Recommender systems

Multi-stage recommender systems

Lecture 5
Fall 2023

Anna Volodkevich

Contents

Recap: recommendation as a general machine learning task on tabular data

Feature vector $\mathbf{x}$ | Target y

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

Conditions:

- We want to use all available features of different types and nature
- Some features depend on user-item pair and thus should be calculated online
- Negatives (not relevant) are often not available
- It is impractical to score all user-item pairs with tabular models
- Production models should be fast
- Possible solution: hand-crafted negative generation and **multi-stage recommender systems**

Multi-stage recommenders: Reasoning and Pipeline

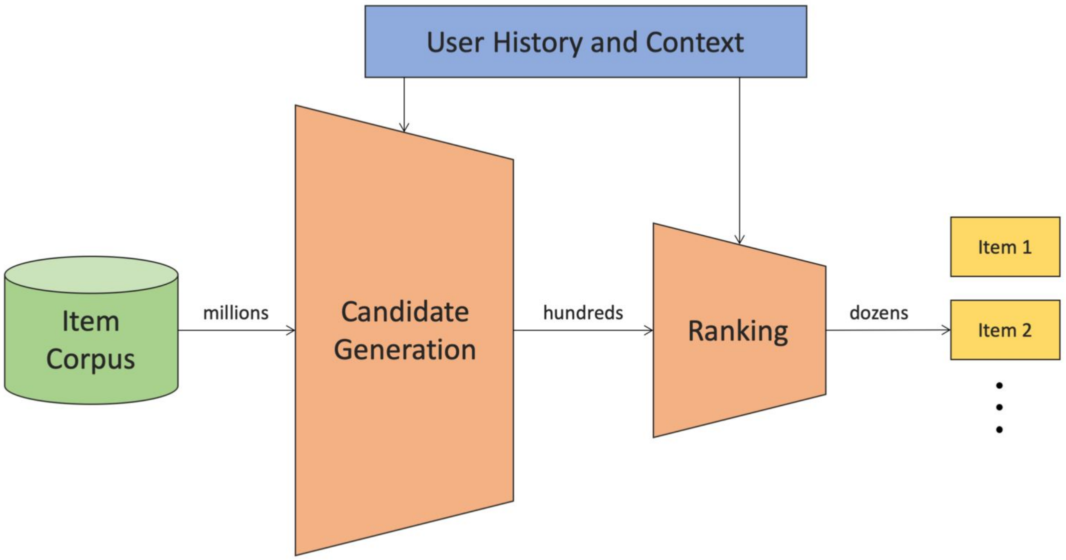


Image source

- collaborative, content-based, non-personalized and the other models has their own advantages and disadvantages
- developers want to **combine recommendations** from different sources
- let's generate a small set of candidates for each user with light and fast models and **rerank** them with the compex one **to get top-k**

General multi-stage recommendation pipeline from Nvidia

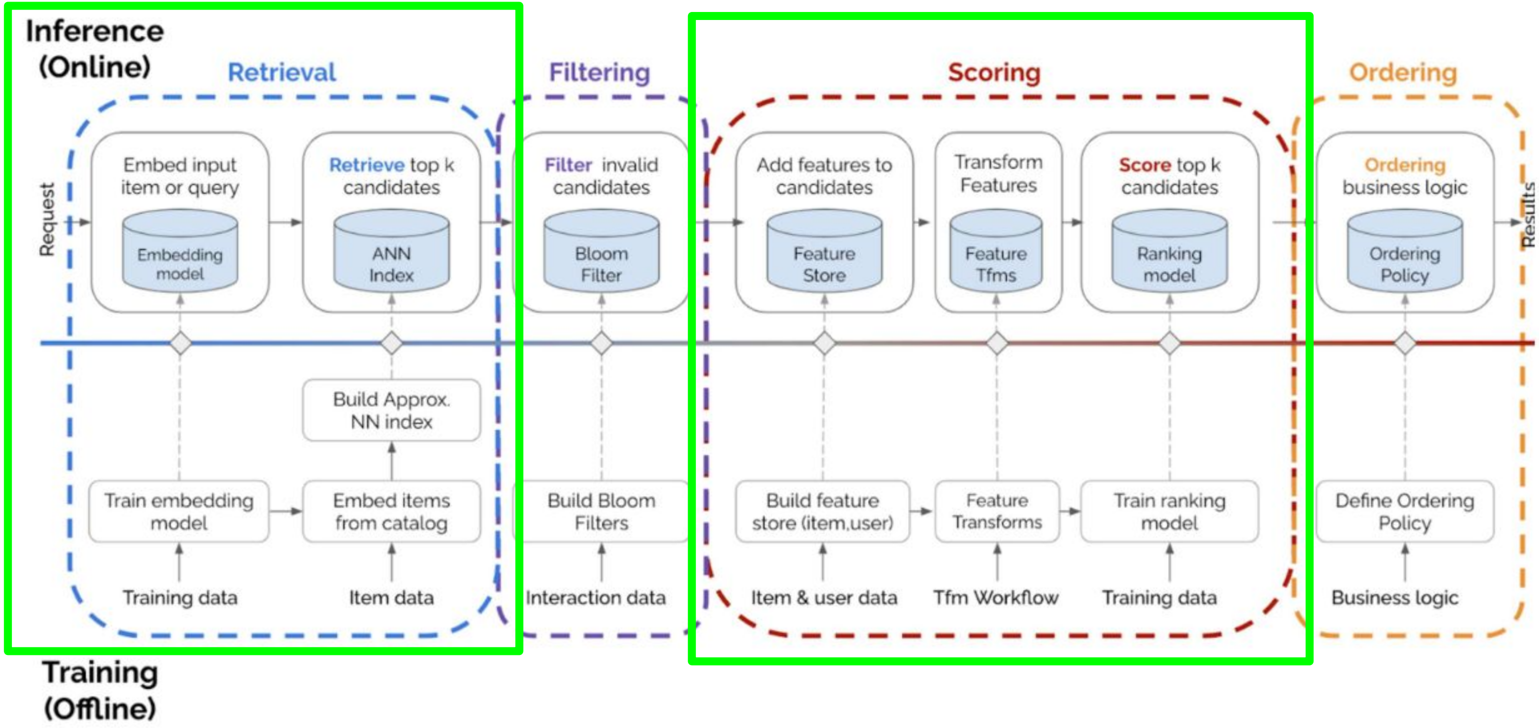


Figure 1: An overview of Four-stage Recommender Systems

Image source

Two-stage model candidate sources

content-based

non-personalized

collaborative

business rules

precision + beyond-accuracy

reranker

Tunable parameters / choices:

- number of candidates from each model
- first-stage model types

Two-stage model feature sources

Item/user features
Context features

First-level outputs

- scores
- ranks
- vectors and biases

User/ item statistics

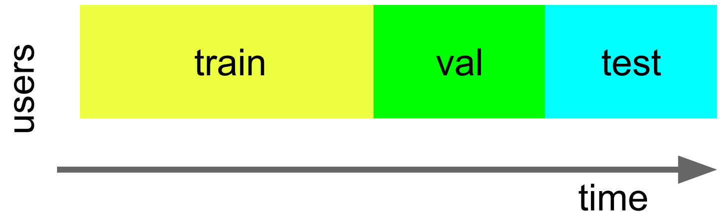
- popularity / ctr over time
- popularity tendency
- time features
- target encoding

Co-occurrences and pair features

- number of interactions with an item/category in the past
- mean/popularity conditioned on feature value (popularity of an item in an age group)
- time features conditioned on feature value (last time bought an item from category)
- co-occurance with the other items from user history

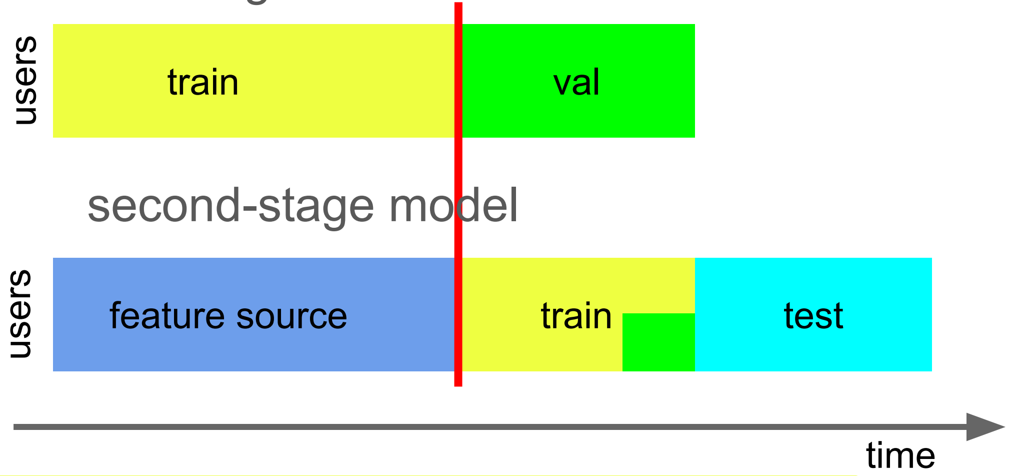
Data split

Simple recommender



- Want to capture the most recent patterns
- Need to avoid overfitting of the second-stage model

Multi-stage recommender



Tunable parameters / choices:
- split ratio
- retraining (attention to possibly unstable vectors and scores)

Second-stage pipeline

- Combine candidates from all first-stage models
- Get scores and ranks from all first-stage models
- Add negative examples and target (usually 0 and 1)
- Add features
- Train the model **or**
- Make prediction, get top-k and calculate metric

Tunable parameters / choices:

- source of negative items: first stage, random, ∝ popularity
- second-stage model type, task and loss function

Deep learning as a promising alternative

For the past few years most published research on recommendation algorithms has been based on deep learning (DL) methods. Following common research practices in our field, these works usually demonstrate that a **new DL method is outperforming other models not based on deep learning in offline experiments**. This almost consistent success of DL based models is however not observed in recommendation-related machine learning competitions like the challenges that are held with the yearly ACM RecSys conference. **Instead the winning solutions mostly consist of substantial feature engineering efforts and the use of gradient boosting or ensemble techniques.**

Why Are Deep Learning Models Not Consistently Winning Recommender Systems Competitions Yet?: A Position Paper RecSys'20

Deep learning models

- can utilize all previously discussed gradient-based models
- can learn embeddings and work with all data types end-to-end (no separate models retraining)
- provide simple sequential data modelling

Next lecture: DL approaches in Recommender Systems

Additional sources

Multi-stage models

Your second recsys: Multi-stage models

Rekko Challenge: second place solution

RecSys Challenge 2018: third place solution

Dzen Recommendation system

OKKO Recommendation system

Other interesting papers and videos

Meetup series “Дзен-митап” (youtube)

Yandex Music Recommendation System