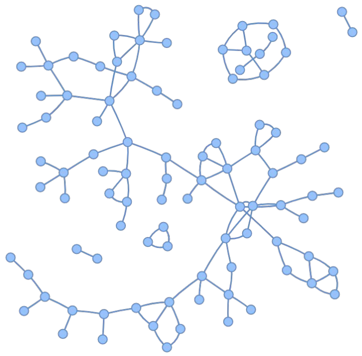


Graph Recommender Systems

It's like ten thousand spoons when all you need is a knife.

- Alanis Morissette

Jay Urbain, PhD - 2/1/2023

References

GCMC: https://arxiv.org/pdf/1706.02263.pdf

PinSAGE: https://arxiv.org/pdf/1806.01973.pdf

Pinterest Engineering Blog:

https://medium.com/pinterest-engineering/pinsage-a-new-graph-convolutional-neural-network-for-web-scale-recommender-systems-88795a107f48

Diffnet++: https://arxiv.org/pdf/2002.00844.pdf

GraphRec: https://arxiv.org/abs/1902.07243

Dynamic GNN for Seq. Rec: https://arxiv.org/pdf/2104.07368.pdf

Google Developers Blog Post:

https://developers.google.com/machine-learning/recommendation/collaborative/basics

Graph Convolutional Neural Networks for Web-Scale Recommender Systems

Rex Ying*†, Ruining He*, Kaifeng Chen*†, Pong Eksombatchai*,

William L. Hamilton†, Jure Leskovec*†

*Pinterest, †Stanford University

{rhe,kaifengchen,pong}@pinterest.com,{rexying,wleif,jure}@stanford.edu

ABSTRACT

Recent advancements in deep neural networks for graph-structured data have led to state-of-the-art performance on recommender system benchmarks. However, making these methods practical and scalable to web-scale recommendation tasks with billions of items and hundreds of millions of users remains a challenge.

Here we describe a large-scale deep recommendation engine that we developed and deployed at Pinterest. We develop a data-efficient Graph Convolutional Network (GCN) algorithm PinSage, which combines efficient random walks and graph convolutions to generate embeddings of nodes (i.e., items) that incorporate both graph structure as well as node feature information. Compared to prior GCN approaches, we develop a novel method based on highly efficient random walks to structure the convolutions and design a novel training strategy that relies on harder-and-harder training examples to improve robustness and convergence of the model.

We deploy PinSage at Pinterest and train it on 7.5 billion examples on a graph with 3 billion nodes representing pins and boards, and 18 billion edges. According to offline metrics, user studies and A/B tests, PinSage generates higher-quality recommendations than comparable deep learning and graph-based alternatives. To our knowledge, this is the largest application of deep graph embeddings to date and paves the way for a new generation of web-scale recommender systems based on graph convolutional architectures.

ACM Reference Format:
Rex Ying*†, Ruining He*, Kaifeng Chen*†, Pong Eksombatchai*, William L. Hamilton†, Jure Leskovec*†. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219890

1 INTRODUCTION

Deep learning methods have an increasingly critical role in recommender system applications, being used to learn useful low-dimensional embeddings of images, text, and even individual users [9, 12]. The representations learned using deep models can be used to complement, or even replace, traditional recommendation algorithms like collaborative filtering. and these learned representations have high utility because they can be re-used in various recommendation tasks. For example, item embeddings learned using a deep model can be used for item-item recommendation and also to recommended themed collections (*e.g.*, playlists, or "feed" content).

Recent years have seen significant developments in this space—especially the development of new deep learning methods that are capable of learning on graph-structured data, which is fundamental for recommendation applications (*e.g.*, to exploit user-to-item interaction graphs as well as social graphs) [6, 19, 21, 24, 29, 30].

Most prominent among these recent advancements is the success of deep learning architectures known as Graph Convolutional Networks (GCNs) [19, 21, 24, 29]. The core idea behind GCNs is to learn how to iteratively aggregate feature information from local graph neighborhoods using neural networks (Figure 1). Here a single "convolution" operation transforms and aggregates feature information from a node's one-hop graph neighborhood, and by stacking multiple such convolutions information can be propagated across far reaches of a graph. Unlike purely content-based deep models (*e.g.*, recurrent neural networks [3]), GCNs leverage both content information as well as graph structure. GCN-based methods have set a new standard on countless recommender system benchmarks (see [19] for a survey). However, these gains on benchmark tasks have yet to be translated to gains in real-world production environments.

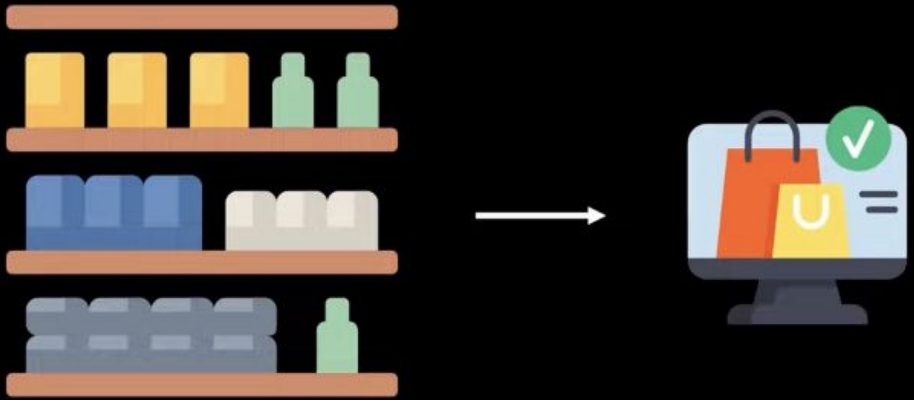
The main challenge is to scale both the training as well as inference of GCN-based node embeddings to graphs with billions of nodes and tens of billions of edges. Scaling up GCNs is difficult because many of the core assumptions underlying their design are violated when working in a big data environment. For example, all existing GCN-based recommender systems require operating on the full graph Laplacian during training—an assumption that is infeasible when the underlying graph has billions of nodes and whose structure is constantly evolving.

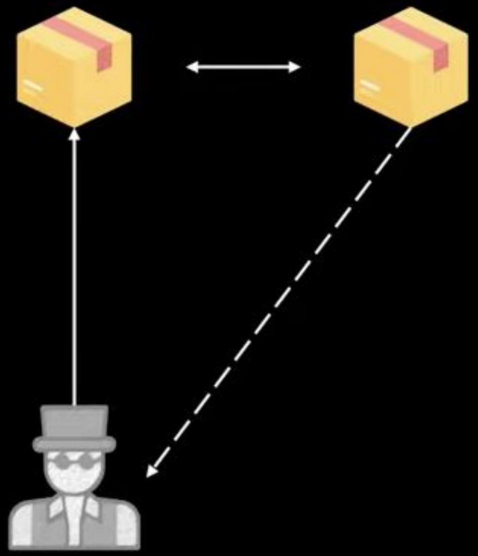
Present work. Here we present a highly-scalable GCN framework that we have developed and deployed in production at Pinterest. Our framework, a random-walk-based GCN named PinSage, operates on a massive graph with 3 billion nodes and 18 billion edges—a graph that is 10,000× larger than typical applications of GCNs. PinSage leverages several key insights to drastically improve the scalability of GCNs:

KDD '18, August 19–23, 2018, London, United Kingdom

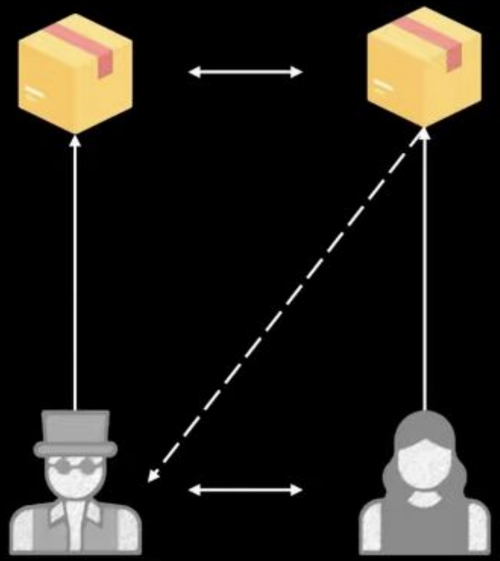
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219890

People are overwhelmed with information and choices.

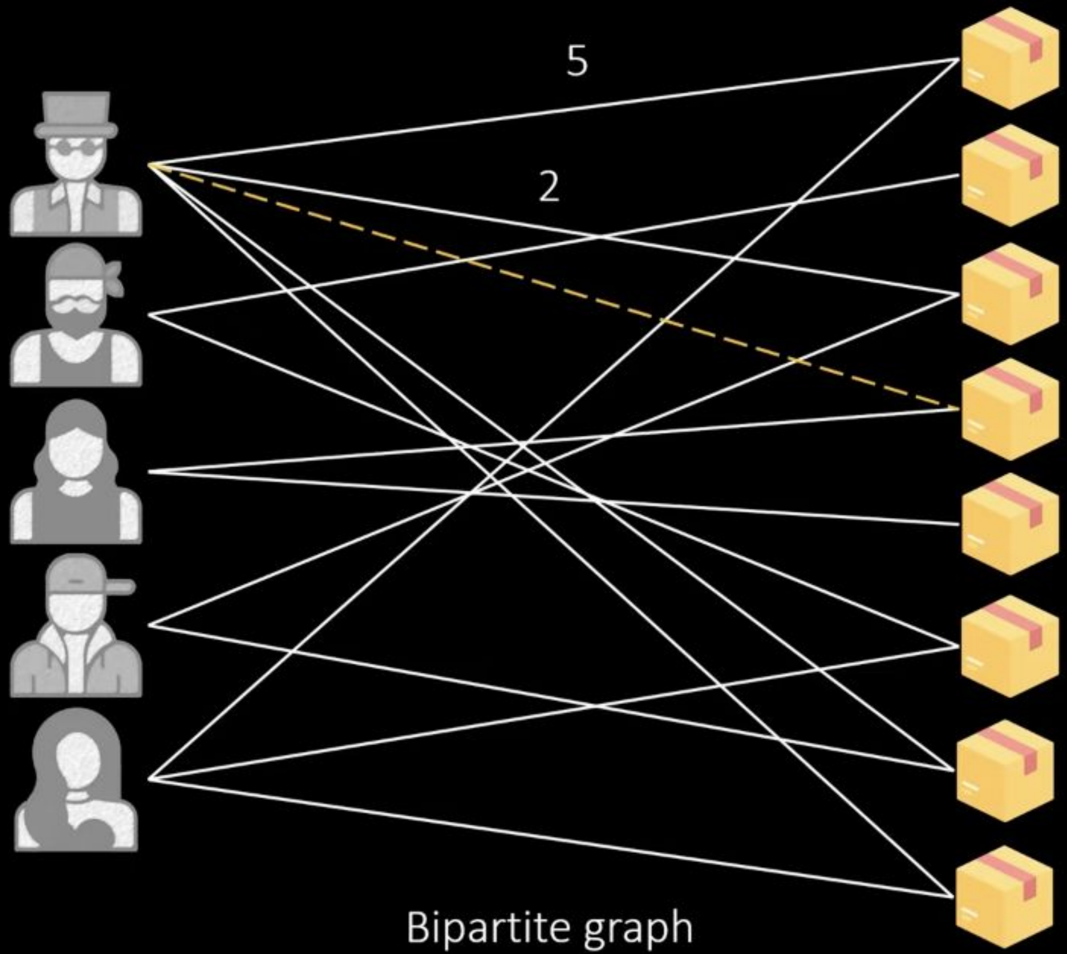




Content-based filtering



Collaborative filtering

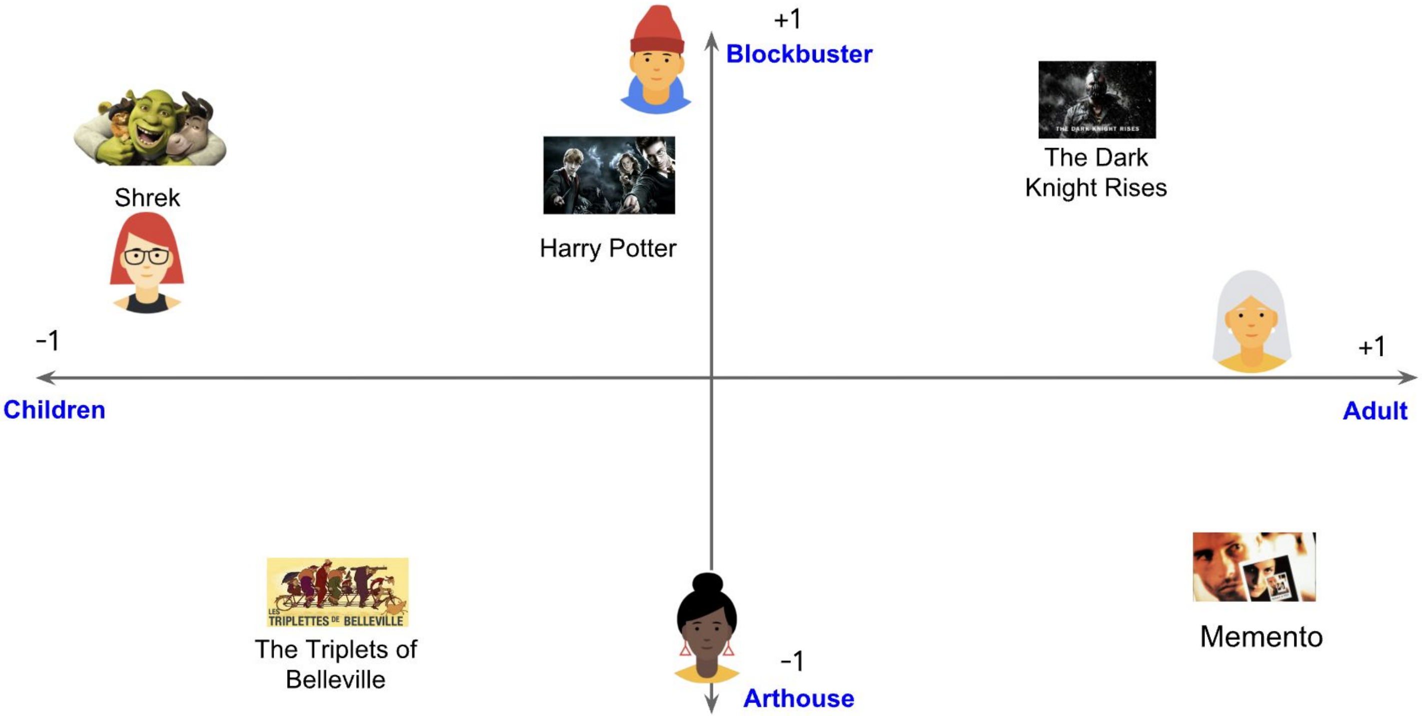


Bipartite graph

In the diagram below, each checkmark identifies a movie that a particular user watched. The third and fourth users hav preferences that are well explained by this feature—the third user prefers movies for children and the fourth user prefer movies for adults. However, the first and second users' preferences are not well explained by this single feature.

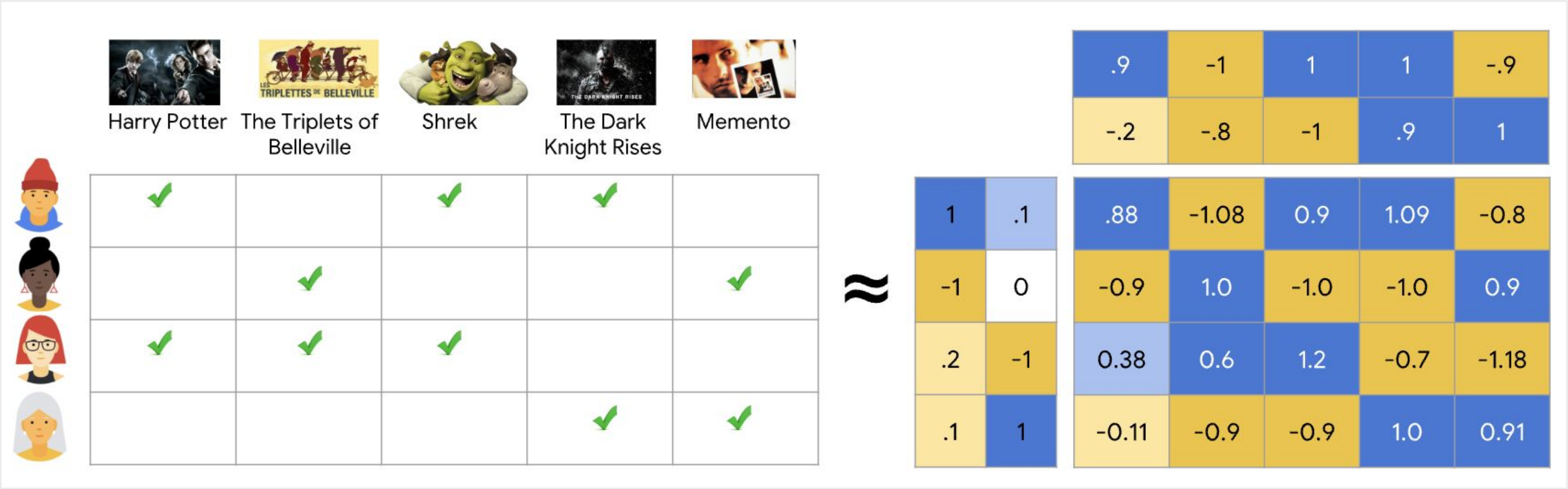
2D Embedding

One feature was not enough to explain the preferences of all users. To overcome this problem, let's add a second feature: the degree to which each movie is a blockbuster or an arthouse movie. With a second feature, we can now represent each movie with the following two-dimensional embedding:



Matrix factorization is a simple embedding model. Given the feedback matrix A $\in R^{m \times n}$, where m is the number of users (or queries) and n is the number of items, the model learns:

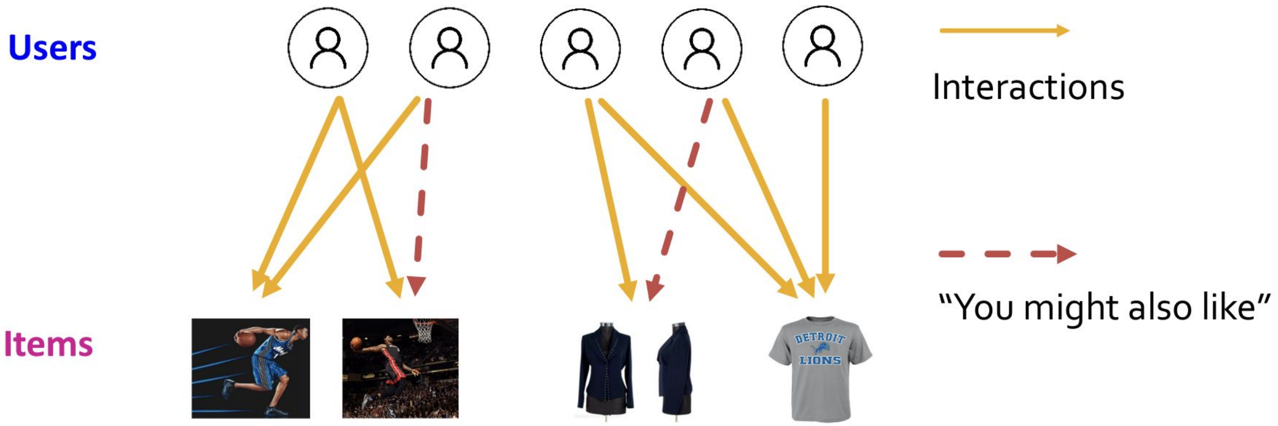
- A user embedding matrix $U \in \mathbb{R}^{m \times d}$, where row i is the embedding for user i.
- An item embedding matrix $V \in \mathbb{R}^{n \times d}$, where row j is the embedding for item j.



The embeddings are learned such that the product UV^T is a good approximation of the feedback matrix A. Observe that the (i, j) entry of $U.V^T$ is simply the dot product $\langle U_i, V_j \rangle$ of the embeddings of user i and item j, which you want to be close to $A_{i,j}$.

PinSage Recommender Systems

- Users interacts with items
  - Watch movies, buy merchandise, listen to music
- Goal: Recommend items users might like
  - Customer X buys Metallica and Megadeth CDs
  - Customer Y buys Megadeth, the recommender system suggests Metallica as well



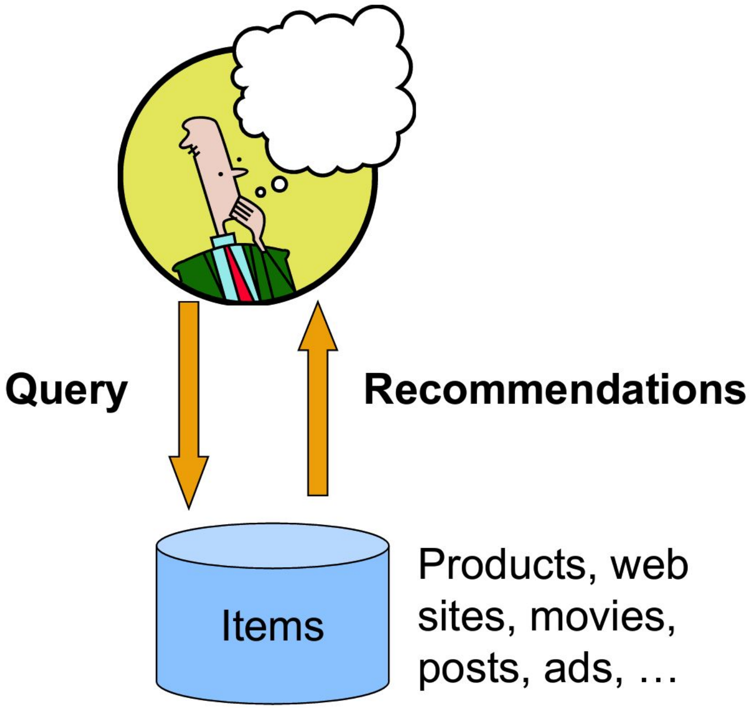
PinSage Recommender System

Goal: Learn what items are related

- For a given query item(s) ***Q***, return a set of similar items that we recommend to the user

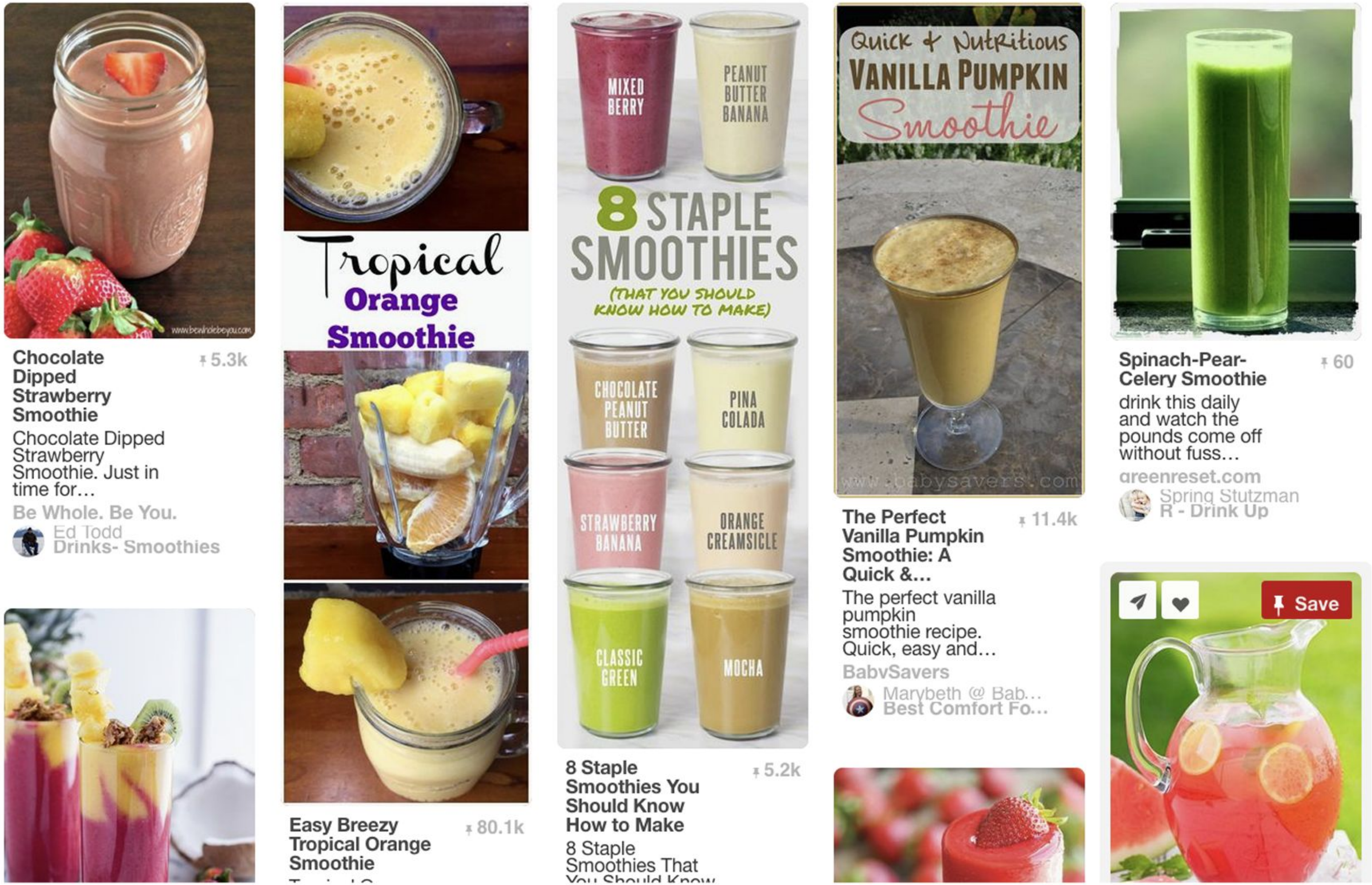
Idea:

- User interacts with a set of items
- Formulate a query Query ***Q***
- Search the items and return recommendations

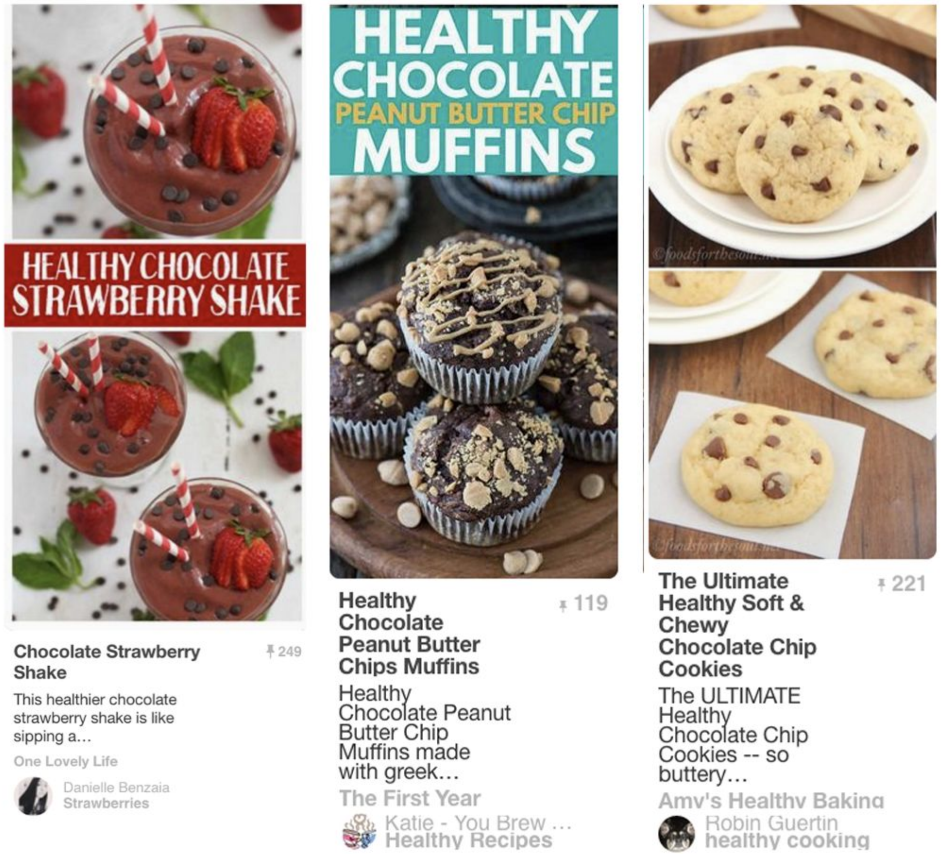


Query:

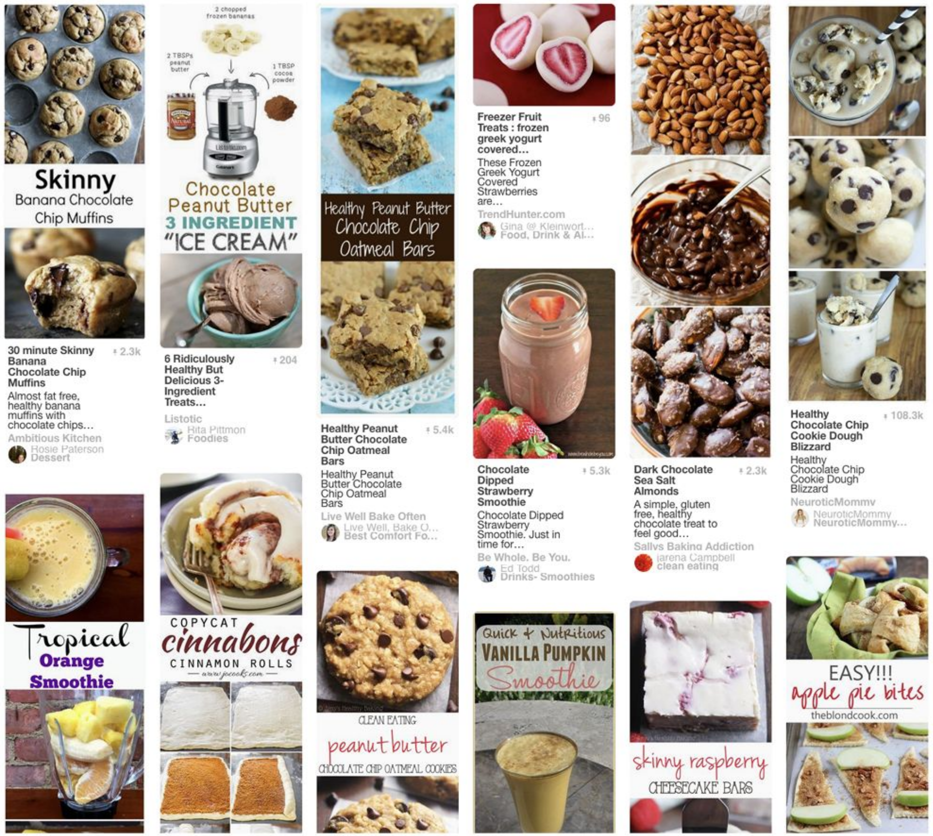
Recommendations:



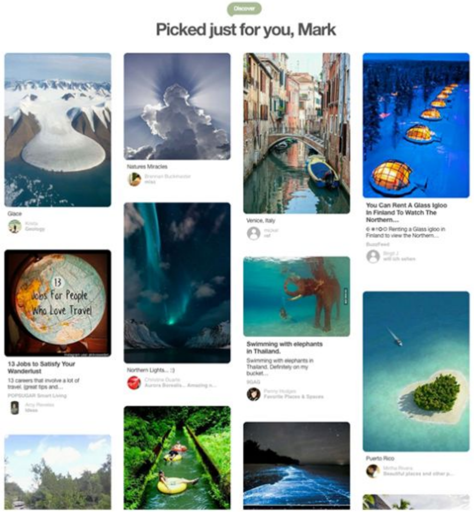
Query:



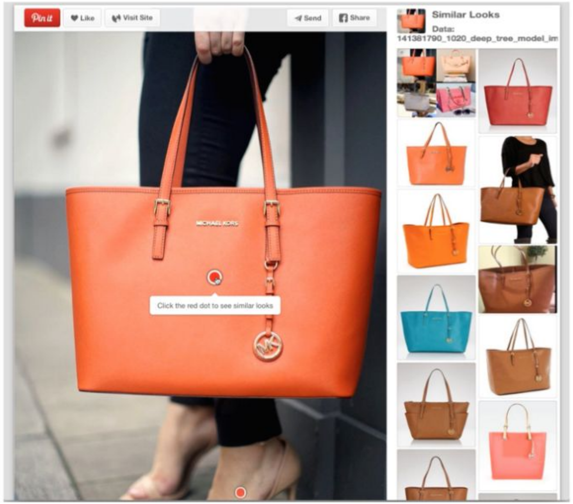
Recommendations:



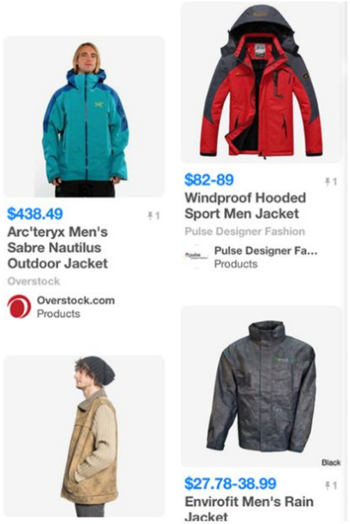
Universal similarity function has many applications



Homefeed
(endless feed of recommendations)



Related pins
(find most similar/related pins)



Ads and shopping
(use organic for the query and search the ads database)

Key Problem: Defining Similarity

1) Content-based: User and item features, in the form of images, text, categories, etc.

2) Graph-based: User-item interactions, in the form of graph/network structure
This is called collaborative filtering:

- For a given user X, find others who liked similar items
- Estimate what X will like based on what similar items others like

Key Problems

How do we define similarity:

(1) Gathering “known” similarities

- How to collect the data about what users like?

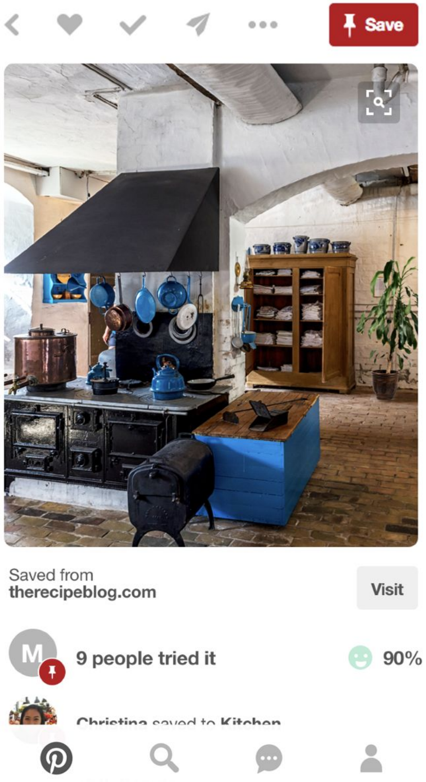
(2) Extrapolating unknown similarities from known ones

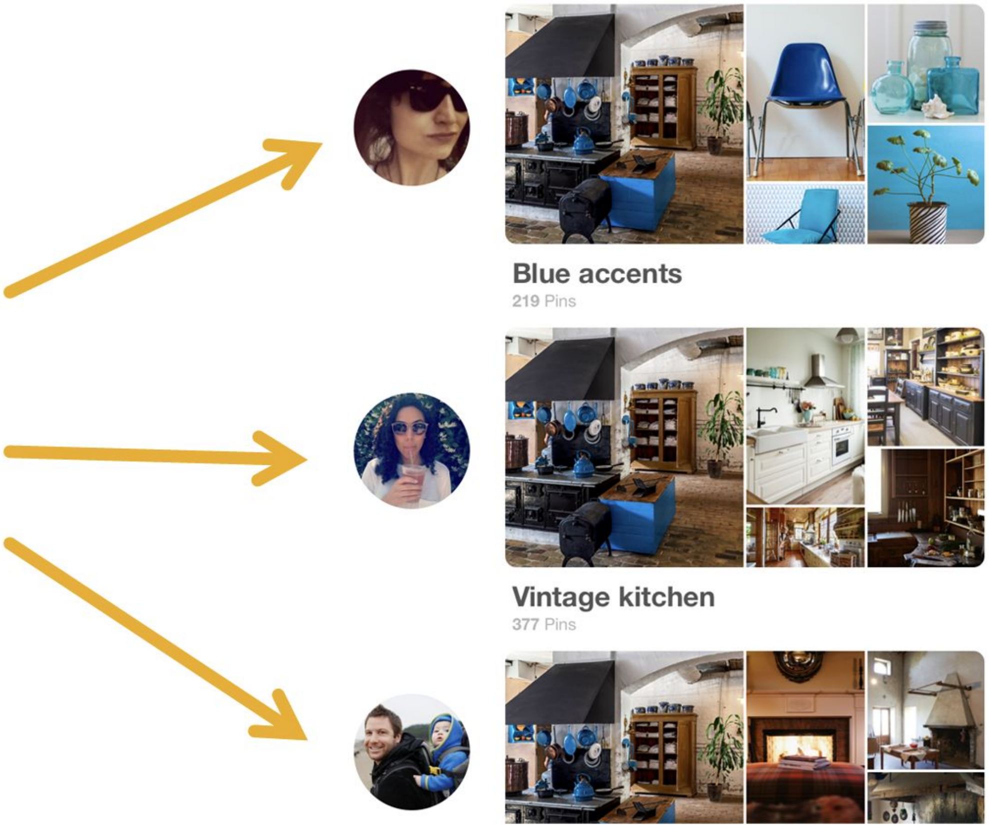
- Mainly interested in unknown similarities
- Not interested in knowing what you don’t like but what you like

(3) Evaluating methods

- How to measure success/performance of recommendation methods

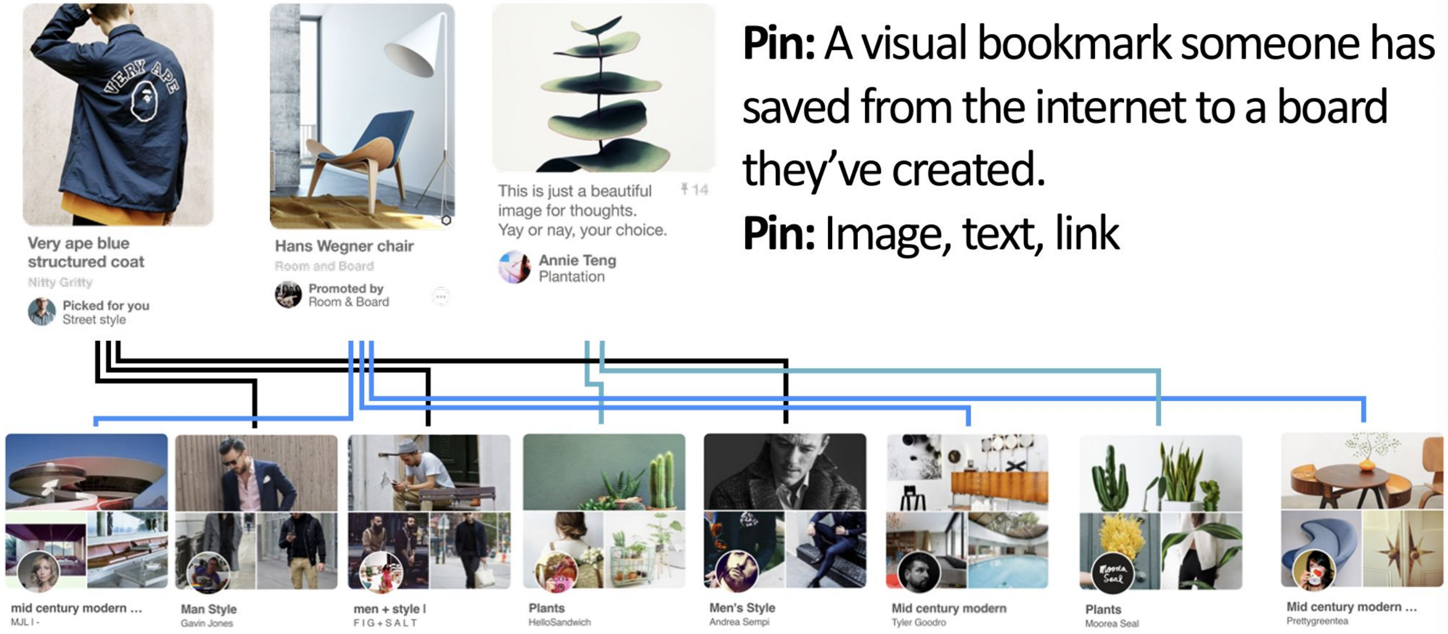
Pinterest





- 300M users
- 4+B pins, 2+B boards

Pinterest: Human curated collection of pins



Pin: A visual bookmark someone has saved from the internet to a board they've created.

Pin: Image, text, link

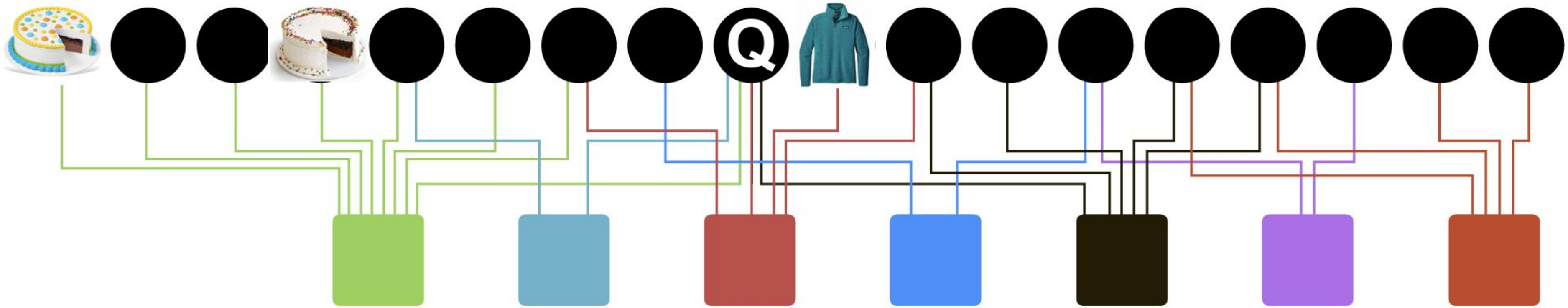
Board: A collection of ideas (pins having something in common)

Pinterest: Signals

Two sources of signal:

- Image and text of each pin

Graph is dynamic: Need to apply to new nodes without model retraining



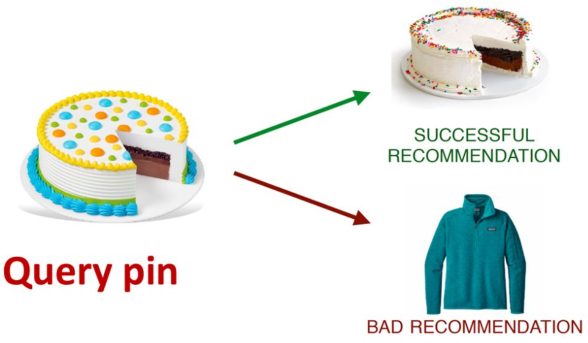
Recommendations via Embeddings

Goal: Learn embeddings for items

- Related Pins Query: Which pin to recommend when a user interacts with a pin v?
- Answer: Find the closest embedding (v) to v by nearest neighbor. Recommend it.

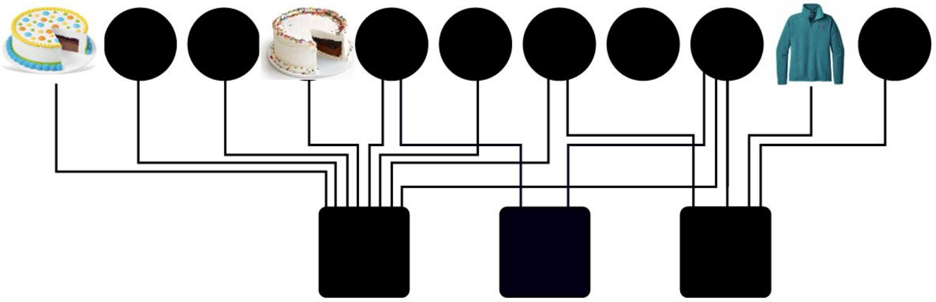
Pin Recommendation

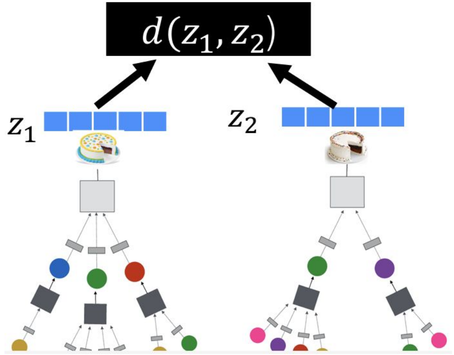
Task: Recommend related pins to users



Task: Learn node embeddings z_i such that

$$d(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$$

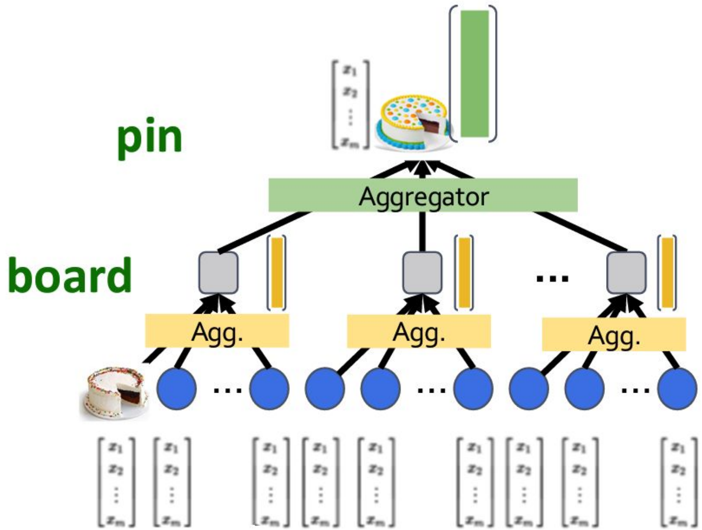




Predict whether two nodes in a graph are related

Approach:

- Pins have embeddings at each layer
- Layer-0 embedding of a node are its features: pin board
  - Text, image, …

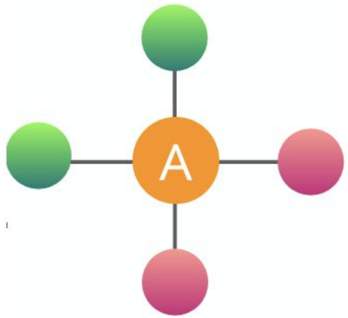


PinSage: graph convolutional network

Goal: Generate embeddings for nodes (e.g., pins) in the Pinterest graph containing billions of objects.

Key Idea: Borrow information from nearby nodes

- Example: Bed rail Pin might look like a garden fence, but fences and beds are rarely adjacent in the graph.



Pin embeddings are essential to many different tasks besides the “find related pins” task.

- Recommend related ads
- Home feed recommendation
- Cluster users by their interest

PinSage Pipeline

1) Collect billions of training pairs from logs.
   - Positive pair: Two pins that are consecutively saved into the same board within a time interval (1 hour)
   - Negative pair: A random pair of 2 pins: With high probability the pins are not on the same board

2) Train GNN to generate similar embeddings for training pairs

3) Inference: Generate embeddings for all pins

4) Nearest neighbor search in embedding space to make recommendations.

Training Objective Function

Train so that pins that are consecutively pinned have similar embeddings.

Max-margin loss:

$$\mathcal{L} = \sum_{(u,v)\in\boxed{\mathcal{D}}} \max(0, -\boxed{\mathbf{z}_u^\top \mathbf{z}_v} + \mathbf{z}_u^\top \boxed{\mathbf{z}_n} + \boxed{\Delta})$$

set of training pairs from user logs

"positive"/true training pair

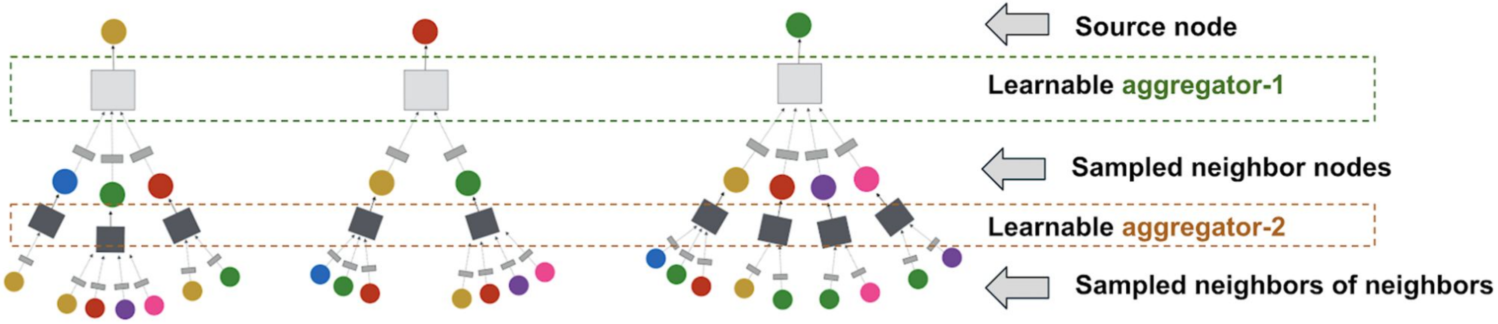
"negative" example

"margin" (i.e., how much larger positive pair similarity should be compared to negative)

Innovations

On-the-fly graph convolutions

- Sample the neighborhood around a node and dynamically construct a computation graph
- Minibatch of neighborhoods:



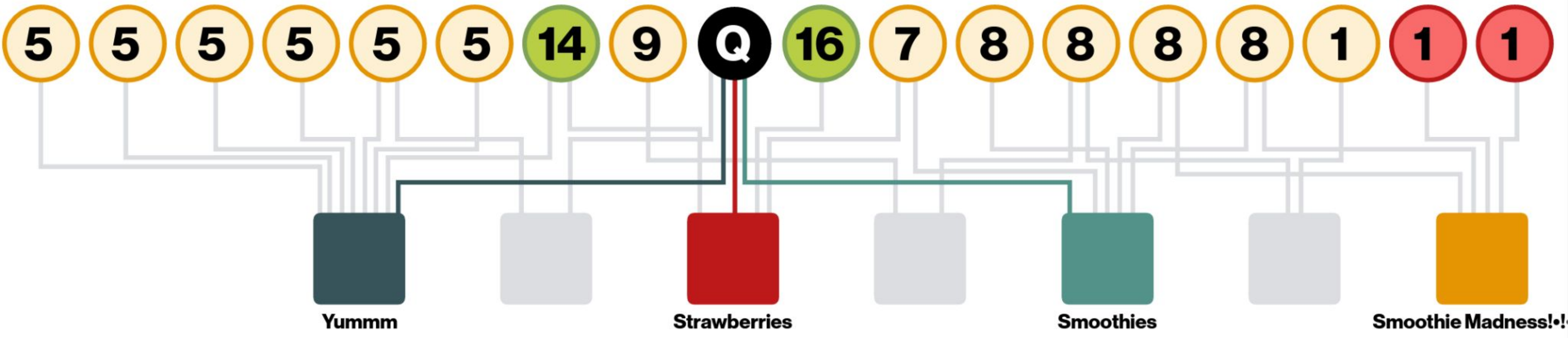
Innovations

Selecting neighbors via random walks

- Performing aggregation on all neighbors is infeasible.
- How to select the set of neighbors of a node to convolve over?
- Personalized PageRank!
- Define Importance pooling: Define importance-based neighborhoods by simulating random walks and selecting the neighbors with the highest visit counts

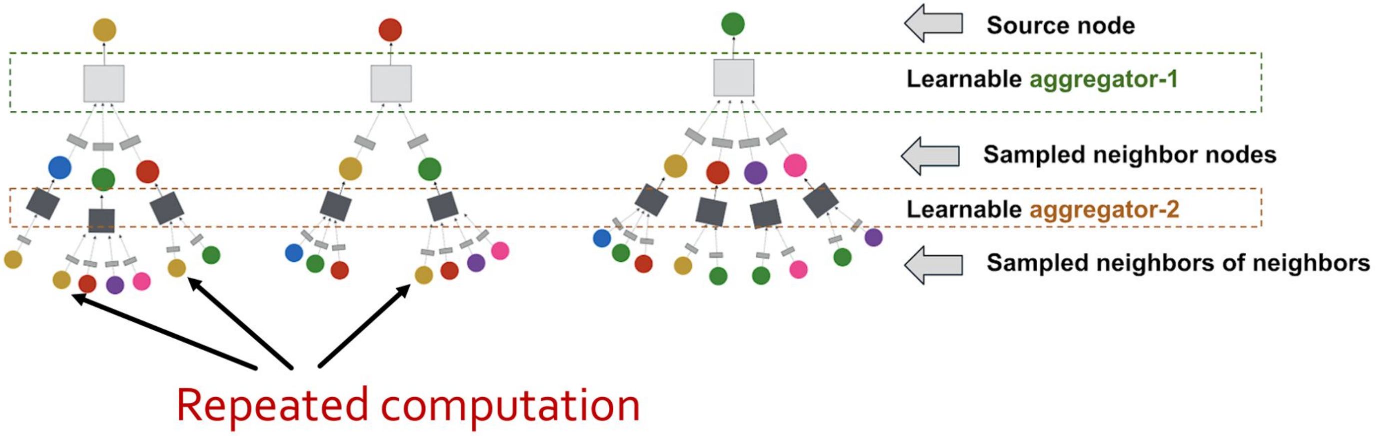
Innovations

- Proximity to query node
- Importance pooling
  - Choose nodes with top K visit counts
  - Pool over the chosen nodes
  - The chosen nodes are not necessarily neighbors



Innovations

- Efficient MapReduce inference
- Problem: Many repeated computation if using localized graph convolution at inference step
- Need to avoid repeated computation



Innovations

- Goal: Identify target pin among 3B pins
- Issue: Need to learn with resolution of 100 vs. 3B
- Massive size: 3 billion nodes, 20 billion edges
- Idea: Use harder and harder negative samples

$$\mathcal{L} = \sum_{(u,v)\in \boxed{\mathcal{D}}} \max(0, -\mathbf{z}_u^\top \boxed{\mathbf{z}_v} + \mathbf{z}_u^\top \boxed{\mathbf{z}_n} + \boxed{\Delta})$$

set of training pairs from logs

"positive"/true example

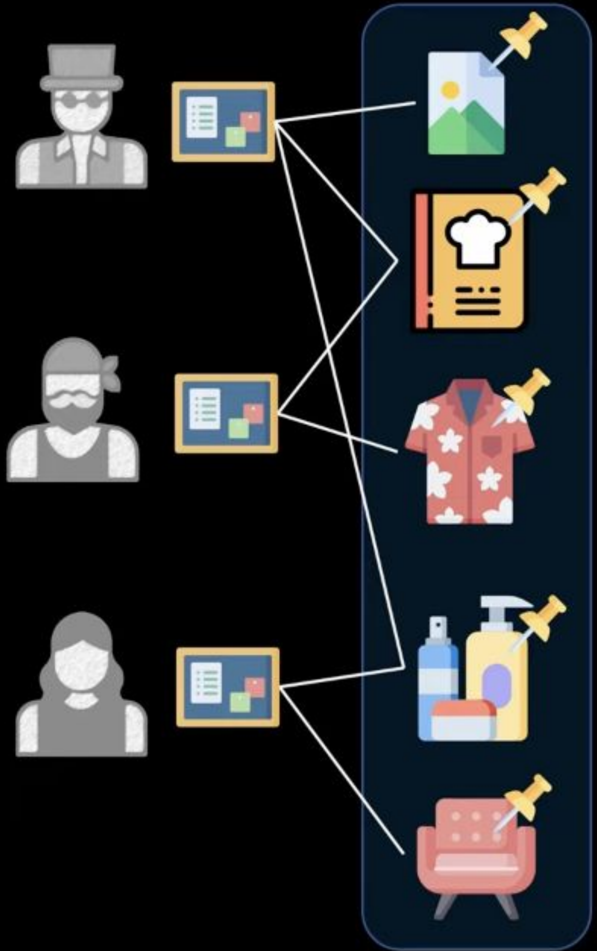
negative examples

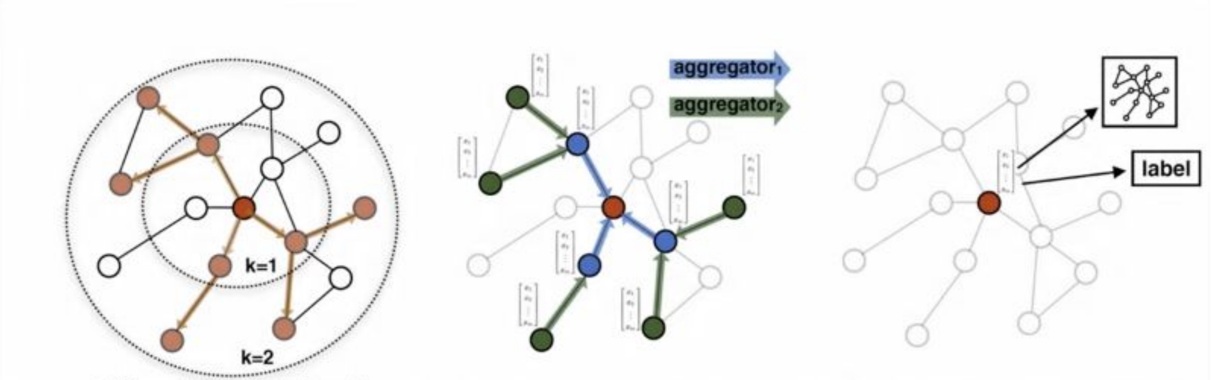
"margin" (i.e., how much larger positive pair similarity should be compared to negative)

Positive Example

Hard Negative

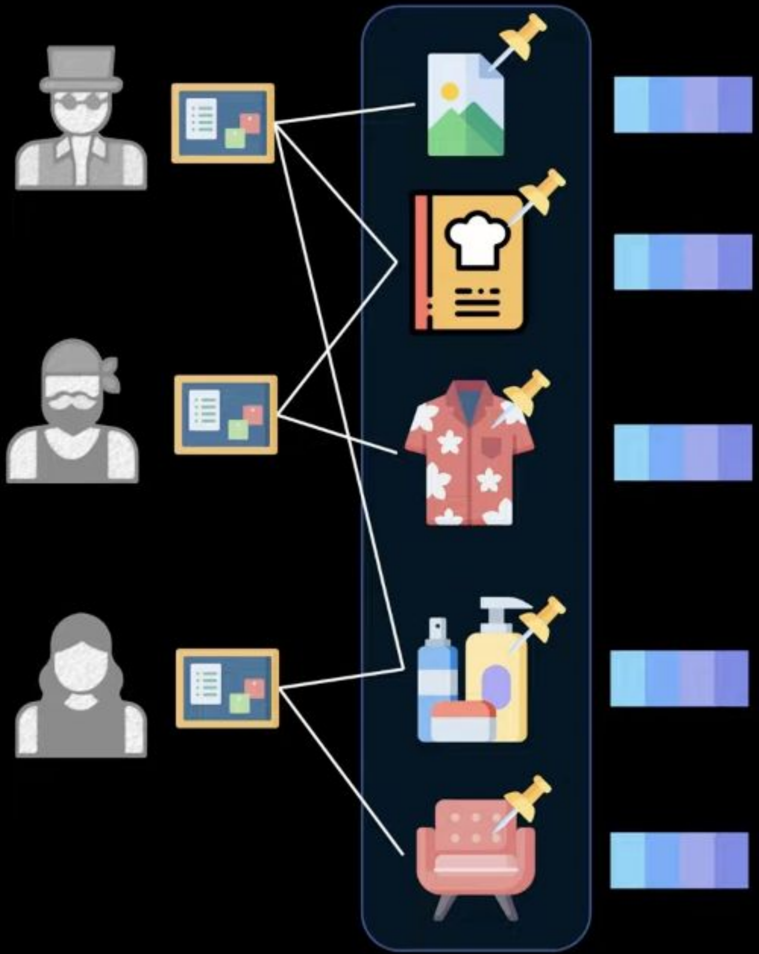
Force model to learn subtle distinctions between pins

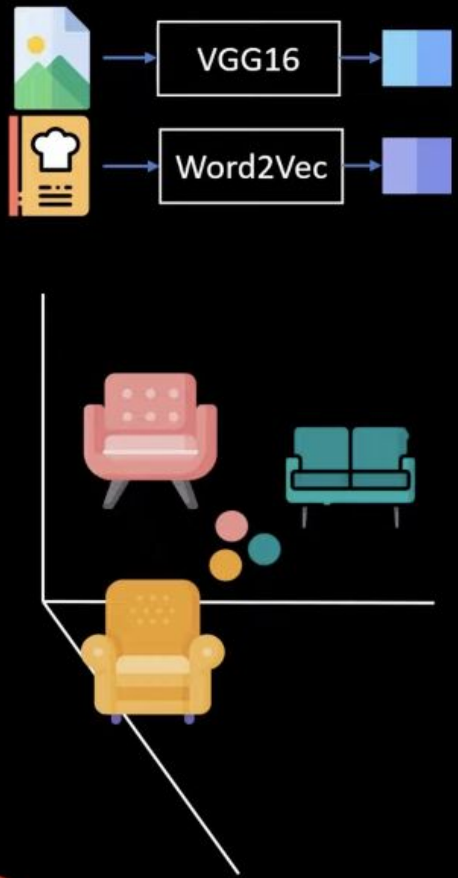




1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

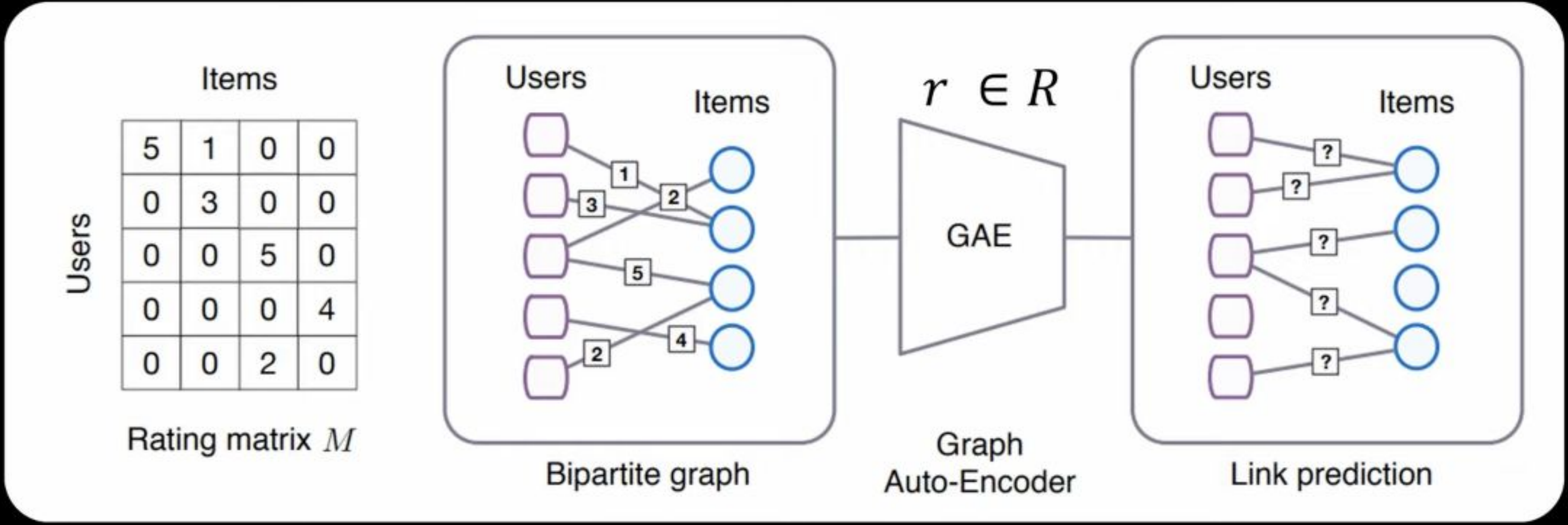
PinSAGE
Ying et al., 2018





PinSAGE

Ying et al., 2018

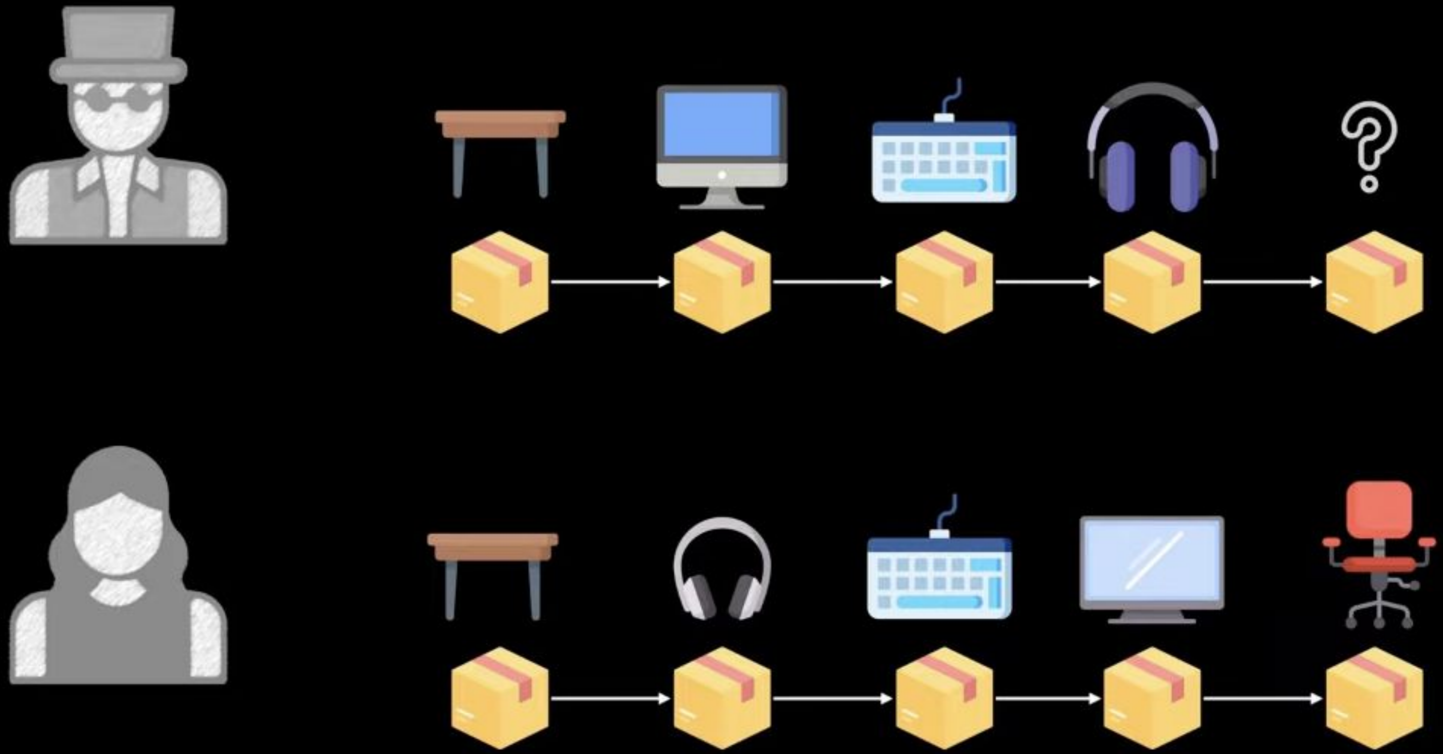


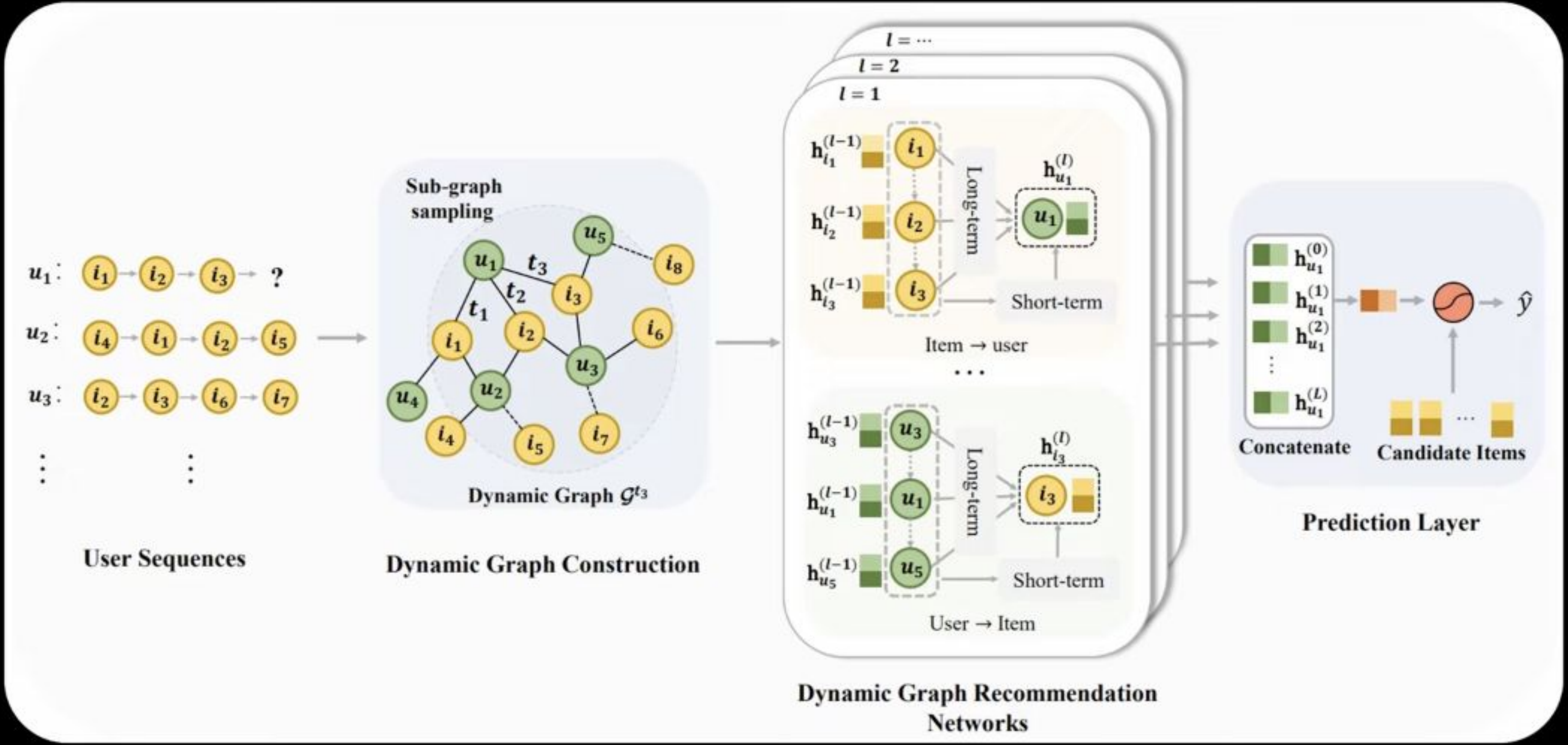
$$p\left(\widetilde{M}_{ij} = r\right) = \frac{e^{u_i^T Q_r v_j}}{\sum_{s \in R} e^{u_i^T Q_s v_j}}$$

5		1	1	1		2	3
5	4		2	1			
1				3			3
1			4	2	3	1	4
1		3	2	2	5		

Graph Convolutional Matrix Completion

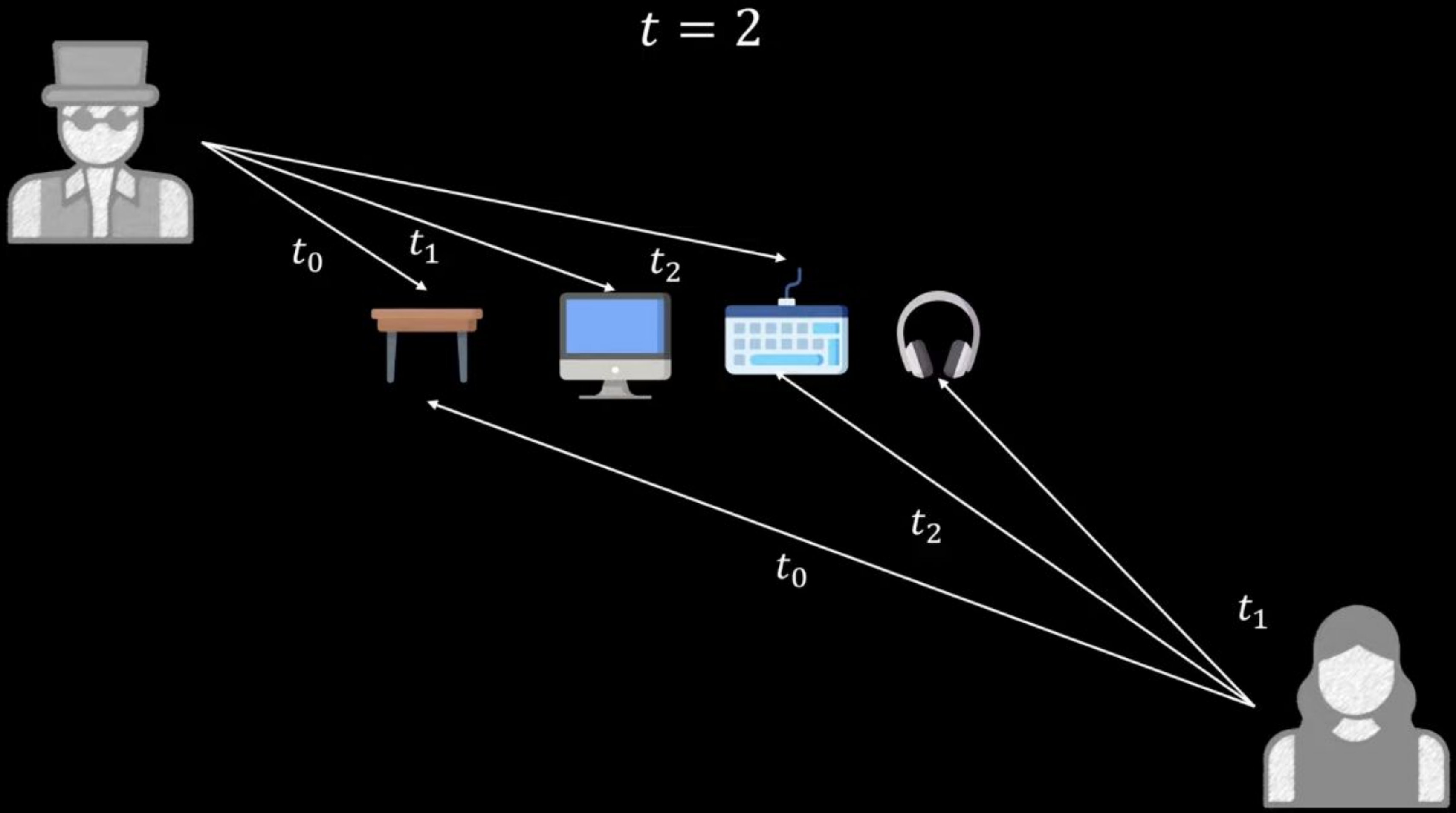
Van den Berg, Kipf and Welling, 2017

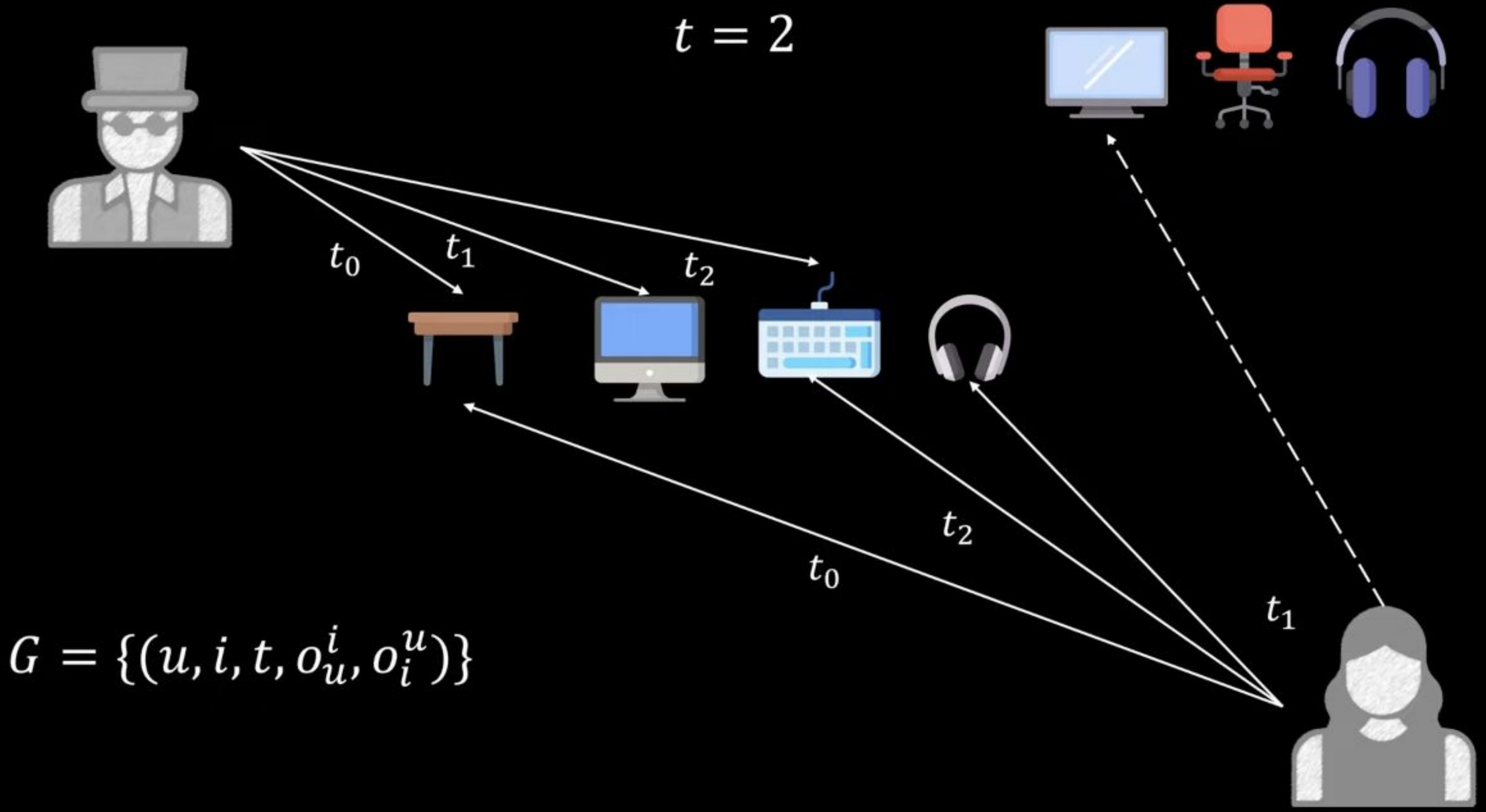




Dynamic GNN for Sequential Recommendation

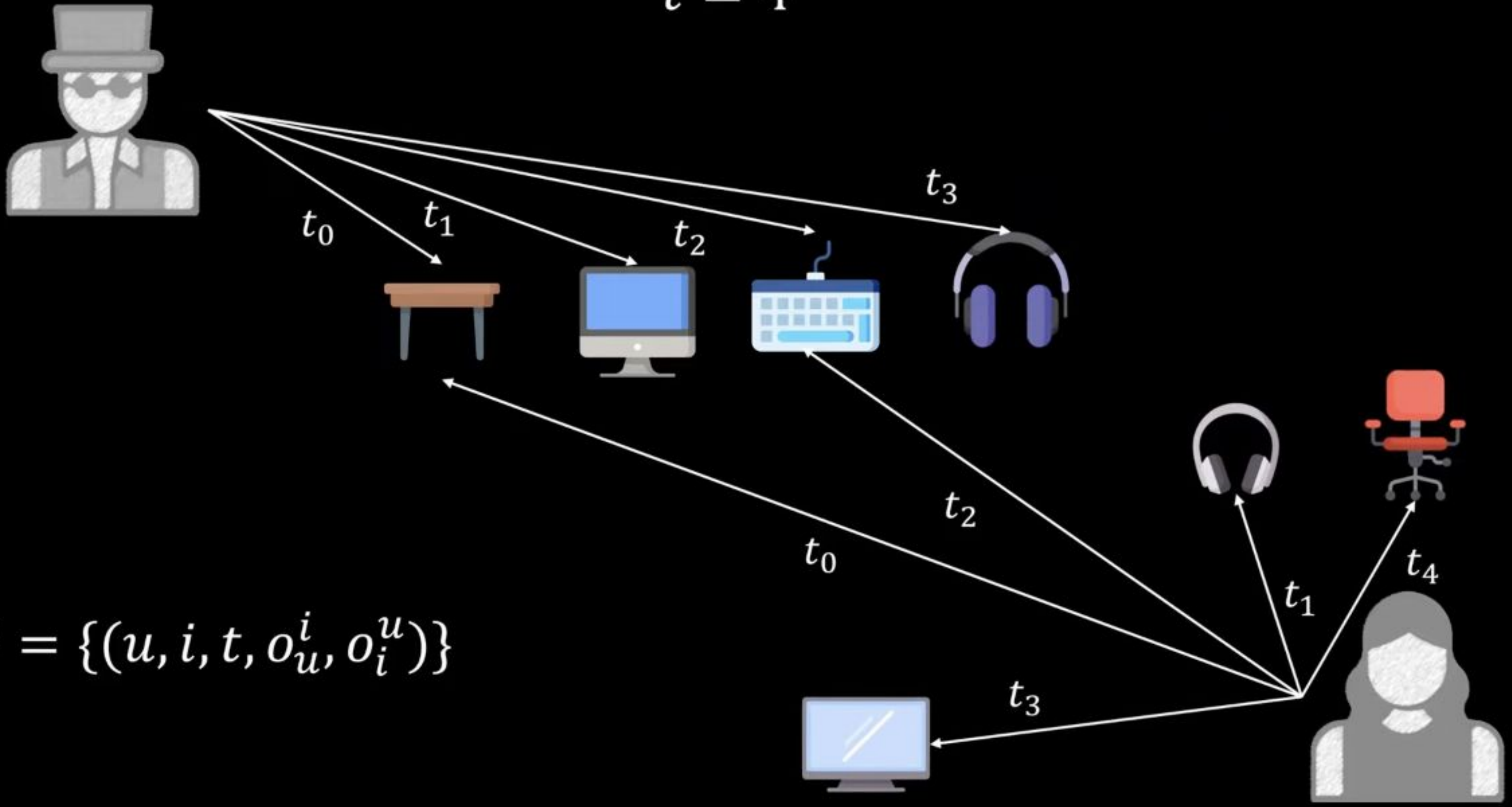
Zhang et al., 2021


$t = 2$
t_0
t_1
t_2
t_2
t_0
t_1

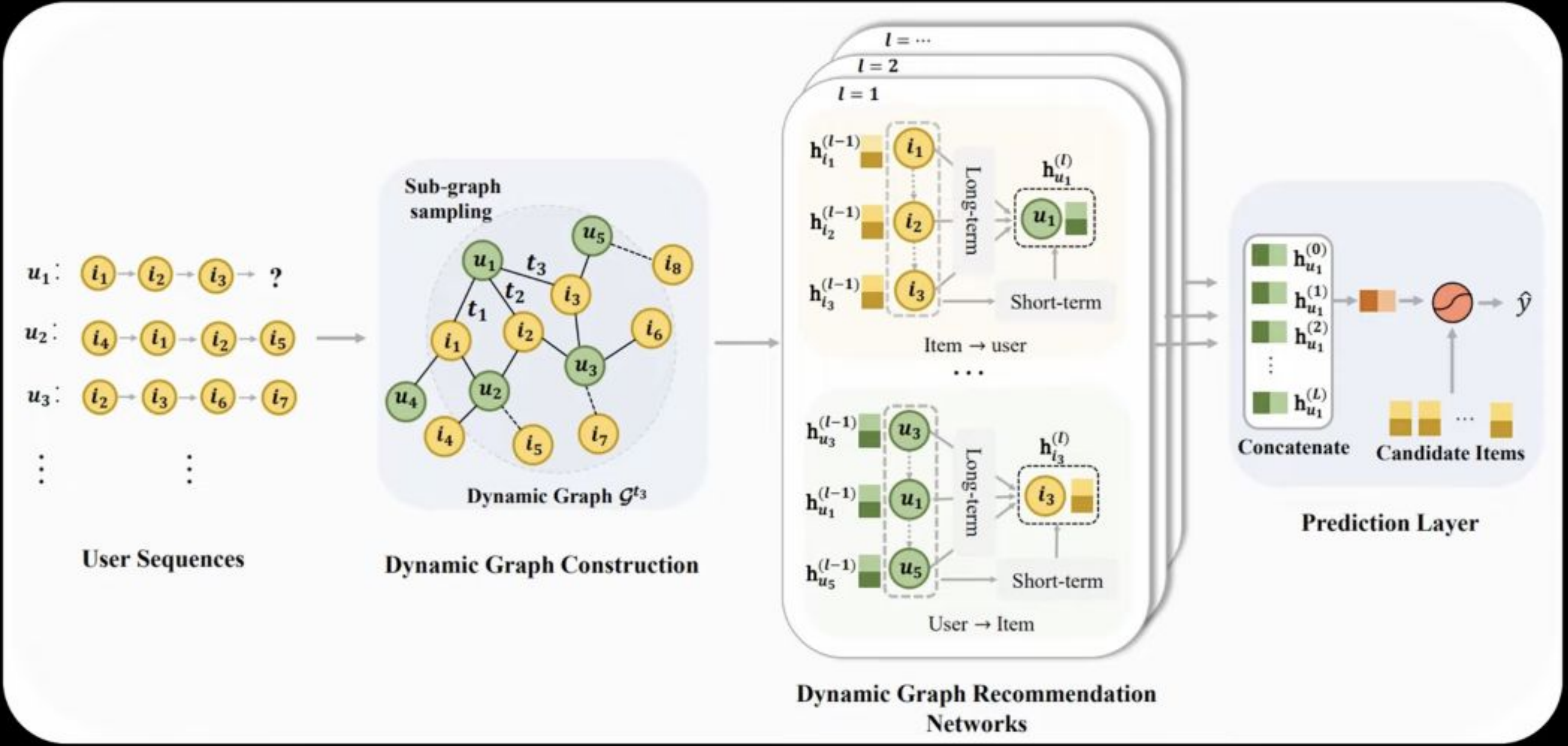


$t = 2$
t_0
t_1
t_2
t_2
t_0
t_1
$G = \{(u, i, t, o_u^i, o_i^u)\}$

$t = 4$

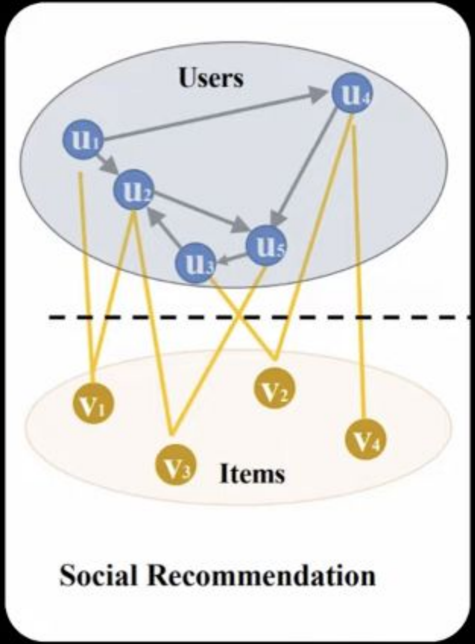


$$G = \{(u, i, t, o_u^i, o_i^u)\}$$

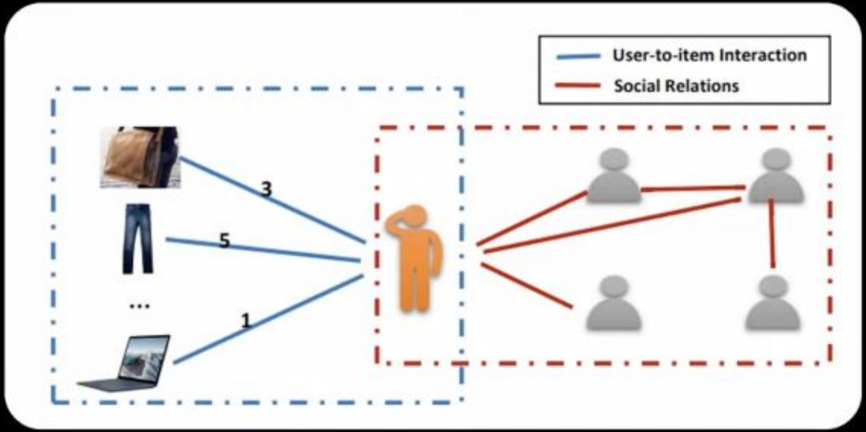


Dynamic GNN for Sequential Recommendation

Zhang et al., 2021



DiffNet++
Wu et al., 2020



GraphRec
Fan et al., 2019