Recommender Forest for Efficient Retrieval

Chao Feng[1]*, Wuchao Li[1]*, Defu Lian[1], Zheng Liu[2], Enhong Chen[1]

[1]School of Computer Science and Technology
University of Science and Technology of China, Hefei, China
[2]Microsoft Research Asia, Beijing, China
{chaofeng,liwuchao}@mail.ustc.edu.cn
{liandefu,cheneh}@ustc.edu.cn, zhengliu@microsoft.com

中国科学技术大学
University of Science and Technology of China

Microsoft

Indexes in recommender system

- Recommender system needs to select the top-n items for users from a massive-scale item set.
- For the sake of efficient recommendation, RS usually calls for the collaboration of representation learning and Approximate Nearest Neighbour search (ANNs) index.
- Two kinds of indexes: Independent training & Joint training

Independent training index

- Process
  1. Users and items are represented by embeddings in the same latent space.
  2. The item embeddings are organized with a specific ANNs index, like SCANN and HNSW.

- Limitation
  - The representation model is independently learned and can be incompatible with the ANNs index

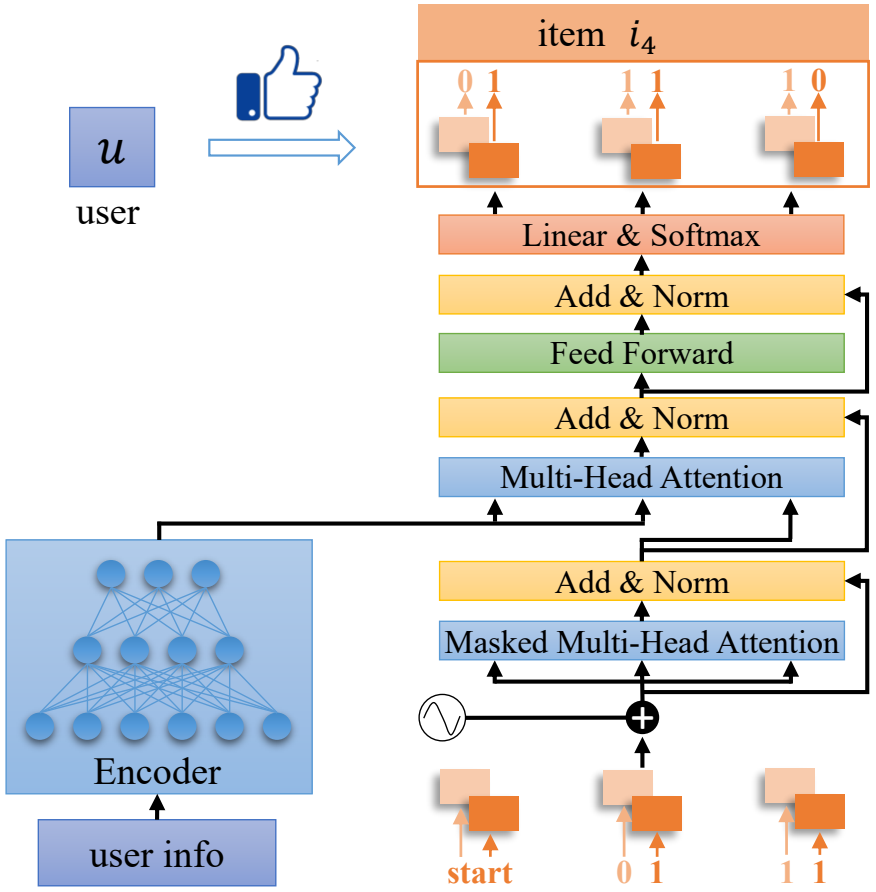

Joint training index: TDM & JTM

- The item set is organized with binary tree structure:
  - internal node: cluster center.
  - leaf node: item.
- A preference model is learned to route from the root to the leaf nodes for the top-n recommendation results.
- Achieve empirical gains over the conventional two-stage methods.

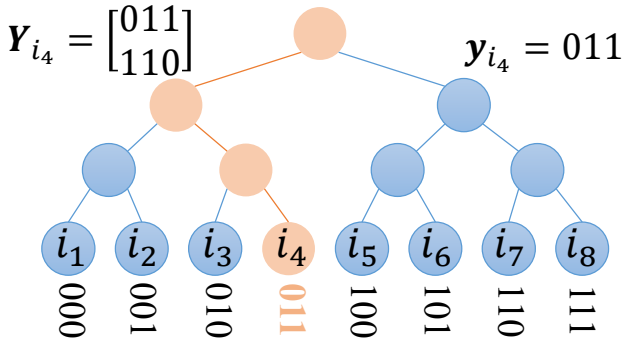
Joint training index: TDM & JTM

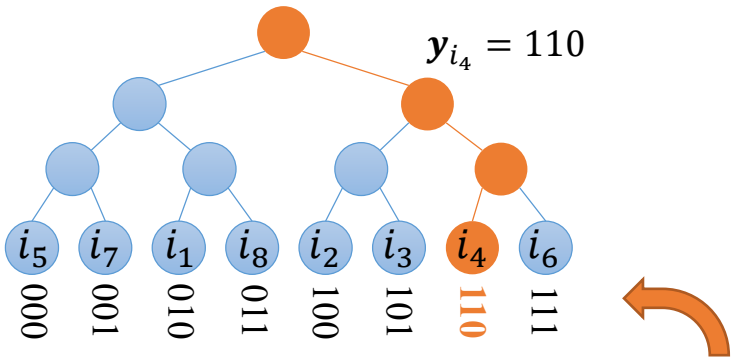
- Limitation
  - It is challenging to route to items located around partition boundaries.
  - A routing decision is made without consideration of the routing trajectory.
  - Memory-consuming, given that the number of internal nodes is at the same magnitude as the leaf nodes.

Introducing RecForest



Each item is represented by a sequence in each tree





The sequence length is $\lceil \log_k N \rceil$

Construction of Tree and Forest

- Tree construction (Blanced-Kmeans)
  - Pre-train item embeddings: DIN or BPR
  - Randomly sample $K^H - N$ items (H is the height of the tree) from the entire item set.
  - For each cluster, the included items are evenly partitioned via Kmeans w.r.t. their embedding similarity.
- Forest construction
  - Different trees are naturally diversified due to the inherent randomness.

Experiments

Comparison with Baselines

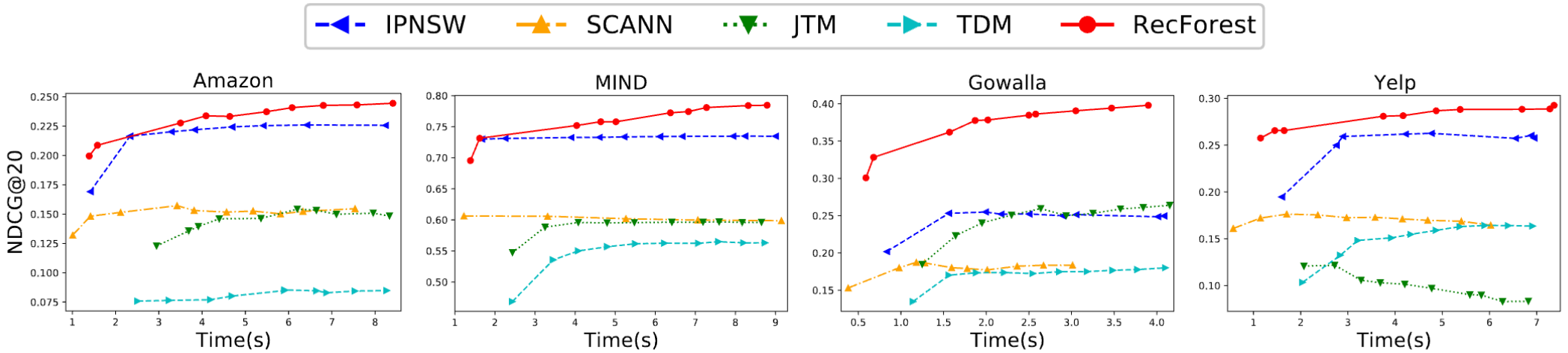
Method	NDCG@20	NDCG@40	Memory	Time	NDCG@20	NDCG@40	Memory	Time
	Movie				**Amazon**			
DIN	**0.5440**	0.5473	-	193.87	**0.2766**	**0.3039**	-	492.64
YoutubeDNN	0.5329	**0.5484**	-	**29.38**	0.2195	0.2491	-	**120.91**
JTM	0.5149	0.5075	10.80	12.05	0.1533	0.1683	75.99	6.64
TDM	0.4684	0.4651	10.80	9.33	0.0856	0.0949	75.99	6.61
SCANN	0.4665	0.4695	3.64	18.64	0.1529	0.1780	14.66	4.48
IPNSW	0.5330	0.5486	10.08	15.52	0.2255	0.2548	66.46	10.28
RecForest	**0.5580**	**0.5682**	**3.21**	**8.33**	**0.2339**	**0.2576**	**7.32**	**3.79**
Method	**Gowalla**				**Tmall**			
DIN	**0.2798**	**0.3095**	-	186.41	**0.2275**	**0.2491**	-	4057.69
YoutubeDNN	0.2312	0.2637	-	**53.55**	0.1736	0.1975	-	**1086.75**
JTM	0.2595	0.2484	77.56	2.64	0.0749	0.0849	151.19	30.11
TDM	0.1723	0.1775	77.56	2.55	0.0257	0.0272	151.19	29.42
SCANN	0.1839	0.2083	15.48	1.86	0.1105	0.1226	28.10	20.88
IPNSW	0.2464	0.2805	70.39	4.73	0.1696	0.1902	132.72	52.90
RecForest	**0.3783**	**0.3963**	**7.39**	**1.82**	**0.2059**	**0.2261**	**9.29**	**18.88**

More accurate

Lighter

Faster

Comparison with Baselines



- RecForest strikes the best balance between query time and retrieval accuracy.
- With the increase of beam size, the accuracy of RecForest can improve more significantly than baselines.

Why is RecForest better ?

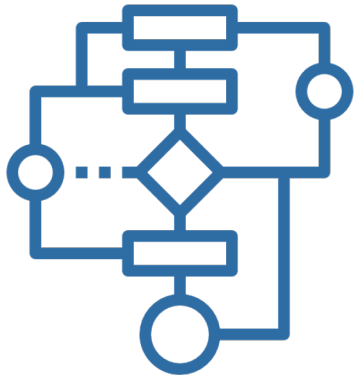
- The retrieval of the near-boundary can be challenging
  - multiple K-ary trees

- Not considering the trajectory history
  - using Transformer Decoder

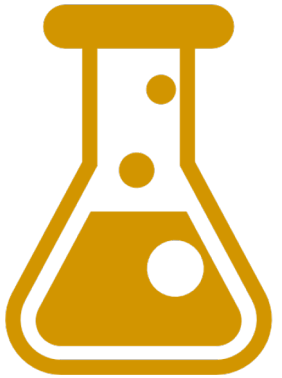
- Memory-consuming:
  - There are mere K vectors (corresponding to the K different branches) in each K-ary tree.

More content in the paper

Complexity analysis



Algorithm flow



More experiments

Thanks