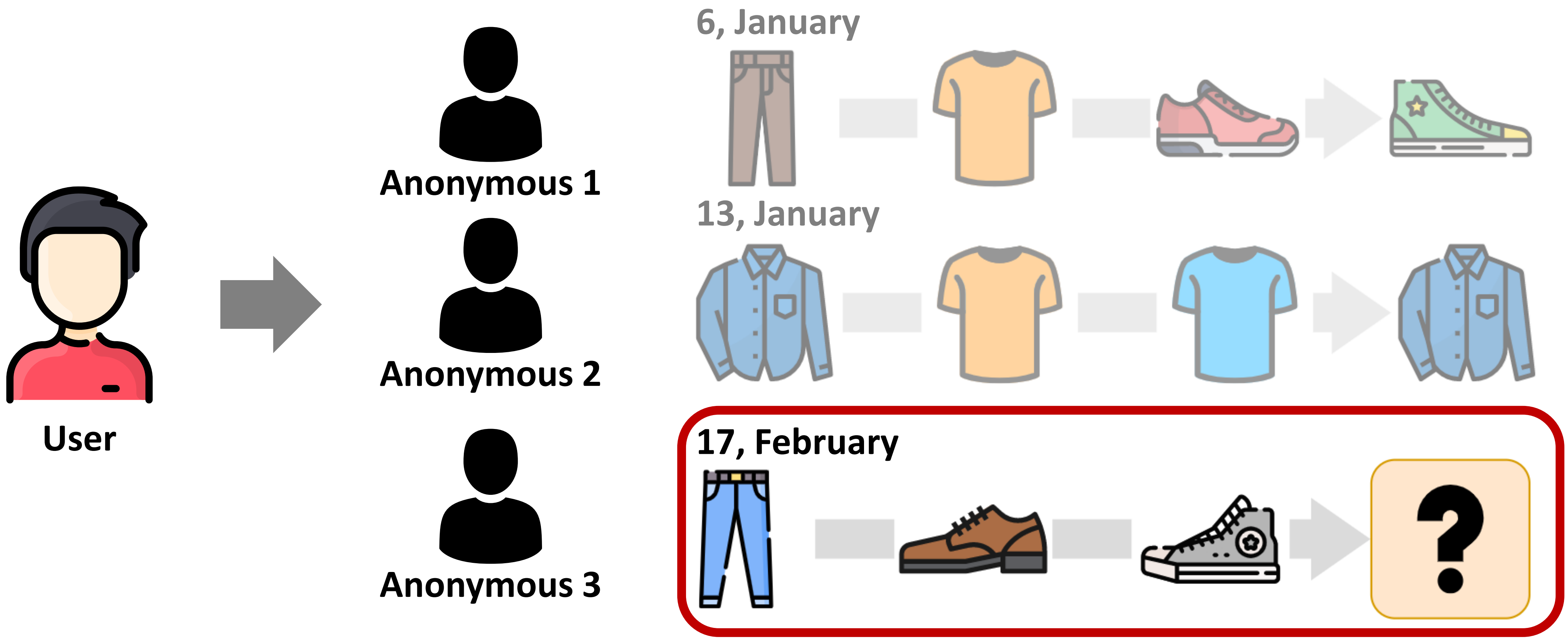
Minjin Choi[1], Jinhong Kim[1], Joonseok Lee[2,3] , Hyunjung Shim[4], Jongwuk Lee[1]

Sungkyunkwan University (SKKU)[1], Google Research[2]
Seoul National University[3], Yonsei University[4]

Motivation

Session-based Recommendation (SR)

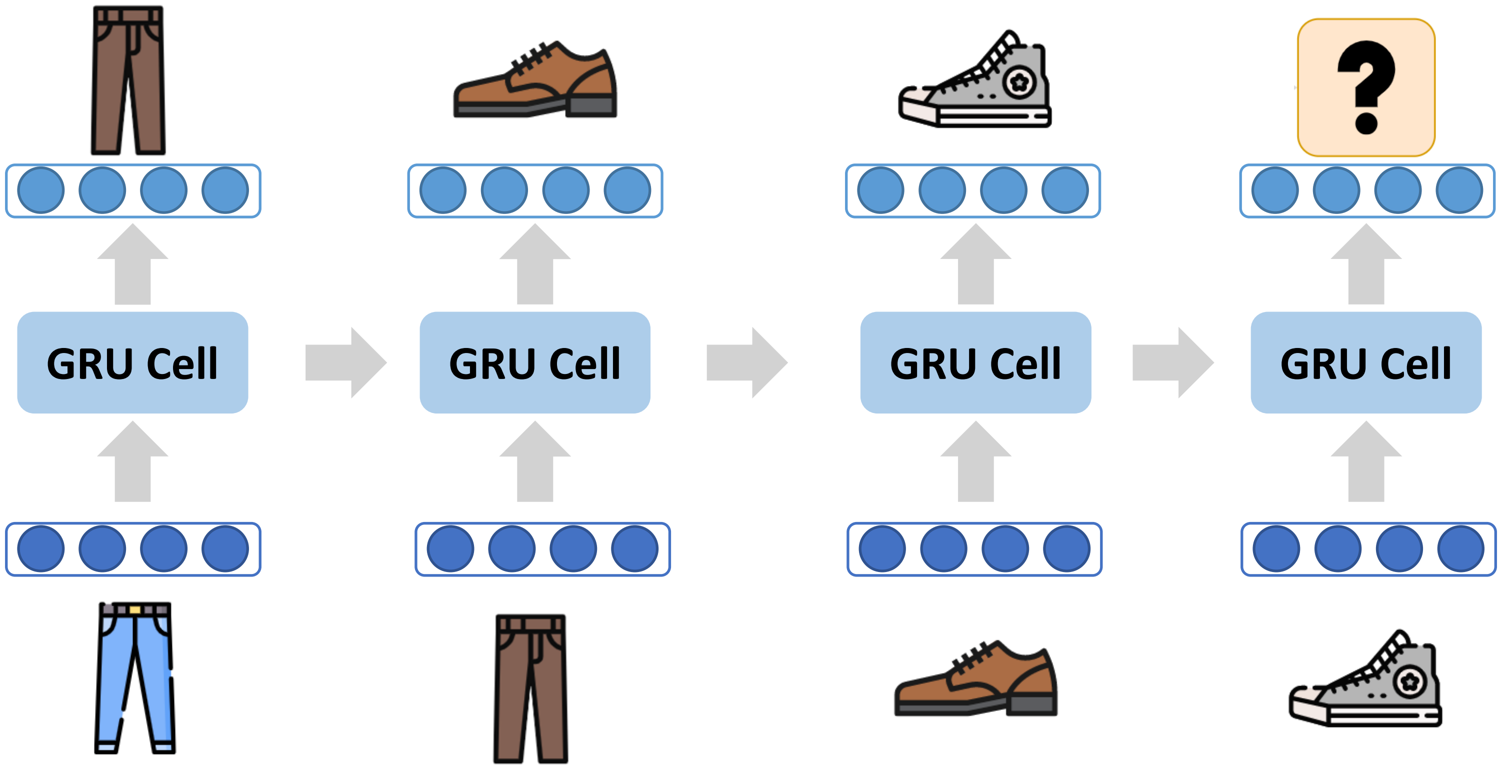
- **Predicting the next item(s) based on only the current session**
  - Usually, session histories are much shorter than user histories.



Limitation of Existing SR Models

➢ **DNN models show good performance but suffer scalability issues.**

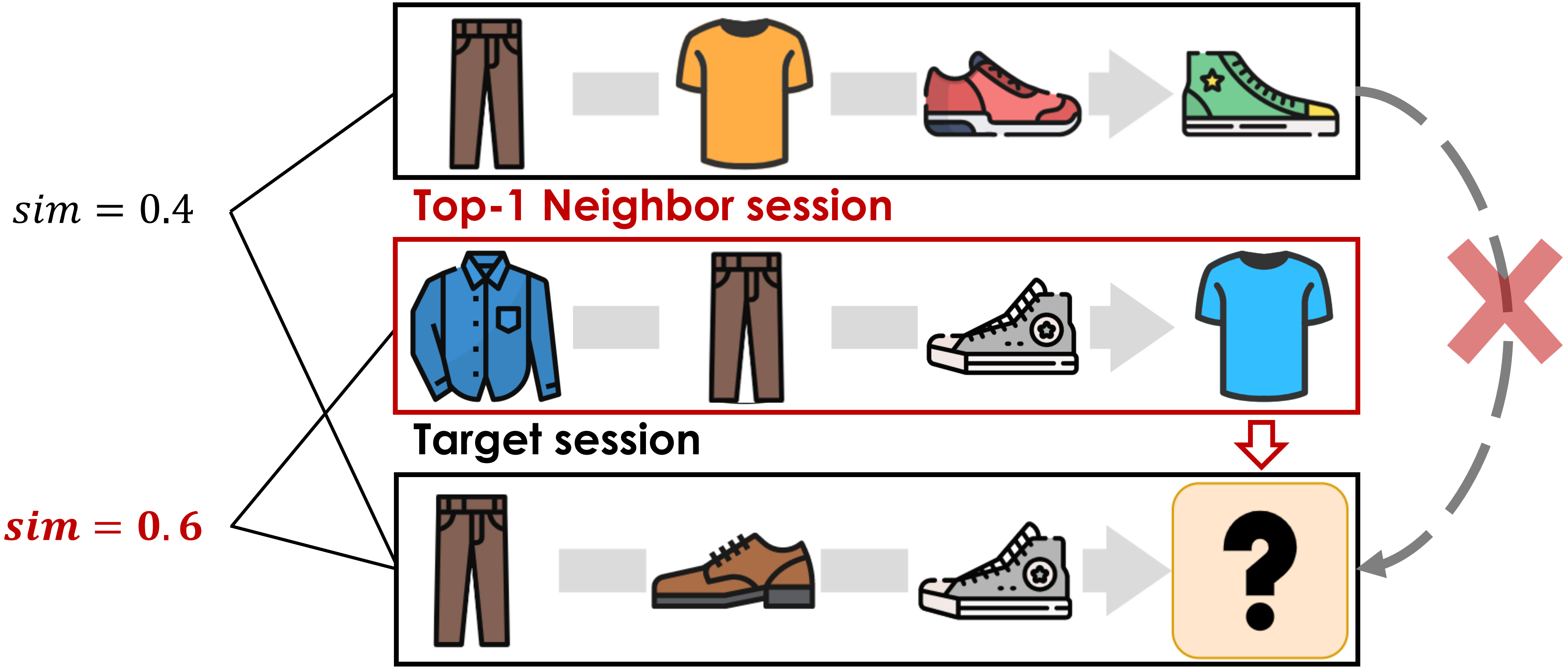
- RNNs and GNNs are mostly used for session-based recommendations.
- When the dataset is too large, the scalability issue arises.



Balázs Hidasi et. al., "Session-based Recommendations with Recurrent Neural Networks", ICLR 2016

Limitation of Existing SR Models

➢ Neighborhood-based models **are difficult to capture complex dependencies between items** in a session.



Diksha Garg et. al., "Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN", SIGIR 2019

Research Question

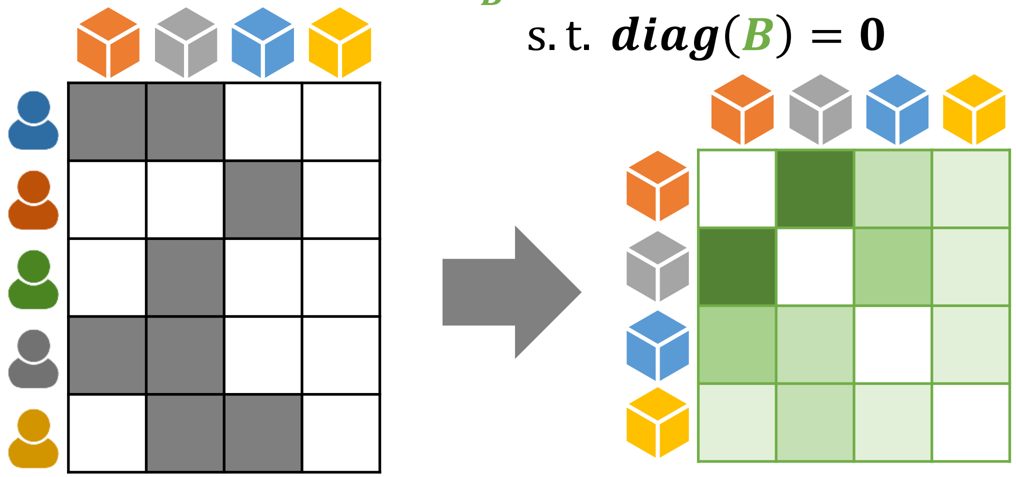
How to build the accurate and scalable session-based recommender model?

Our Key Contributions

- **We utilize linear models for session-based recommendations.**
  - Linear models show great performance in traditional recommendations.
  - However, simply applying them to session-based recommendations does not show an effective performance.

$$\underset{B}{\mathbf{argmin}} \|X - XB\|_F^2 + \lambda \|B\|_F^2$$

$$\text{s.t. } \boldsymbol{diag}(B) = 0$$



User-item matrix X — $X \in \mathbb{R}^{m \times n}$

Item-item model B — $B \in \mathbb{R}^{n \times n}$

m: # of users

n: # of items

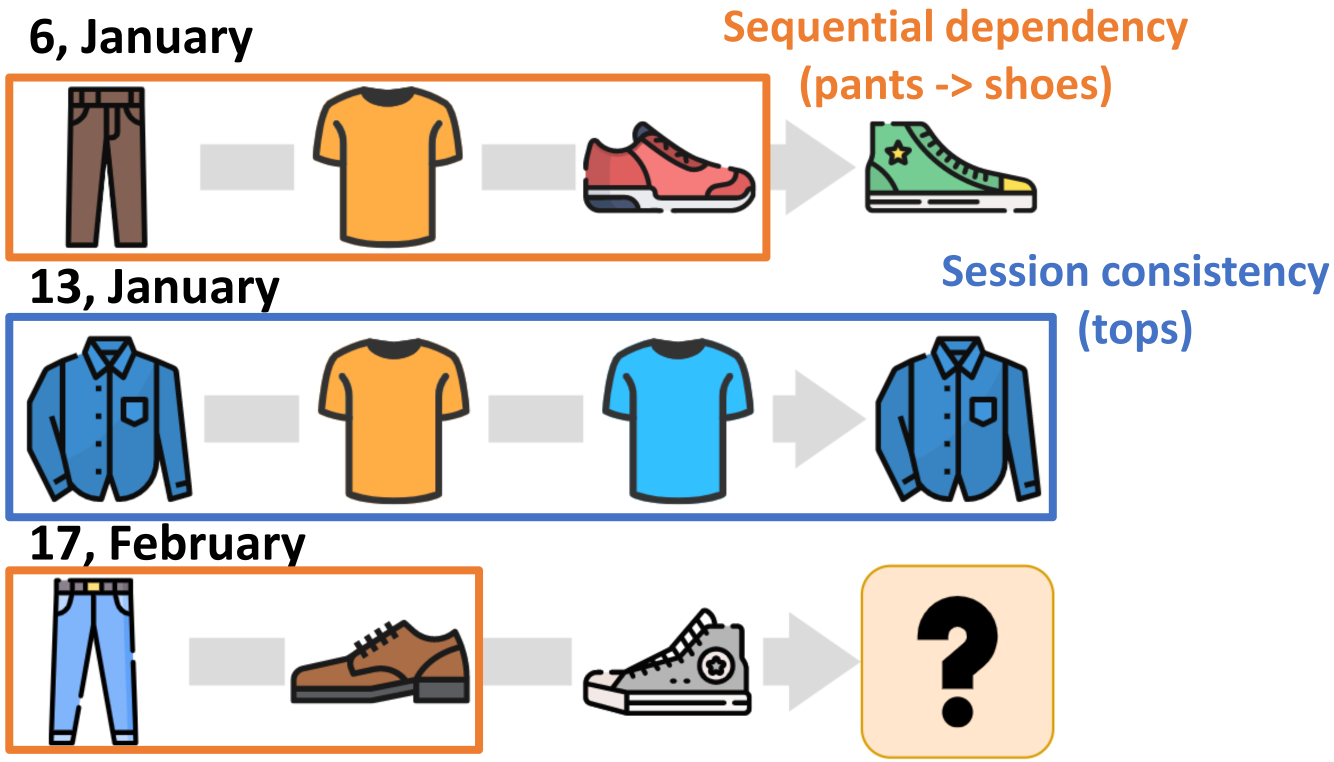
Dataset: YC-1/64		
Models	**HR@20**	**MRR@20**
GRU4Rec+	**0.6528**	**0.2752**
SKNN	0.6423	0.2522
SEASE^R	<u>0.5443</u>	<u>0.1963</u>

Worse than existing models!

Harald Steck, "Embarrassingly Shallow Autoencoders for Sparse Data", WWW 2019

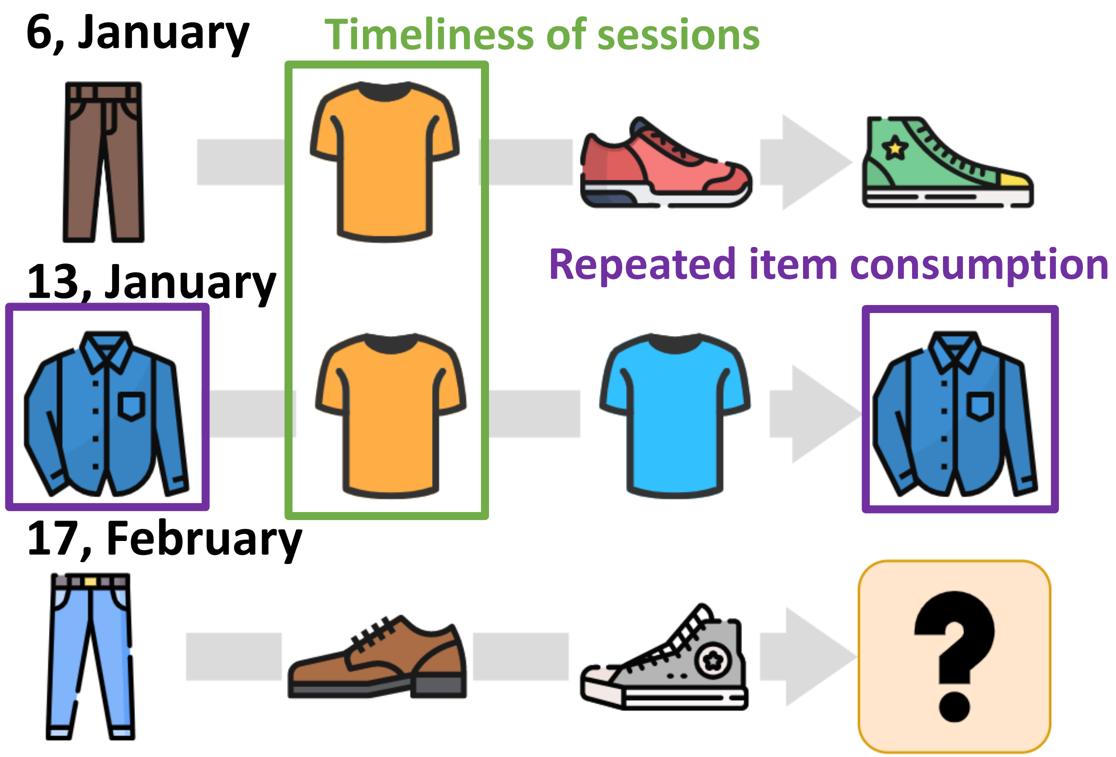
Our Key Contributions

- **We design linear models to utilize some unique characteristics of sessions.**
  - Some items tend to be consumed in a **specific order**. (dependency)
  - Items in a session are often **highly coherent**. (consistency)



Our Key Contributions

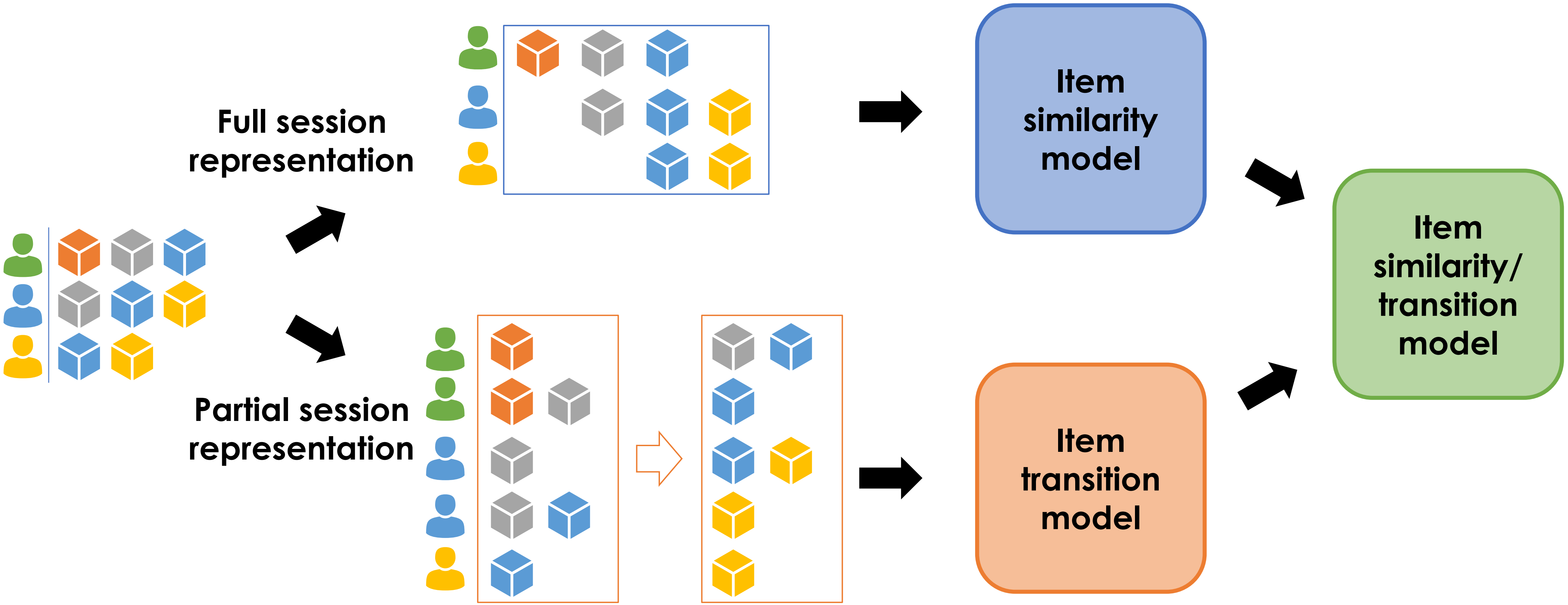
➢ **We design linear models to utilize some unique characteristics of sessions.**
- Sessions often reflect a **recent trend**. (timeliness)
- The user might **consume the same items** in a session. (repeated item)



Proposed Model

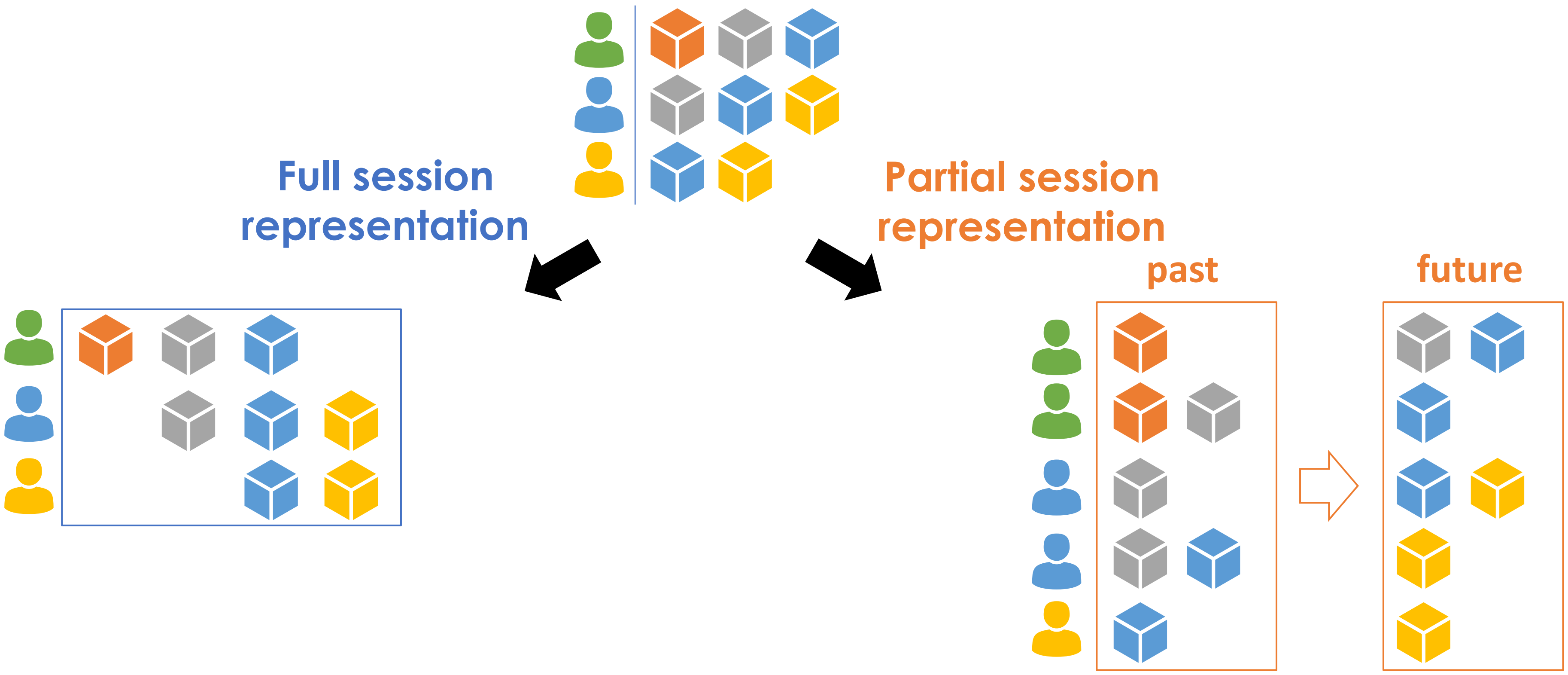
Overview of the Proposed Model

- We devise **two linear models** to learn item similarity/transition and **unify** them to fully capture the complex relationships of items.



Two Session Representations

- **We deal with two session representations for our linear models.**
  - (1) **A single vector** to capture **the correlations between items** while ignoring the order.
  - (2) **Past and future vectors** to represent sequential dependency of items.

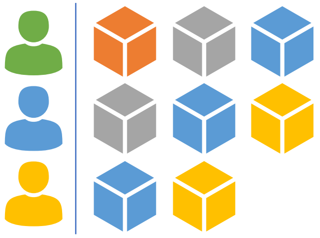


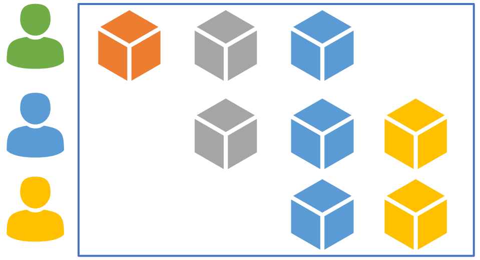
Item Similarity Model (SLIS)

- **We learn the linear model using full session representation, to represent the similarity between items.**
  - The entire session is represented as a single vector.
  - B_{ij} indicates the learned item similarity between i and j.
    - Note: the diagonal constraint is relaxed.

$$\underset{\boldsymbol{B}}{\mathbf{argmin}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{B}\|_F^2 + \boldsymbol{\lambda}\|\boldsymbol{B}\|_F^2$$

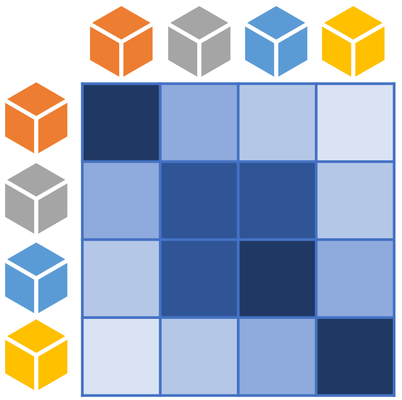
$$\text{s.t. } \boldsymbol{diag}(\boldsymbol{B}) \leq \boldsymbol{\xi}$$





Full session representation $\boldsymbol{X}$

$\boldsymbol{X} \in \mathbb{R}^{|\boldsymbol{S}| \times \boldsymbol{N}}$



Item similarity model $\boldsymbol{B}$

$\boldsymbol{B} \in \mathbb{R}^{\boldsymbol{N} \times \boldsymbol{N}}$

Dark cells mean high similarities.

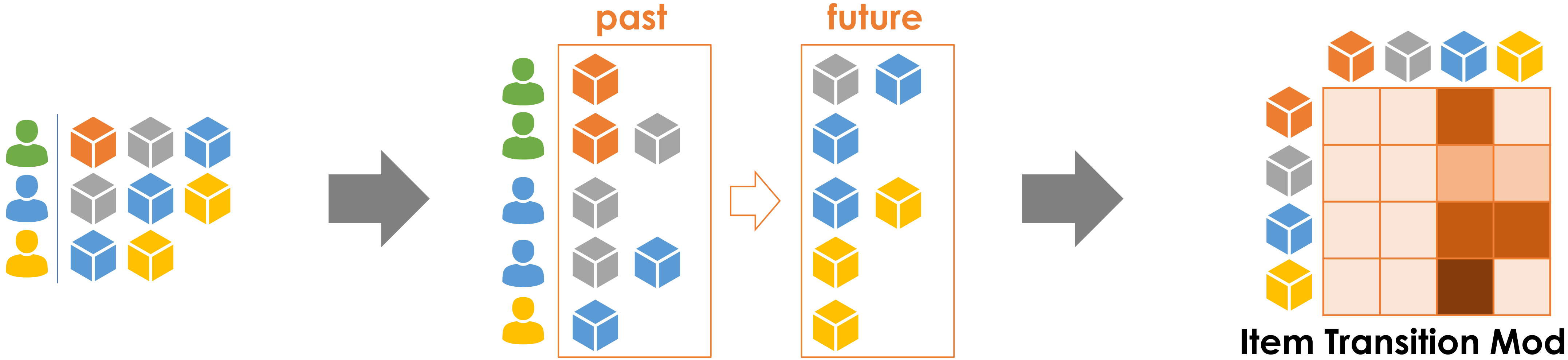
$|\boldsymbol{S}|$: # of sessions

$\boldsymbol{n}$: # of items

Item Transition Model (SLIT)

- **We learn the linear model using partial session representation that captures the sequential dependency across items.**
  - A session is divided into past vectors and future vectors.
  - B_{ij} indicates the item transitions from i to j.

$$\underset{B}{\operatorname{argmin}} \|T - SB\|_F^2 + \lambda \|B\|_F^2$$



Partial session representation $S \rightarrow T$

$S, T \in \mathbb{R}^{|S'| \times N}$

Item Transition Model B

$B \in \mathbb{R}^{N \times N}$

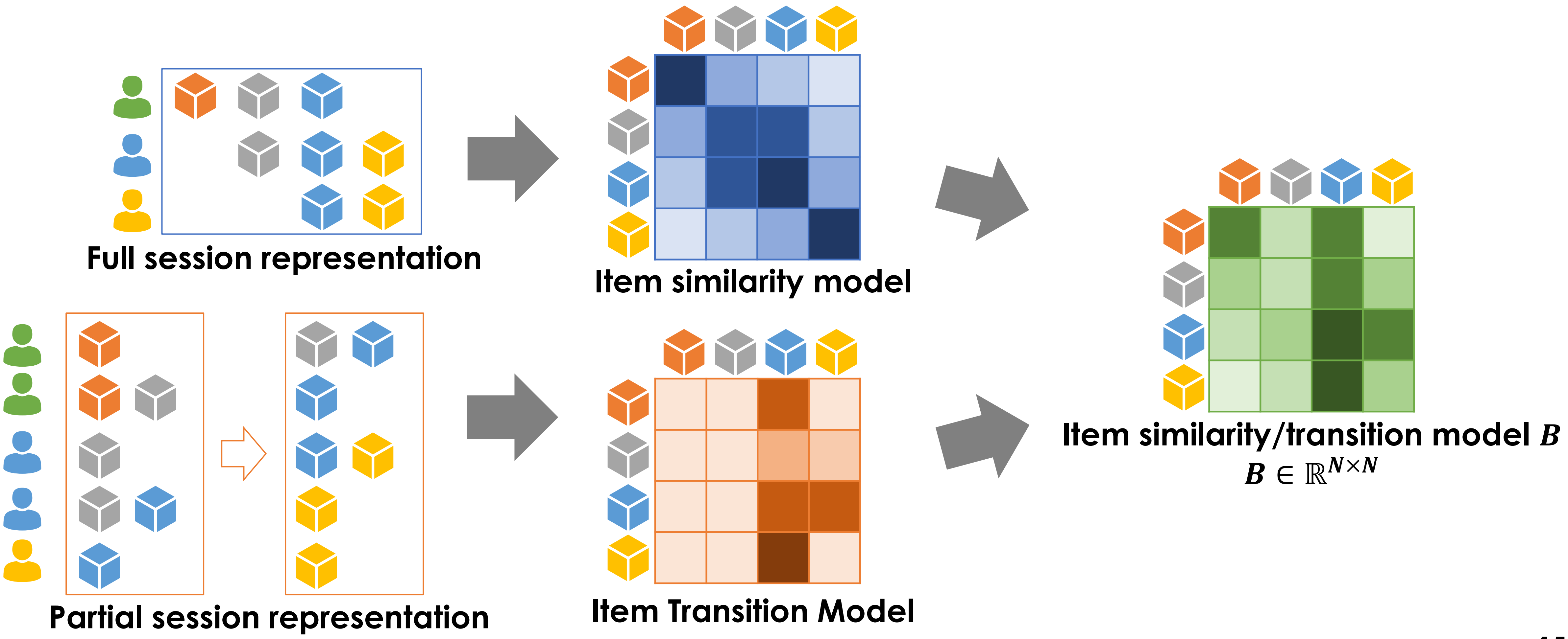
Dark cells mean the high transition probability.

$|S'|$: # of partial sessions

n: # of items

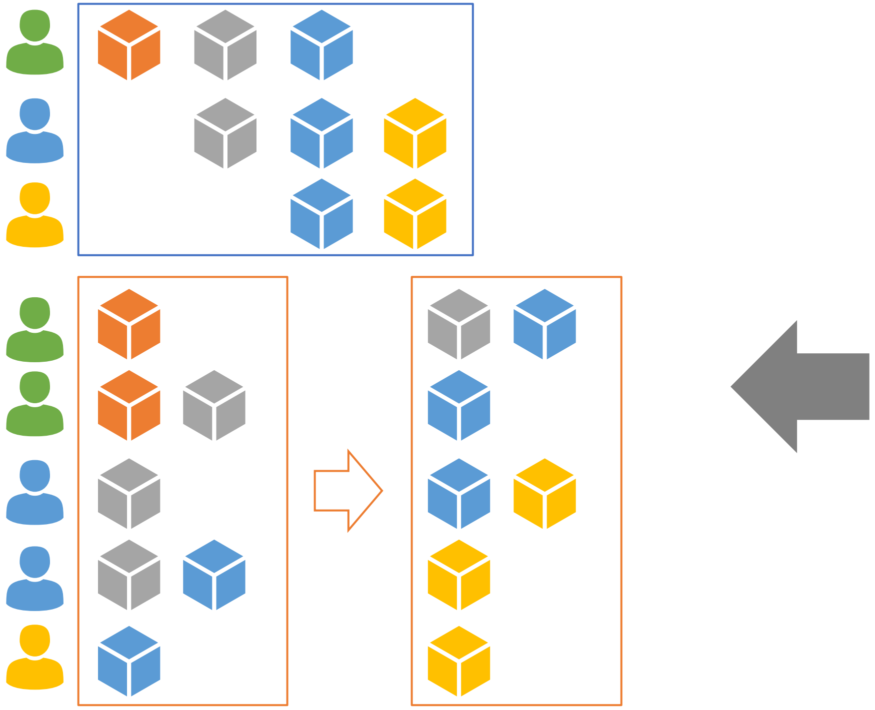
Item Similarity/Transition Model: SLIST

- We **unify two linear models** by jointly optimizing both models.

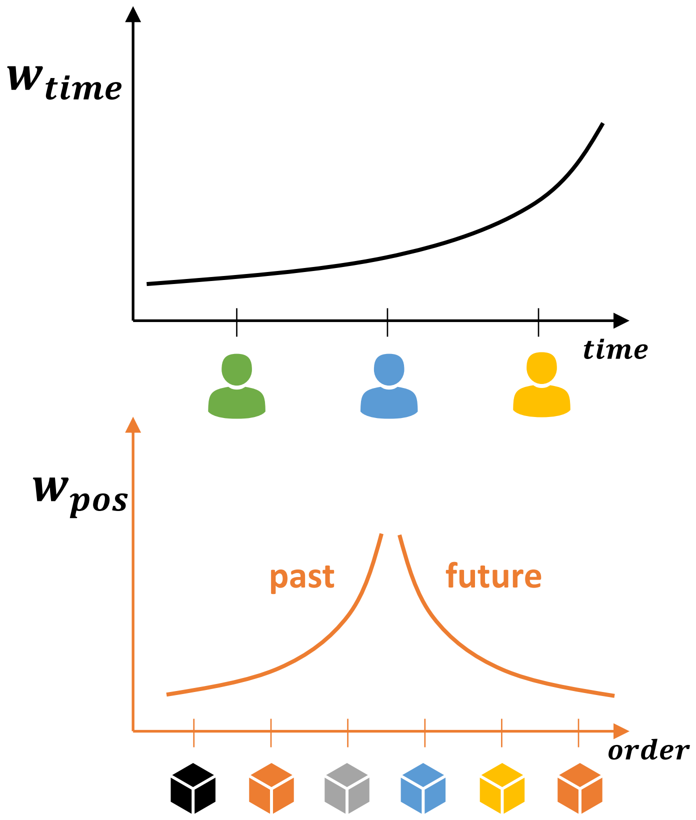


Weight Decay of Items in a Session

- We assign higher weights for more recent sessions (w_{time}) and current items (w_{pos}, w_{inf}) in the sessions.



Full and partial session representation



Detail: Training and Inference

➢The objective function of SLIST

Item similarity model (SLIS) Item transition model (SLIT)

$$\underset{B}{argmin}\,\alpha\left\|W_{full}\odot(X-XB)\right\|_F^2+(1-\alpha)\left\|W_{par}\odot(T-SB)\right\|_F^2+\lambda\|B\|_F^2$$

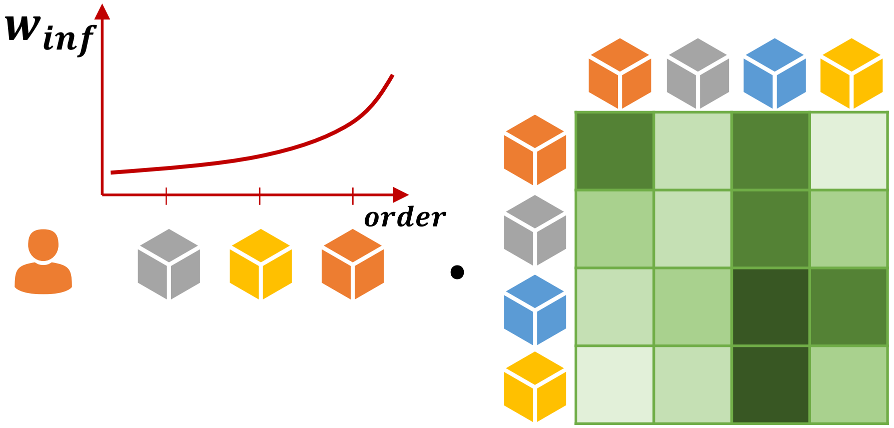
Closed form solution

$$\hat{B}=I-\lambda\hat{P}-(1-\alpha)\hat{P}S^{\top}diagMat(w_{par}^2)(S-T)$$

➢Model inference

Partial representation vector Learned item-item model

$$\hat{t}=s\cdot\hat{B}$$



Experiments

Experimental Setup: Dataset

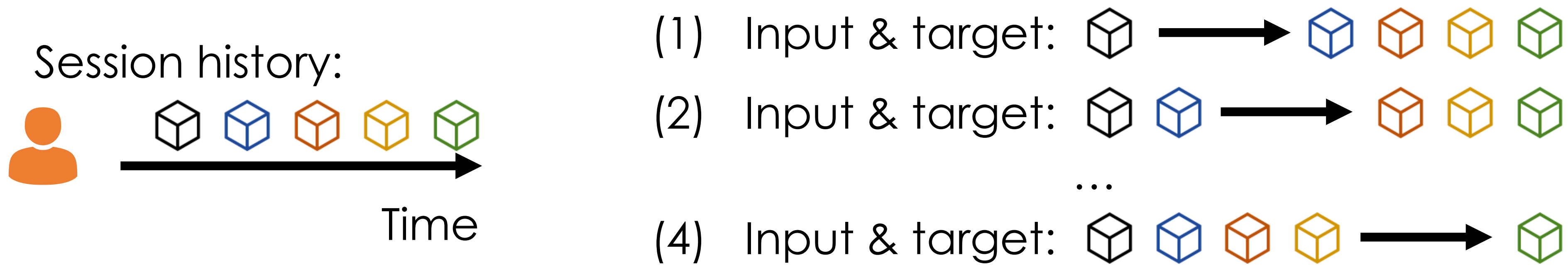
➢ **We evaluate the accuracy and scalability of SLIST over public datasets.**
- For a fair comparison, we evaluate both on 1-split and 5-split datasets.

Split	Dataset	# of actions	# of sessions	# of items
1-split	YooChoose 1/64 (YC-1/64)	494,330	119,287	17,319
	YooChoose 1/4 (YC-1/4)	7,909,307	1,939,891	30,638
	DIGINETICA (DIGI1)	916,370	188,807	43,105
5-split	YooChoose (YC5)	5,426,961	1,375,128	28,582
	DIGINETICA (DIGI5)	203,488	41,755	32,137

Evaluation Protocol and Metrics

➢Evaluation protocol: iterative revealing scheme

- We iteratively expose the item of a session to the model.



➢Evaluation metrics

- HR@20 and MRR@20
  - To predict only the next item in a session
- R@20 and MAP@20
  - To consider all subsequent items for a session

Competitive Models

- **Two neighborhood-based models**
  - **SKNN**: a session-based KNN algorithm, a variant of user-based KNN.
  - **STAN**: an improved version of SKNN by considering sequential dependency and timeliness of sessions.

- **Four DNN-based models**
  - **GRU4Rec+**: an improved version of GRU4REC using the top-k gain.
  - **NARM**: an improved version of GRU4REC+ using an attention mechanism.
  - **STAMP**: an attention-based model for capturing user's interests.
  - **SR-GNN**: a GNN-based model to capture complex dependency.

Dietmar Jannach et. al., "When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation", RecSys 2017
Diksha Garg et. al., "Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN", SIGIR 2019
Balázs Hidasi et. al., "Session-based Recommendations with Recurrent Neural Networks", ICLR 2016.
Jing Li et. al., "Neural Attentive Session-based Recommendation" CIKM 2017.
Qiao Liu et. al., "STAMP: ShortTerm Attention/Memory Priority Model for Session-based Recommendation" KDD 2018.
Shu Wu et. al., "Session-Based Recommendation with Graph Neural Networks", AAAI 2019.

Accuracy: Ours vs. Competing Models

- **Our models consistently show competitive performance.**
  - No existing models show state-of-the-art performance for all datasets.

Dataset	Metric	SKNN (Neighborhood-based)	STAN (Neighborhood-based)	GRU4Rec+ (DNN-based)	NARM (DNN-based)	STAMP (DNN-based)	SR-GNN (DNN-based)	SLIS (Ours)	SLIT (Ours)	SLIST (Ours)	Gain(%)
YC-1/64	HR@20	0.6423	0.6838	0.6528	0.6998	0.6841	<u>0.7021</u>	0.7015	0.6968	**0.7088**	0.95
	R@20	0.4780	0.4955	0.4009	<u>0.5051</u>	0.4904	0.5049	0.5051	0.4976	**0.5080**	0.57
YC-1/4	HR@20	0.6329	0.6846	0.6940	0.7079	0.7021	<u>0.7118</u>	0.7119	0.7149	**0.7175**	0.80
	R@20	0.4756	0.4952	*0.4887*	<u>0.5097</u>	0.5008	0.5095	0.5121	0.5110	**0.5130**	0.65
DIGI1	HR@20	0.4846	<u>0.5121</u>	0.5097	0.4979	0.4690	0.4904	0.5247	0.4729	**0.5291**	3.32
	R@20	0.3788	<u>0.3965</u>	0.3957	0.3890	0.3663	0.3811	0.4062	0.3651	**0.4091**	3.18
YC5	HR@20	0.5996	0.6656	0.6488	<u>0.6751</u>	0.6654	0.6713	0.6786	0.6838	**0.6867**	1.72
	R@20	0.4658	0.4986	0.4837	<u>0.5109</u>	0.4979	0.5060	0.5074	0.5097	**0.5122**	0.25
DIGI5	HR@20	0.4748	<u>0.4800</u>	0.4639	0.4188	0.3917	0.4158	0.4939	0.4193	**0.5005**	4.27
	R@20	0.3715	<u>0.3720</u>	0.3617	0.3254	0.3040	0.3232	0.3867	0.3260	**0.3898**	4.78

Scalability: Ours vs. Competing Models

- **Training SLIST is much faster than training DNN-based models.**
  - Computational complexity is mainly proportional to the number of items.
  - It is highly desirable in practice, especially on popular online services, where millions of users create billions of session data every day.

Models	The ratio of training sessions on YC-1/4				
	5%	10%	20%	50%	100%
GRU4Rec+	177.4	317.8	614.0	1600.2	3206.9
NARM	2137.6	3431.2	7454.2	27804.5	72076.8
STAMP	434.1	647.5	985.3	2081.2	4083.3
SR-GNN	6780.5	18014.7	31444.3	68581.9	185862.5
SLIST	202.7	199.0	199.9	227.0	241.9
Gains (SLIST vs. SR-GNN)	**33.5x**	**90.6x**	**157.3x**	**302.1x**	**768.2x**

Ablation Study: Weight components

- **SLIST with all components is always better than others.**
  - For three datasets, considering recent items at the inference phase (w_{inf}) is the most influential factor.

	w_{inf}	w_{pos}	w_{time}	YC-1/64		YC-1/4		DIGI1	
				HR	MRR	HR	MRR	HR	MRR
All Weights	O	O	O	**0.7088**	**0.3083**	**0.7175**	**0.3161**	**0.5291**	**0.1886**
Two Weights	O	O		0.7078	0.3077	0.7096	0.3115	0.5253	0.1880
	O		O	0.6761	0.2697	0.6841	0.2753	0.5145	0.1820
		O	O	0.6880	0.2973	0.7004	0.3031	0.5202	0.1855
One Weights	O			0.6764	0.2697	0.6784	0.2723	0.5111	0.1817
		O		0.6898	0.2980	0.6947	0.3005	0.5173	0.1852
			O	0.6570	0.2651	0.6675	0.2693	0.5055	0.1789
No Weights				0.6592	0.2652	0.6626	0.2671	0.5025	0.1786

Conclusion

Conclusion

- **We propose session-aware linear item models to fully capture various characteristics of sessions.**
  - SLIST: session-aware linear similarity/transition model

- **Despite its simplicity, SLIST achieves competitive or state-of-the-art performance on various datasets.**

- **SLIST is highly scalable thanks to closed-form solutions of the linear models.**
  - The computational complexity of SLIST is proportional to the number of items, which is desirable in commercial systems.

Q&A

Email: zxcvxd@skku.edu

Code: https://github.com/jin530/SLIST