Neural Recommender Systems

Aleksandr “Sasha” Petrov
University of Glasgow

15th European Summer School on Information Retrieval
Amsterdam 2024

About me

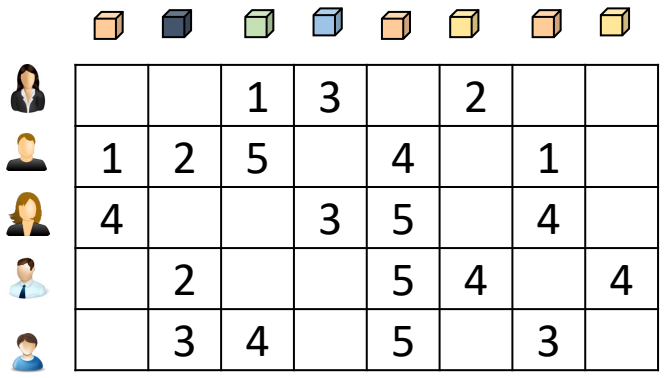
- PhD Student, University of Glasgow
- Focus: Effective and Efficient Transformer models for Recommender Systems
- >10 years of industrial experience
- Ex. Sr. Software Engineer @ Amazon
- Ex. CTO @ E-Contenta

Plan for today

- From Matrix Factorisations to Neural Networks
- Variational Autoencoder
- Sequential Recommendation
- Transformer-based models
- Generative recommender systems for beyond-accuracy goals
- Large scale sequential recommendation
- LLMs for RecSys

Before Neural Networks we had Matrix factorisation…

Matrix Factorisation (MF) Methods*



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

≈

0.641	-1.370	-1.136
2.026	0.131	-1.172
0.796	1.804	-1.000
1.516	1.347	-0.933
0.851	1.167	-1.691

User embeddings

×

0.074	0.103	1.950	1.595	1.102	1.388	0.171	1.013
1.901	0.336	0.846	-0.003	1.546	0.398	1.954	1.090
-0.513	-1.490	-0.800	-1.737	-1.335	-1.457	-0.339	-1.066

Item embeddings (transposed)

=

-2.0	1.3	1.0	3.0	0.1	2.0	-2.2	0.4
1.0	2.0	5.0	5.3	4.0	4.6	1.0	3.4
4.0	2.2	3.9	3.0	5.0	3.3	4.0	3.8
3.2	2.0	4.8	4.0	5.0	4.0	3.2	4.0
3.1	3.0	4.0	4.3	5.0	4.1	3.0	3.9

Recovered

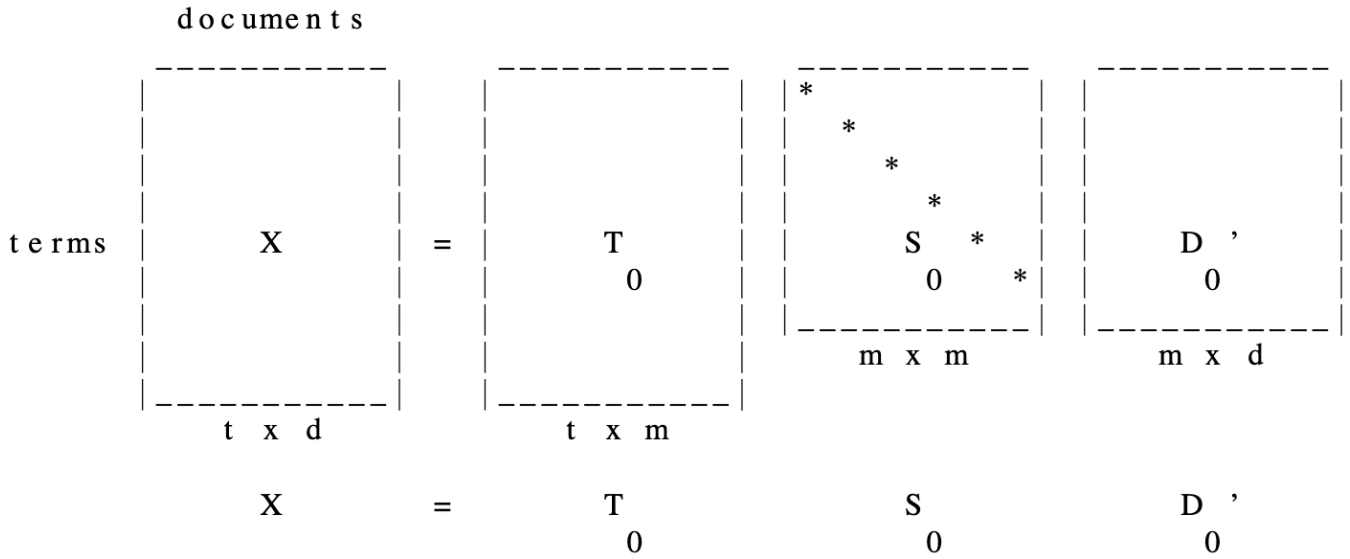
Predicted

Loss Functions:
–Mean Square Error on recovery of the training ratings
- BPR (for implicit data): pairwise loss between the score of a positive item for a given user, and a randomly sampled negative item

*Koren and Bell, Advances in Collaborative Filtering, in Recommender Systems Handbook, 2010

MF Models Were also Popular in IR

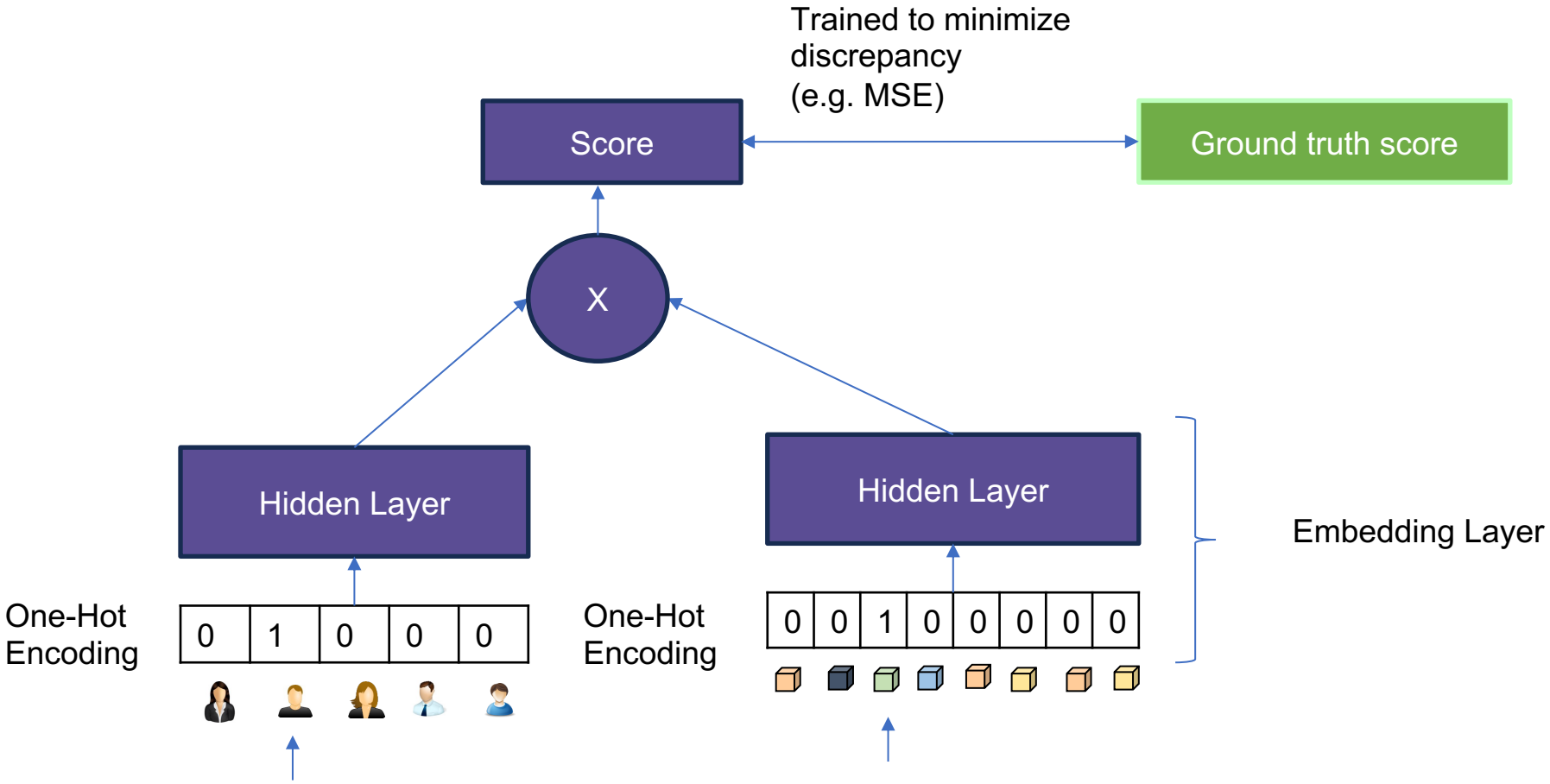
E.g. Latent Semantic Indexing (LSI):



... But now most of IR is Deep Learning.

* Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), pp.391-407.

Matrix Factorisation is a Neural Network!



It is the same MF, just redrawn as a Neural Network.

- Can be trained as a Neural Network, e. g. Using Back Propagation
- Can utilize modern Neural Network Libraries, e. g. PyTorch and TensorFlow

MF In PyTorch

Very simple model*:

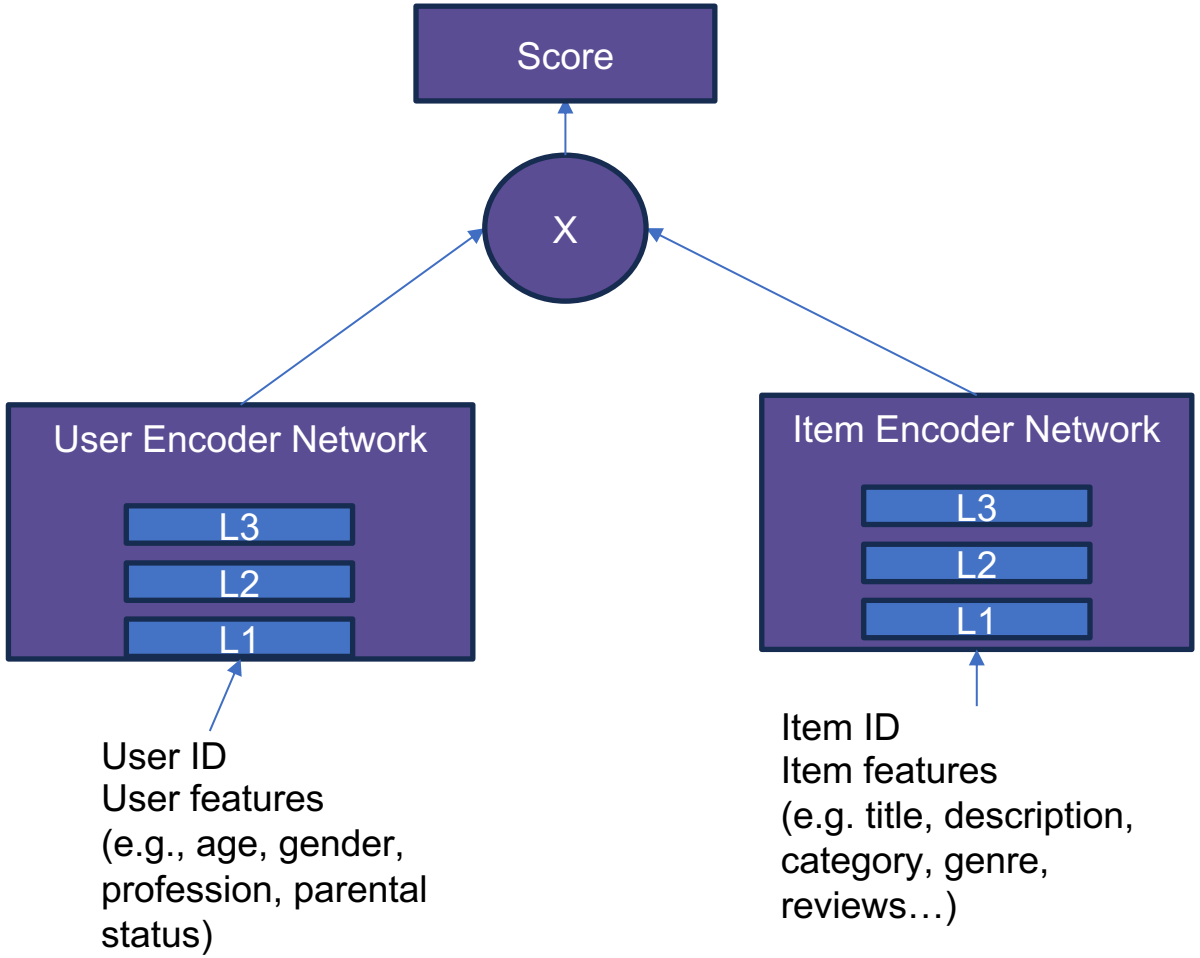
```
import torch

class MFModel(torch.nn.Module):
    def __init__(self, n_users, n_items):
        super().__init__()
        self.user_embeddings = torch.nn.Embedding(n_users, 128)
        self.item_embeddings = torch.nn.Embedding(n_items, 128)

    def forward(self, user_id, item_id):
        user_embedding = self.user_embeddings(user_id)
        item_embedding = self.item_embeddings(item_id)
        return torch.dot(user_embedding, item_embedding)
```

Standard Training Loop

```
n_samples = 10
users = torch.tensor([0,0,0,1,1,1,2,2,3,3])
items = torch.tensor([1,2,3,0,1,0,2,3,2,1])
ratings = torch.tensor([2,3,0,4,3,2,3,1,3,0],
dtype=torch.float32)

optimiser = torch.optim.Adam(mf.parameters())

for epoch in range(10):
    loss_sum = 0.0
    for i in range(n_samples):
        score = mf(users[i], items[i])
        loss = torch.nn.functional.mse_loss(score, ratings[i])
        loss.backward()
        optimiser.step()
        loss_sum += loss.item()
    print("Train MSE Loss: ", loss_sum / n_samples)
```

* This is a *very* simplified example. Do not use in production.

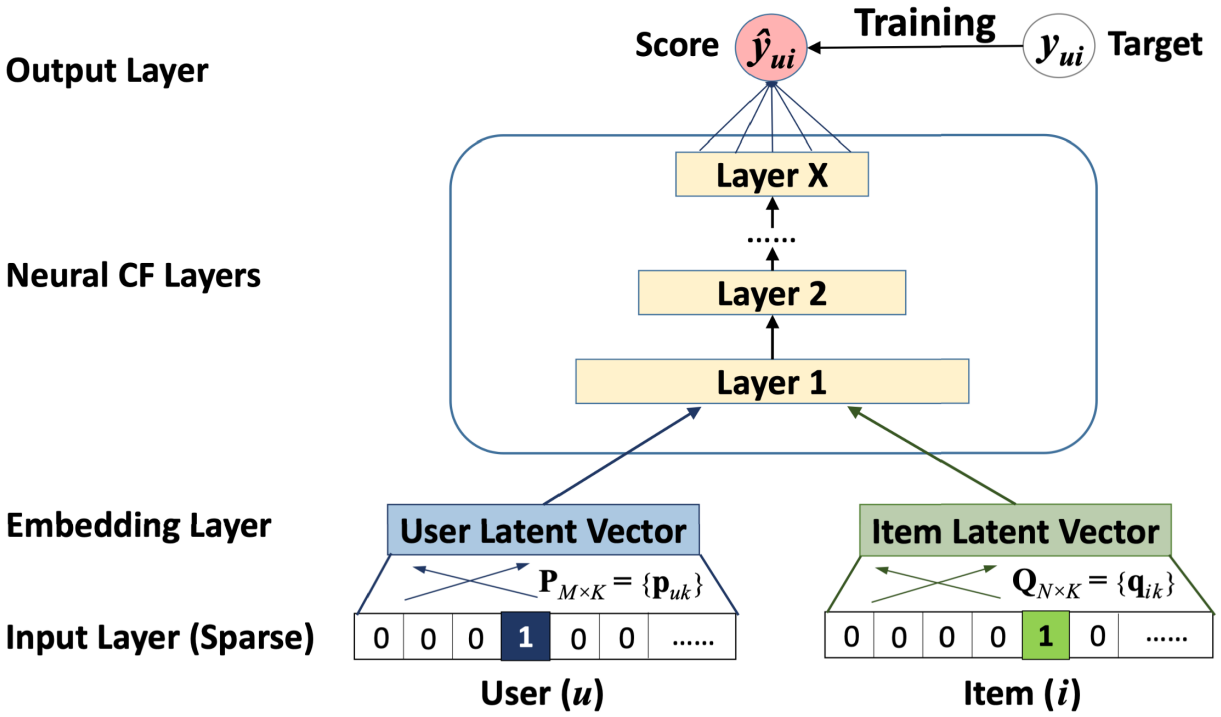
Two-Tower models



- Generalisation over MF.
- User Encoder and Item Encoder can be very complex (e.g. use Transformers as a submodule to encode text).
- Very similar to the Bi-Encoder models in classic IR

https://towardsdatascience.com/two-tower-networks-and-negative-sampling-in-recommender-systems-fdc88411601b

Neural Collaborative filtering *



- Is dot product as the scoring function enough?

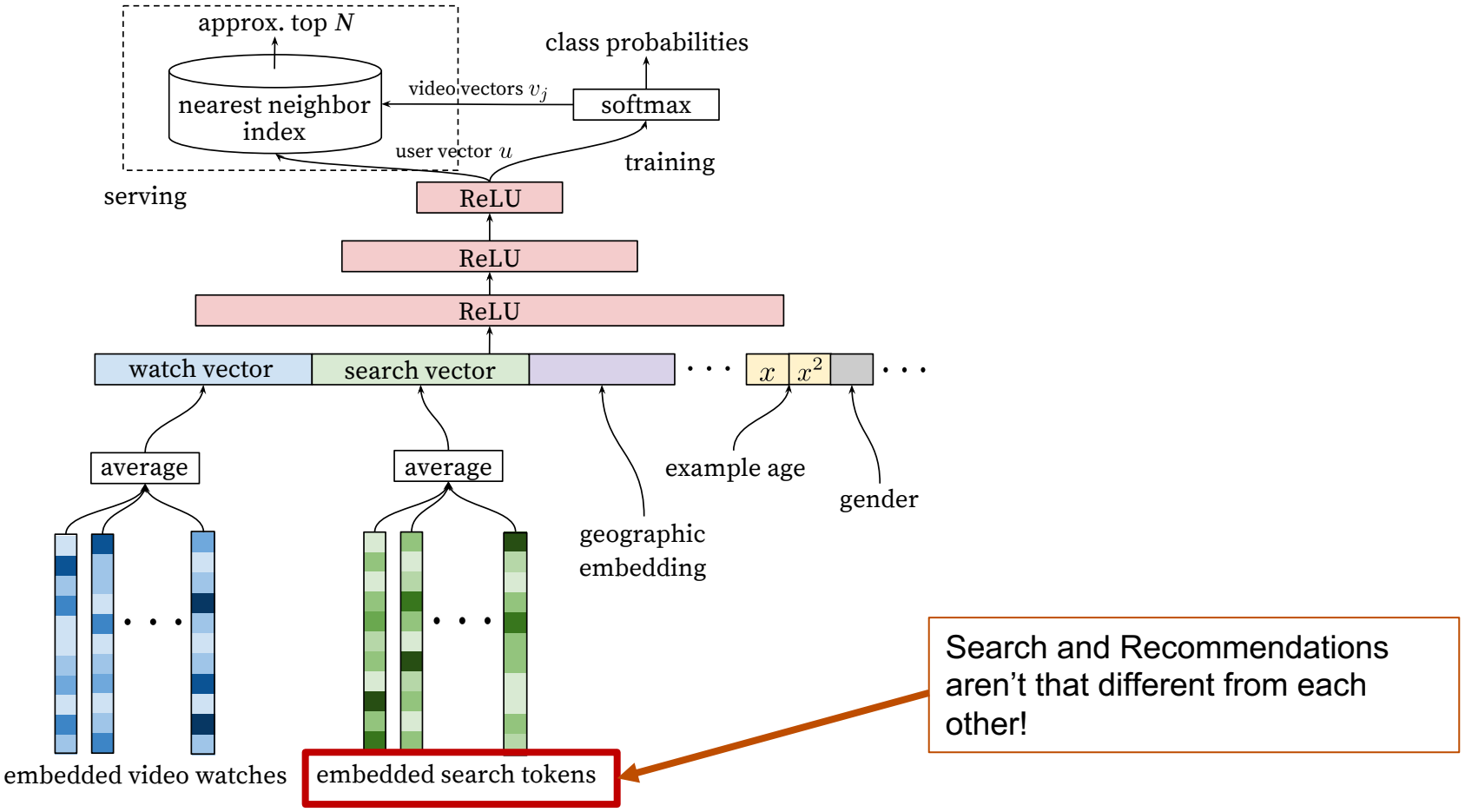
- He et al.* argue that we can do better!

- Complex learnable similarity functions can capture non-linear dependencies between latent factors.

Somewhat similar to the Cross-Encoder models in classic IR

* He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T.S., 2017, April. Neural collaborative filtering. In Proceedings of the 26th international conference on World Wide Web (pp. 173-182).

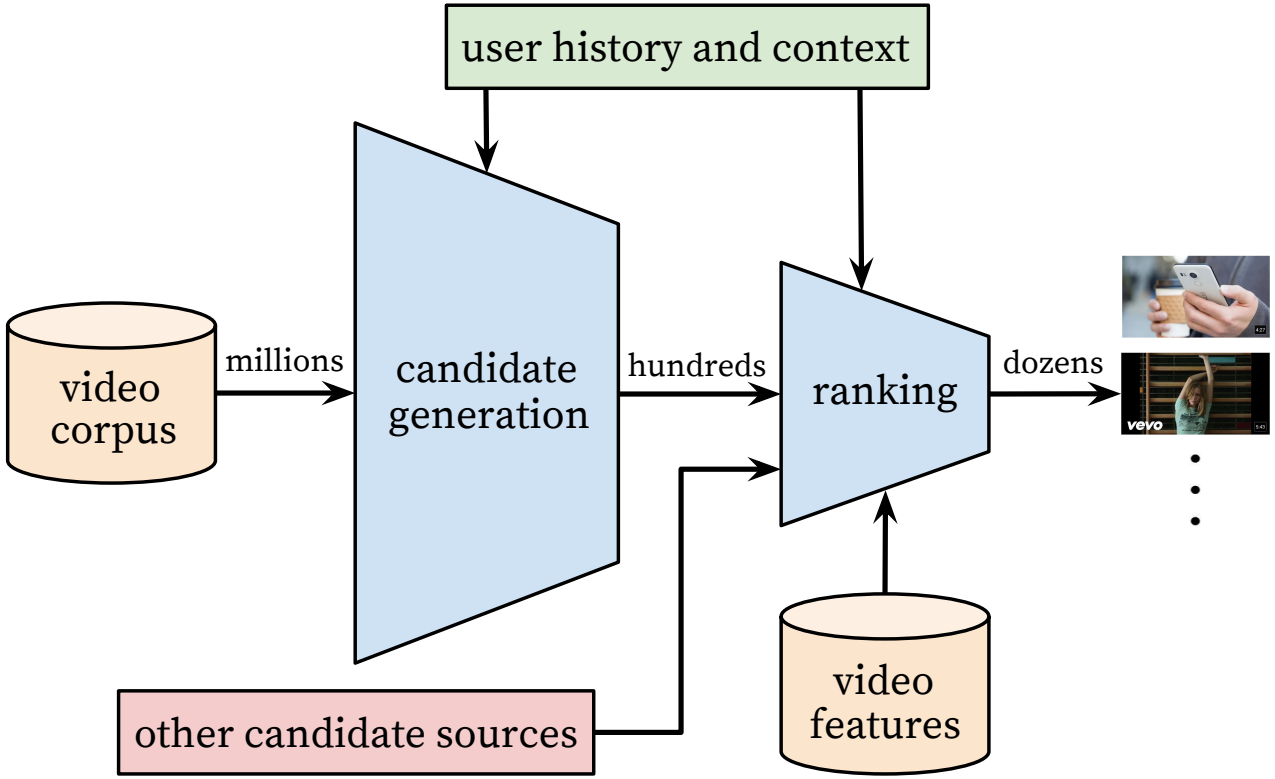
Example: YouTube Recommendations*



* Covington, P., Adams, J. and Sargin, E., 2016, September. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems (pp. 191-198).

Heavy Models require Re-Ranking

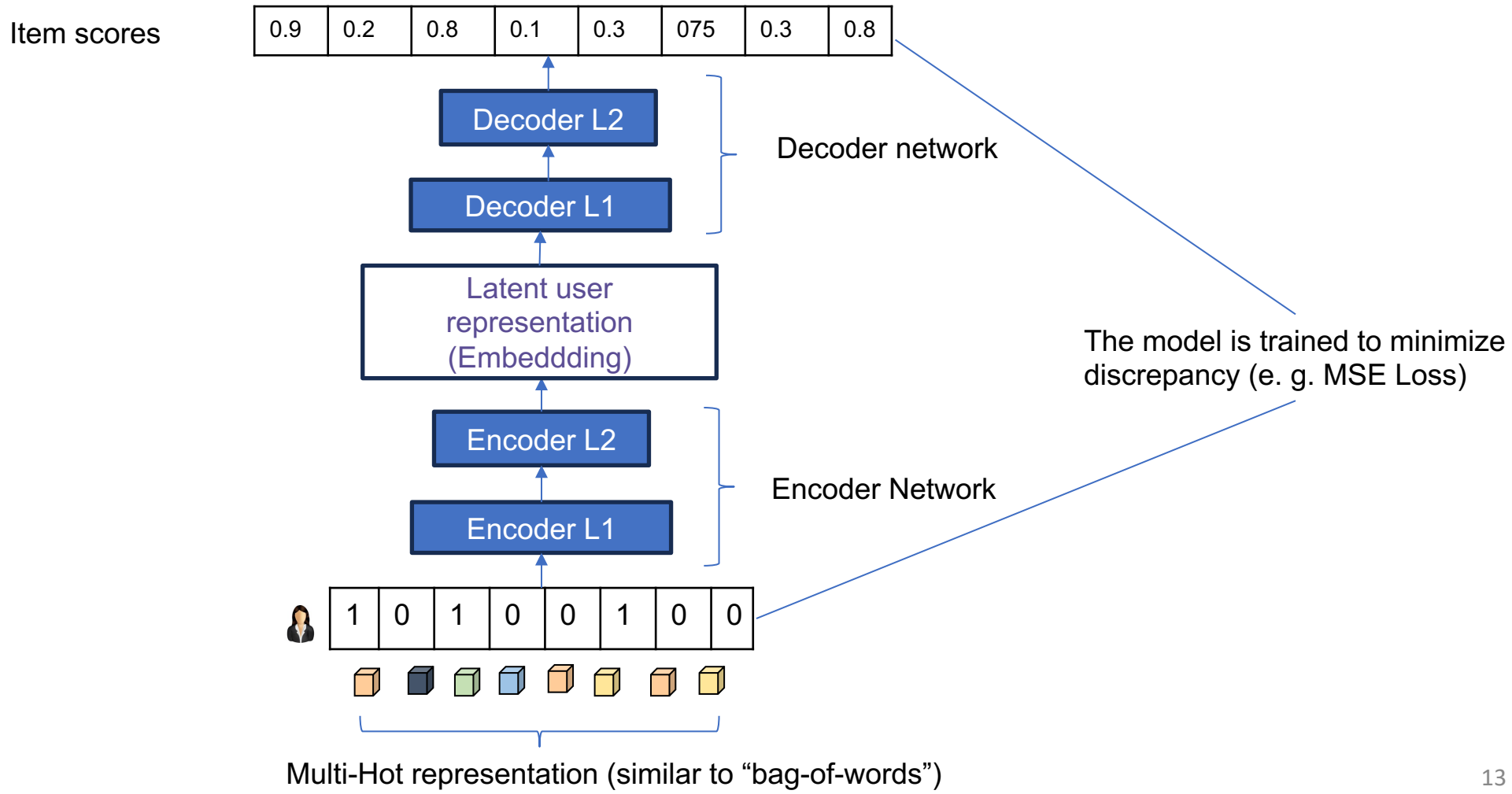
Example: YouTube recommendation pipeline*:



* Covington, P., Adams, J. and Sargin, E., 2016, September. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems (pp. 191-198).

Autoencoders

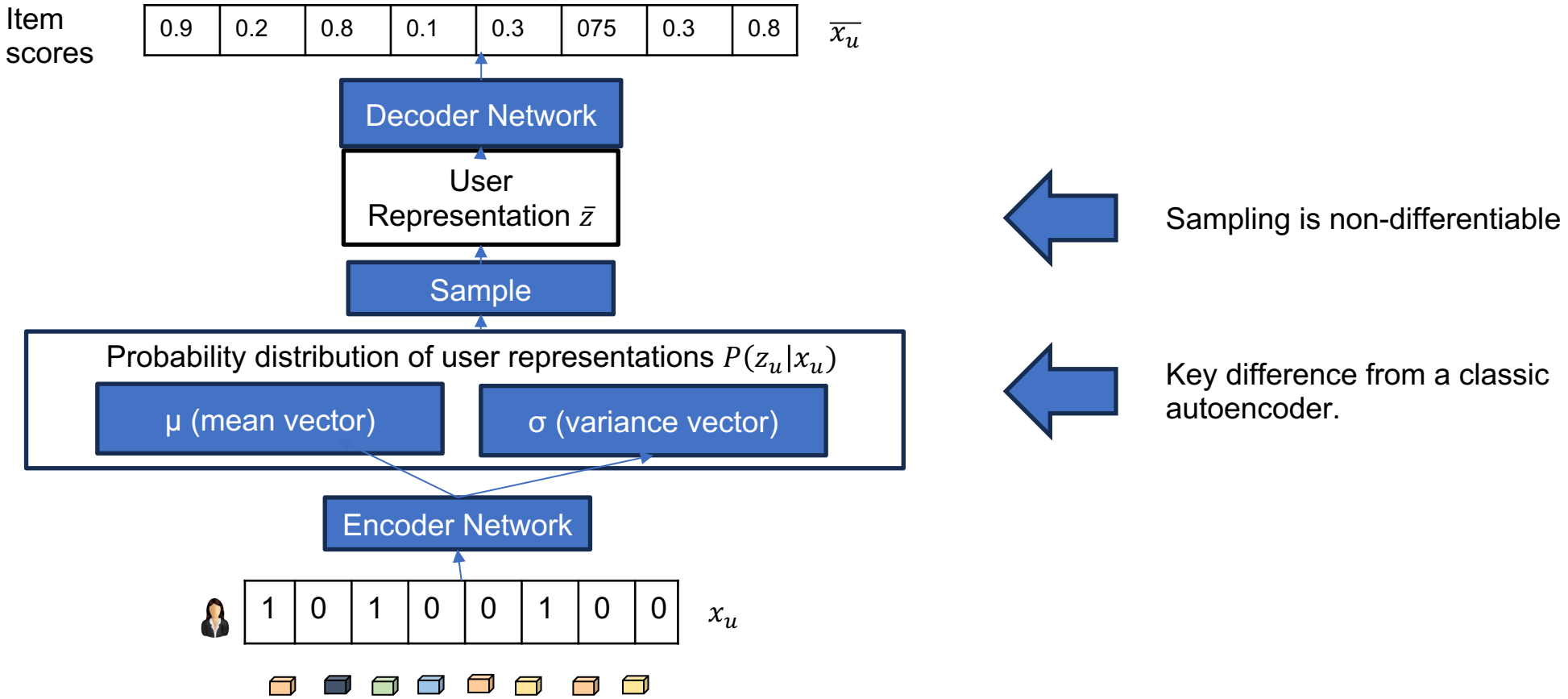
Matrix Factorisation models learn embeddings for both users and items.
… But we can represent users as sets of items they consumed.



Variational Autoencoder (VAE)*

- Usually, recommender systems represent a user using a single vector.
- Shouldn't it be probabilistic?

  E.g., model knowledge about a user increases with the amount of information it knows about the user.



*Liang, D., Krishnan, R.G., Hoffman, M.D. and Jebara, T., 2018, April. Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).

Training VAE

- Reparametrisation Trick
  - Sampling is non-differentiable, but we can sample from the normal distribution $\mathcal{N}(0,1)$ (non-differentiable) and then offset/scale according to μ and σ

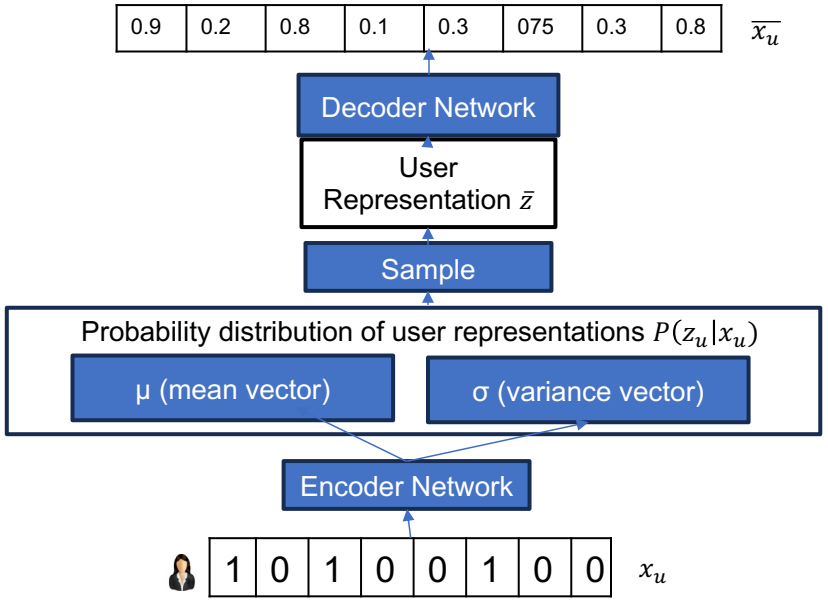
- Loss function: **Evidence Lower Bound ("ELBO")**

$$\mathcal{L}(x_u) = -\mathbb{E}_{P(x_u|z_u)}[\log P(x_u|z_u)] + \beta \cdot D_{KL}(P(x_u|z_u) \parallel \mathcal{N}(0,1))$$

Reconstruction loss
("how far away are the decoded item scores from the encode scores ")

How far away is the user representations Probability Distribution from the normal distribution

Regularisation strength



*Liang, D., Krishnan, R.G., Hoffman, M.D. and Jebara, T., 2018, April. Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).

Is VAE any good?

- In the famous reproducibility paper “Are We Really Making Much Progress”* paper, the authors found that VAE was the only deep-learning-based method that actually outperformed classic Matrix Factorisation.

- A recent reproducibility paper** confirms that VAE-based methods outperform all other methods for *classic* (no item order, no side information) recommender systems tasks.

💡 Variational Auto Encoder (VAE)-based methods are State-Of-The-Art for classic RecSys

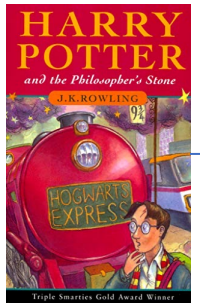
*Ferrari Dacrema, M., Cremonesi, P. and Jannach, D., 2019, September. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In Proc. RecSys

**Rendle, S., Krichene, W., Zhang, L. and Koren, Y., 2022, September. Revisiting the performance of ials on item recommendation benchmarks. In Proc. RecSys.

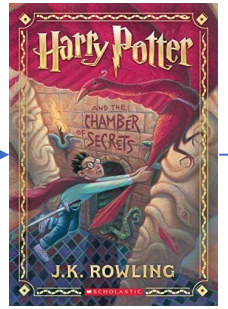
Sequential Recommendation

Goal: predict the next (future) interaction in a chronologically an ordered sequence of user-item interactions (historical).

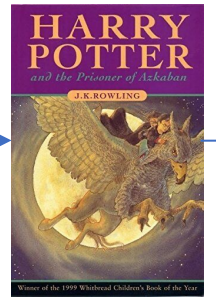
Example:



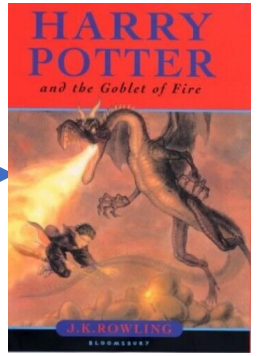
Harry Potter and the Philosopher's Stone



Harry Potter and the Chamber of Secrets



Harry Potter and the Prisoner of Azkaban



Recommended: Harry Potter and the Goblet of Fire

When does Classic RecSys fail?

MF does not consider the order of interactions
In reality, the **order** of interactions may be very important in some recommendation scenarios:

- **Natural sequential patterns:** Choose a case for the phone after buying a phone, not the other way around.
- **Series of items:** Star Wars IV → Star Wars V → Star Wars VI → Star Wars I
- **Evolving user interests**: 10 years ago, a user used to listen to pop music, but now they prefer rock.
- **Geographical proximity** defines the order of places to visit

  London → Newcastle → Edinburgh → Glasgow → Manchester → London
  OR
  Glasgow → London → Newcastle → London → Manchester → Edinburgh

- **No repeating interactions in MF:** A user buys a pack of coffee every month.

Sequential Recommendation

Goal: Predict the next (future) interaction in a chronologically ordered sequence of user-item interactions (historical).

What exactly does “**predict**“ mean?

Formalising sequential recommendation

Let's denote a set of items as I

A sequential recommendation model is a function:

$$f(h) \mapsto g$$

$h \in I^n = \langle h_1, h_2, h_3, \dots h_n \rangle$

Chronologically ordered sequence of past interactions

$g \in I^m = \langle g_1, g_2, g_3, \dots g_m \rangle$

Suggested candidates for becoming the next interaction h_{n+1}

Utility function

How do we know whether our sequential model $f(h) \mapsto g$ is *good*?

💡 We need a utility function:

$$L_u(h, g, h_{n+1})$$

$h \in I^n$
User history

$g^m \in I^n$
Suggested recommendations

$h_{n+1} \in I$
Ground truth next interaction

Examples:

- **Ranking accuracy:** recommendations g contain ground truth h_{n+1} on high positions (NDCG, MAP, MRR, Recall@K).

"accuracy-based" recommendation

- **Diversity:** recommended items in g are not too similar to each other.
- **Novelty:** recommended items in g are not too similar to the items in the user history h.
- **Long tail promotion:** recommended items in g are not too popular.

"beyond accuracy" recommendation

Accuracy-based recommendation and the Probability Ranking Principle

- **Probability Ranking Principle (Maron and Kuhns, 1960; Robertson, 1977):**

> **The best that a recommender system can do (in terms of ranking accuracy) is to rank items according to the descending order of estimated interaction probability.**

Top-K recommendation ("score-and-rank"):

- For a given user, estimate the probability of interaction $P(i)$ with *every* item in the catalogue*.
- Rank items according to estimated probability.
- Return K items with the highest estimated probability.

*Instead of estimating the probability, sometimes it is more convenient to estimate a monotonic transformation, which does not break the order. For example, estimating $\log(P(i))$ is a popular choice.

Deep Learning Models for Sequential Recommendation

- Most state-of-the-art sequence recommendation models are based on Deep Learning
- Such recommender models borrow ideas from other domains
  - Computer Vision – e.g. NextItNet (Yuan et al.) and Caser (Tang and Wang) are based on Convolutional Neural Networks.
  - Natural Language Processing
    - RNNs – e.g. GRU4Rec (Hidasi et al.)
    - **Transformers** – e.g. SASRec (Kang et al.), BERT4Rec (Sun et al.)
- No learning of user embeddings offline, just sequence characteristics

Parallel to Language Modelling

Language modelling is essentially the same thing… predicting the probability of the next token, given a sequence of k preceding tokens, i.e. $P(t_{n+1} \mid t_{n-k} \ldots t_n)$

Sequential Recommendation: Next Item Prediction

Language Modelling: Next Token Prediction

∴ It makes sense to adapt language models to sequential recommendation

A Recipe for Sequential Recommendation

1. We have a deep learning **model (architecture)** that learns how to complete a sequence
   - Is it RNN, is it Attention? How many layers?
   - Each item in the model has a learnable embedding.

NB: We don't need user embeddings, we simply feed the recent interactions of the user into the mode each time

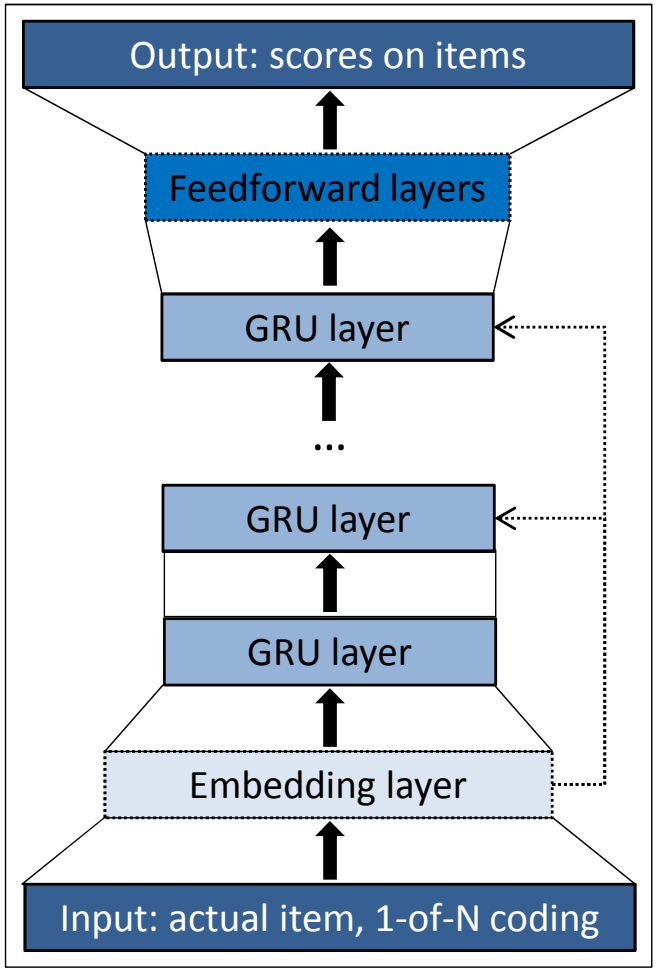
2. We have a **training objective** that defines how we show training sequences (including positive items) to the model, which the model aims to correctly complete

3. Do we treat all unseen items as negative, or do we **sample negative items?**

4. We have a **loss function,** which measures how correct the model is at predicting the relevant items.
   - e.g. RMSE, BPR
   - Did the model correctly identify the relevant item (pointwise) etc.

GRU4Rec*

- One of the first successful adaptations of the Deep Learning-based Language Models for sequential recommendation
- Uses Gated Recurrent Unit (“GRU”) – a special type of Recurrent Neural Network architecture
- First to come up with the idea of replacing tokens with item IDs
- A good demonstration of the high-level structure shared by most of the NLP-based models:
    - Input: a sequence of item IDs
    - Embedding layer forward layer to obtain dense representations of the models
    - “Transformation” network that is used to obtain item embeddings (or *sequence embeddings*) (Stack of GRU layers with skip connections)
    - Feedforward layer to obtain score distributions per item



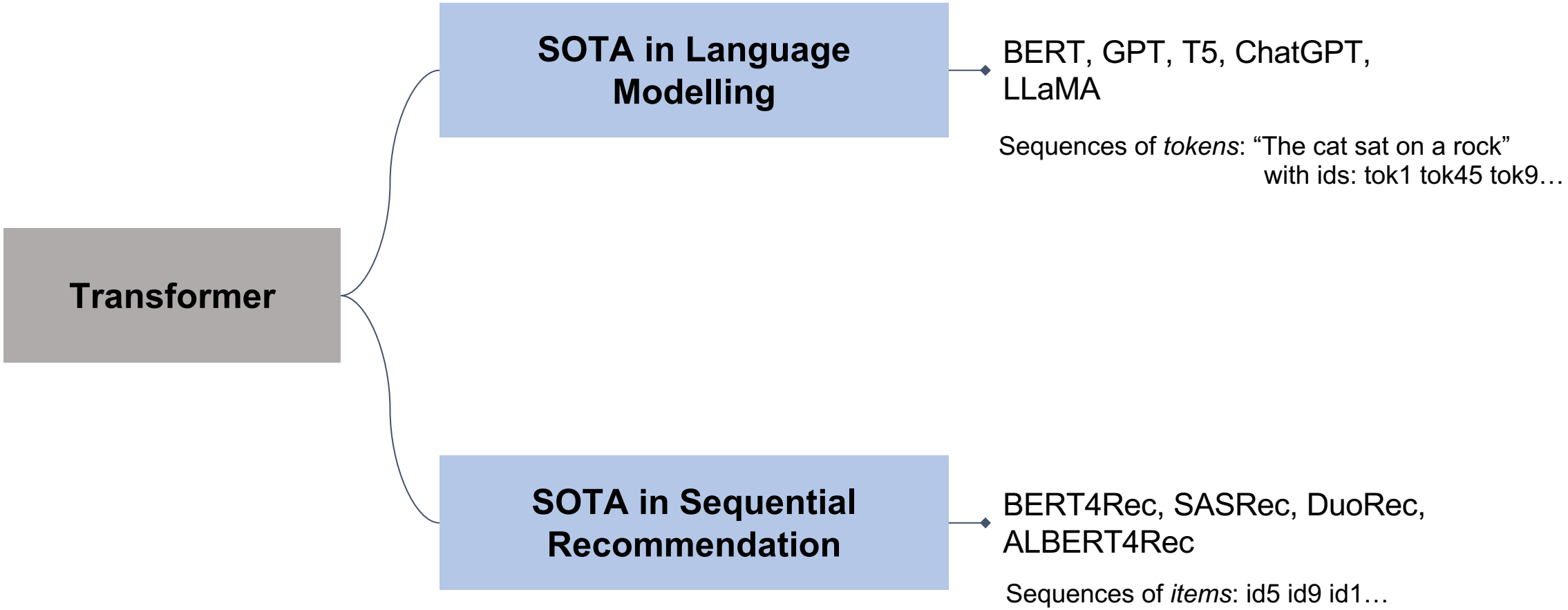
Cons:

- Shares “vanishing” gradients problem with other RNNs (hard to train on long sequences)
- The idea is simple, but there are many technical details in the model that led to replicability problems**

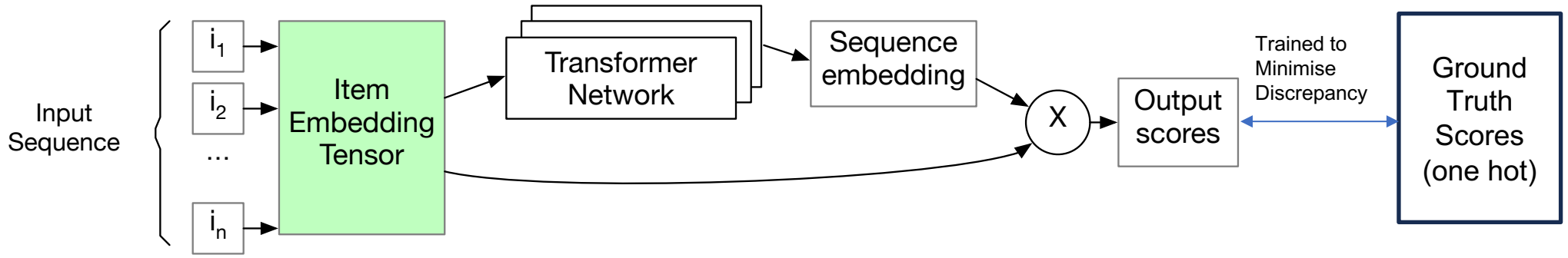
*Hidasi, B., Karatzoglou, A., Baltrunas, L. and Tikk, D., 2015. Session-based recommendations with recurrent neural networks. *ICLR 2016*.

**Hidasi, B. and Czapp, Á.T., 2023, September. The effect of third party implementations on reproducibility. *ACM RecSys 2023*

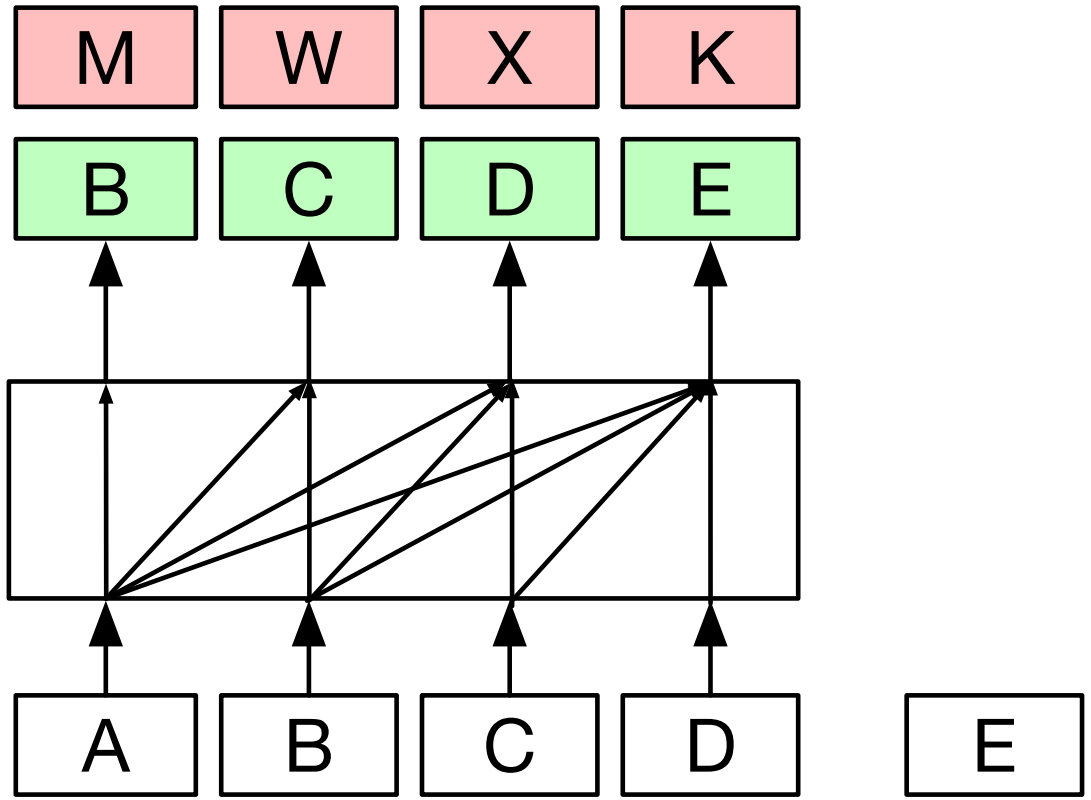
Transformers



Architecture of a typical Transformer-based **recommendation** model



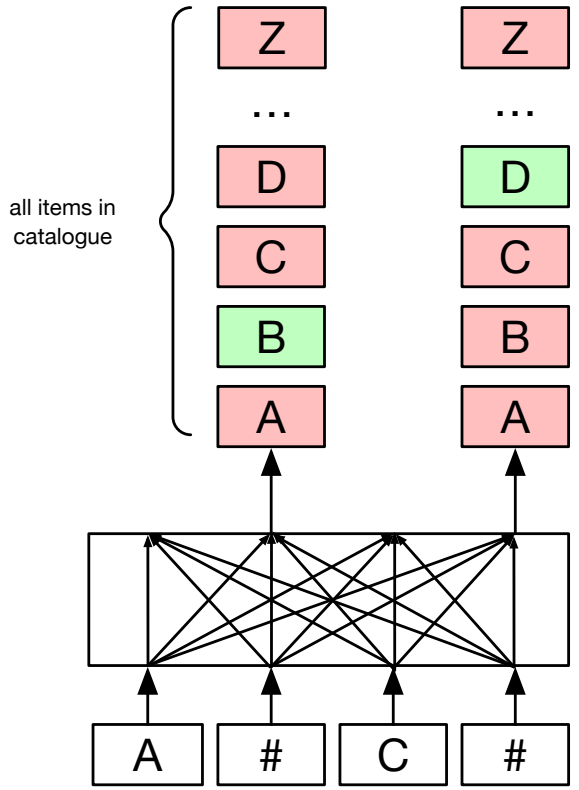
SASRec*



- First Transformer-based model.
- Uses the **Decoder** part of the Transformer architecture (similar to GPT).
- **Causal** self-attention.
- Sequence shifting training task.
- One randomly sampled negative per positive.
- **Binary Cross-Entropy** loss.
- A version of SASRec can be easily implemented using `torch.nn.TransformerEncoder` class

*Kang WC, McAuley J. Self-attentive sequential recommendation. In ICDM 2018

BERT4Rec*



- Based on the **Encoder** part of the Transformer architecture (more specifically, based on BERT**).
- Item masking training task.
- Trained to distinguish **all** negative items from positive ones.
- **Full** self-attention
- A version of BER4Rec can be easily implemented using the `BertForMaskedLM` Model from the HuggingFace Transformers library.

The original BERT4Rec paper claimed superiority. However, we found inconsistencies in the literature. Which of the two is the best?

*Sun F, Liu J, Wu J, Pei C, Lin X, Ou W, Jiang P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. *In CIKM 2019*

**Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *In NAACL 2019*

Beyond-Accuracy goals

- **Diversity:** recommended items are not too similar to each other.
- **Novelty:** recommended items are not too similar to the items in the user history.
- **Long tail promotion:** recommended items in are not too popular.

Most of the existing methods are based on Re-Ranking heuristics:

1. Generate recommendations using a standard Top-K model.
2. Re-Rank to promote items that increase secondary beyond-accuracy metric.

Examples: **Maximal Marginal Relevance (MMR) ***, **Serendipity Oriented Greedy (SOG)****

*Carbonell J, Goldstein J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. *In SIGIR 1998*

**Kotkov D, Veijalainen J, Wang S. A serendipity-oriented greedy algorithm for recommendations. *In WEBIST 2017*

Why does the Probability Ranking Principle fail for the Beyond-Accuracy goals?

- **Probability Ranking Principle (Maron and Kuhns, 1960; Robertson, 1977):**

> The best that a recommender system can do (in terms of ranking **accuracy**) is to rank items according to the descending order of estimated interaction probability.

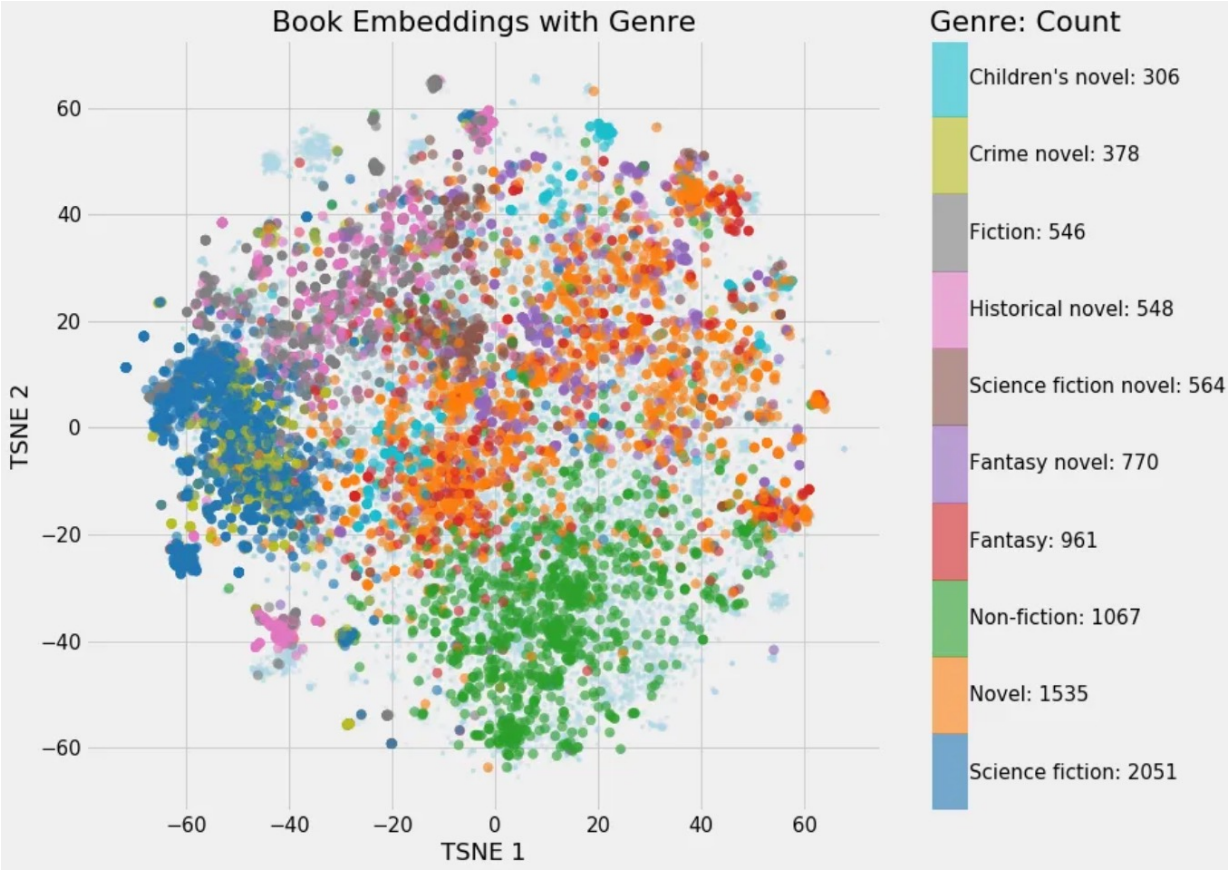
- Frequently, beyond-accuracy goals include components that are defined in terms of whole *lists* of items.

Example: In music recommendation, we may want a recommender system to recommend a set of songs that contains music from different artists rather than music from a single artist.

💡 We need a better way to **generate** recommendations than “Score and Rank.”

Score-and-rank recommenders do not produce diverse recommendations **by design**

In a trained recommender system, similar items have similar embeddings.
Example*:



I like reading:

- Scientific books about Machine learning (" Non-Fiction") for my PhD Research
- Historical novels to relax
- From time to time, I buy books for my 5-year-old daughter

What point in this space best represents my interests?

Top-K recommendation in embedding-based methods (including Transformers):

1) Find a point in the embedding space representing a user.

2) Recommend items in close proximity to user representation in the embedding space.

3) Items in close proximity to a specific point are likely to be also *similar to each other* (non-diverse)

* Will Koehrsen, Building a Recommendation System Using Neural Network Embeddings, in *Towards Data Science blog. https://towardsdatascience.com/building-a-recommendation-system-using-neural-network-embeddings-1ef92e5c80c9*

Generative Language Models

For PhD Research on Machine Learning:

1. **"Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville** - This book is a deep dive into deep learning, covering the fundamentals, methodologies, and advanced topics in machine learning and artificial intelligence.
2. **"Pattern Recognition and Machine Learning" by Christopher M. Bishop** - An essential read for understanding the conceptual framework behind modern machine learning, focusing on pattern recognition.
3. **"The Hundred-Page Machine Learning Book" by Andriy Burkov** - Although concise, this book covers all the necessary basics and some advanced topics in machine learning, making it perfect for PhD students who need a quick reference or overview.

For Relaxation with Historical Novels:

1. **"The Nightingale" by Kristin Hannah** - A gripping tale set in France during World War II, focusing on the lives and resilience of two sisters.
2. **"All the Light We Cannot See" by Anthony Doerr** - Another World War II novel, this beautifully written book intertwines the lives of a blind French girl and a German soldier.
3. **"The Pillars of the Earth" by Ken Follett** - A richly woven novel set in the 12th century, focusing on the building of a cathedral in the fictional town of Kingsbridge, England.

For Your 5-Year-Old Daughter:

1. **"The Day the Crayons Quit" by Drew Daywalt and Oliver Jeffers** - A humorous and colorful story about crayons coming to life, perfect for sparking creativity.
2. **"Ada Twist, Scientist" by Andrea Beaty and David Roberts** - Encourages curiosity and a love for science through the story of Ada Twist, a very curious and persistent young scientist.
3. **"Where the Wild Things Are" by Maurice Sendak** - A classic tale of adventure and imagination as young Max travels to the land of the Wild Things.
4. **"Goodnight Moon" by Margaret Wise Brown and Clement Hurd** - A calming bedtime story that has been a favorite among children for generations.

- Generative language models can produce text that makes sense as a whole rather than a combination of words.
- These models generate text **autoregressively** (token-by-token)
- When generating the next token, the model already knows what items were generated previously.

💡 Can the same technique (**autoregressive generation**) work for optimising complex goals in sequential recommendation?

Sequential Recommendation as a generation task

Parallels with Language Models:

Item ids = Tokens

Text = Sequence of item ids

Language models (e.g. GPT)

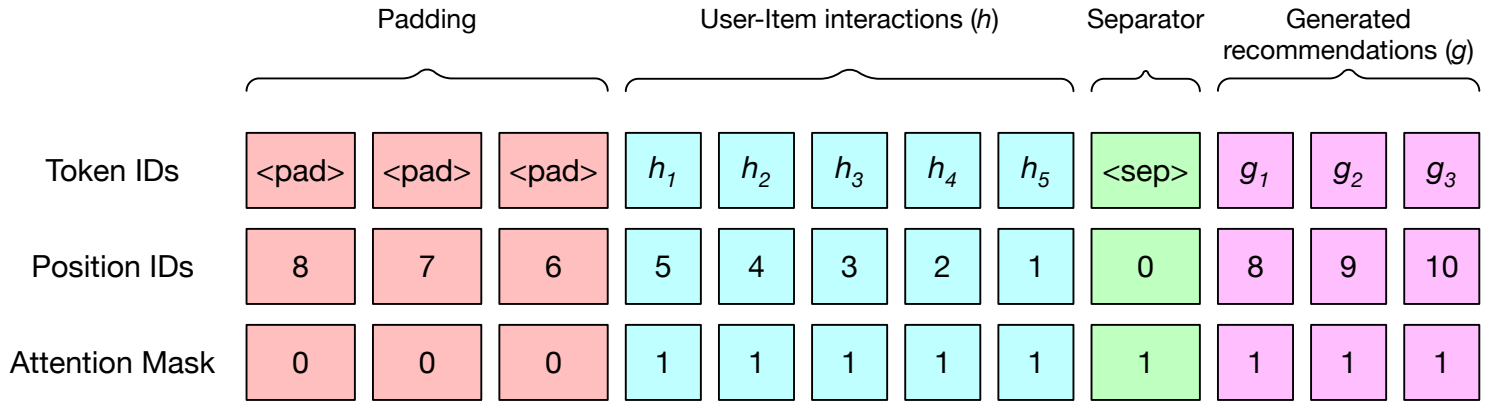
Given an input text (“Prompt”), generate an output text (model’s response) that satisfies complex human needs

Sequential Recommender Systems

Given a sequence (“history”), generate an output sequence (recommendations) that satisfies complex human needs

GPTRec*

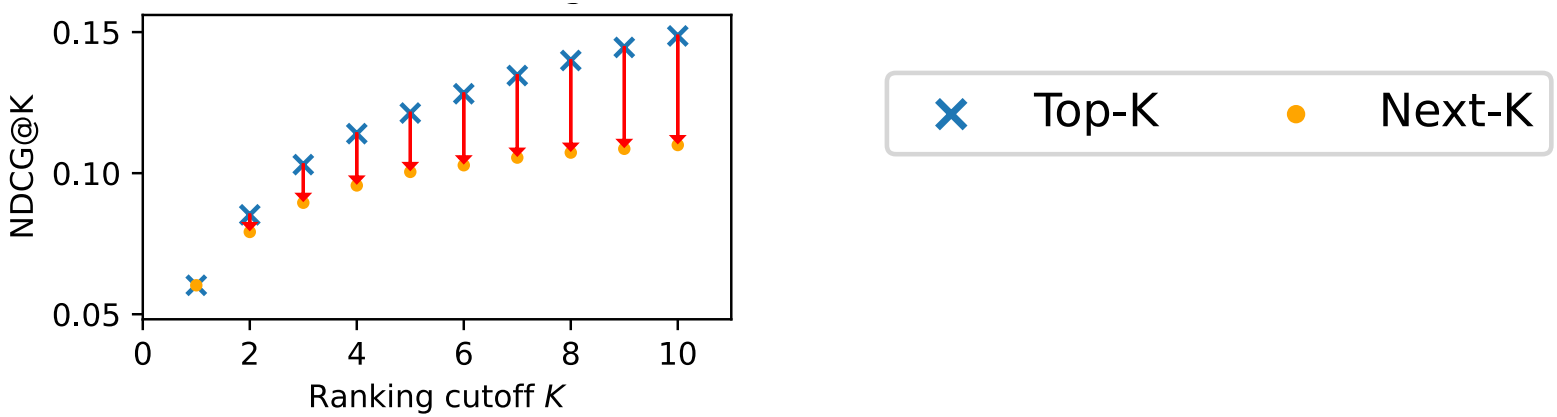
- A Transformer decoder-based recommendation model
  - Based on GPT-2 architecture (but not pre-trained weights)
  - Architecturally, similar to SASRec
  - Generates recommendations using the **Next-K strategy** (autoregressively, item-by-item) – ala `.generate()`
  - Input and output have the same structure:



* Petrov, A.V. and Macdonald, C., 2023. Generative sequential recommendation with GPTRec. *Presented at GenIR@SIGIR2023*

Generative recommendation requires special training

- Autoregressive (“Next-K”) generation is an inference strategy.
- It is possible to train GPTRec using “(g)SASRec’s” training and infer using Next-K strategy, but…



… model quality degrades at higher ranking cutoffs (ML-1M dataset example).

In order to **generate** recommendation lists, the model needs to **see recommendation lists**.

Teacher-Student training scheme*

Where to get a dataset ⟨**history, recommendations**⟩ pairs?

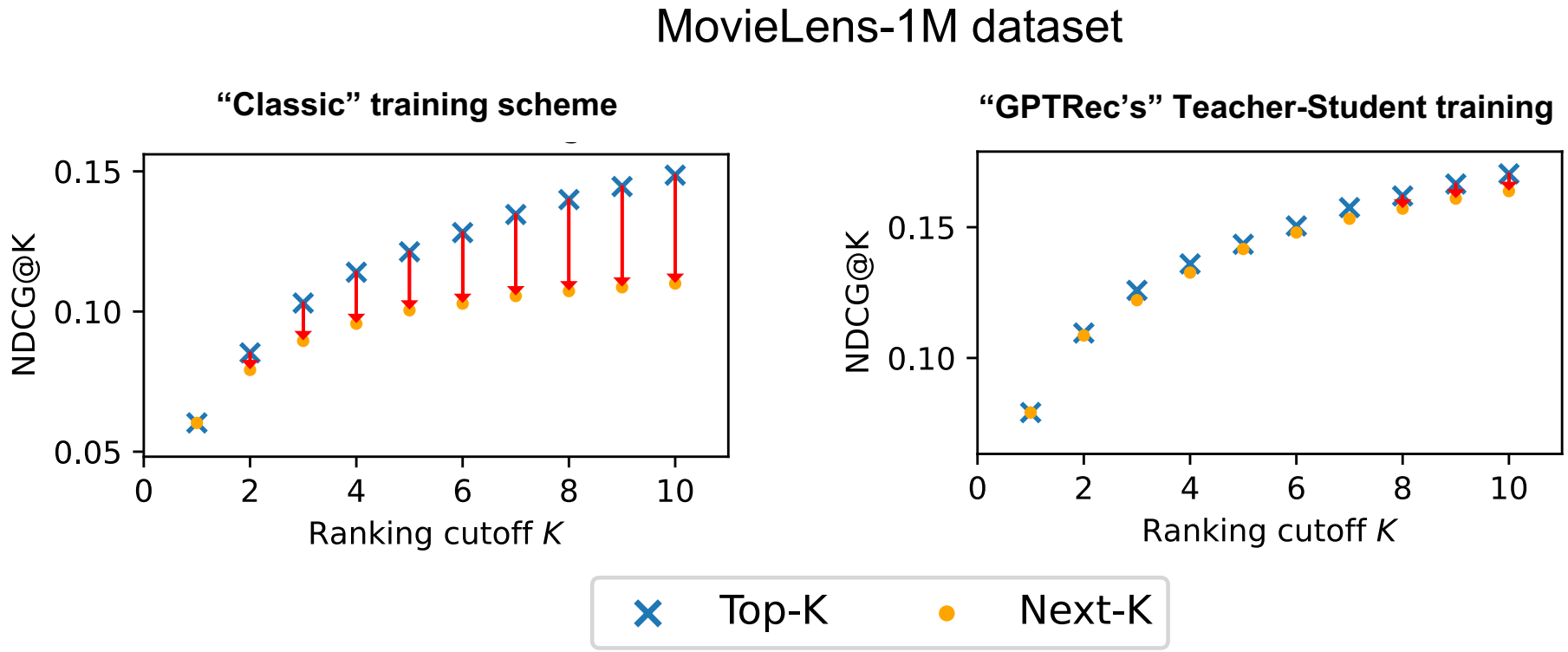
1. Train a “classic” model (e. g. BERT4Rec)

2. Using the “classic” model, generate a training set of ⟨**history, recommendations**⟩ pairs

3. Using this training set, train GPTRec to mimic BERT4Rec’s behaviour (but in generative mode).

*Petrov, A. and Macdonald, C., 2024. Aligning GPTRec with Beyond-Accuracy Goals with Reinforcement Learning. Accepted by GenRec@WWW2024

Teacher-Student training scheme, evaluation results



Teacher-student training allows for achieving SOTA results in autoregressive (Next-K mode)

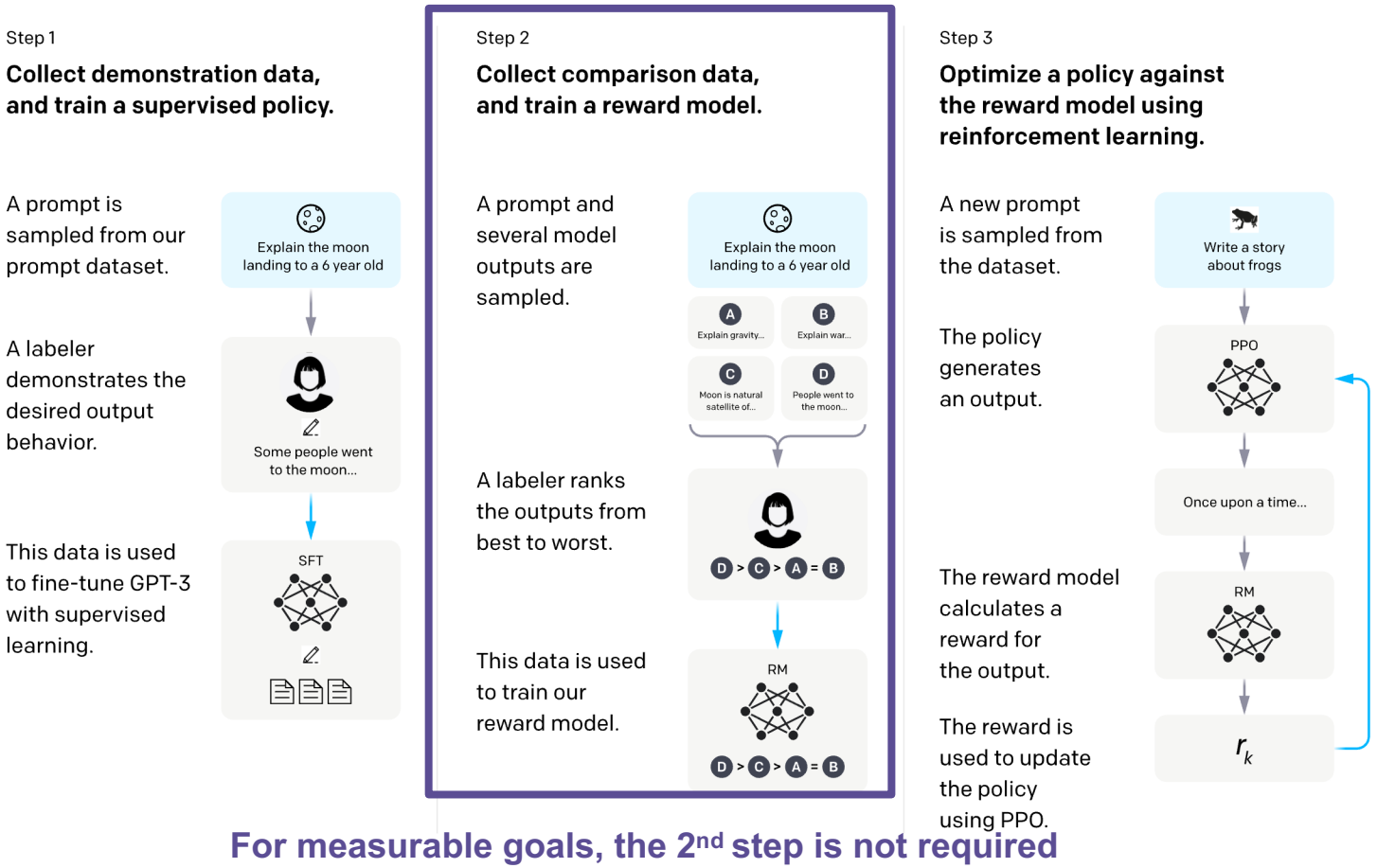
..but our goal is not merely NDCG@K

Misalignment Problem

- The training set generated by the teacher model does not include “beyond-accuracy” training samples.
- The source of training samples for beyond-accuracy goals is usually unavailable.
- The discrepancy between the training set and our real goal is known as **misalignment**.

RLHF: Misalignment Solution in Language Models

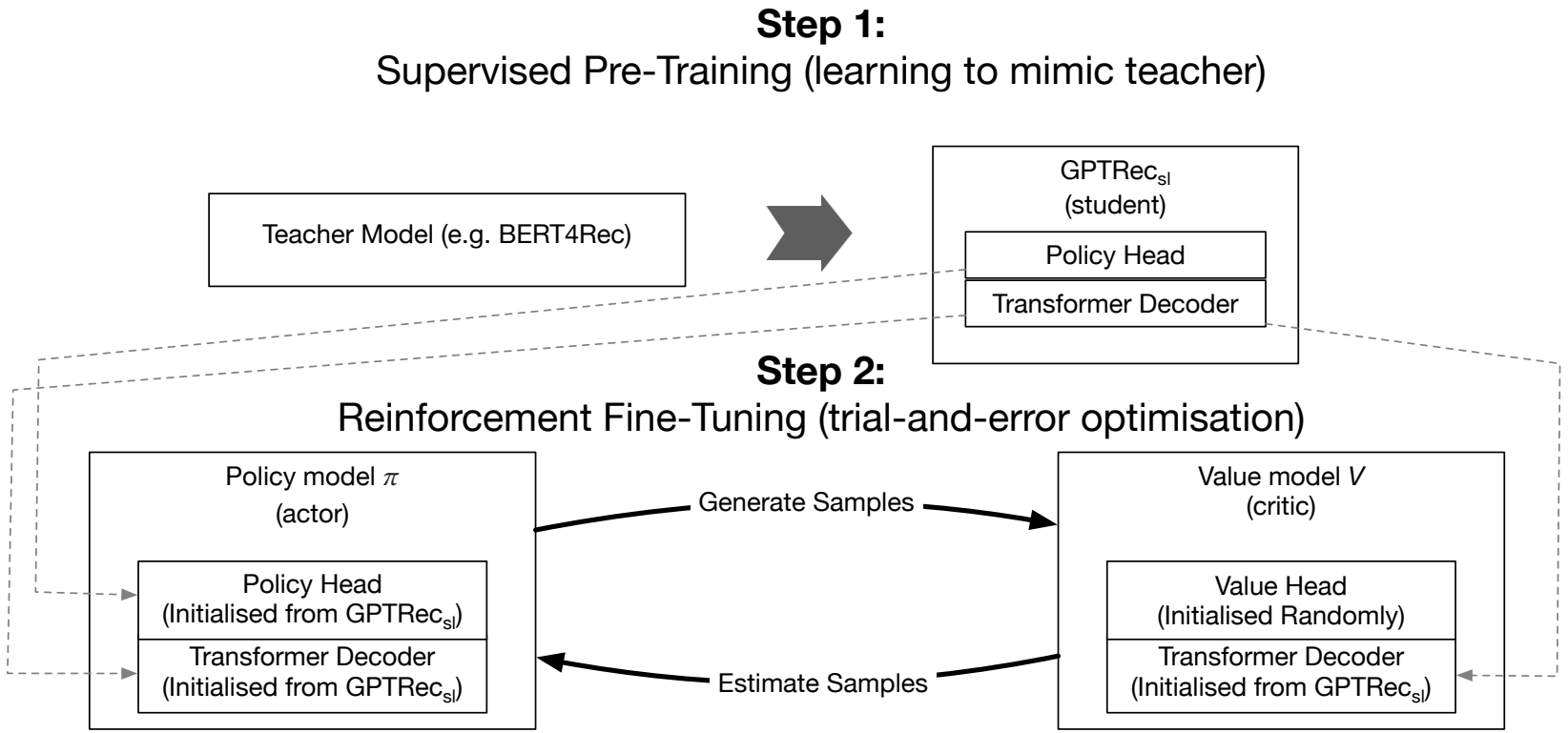
- Misalignment exists in language models.
- Typical Solution: Reinforcement Learning with Human Feedback (RLHF)*



For measurable goals, the 2nd step is not required

*Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J. Training language models to follow instructions with human feedback. *In NeurIPS 2022*

Aligning GPTRec with beyond-accuracy goals

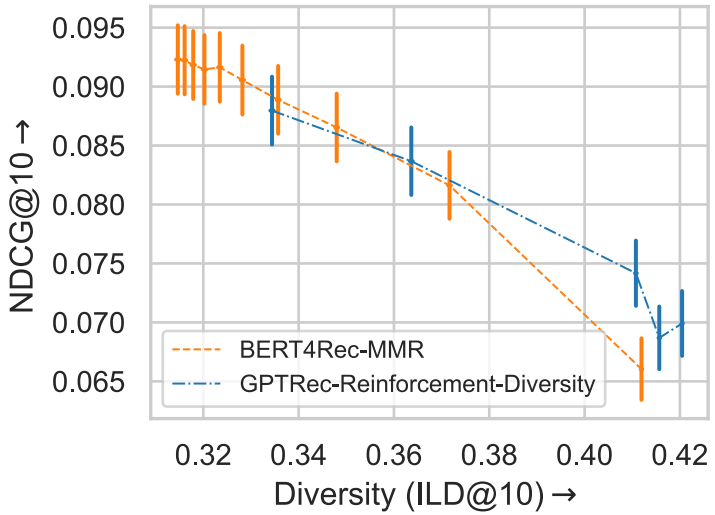


RLHF can be adapted to RecSys to align recommender models with beyond-accuracy goals

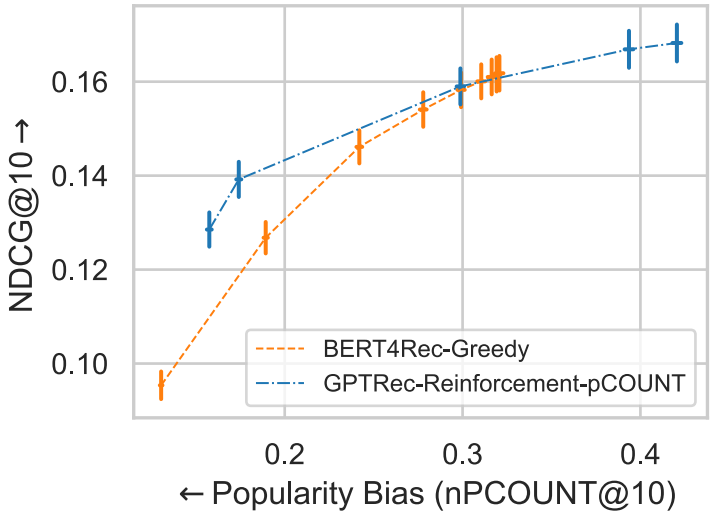
*Petrov, A. and Macdonald, C., 2024. Aligning GPTRec with Beyond-Accuracy Goals with Reinforcement Learning. Accepted by GenRec@WWW2024

Does RL-based alignment with work?

Steam, accuracy-diversity tradeoff.



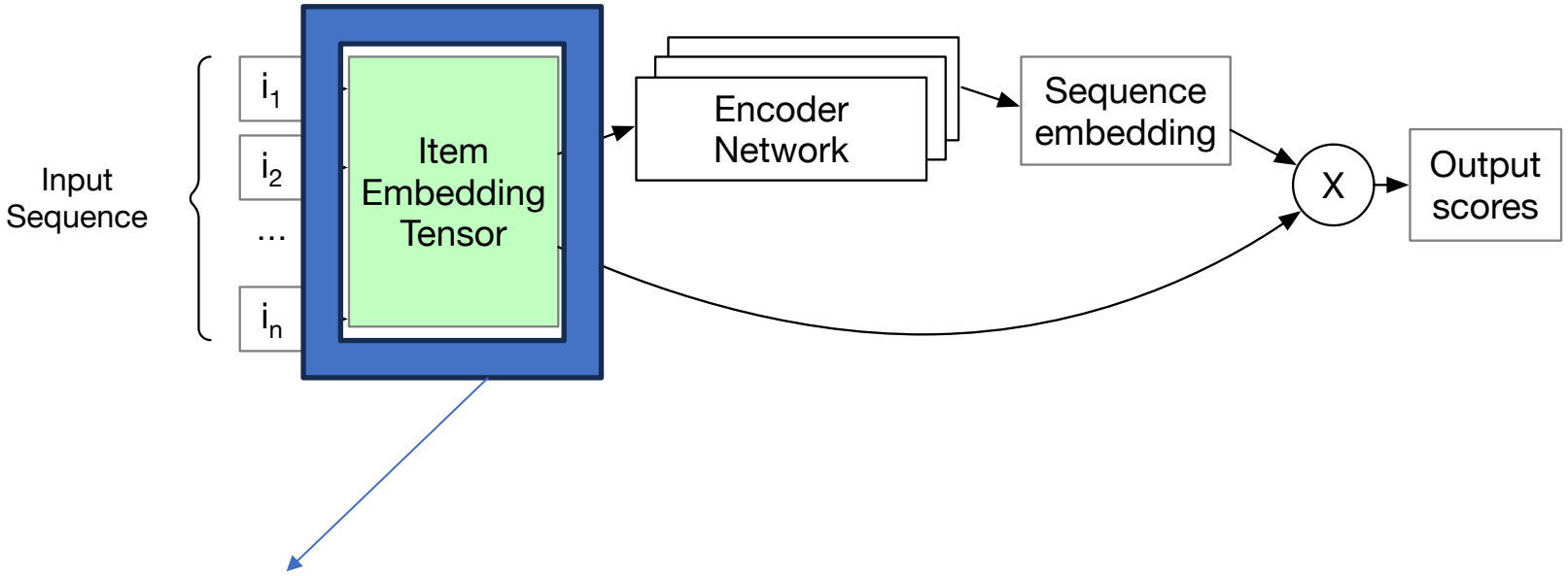
MovieLens-1M, accuracy-popularity tradeoff



Generative recommendation models can be aligned with beyond-accuracy goals using Reinforcement Learning.

The results are better compared to greedy re-ranking heuristics.

Large embeddings tensor



- The large number of parameters require more GPU memory, especially during training (for gradients)
- The large number of parameters increases the chances of model overfitting

Solution from Language Modelling: Tokenisation

To reduce memory consumption, Language Models (e.g. BERT, GPT) split (infrequent) words into sub-word tokens:

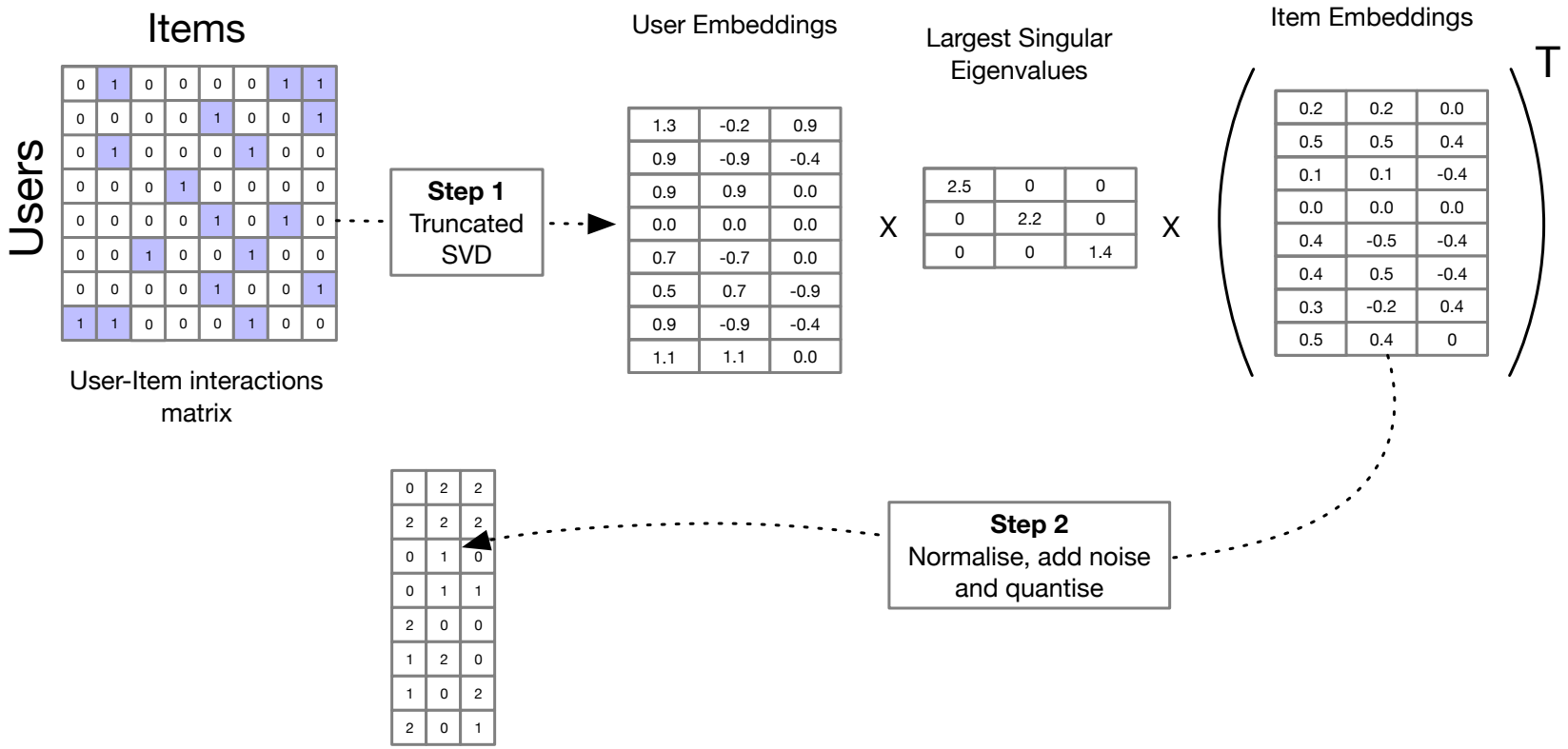
```
In [9]: tokenizer = AutoTokenizer.from_pretrained("gpt2")

In [10]: tokenizer.tokenize("aeroplane")
Out[10]: ['aer', 'opl', 'ane']
```

Any word can be built from tokens.

A small number of tokens → Moderate GPU memory consumption.

💡Can we split items into sub-items?

Sub-item “code” assignment

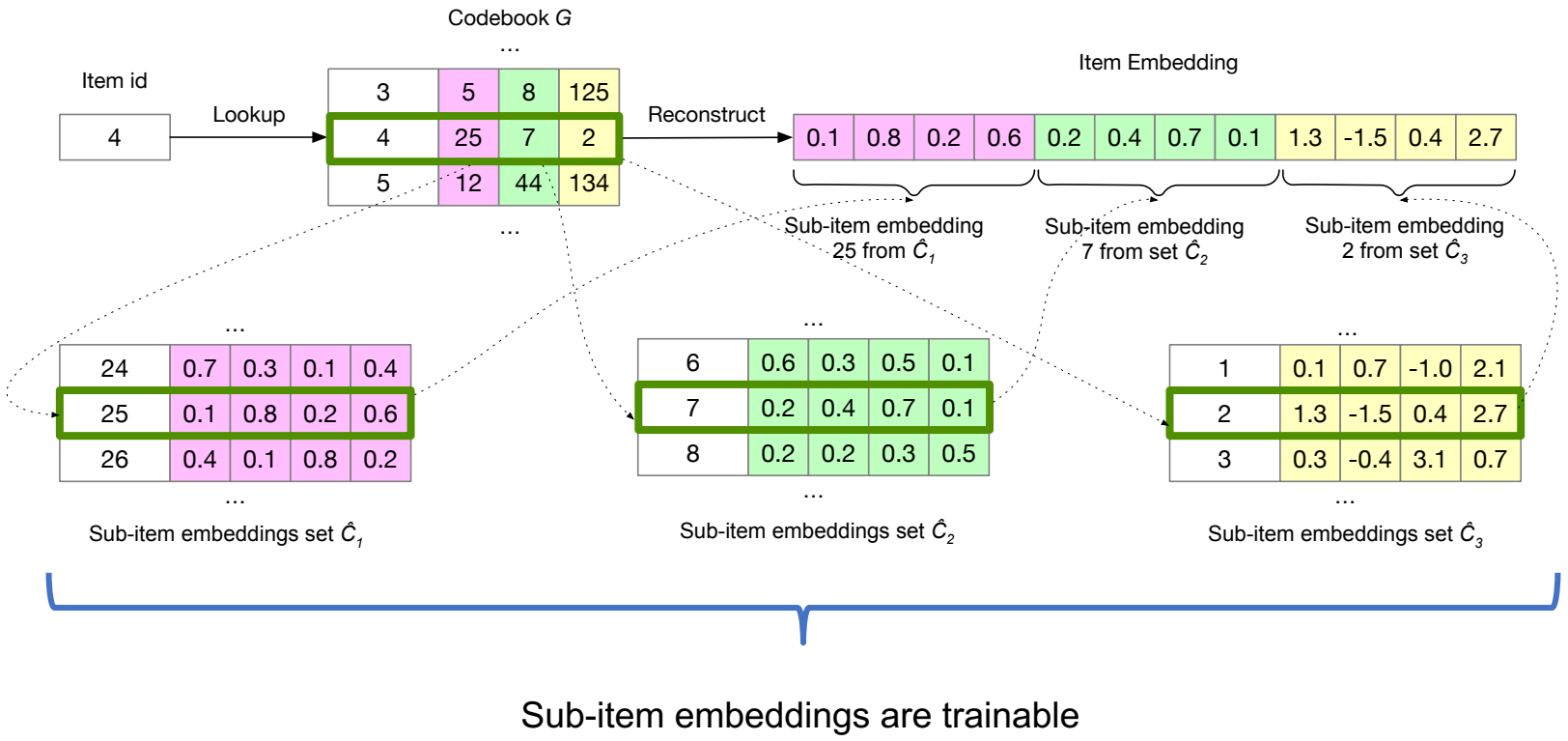


- Similar items have similar codes.
- Truncated SVD is very cheap, does not require GPU

Codebooks of sub-item ids for each item

We call this “SVD tokenisation”

RecJPQ: Quantising item embeddings



Product Quantisation (PQ):

1. Instead of storing embeddings of items, we store embeddings of sub-items, (equivalent of tokens), which we have grouped into sets
2. Store *code* (list of sub-item ids) for items, instead of storing embeddings
3. Recover item embeddings by concatenating relevant sub-item embeddings

PQ vs. JPQ

Product Quantisation (PQ) *

- Codes are assigned *after* full embeddings are built.
- It is effective for compressing an existing model but not for model training.

Joint Product Quantisation (JPQ) **

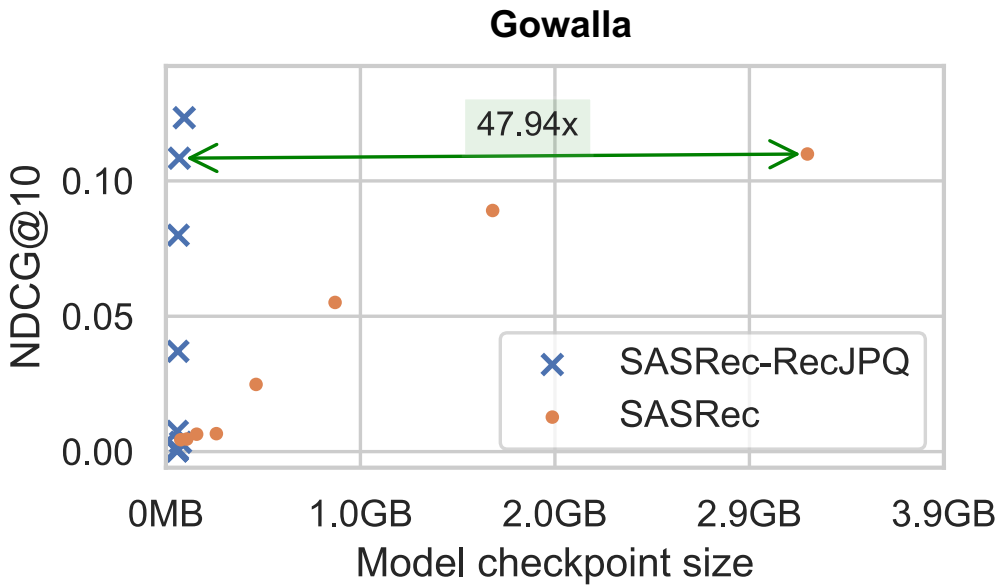
- Codes are assigned *before* full embeddings are built.
- Directly training full compressed model, moderate GPU memory requirements.
- Require code assignment strategy – in RecJPQ, we use Truncated SVD or BPR

No prior work has used JPQ for recommendation

*Jegou, H., Douze, M. and Schmid, C., 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, *33*(1), pp.117-128.

**Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M. and Ma, S., 2021, October. Jointly optimizing query encoder and product quantization to improve retrieval performance. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (pp. 2487-2496).

RecJPQ effect on the size/effectiveness tradeoff



- The model size can be reduced up to 50x without effectiveness loss
- Indeed, model effectiveness can actually be improved by RecJPQ, due to regularisation effect
- Efficiency:
  - Training time is similar (despite SVD)
  - Inference time is much faster

RecJPQ: Other applications

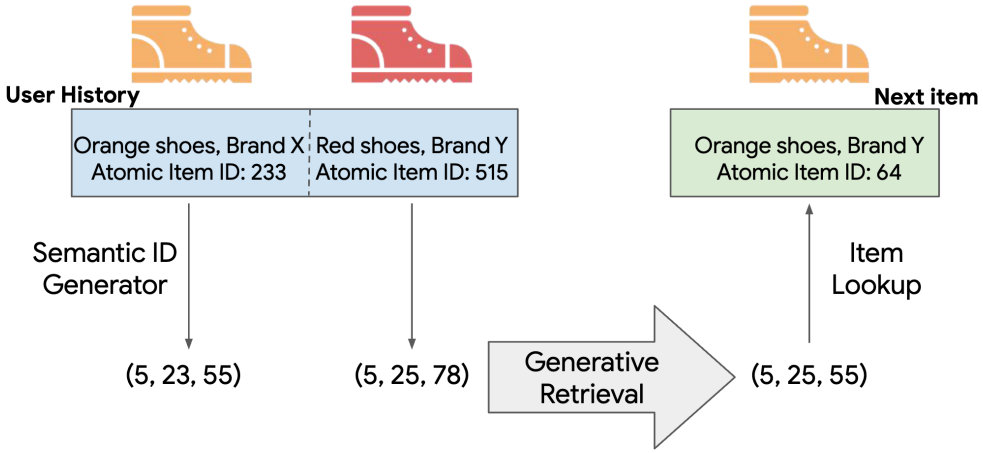
- **Efficient inference** (sub-id scores can be pre-computed).
- **Side information injection** (sub-ids can be based not only on SVD, but also use side information)
- **Generative RecSys** – same technique can be used for item tokenisation in Generative recommendation models.

Item tokenisation in Generative Recommender models

GPTRec**

- RecJPQ-style tokenisation using SVD
- Beam Search for generating item ids

TIGER**: Semantic IDs using item attributes (category, colour, etc.)



* Petrov, A.V. and Macdonald, C., 2023. Generative sequential recommendation with GPTRec. *Presented at GenIR@SIGIR2023*

** Rajput S, Mehta N, Singh A, Hulikal Keshavan R, Vu T, Heldt L, Hong L, Tay Y, Tran V, Samost J, Kula M. Recommender systems with generative retrieval. In NeurIPS

LLMs are Transformer-based sequential recommenders

- ChatGPT, Llama, and Claude are just big Transformer decoder models.
- They can be used for sequential recommendation in “Zero-Shot” mode.

You

While in the UK, the user visited cities in the following order:
London, Newcastle, Edinburgh.

Predict the next city in their trip and rank them according to probability

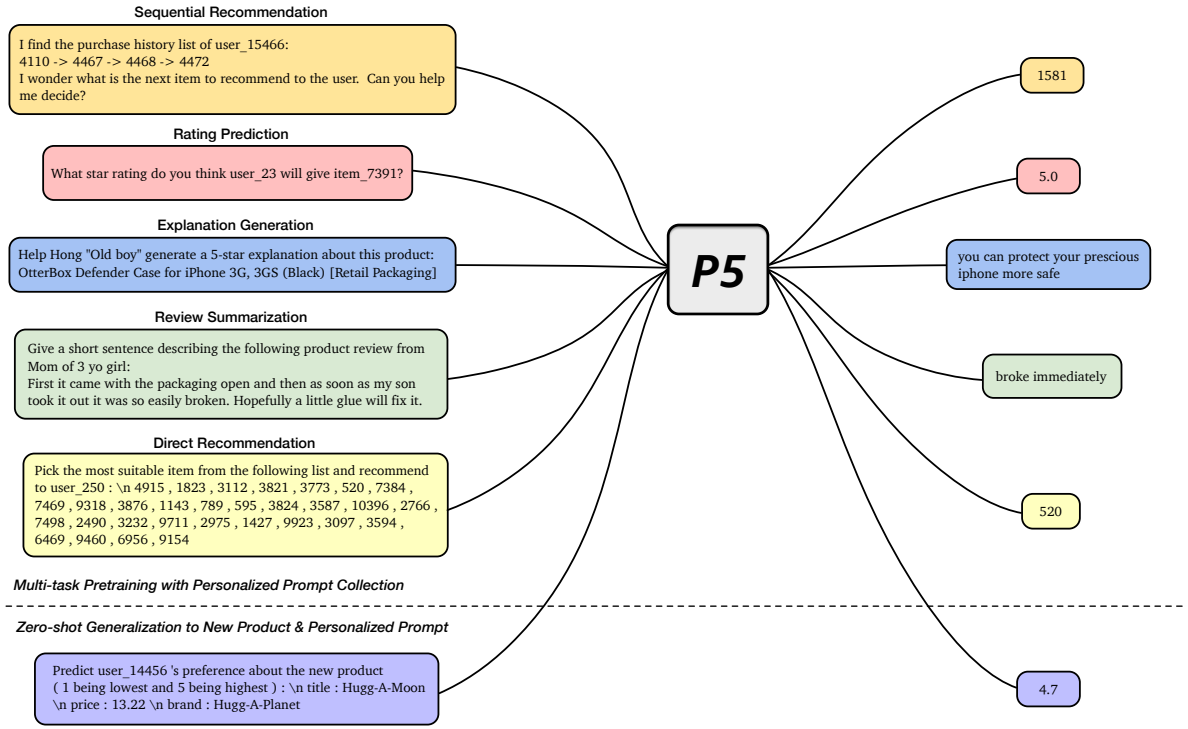
ChatGPT

To predict the next city in your trip based on the pattern of moving from London to Newcastle, then to Edinburgh, we can consider several factors such as geographical progression, cultural or historical significance, and transportation links. The pattern suggests a northward trajectory starting from the south of England (London), moving to the northeast (Newcastle), and then crossing into Scotland (Edinburgh). Assuming the journey continues in a similar direction or pattern, here are my predictions for the next city in the trip, ranked according to probability:

1. **Glasgow** - Located west of Edinburgh, Glasgow is Scotland's largest city and offers a rich cultural scene, making it a logical next destination if the trajectory continues through Scotland. It's well-connected by train and bus services from Edinburgh.
2. **Inverness** - If the journey continues further north within Scotland, Inverness could be the next stop. It's known as the gateway to the Scottish Highlands and offers access to scenic landscapes and historic sites, including Loch Ness.
3. **Aberdeen** - Situated northeast of Edinburgh, Aberdeen offers a blend of maritime history and a robust cultural scene. Its position makes it a potential next step if the trip aims to explore more of Scotland's eastern coastline.
4. **Dundee** - Located on the east coast, north of Edinburgh, Dundee could be a consideration for its burgeoning cultural institutions and proximity to the Scottish Highlands. It's relatively close to Edinburgh and offers unique attractions like the V&A Museum.
5. **Stirling** - Closer to Edinburgh than the other cities listed, Stirling offers historical significance with its castle and battle sites. It could be a shorter trip aimed at exploring more of Scotland's rich history.

These predictions are based on a combination of the geographical trajectory of your trip so far and the assumption that you are exploring major cities with cultural, historical, or natural significance. The actual next city could vary based on personal interests, time constraints, and other factors.

Pretrain, Personalized Prompt & Predict Paradigm (P5)*



- Cast many recommendation tasks as a text prediction.
- Use item IDs and user IDs as text strings.
- Many recent works followed the same idea, but the results are not consistent.

*Geng, S., Liu, S., Fu, Z., Ge, Y. and Zhang, Y., 2022, September. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems* (pp. 299-315).

Large Language Models are Zero-Shot Rankers for Recommender Systems*

Observation 1. LLMs struggle to perceive the order of the given sequential interaction histories. By employing specifically designed promptings, LLMs can be triggered to perceive the order of historical user behaviors, leading to improved ranking performance

Observation 2. LLMs suffer from position bias and popularity bias while ranking, which can be alleviated by bootstrapping or specially designed prompting strategies.

Observation 3. LLMs have promising zero-shot ranking abilities, especially on candidates retrieved by multiple candidate generation models with different practical strategies.

* Hou, Y., Zhang, J., Lin, Z., Lu, H., Xie, R., McAuley, J. and Zhao, W.X., *In ECIR 2024*

Topics not covered in this tutorial

- Graph-Based Recommender Systems
- Conversational Recommender Systems
- Multi-modal Recommender systems
- Reciprocical Recommenders

And So On…

Must Read Papers

Generalisation of MF

- He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T.S., 2017, April. Neural collaborative filtering. In Proc. WWW
- Xue, H.J., Dai, X., Zhang, J., Huang, S. and Chen, J., 2017, August. Deep matrix factorization models for recommender systems. In Proc. IJCAI

Challenging Deep Learning

- Ferrari Dacrema, M., Cremonesi, P. and Jannach, D., 2019, September. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. Proc. RecSys

Variational Autoencoders

- Liang, D., Krishnan, R.G., Hoffman, M.D. and Jebara, T., 2018, April. Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).
- Shenbin, I., Alekseev, A., Tutubalina, E., Malykh, V. and Nikolenko, S.I., 2020, January. RecVAE: A new variational autoencoder for top-n recommendations with implicit feedback. In Proc. WSDM.

Practical Deep Learning Models

- Covington, P., Adams, J. and Sargin, E., 2016, September. Deep neural networks for youtube recommendations. In Proc. RecSys
- Wang, R., Fu, B., Fu, G. and Wang, M., 2017. Deep & cross network for ad click predictions. Proc. AKDD

Must Read Papers

RNNs for RecSys

- Hidasi, Balázs, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 'Session-Based Recommendations with Recurrent Neural Networks', 2016. In Proc. ICLR

Convolutional Neural Networks for RecSys

- Tang, J. and Wang, K., 2018, February. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proc. WSDM
- Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M. and He, X., 2019, January. A simple convolutional generative network for next item recommendation. In Proc. WSDM

Transformers for RecSys

- Kang, W.C. and McAuley, J., 2018, November. Self-attentive sequential recommendation. In Proc. ICDM
- Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W. and Jiang, P., 2019, November. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proc. CIKM

LLMs for RecSys

- Geng, S., Liu, S., Fu, Z., Ge, Y. and Zhang, Y., 2022, September. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In Proc. RecSys
- Hou, Y., Zhang, J., Lin, Z., Lu, H., Xie, R., McAuley, J. and Zhao, W.X., 2024, March. Large language models are zero-shot rankers for recommender systems. In Proc. ECIR
- Zhu, Y., Wu, L., Guo, Q., Hong, L. and Li, J., 2024, May. Collaborative large language model for recommender systems. In Proc. WWW

Key Take Aways

- The most advanced models are Neural Networks based these days
- With neural architectures, there is no much difference between IR and RecSys
- SOTA models for “classic” recommendation are based on Variational Autoencoder
- Users’ behaviour can be seen as a specific “language” with the interactions being “words”
- LLMs are coming, but many unsolved problems.

RecSys Summer School 2024

Supported by ACM RecSys, SIGCHI, ACM Europe, EURAI, and University of Cagliari

October 8-12, 2024, Bari, Italy

Q&A

Acknowledgments

Some of the slides were prepared by prof. Craig Macdonald

@asash