Dynamic Personalised Recommendations

Data and Algorithms

About Me

https://www.linkedin.com/in/yxtay/

https://github.com/yxtay/

Tay Yu Xuan

Data Scientist, RecSys, Traveloka

Previously, Data Scientist, Lazada

Experience in Public Service, Ecommerce, Travel

Work with problems in language, recommendations

BSc Statistics, National University of Singapore

Agenda

Why Recommendations

Use cases

RecSys Architecture

Smart Adaptive Recommendations

Similarity Model

Affinity Model

Personalised Recommendations

Implementation

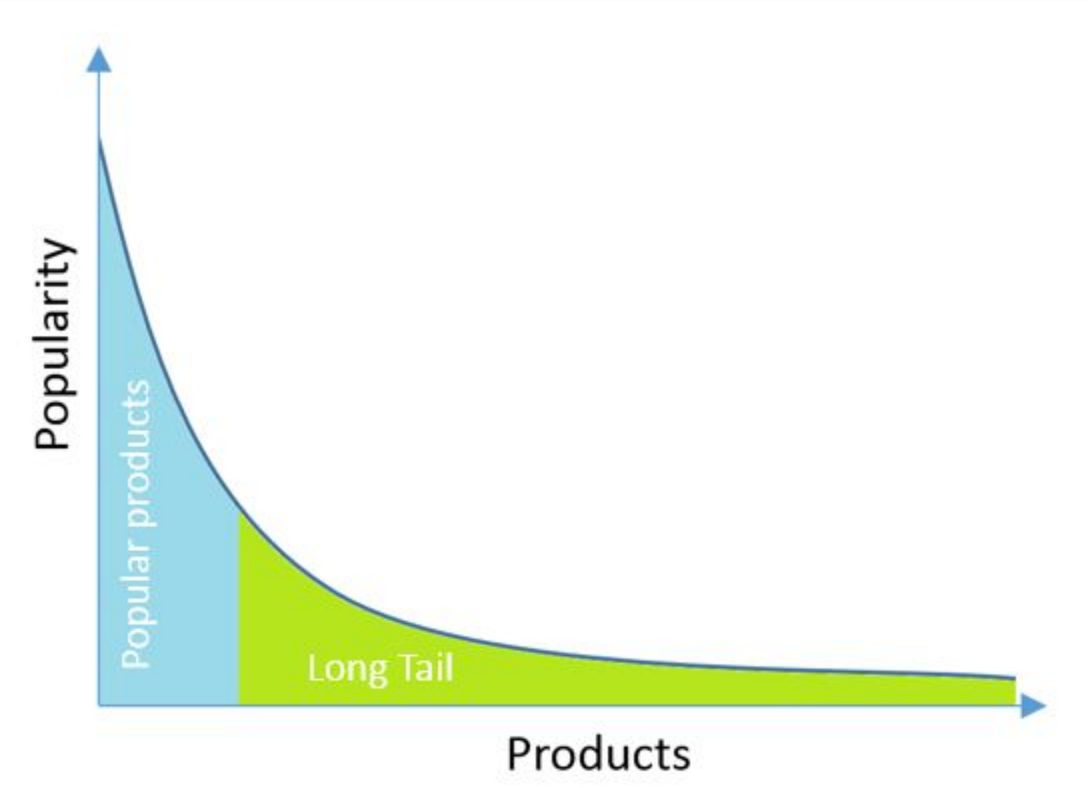
Conclusion

Why Recommendations?

Assists Discovery and Decision

Long tail: **pareto principle**

Promote discovery of niche products



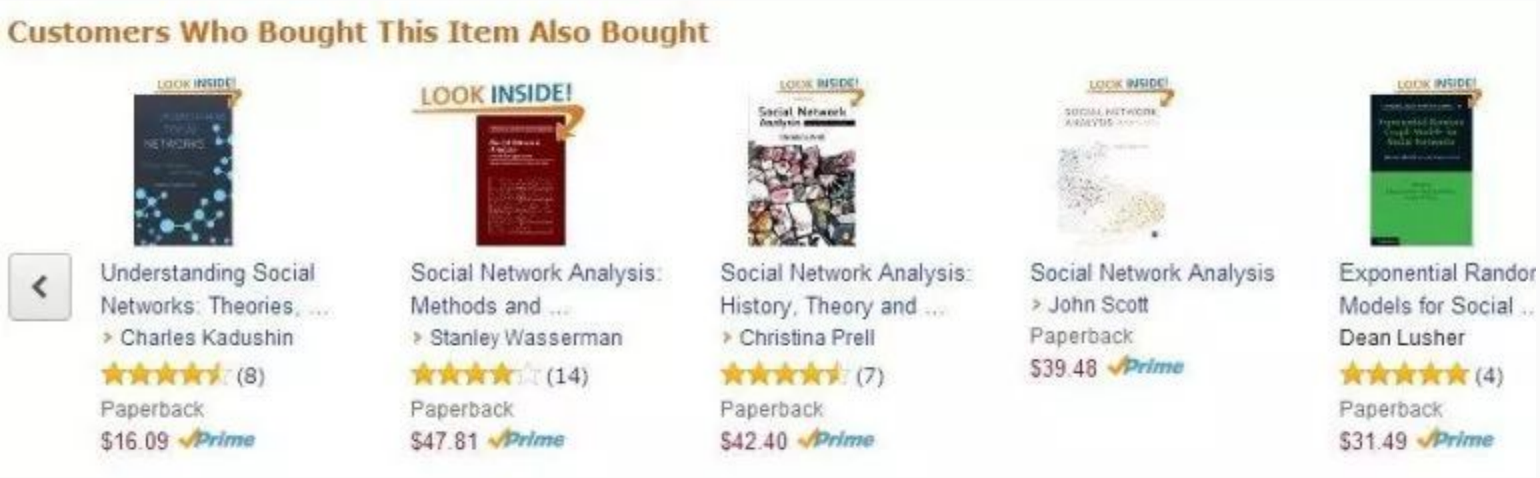
Personalisation Promotes Satisfaction

Increase customer satisfaction by **recommending relevant products**

Real-time personalisation further encourages user interaction

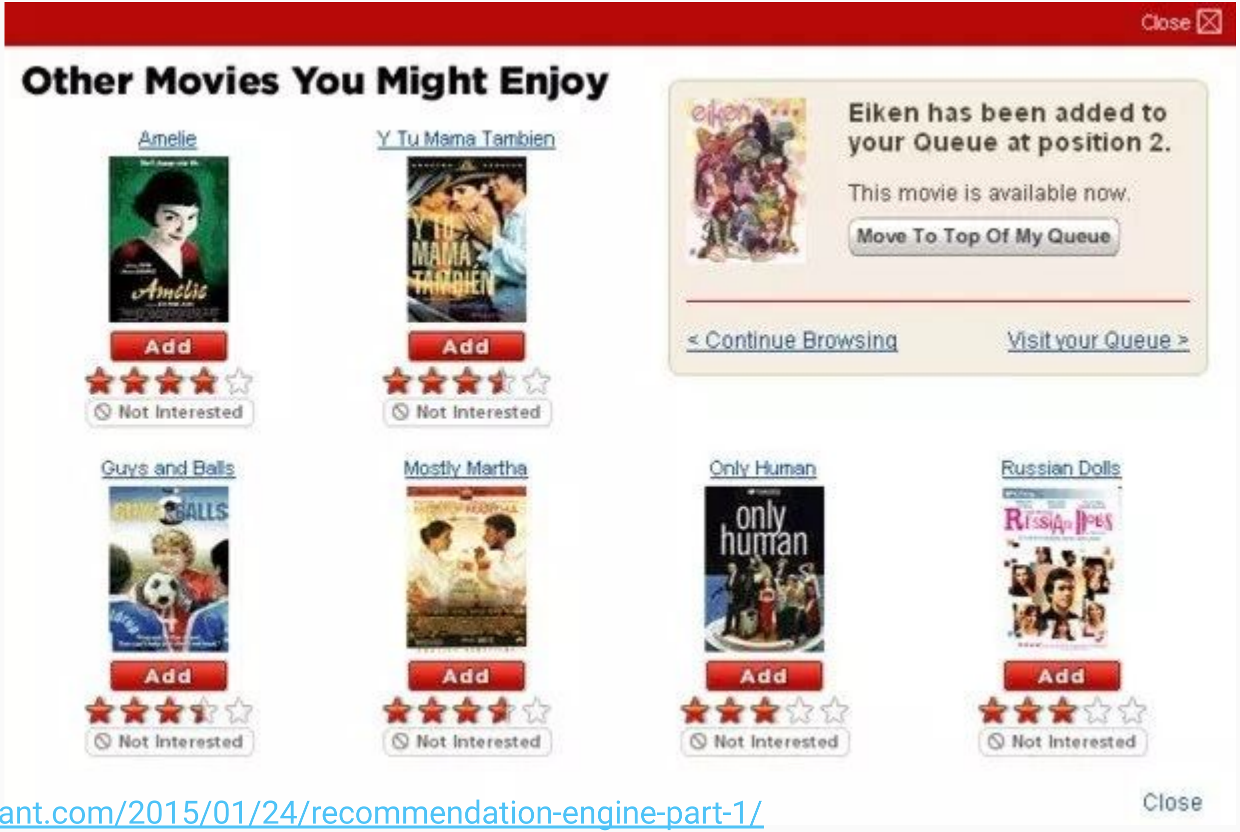
Use cases

Item Similarity Recommendations



http://dataaspirant.com/2015/01/24/recommendation-engine-part-1/

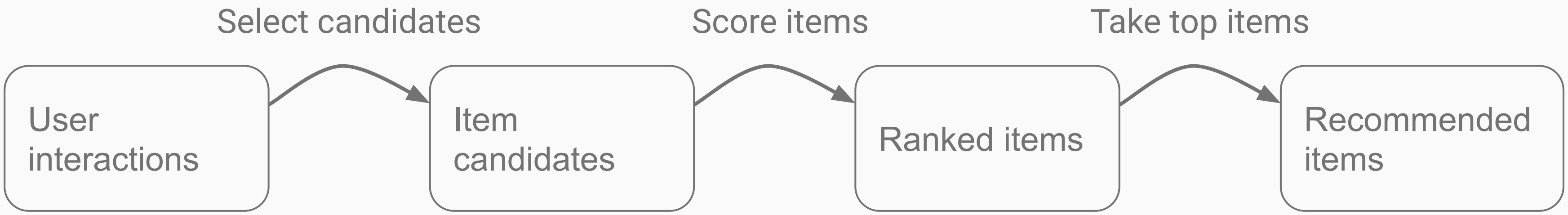
Personalised Recommendations



http://dataaspirant.com/2015/01/24/recommendation-engine-part-1/

RecSys Architecture

Simplified Overview



User segmentation

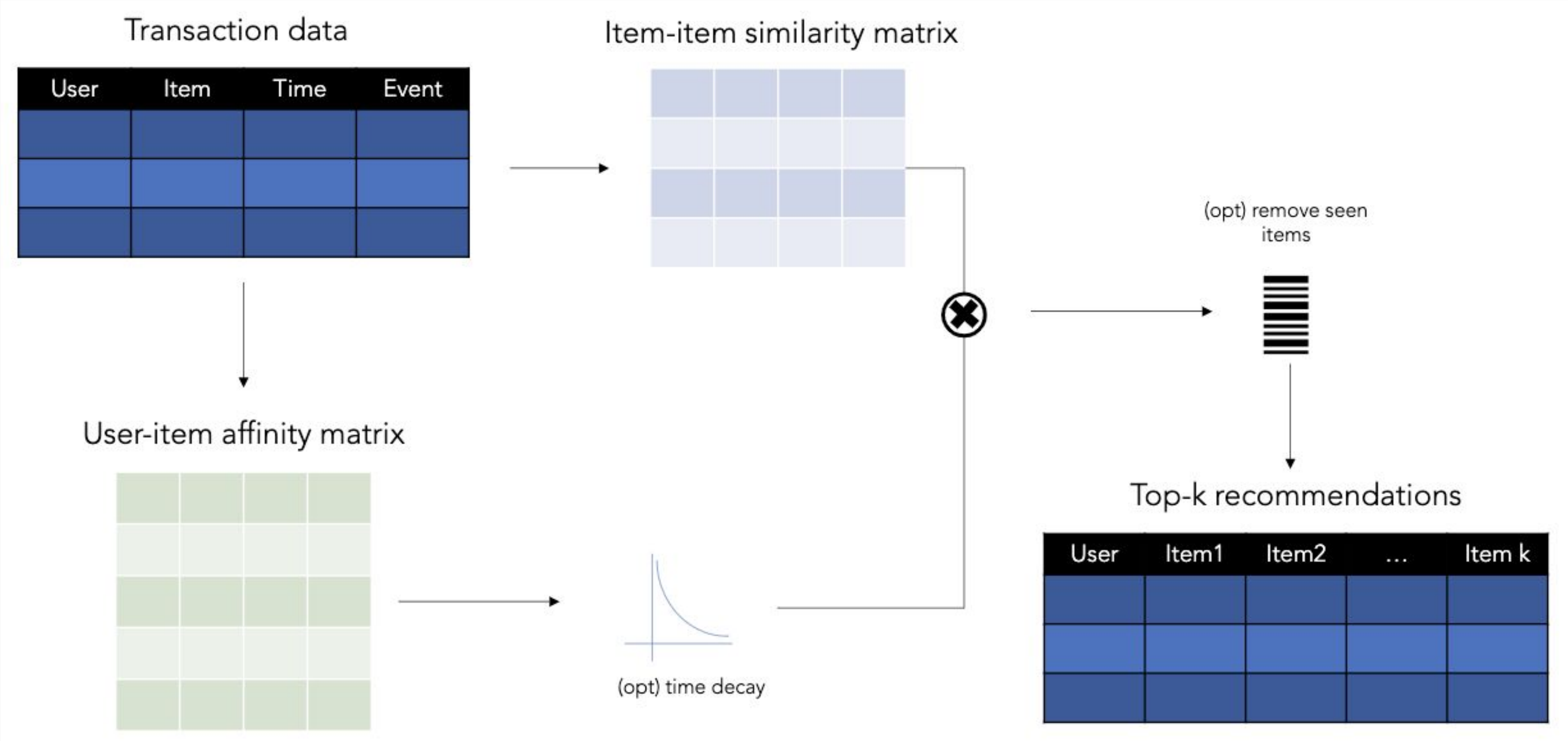
User preferences

Item similarity

User-item relevance

Smart Adaptive Recommendations (SAR)

Model Overview



https://github.com/microsoft/recommenders/blob/master/notebooks/02_model/sar_deep_dive.ipynb

Characteristics

Based on **item similarity** and **recency**

Collaborative filtering on implicit feedback

Real-time personalisation

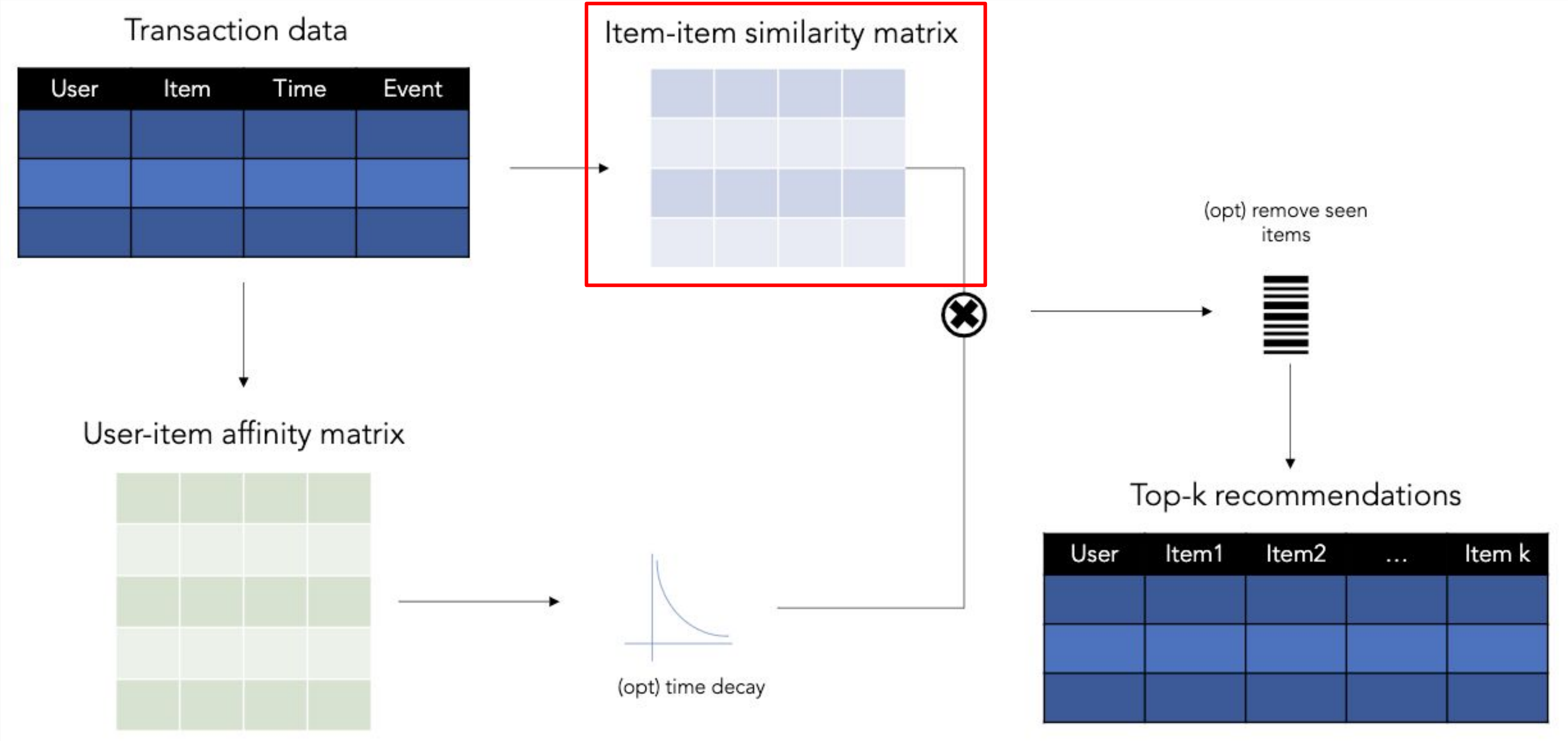
Works on **new users**

Cold start problem for items

Serves both item similarity and personalised recommendations

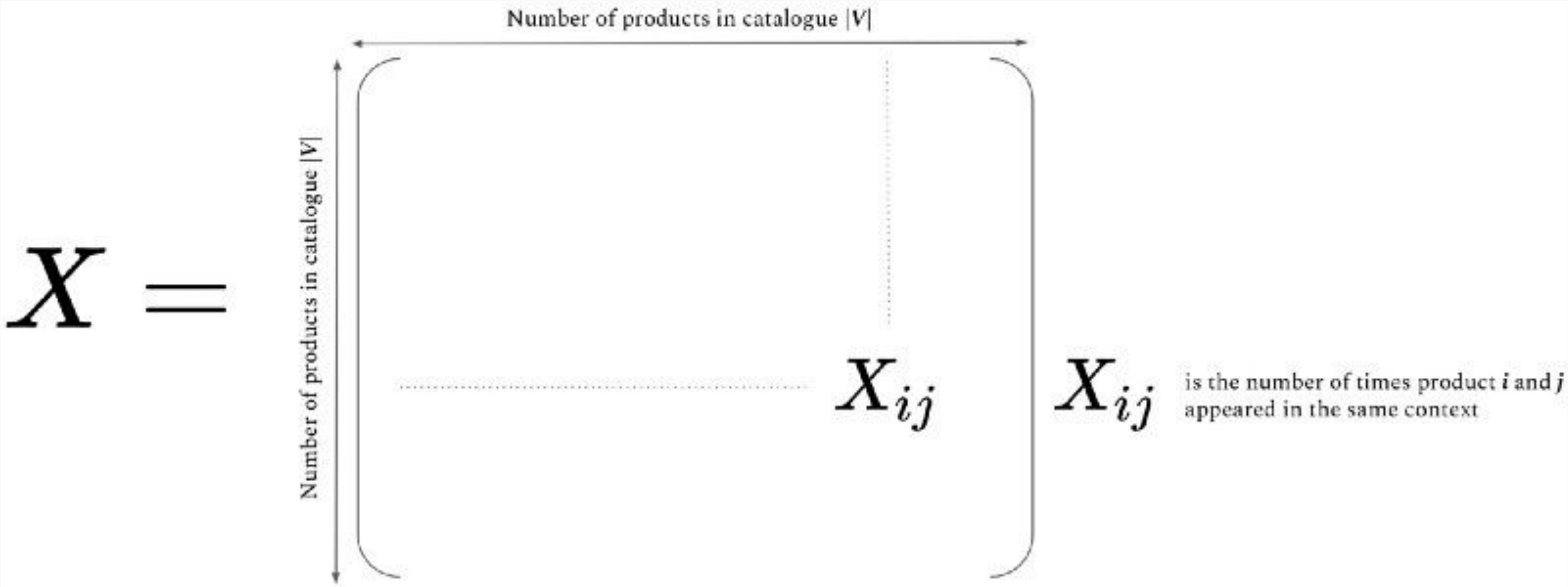
Similarity Model

Model Overview



https://github.com/microsoft/recommenders/blob/master/notebooks/02_model/sar_deep_dive.ipynb

Co-Occurrence Matrix



Count distinct users that have interacted both items i and j within a predefined period (e.g. 1 week)

Co-Occurrence Matrix

User interaction data

user	item	time
user_1	item_1	x
user_1	item_2	y
user_1	item_3	z
...	...	...

Self-join by user

user	item_i	time_i	item_j	time_j
user_1	item_1	x	item_2	y
user_1	item_1	x	item_3	z
user_1	item_2	y	item_1	x
...	...	...	...	...

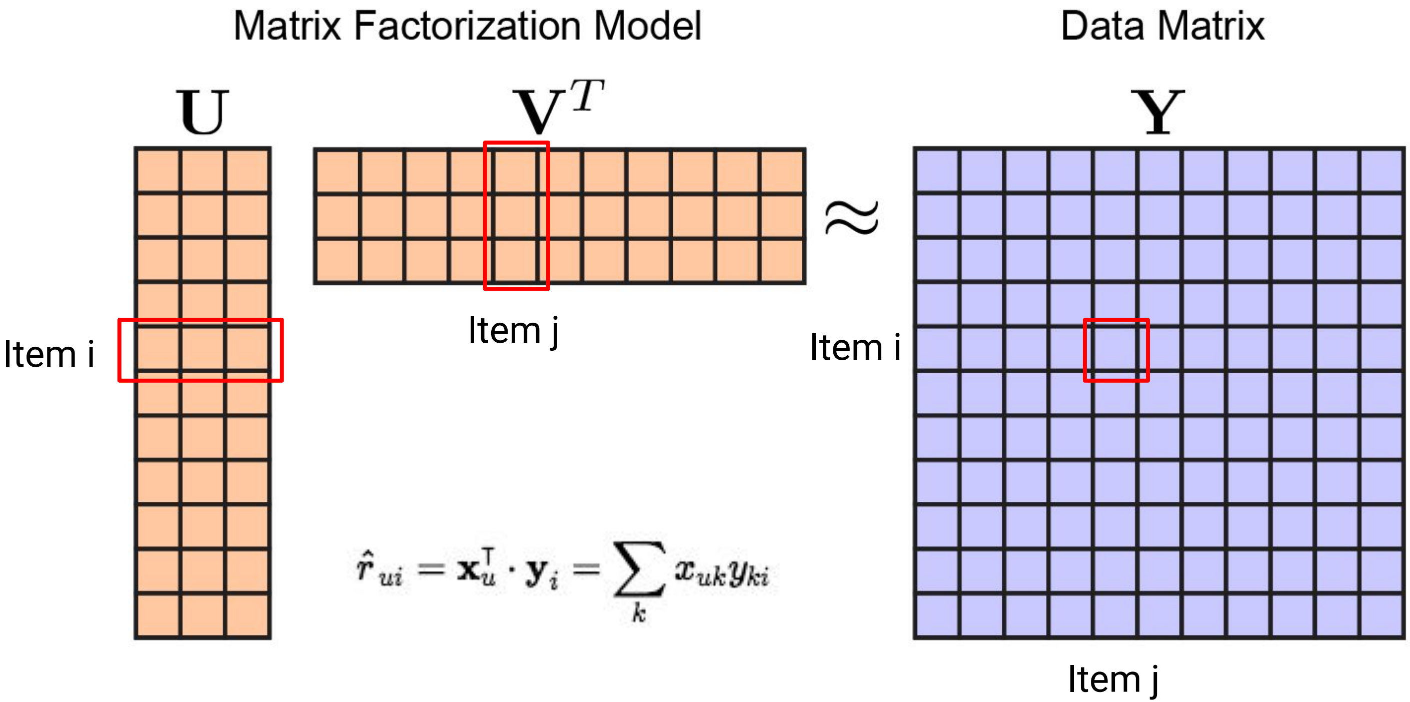
Filter by duration between events

Group by item_i, item_j

Count distinct user

item_i	item_j	count
item_1	item_2	#
item_1	item_3	##
item_2	item_3	###
...	...	...

Matrix Factorisation



$$\hat{r}_{ui} = \mathbf{x}_u^\top \cdot \mathbf{y}_i = \sum_k x_{uk} y_{ki}$$

Matrix Factorisation

$$L = \sum_{u,i \in S} (r_{ui} - \mathbf{x}_u^{\intercal} \cdot \mathbf{y}_i)^2 + \lambda_x \sum_u \|\mathbf{x}_u\|^2 + \lambda_y \sum_u \|\mathbf{y}_i\|^2$$

Alternating least squares

Stochastic gradient descent

GloVe

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

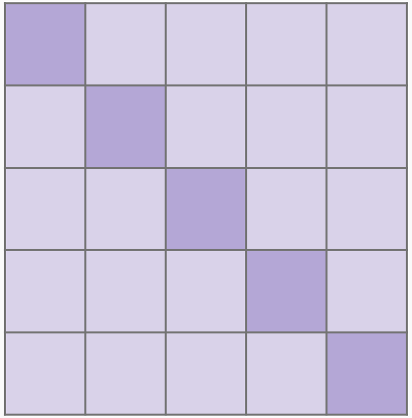
Item Similarity Matrix

Cosine similarity on embeddings

Assumption: similar items are more likely to be **viewed close together** by **many users**

Can be precomputed in batch

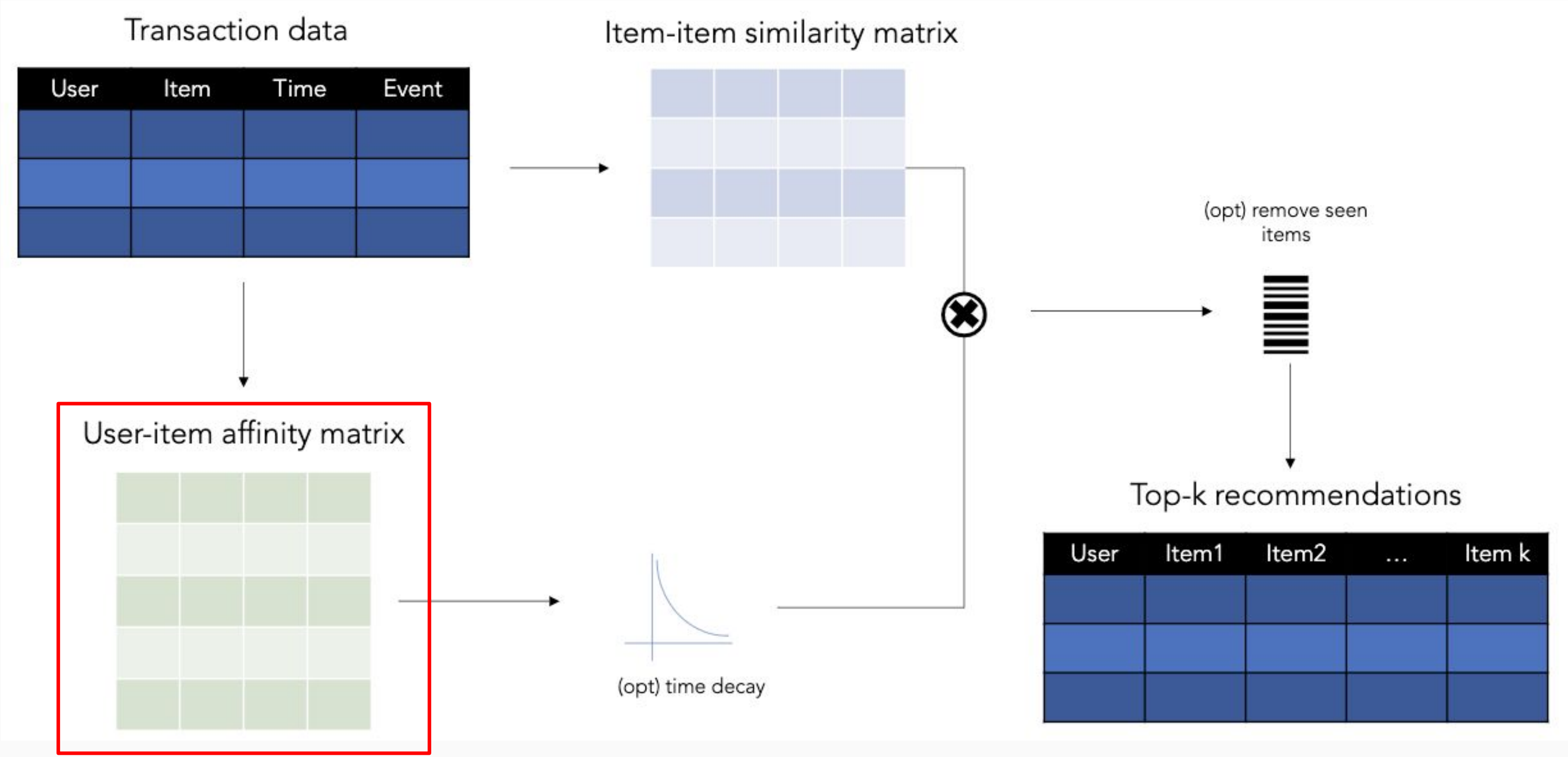
Can be used directly to serve item similarity recommendations



s_{ij}

Affinity Model

Model Overview



https://github.com/microsoft/recommenders/blob/master/notebooks/02_model/sar_deep_dive.ipynb

Item Affinity

User interactions data

user	item	time	event
user_1	item_1	x	view
user_1	item_2	y	purchase
user_1	item_3	z	add-to-cart
...	...	...	

user	item	time weight	event weight	affinity score
user_1	item_1	0.9	1.0	0.9
user_1	item_2	0.6	3.0	1.8
user_1	item_3	0.5	2.0	1.0
...	...	...		

Takes into **type** and **recency** of interaction

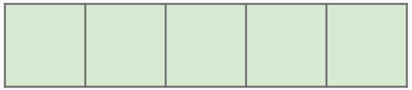
$$a_{ui} = w_{ei}\, 2^{-\frac{t_0 - t_i}{T}}$$

Item Affinity Vector

Assumption: user have preferences on **items recently interacted** with

Used to derive candidate items for recommendations

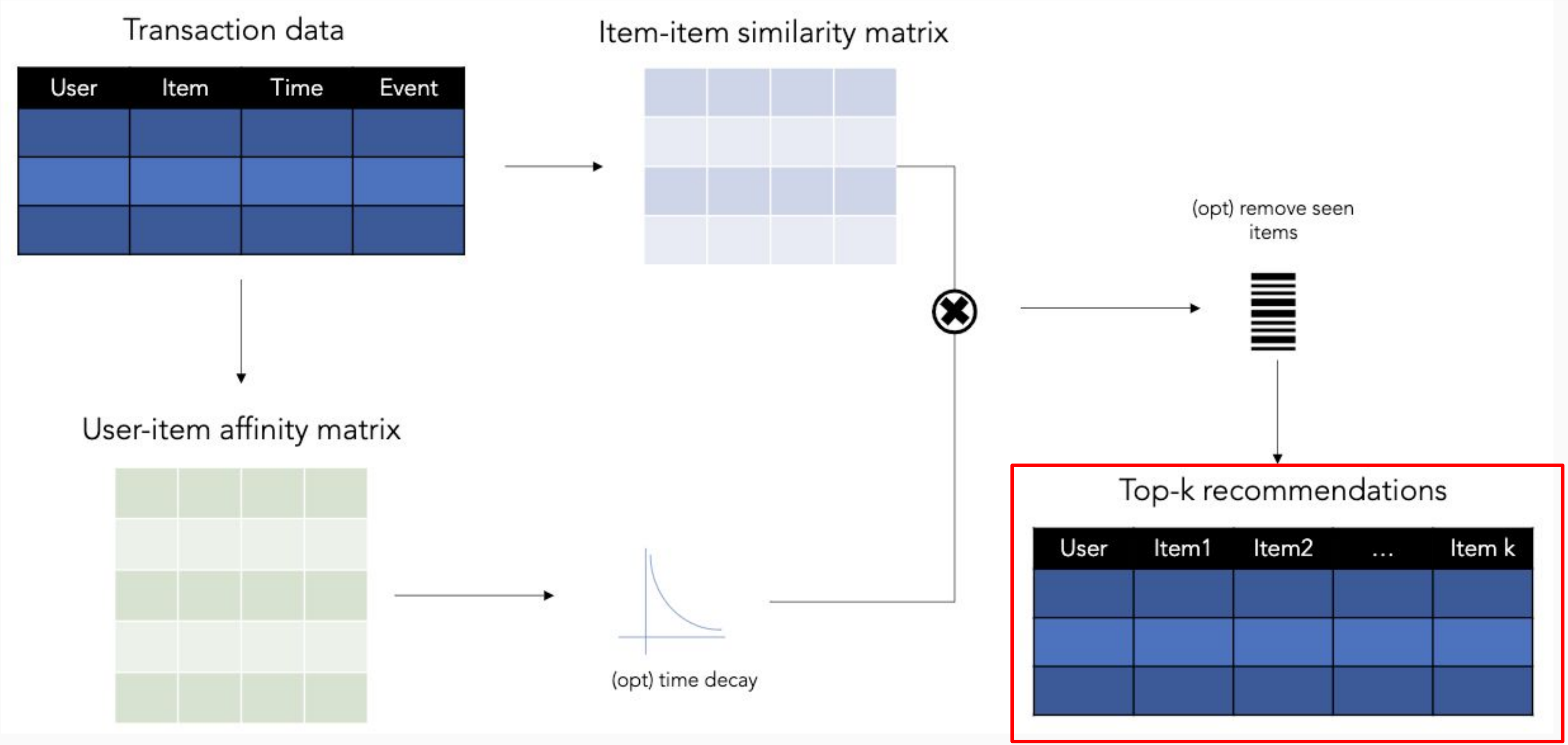
Computed in real-time



a_{ui}

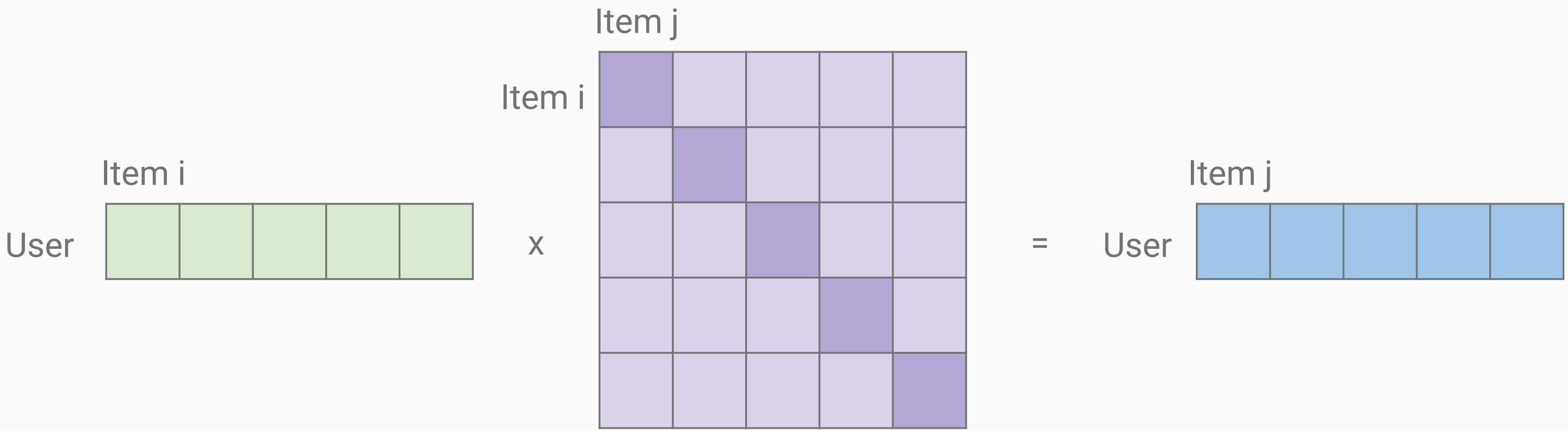
Personalised Recommendations

Model Overview



https://github.com/microsoft/recommenders/blob/master/notebooks/02_model/sar_deep_dive.ipynb

Relevancy Scoring



a_{ui}

affinity

s_{ij}

similarity

$r_{uj} = \sum_i a_{ui} \, s_{ij}$

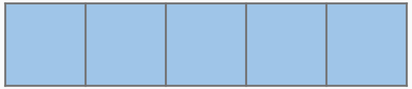
relevancy

Personalised Recommendations

High scores: Very similar to many recent items

Assumption: items **similar to last interacted items** are relevant to user

Computed in real-time



$$r_{uj} = \sum_{i} a_{ui}\, s_{ij}$$

Implementation

Challenges

Real-time recommendations

- **Accessible live tracking of user interaction logs**
- Efficient computation of item similarity and relevance scores

Improvements

Algorithm is generic to apply to any kind of products

Improvements can be made to component models separately

Specific user-item relevance scoring can be added to improve the recommendations

Conclusion

Dynamic Personalised Recommendations

Conceptually straightforward recommendation algorithm

Basic data requirements

Useful baseline for real-time recommendations

Challenge is mostly in engineering

The End, Questions?

Data Team is hiring! https://www.traveloka.com/en-sg/careers

- **Data Scientist**, RecSys & Experimentation teams
- **Data Engineer,** Data team
- **Software Engineer,** Data & RecSys teams
- **DevOps Engineer,** Data team
- **Technical Product Manager,** Data team

References

- https://azure.microsoft.com/en-us/blog/building-recommender-systems-with-azure-machine-learning-service/
- https://github.com/microsoft/recommenders/blob/master/notebooks/02_model/sar_deep_dive.ipynb
- https://github.com/microsoft/Product-Recommendations/blob/master/doc/sar.md
- https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed
- https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-2-deep-recommendation-sequence-prediction-automl-f134bc79d66b
- https://towardsdatascience.com/collaborative-embeddings-for-lipstick-recommendations-98eccfa816bd