University of Glasgow | Terrier Team

ECIR 2024 GLASGOW

Transformers for Sequential Recommendation

Aleksandr "Sasha" Petrov and **Craig Macdonald**
University of Glasgow

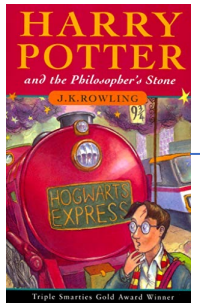
The 46th European Conference on Information Retrieval
Glasgow, Scotland

Sequential Recommendation

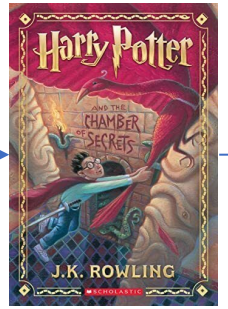
University of Glasgow | Terrier Team

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

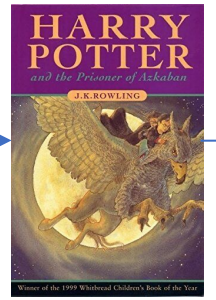
Example:



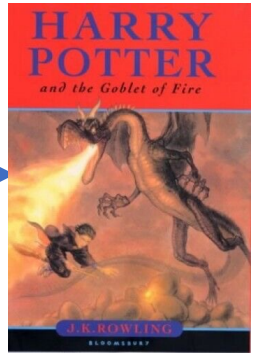
Harry Potter and the Philosopher's Stone



Harry Potter and the Chamber of Secrets



Harry Potter and the Prisoner of Azkaban



Recommended: Harry Potter and the Goblet of Fire

Tutorial Goals

To...

- … provide attendees a **theoretical overview** and **practical recommendations** for building sequential recommender systems using recent advancements in deep learning, language models and their adaptations to recommendation problems

- … provide information on the practical aspects of **scaling sequential recommender systems** to **large catalogues** with millions of items, which are common in industrial applications such as e-commerce websites and music streaming services

- ... cover the most recent advancements in **generative recommender models** and intends to provide the attendees with the vision and ideas for further research.

Intended Learning Objectives

University of Glasgow | Terrier Team

Part 1: Classic models ("score-and-rank")

→ **ILO 1A.** Describe the problem of Sequential Recommendation and draw the parallels with the problem of language modelling.

→ **ILO 1B.** Understand the Transformer architecture, Transformer Encoder-based models (BERT) and Transformer Decoder-based models (GPT).

→ **ILO 1C.** Describe classic transformer-based sequential recommender models, SASRec and BERT4Rec, and understand their similarities and differences.

→ **ILO 1D.** Understand the techniques of training the models for large catalogues: training objectives, negative sampling, and model compression.

Part 2: Generative models

→ **ILO 2A.** Describe sequential recommendation as a generative task and draw parallels with language generation and generative transformers.

→ **ILO 2B.** Describe the whole recommendation list generation problem and optimisation for beyond-accuracy goals and describe listwise generative models, such as GPTRec and P5.

→ **ILO 2C.** Understand the role of Large Language Models (LLMs) in the future development of recommender systems.

PDF slides available at https://github.com/asash/transformers-for-recsys-tutorial

Presenter Biographies

Aleksandr “Sasha” Petrov

PhD candidate at the University of Glasgow
Thesis title: Effective and Efficient Transformer Models for Sequential Recommendation
Publications in RecSys, ECIR, WSDM, ToRS.

Came to academia from industry; over 10 years industrial experience, including Senior Engineer @ Amazon and CTO of a RecSys startup.

Craig Macdonald

Professor of Information Retrieval at the University of Glasgow

>250 publication in information retrieval and recommendation systems venues. Has taught courses on both information retrieval and recommender systems at undergraduate and postgraduate level. Given various tutorials at IR conferences, including ECIR, CIKM.

Joint line of work on Transformers for Sequential RecSys: ~10 published papers, Best Paper Award @ RecSys 2023, Best Student Paper Nomination @ RecSys 2022

Schedule

- Organisational information [5 minutes] ✓
- Introduction to sequential recommendation and early RecSys models [35 minutes]
- Transformer architecture and key Transformer-based RecSys models [25 minutes]
- Training Objectives [15 minutes]
- Q&A [10 minutes]

☕ **30 minutes break**

- Loss Functions, Negative sampling for large-scale recommendations [30 minutes]
- Item tokenisation [15 minutes]
- Generative Sequential Recommendation for Beyond-Accuracy goals [20 minutes]
- LLMs for sequential recommendation [15 minutes]
- Q&A [10 minutes]

Part 1: Sequential Recommendation: Motivation and early models

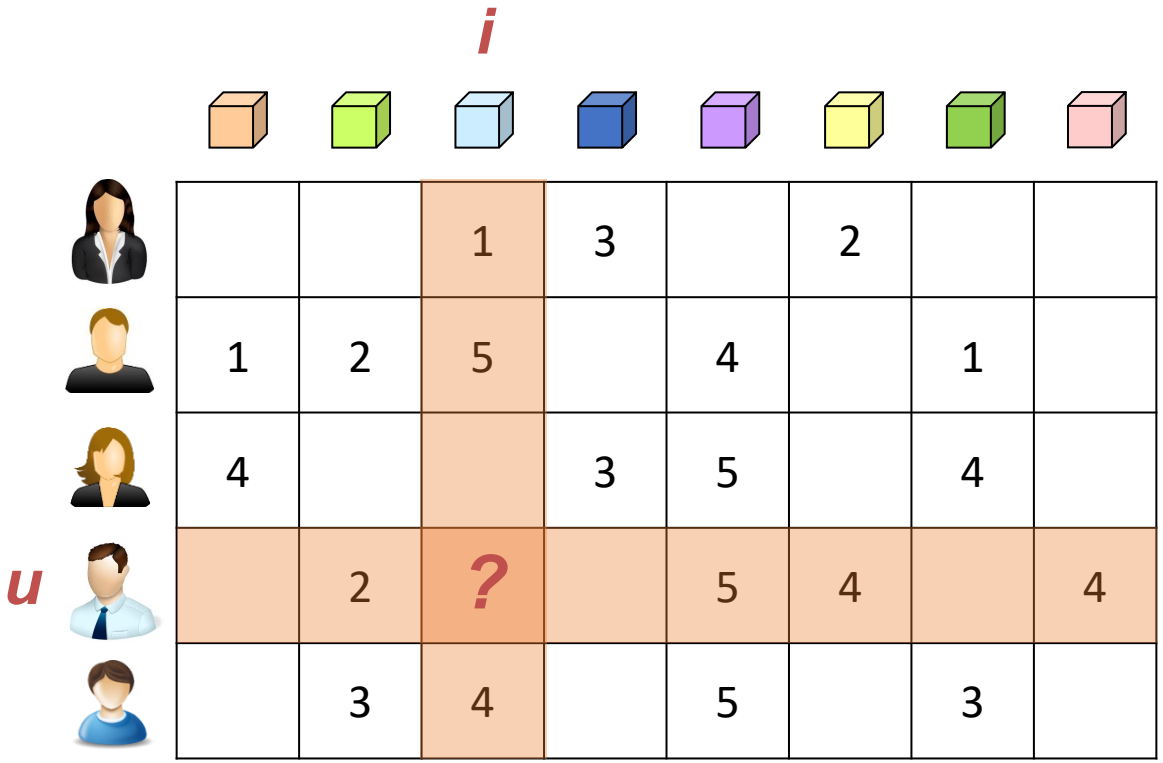
Traditional Recommendation

- Recommender systems have access to many forms of data
  - *Collaborative* information, i.e. interactions by users on items
    - Such interactions can be *explicit* actions indicating preferences (e.g. ratings) or *implicit* feedback (clicks, purchases, views vs. skips, fast forwards)
  - *Content* information about items
    - Textual features, genre (GPS position)
  - User *context* (as much as can be observed)…
    - Time of day, device, GPS position etc.

- Our focus today is entirely on **collaborative** information
  - Which items did a user interact with, (and in what order…)
  - There are many extended recommendation models for context and content *side information*

Recommendation as Matrix Completion*

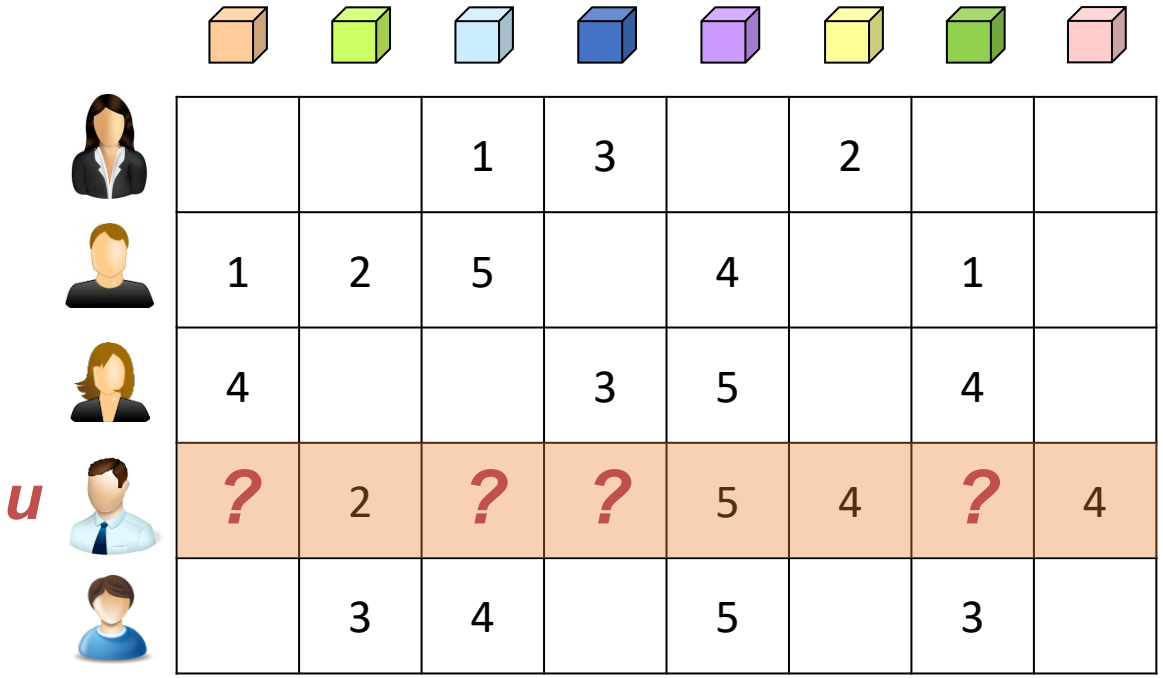
- *Predict how much user **u** will like item **i***



buys
clicks
views
rates
reviews
…

Recommendation as Matrix Completion

- *Recommend items that user **u** will like*

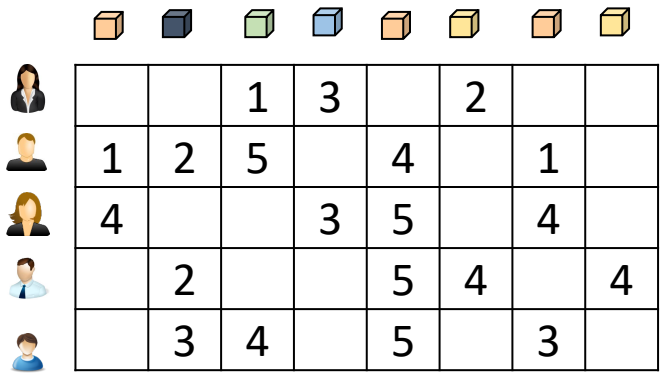


buys
clicks
views
rates
reviews
…

Classical Methods

- How will I like item *i*?
  - User-based Collaborative Filtering
    - *How do similar users like item i?*
    - Find neighbours for the target user – predict the items that a user might like on the basis of ratings given to that item by other users who have similar taste with that of the target user.
    - Recommendation must be done online – no pre-offline computation
  - Item-based Collaborative Filtering
    - *How do I like items similar to i?*
    - Look for similar items (rated by the same people): predicted rating of target item is the item-item similarity weighted average rating of neighbours of items already rated by the target user
- Item-based CF is preferred as item neighbourhoods are more stable, and can be pre-computed
  - This led to matrix factorisation methods…

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

CF

- Problem?
  - Predicted utility of *i* ignores context of the user

Our best selling sauce

Causality

Frequently bought together

This item: Einhell 4430840 230 mm Angle Grinder TE-AG 230/2000, 2000W, Soft Start,…

£54.49 ✓prime

Silverline 589673 Concrete and Stone Cutting Diamond Blade 230 x 22.23 mm Segmented Rim

£11.92 ✓prime

Blue Spot 2 Piece 9 inch Diamond Cutting Disc Set

£14.99 ✓prime

Total price: £81.40

Add all 3 to Cart

Causality is because previous interactions by users follow a *sequence*…

When does MF 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 → 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.

Early works on Sequential Recommendation

Early Approaches

Association Mining*: count frequency of association between items:

- How many transactions contain “Ice Cream” that also contain “Raspberry Sauce”?
- Measures:
  - Support (fraction of transactions containing both X and Y – c.f. joint probability)
  - Confidence (fraction of transactions containing Y that also contain X, c.f. conditional probability)
  - Lift**: Are X and Y more likely to occur together than separately? A measure of dependency

Non-personalised…

$$\hat{r}_{xy} \propto \frac{P(X \cap Y)}{P(X)P(Y)}$$

*Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. *In SIGMOD 1993*

**McNicholas, P.D., Murphy, T.B. and O’Regan, M., 2008. Standardising the lift of an association rule. *Computational Statistics & Data Analysis*, *52*(10), pp.4712-4721.

Markov Chains*

- More generally, consider the conditional probability

$$P(i_2 \mid i_1)$$

c.f. what is the likelihood that item i_2 is interacted with after interaction with item i_1 … *confidence*.

- More generally, given a sequence of n interactions $\langle i_1, i_2, \ldots i_n \rangle$, find the user interacting with the item i_{n+1}: $P(i_{n+1} \mid i_1 .. i_n)$

- If the model only considers last k interactions, then it is a k-order Markov chain:

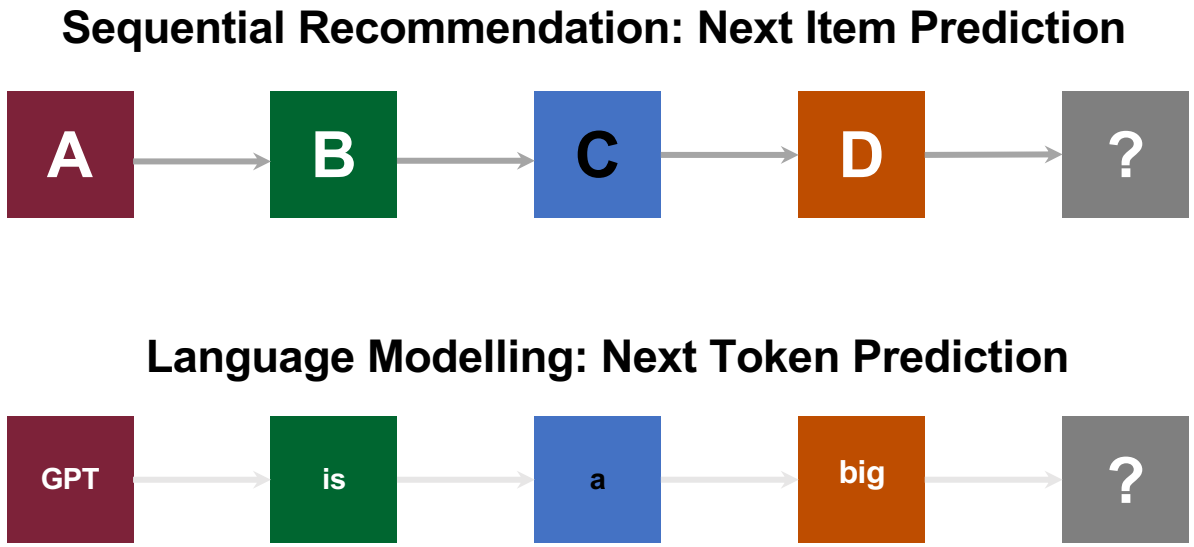
$$P(i_{n+1} \mid i_{n-k} .. i_n)$$

- We can calculate all of these by counting observations in our training data…, but if *k* is large, the training data becomes very sparse

A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In UAI '01

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. $P(t_{n+1} \mid t_{n-k} \ldots t_n)$



∴ It makes sense to adapt language models to sequential recommendation

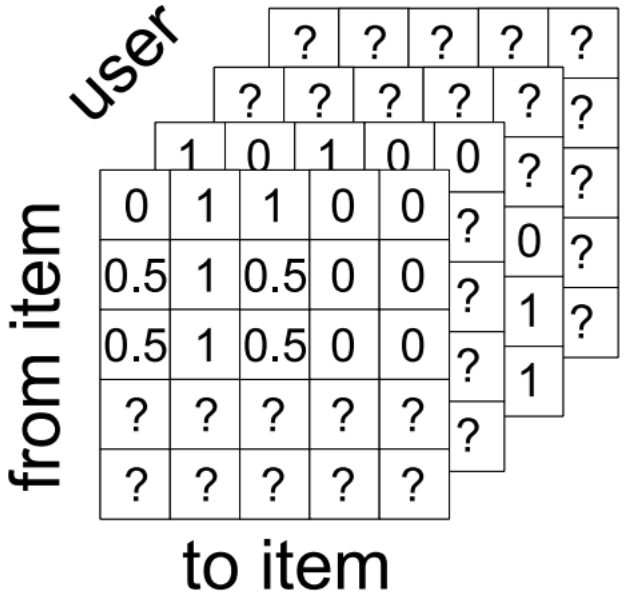
Consequences

- Early language models were counting based…
  E.g. N-gram models, perhaps with smoothing

- In recent years, there has been a move from counting-based language models, to those that can *predict,* based on a model that has learned some language characteristics
  RNNs
  Transformers

- The same transition has occurred in sequential recommendation models (where tokens = items)
  What works for language models often works for sequence recommendation…
- But what else did people try before adapting language models…

Factorising Personalized Markov Chains



- 1-st order Markov chain computes the probability of the next item given a previous item.
- This probability is user-dependent
- The FPMC algorithm is a generalisation over MF, that factorises the 3d tensor (source item, destination item, user) cube.
- **Cons:** No dependencies beyond 1st order, need to re-train model for every new user.

Rendle et al., Factorizing Personalized Markov Chains for Next-Basket Recommendation, in WWW 2010

Summary

- Early models were simplistic and could not model complex non-linear dependencies (e.g. only first order Markov chains)
  - E.g – buying green trousers is a weak indication of preference, but buying green trousers and a green shirt may indicate a preference or trend

- Some still remain usable as strong baselines.
- Since 2016, insights from Language Modelling have offered increasingly effective recommendation models.
  - Items =~ Tokens
- However, as we will show, the parallels are not all-encompassing; by thinking more carefully about the differences, further improvements can be attained

Img Src: Openclipart

ILO 1A. Sequential Recommendation and the Parallels with Language Modelling.

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:

$$f(h) \mapsto g$$

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

Chronologically ordered sequence of past interactions

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

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

University of Glasgow | Terrier Team

- **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.

What models are the best in predicting probabilities?

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

A Recipe for Sequential Recommendation

University of Glasgow | Terrier Team

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

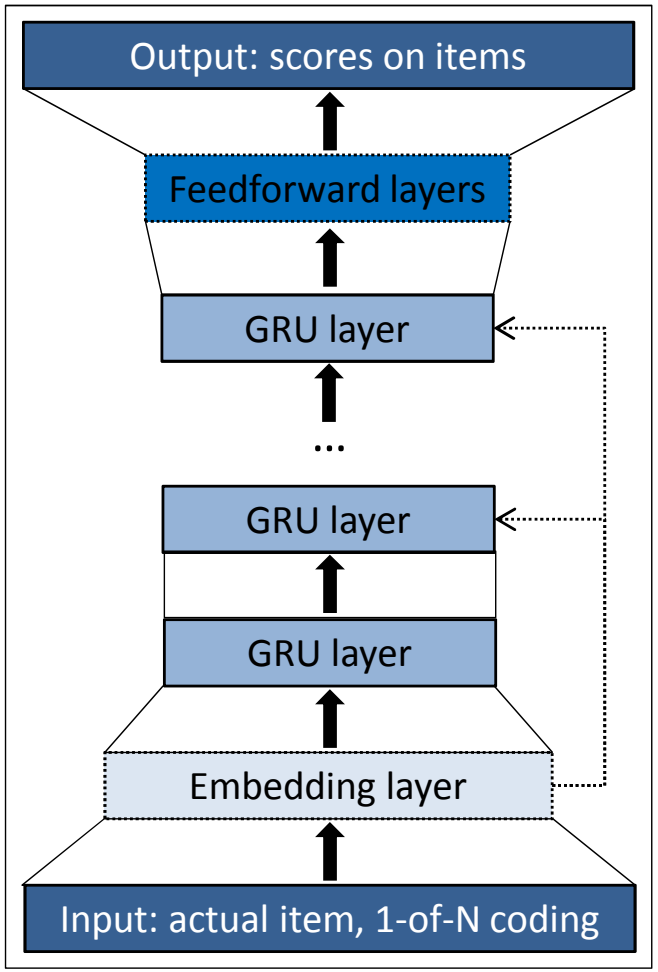
2. We have a **training objective** that defines how we show training sequences (including positive items) to the model, which the model aims to correctly complete

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



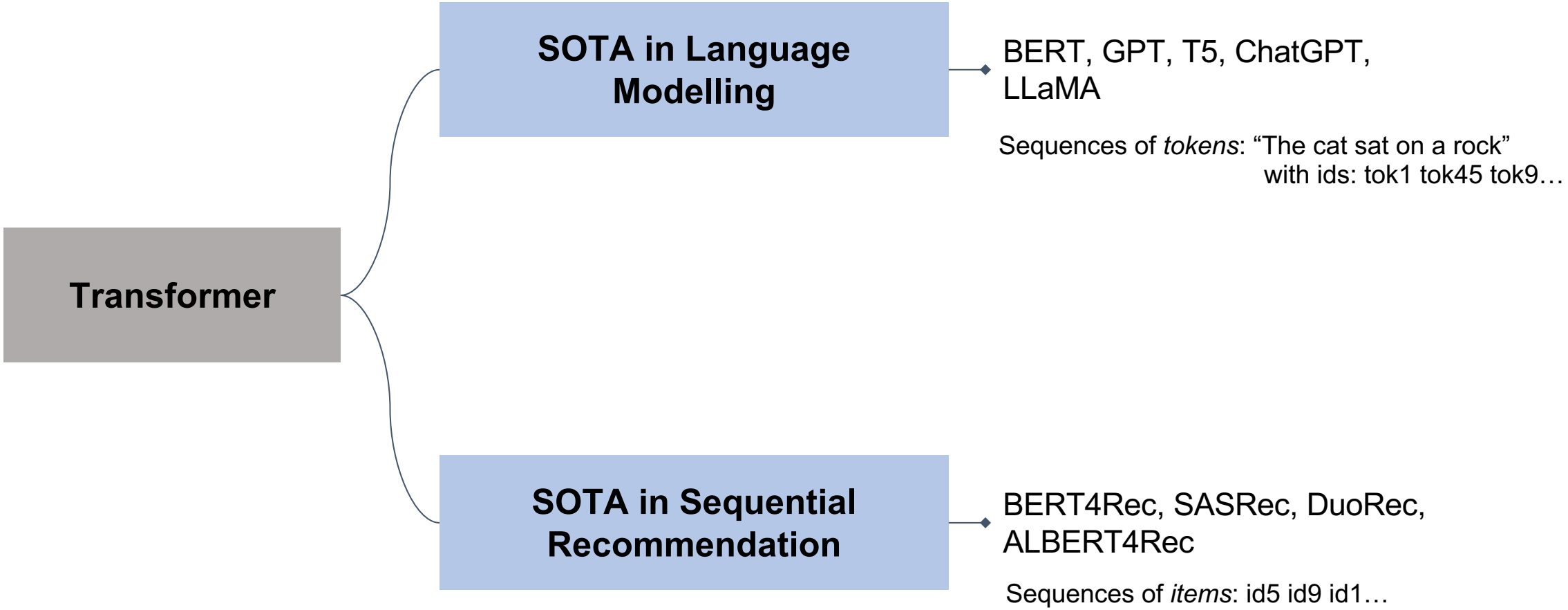
Cons:

- 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*

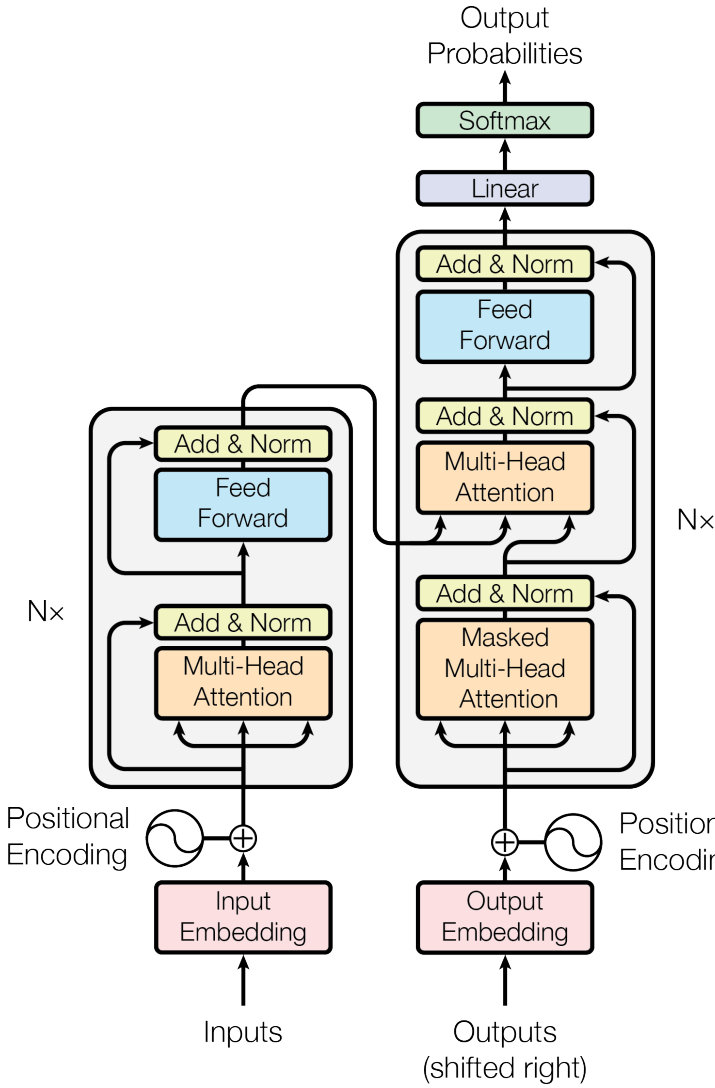
Transformers



ILO 1B. Transformer architecture.

Original Transformer* Model

- Originally developed for machine translation.
- It consists of 2 parts:
  Encoder - builds a representation of input text in the source language.
  Decoder - generates output in the destination language autoregressively (token-by-token).
- The architecture has been shown to outperform other architectures in most language-related tasks.
- Achieved markable improvements over the previous generation of the models by using the *Self-Attention mechanism*



* Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *NeurIPS*.

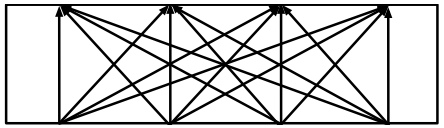
Self-Attention

Self-Attention is a key mechanism within Transformer architecture

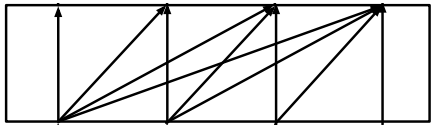
- **Input:** a sequence of embeddings
- **Output:** a sequence of embeddings (usually the same size as input)

Main idea: compute new embeddings as a weighted sum of old embeddings. **Similar** embeddings have higher weights.

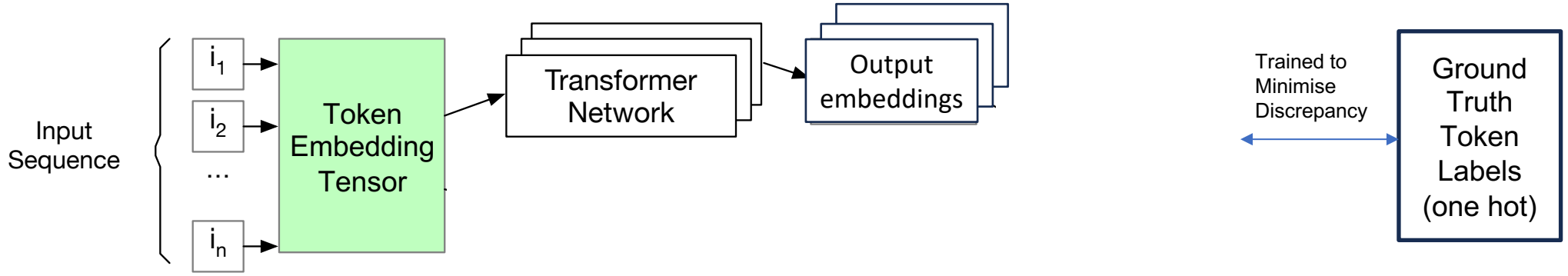
- **Full Self-Attention**: All items in sequence have a non-zero weight in every item's new representation (used in Transformer Encoder, e.g. BERT)



- **Causal Self-Attention**: Only preceding items have non-zero weight in the item's new representation. (used in Transformer Decoder, e.g. GPT)

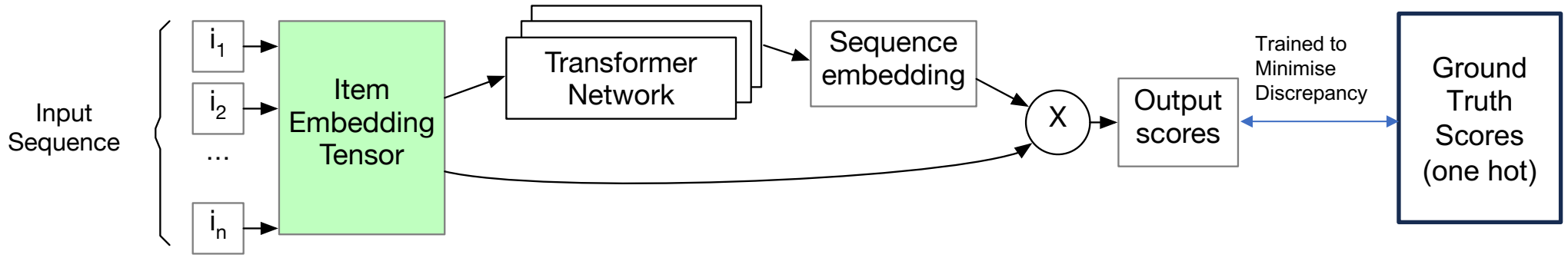


Architecture of a typical Transformer-based model

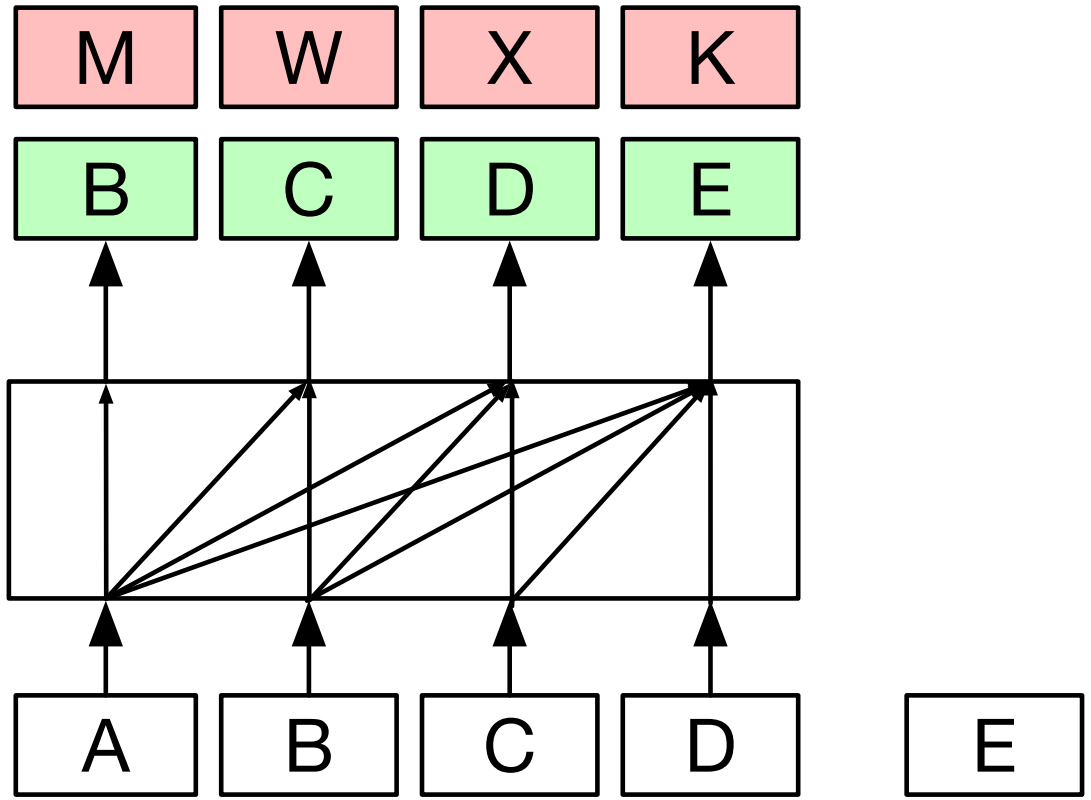


ILO 1C. Transformer-based recommendation models.

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



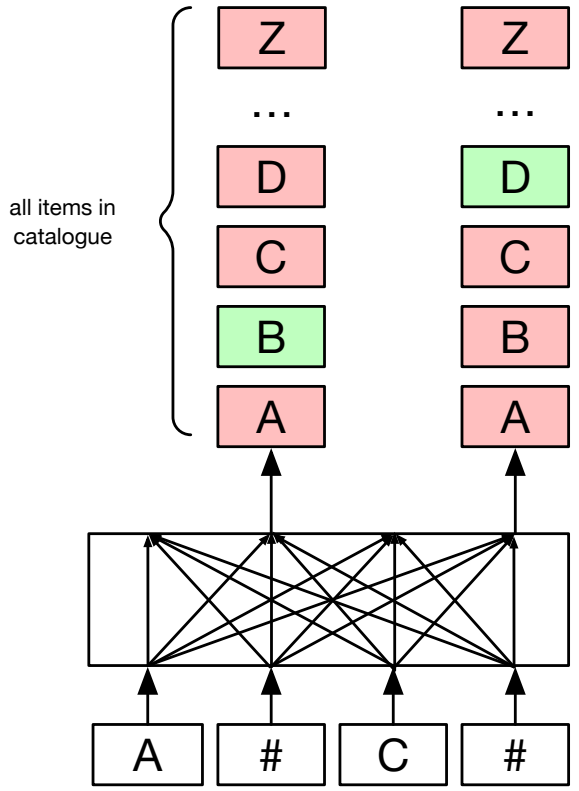
SASRec*



- 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.TransformerDecoder` 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*

Example: BERT4Rec with Hugging Face Transformers*

https://huggingface.co/docs/transformers/en/index

BERT4Rec with HuggingFace: Creating the Model

```
from transformers import BertConfig, BertForMaskedLM
import torch
num_items = 100
embedding_size = 144 # } (768)
sequence_length = 5  # } (512)
num_layers = 3       # } much smaller than bert-base, say (12)
num_heads = 4        # } (12)

mask_item_id = 0     # what we want to predict

config = BertConfig(
    vocab_size = num_items + 2, #+1 for padding; +1 for masking
    hidden_size = embedding_size,
    num_hidden_layers = num_layers,
    num_heads = num_heads,
    max_position_embeddings = sequence_length,
)
bert = BertForMaskedLM(config)
```

BERT4Rec with HuggingFace: computing loss

Sequence of User-Item interactions: ⟨4, 1, 6, 5 ⟩

Selected for masking

Padding

Masked items

-100 means that this label should be ignored for loss (e.g. paddings, masks)

```
masked_seq = torch.tensor([[100, 0, 1, 0, 5]])
labels = torch.tensor([[-100, 4, -100, 6, -100]])
bert_output = bert(input_ids=masked_seq, labels=labels)
bert_output.loss
```

>> tensor(4.8681, grad_fn=<NllLossBackward0>)

BERT4Rec with HuggingFace: Computing Recommendations

Sequence of User-Item interactions: ⟨1, 6, 5 ⟩

For inference, we add “mask” (id 0) to the end of the input sequence and pad (id 100) the sequence to the correct length: ⟨**100**, 1, 6, 5, **0** ⟩

```
masked_seq = torch.tensor([[100, 1, 6, 5, 0]])
bert_output = bert(input_ids=masked_seq)
bert_output.logits.shape
```

>>torch.Size([1, 5, 102]) #[batch_size, sequence_length, num_items + 2]

Recommendation scores are the logits corresponding to the last (masked) element:

```
recommendations = bert_output.logits[:,-1]
recommendations.shape
```

>>torch.Size([1, 102]) #[batch_size, num_items+2]

Post-processing:

- Sort by score
- Remove special items (mask, pad)
- Optional: remove already interacted items

Systematic Review and Replicability Study of BERT4Rec vs. SASRec

Systematic Review of BERT4Rec *

- We studied 370 papers citing BERT4Rec, keeping only those including SASRec, and excluding unpublished/MSc theses etc. Final analysis on 40 papers

Dataset	Number of papers	BERT4Rec wins	SASRec wins	Ties
Amazon Beauty*	19	12 (63%)	5 (26%)	2 (11%)
ML-1M*	18	13 (72%)	3 (17%)	2 (11%)
Yelp	10	6 (60%)	4 (40%)	0 (0%)
Steam*	8	7 (88%)	1 (12%)	0 (0%)
ML-20M*	8	7 (88%)	0 (0%)	1 (12%)
Total (including all other datasets)	134	86 (64%)	32 (23%)	16 (12%)

- ✅ H1: BERT4Rec is not consistently superior compared to SASRec in the published literature
- ✅ H2: A large number of papers fail to replicate BERT4Rec's significant enhancement of SASRec on several datasets from original publication

* - dataset used in the original BERT4Rec publication

*Petrov A, Macdonald C. A systematic review and replicability study of bert4rec for sequential recommendation. *In RecSys 2022*

Replicability analysis of available BERT4Rec implementations

University of Glasgow | Terrier Team

Can we replicate originally reported popularity-sampled Recall@10 using the available open-source implementations with default configuration?

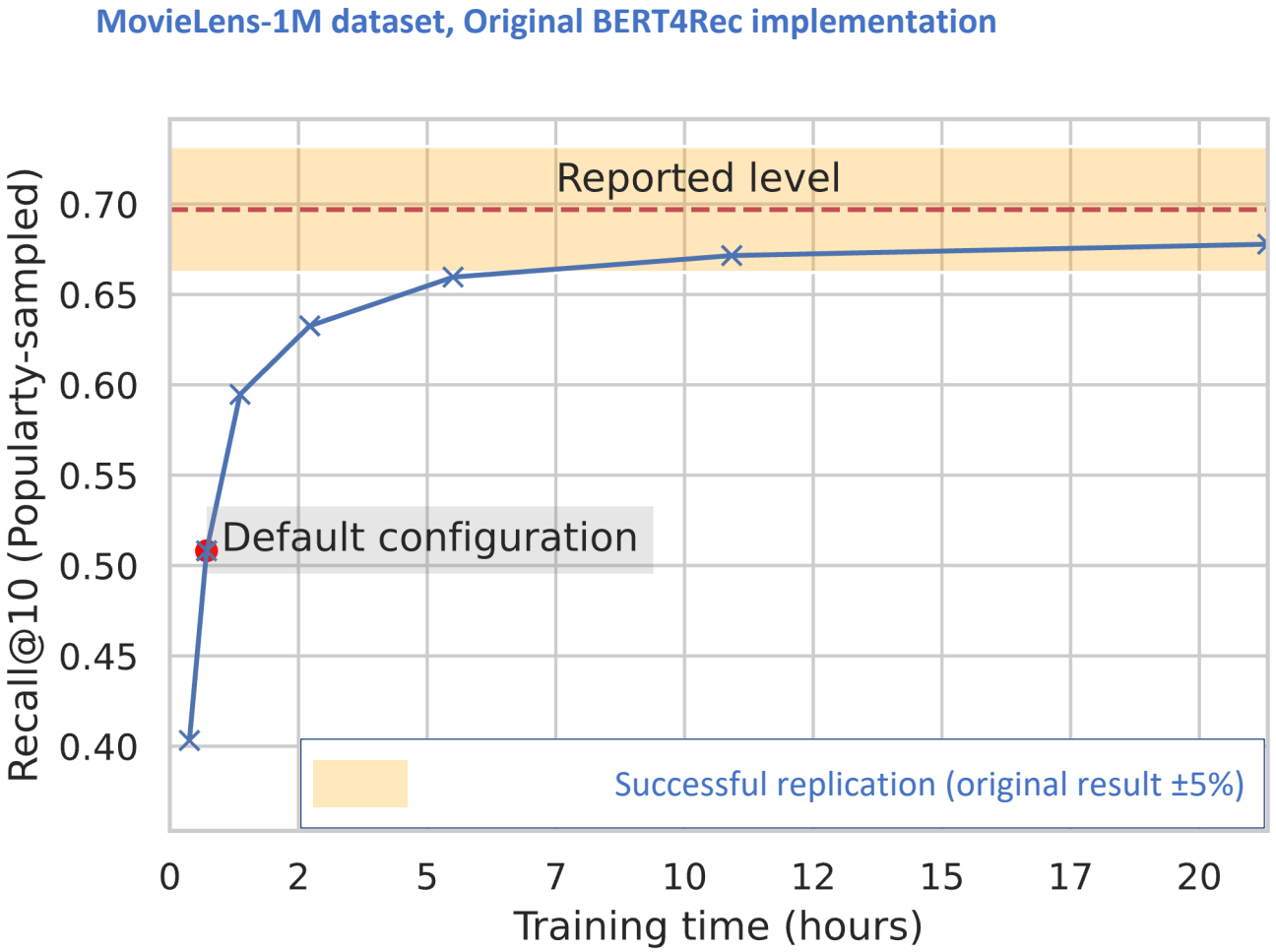
✅ - Successful replication within 5% tolerance

❌ Unsuccessful replication

Implementation	MovieLens-1M	Steam	Amazon Beauty	MovieLens-20M
Original	❌	❌	❌	❌
RecBole	❌	❌	❌	❌
BERT4Rec-VAE	✅	❌	❌	✅
Ours*	✅	✅	❌	✅

* Based on Hugging Face Transformers, Inspired by Transformers4Rec (Moreira et al.)

Effect of training time on BERT4Rec performance



- The original BERT4Rec paper doesn't describe how much training is needed to train the model.
- Number of training steps specified in the original BERT4Rec code is too small and leads to underfitting.
- Replicating the originally reported results with the original code requires 10-20 hours on the MovieLens-1M dataset.

Empirical Reproducibility Findings

- The default configuration of BERT4Rec implementations lead to underfitting, and the original BERT4Rec paper was not clear on the required level of training
- Our HuggingFace Transformers version of BERT4Rec replicates originally reported results on 3/4 datasets. It can be further improved by adapting other available architectures from the Hugging Face Transformers library
- When trained for a **sufficient amount of time**, BERT4Rec exhibits state-of-the-art performance comparable to the best results reported in the literature
- Other BERT-like encoder architectures can outperform outperform BERT4Rec, e.g. ALBERT, DeBERTa

Models that build upon SASRec and BERT4Rec

Contrastive Learning Methods

Augmentation – usually more input data out of the same training instances

Then provide an additional loss function that ensures similar things are similar and different things are different – i.e. robust to noise

Using a contrastive objective, there is less overfitting, acting as a regularisation

For example:

- Different (augmented) representations of the same sequence are similar to each other.
- Representations of different sequences aren't similar to each other.
- Contrastive objective helps to stabilise training and converge faster.

Examples:

CL4SRec*: data-level augmentation (cropping, masking, reordering)

DuoRec**: model-level augmentation (dropout-based)

CBiT*:** model & masking augmentation (dropouts and masking)

*Xie, X. et al., Contrastive learning for sequential recommendation. *In ICDE 2022*

**Qiu, R., Huang, Z., Yin, H. and Wang, Z., Contrastive learning for representation degeneration problem in sequential recommendation. In WSDM 2022

***Du, H., Shi, H., Zhao, P., Wang, D., Sheng, V.S., Liu, Y., Liu, G. and Zhao, L. Contrastive learning with bidirectional transformers for sequential recommendation. In CIKM 2022.

Side Information

BERT4Rec & SASRec learn item representations from scratch.

However, often there is additional information & context that is available.

- **Time intervals between consecutive interactions**: TiSASRec*
- **Item categories**: CaFe **
- **Textual descriptions of the items:** Recformer ***
  - The titles of the items in the sequence are input to a BERT models; the CLS embedding is the sequence representation
  - Biencoder – items have already been encoded
- **Transformers4Rec** **** – a framework from NVIDIA for integrating side information in transformers

* Li J, Wang Y, McAuley J. Time interval aware self-attention for sequential recommendation. In WSDM 2020
** Li J, Zhao T, Li J, Chan J, Faloutsos C, Karypis G, Pantel SM, McAuley J. Coarse-to-fine sparse sequential recommendation. In SIGIR 2020
*** Li J, Wang M, Li J, Fu J, Shen X, Shang J, McAuley J. Text is all you need: Learning language representations for sequential recommendation. In KDD2023
**** de Souza Pereira Moreira G, Rabhi S, Lee JM, Ak R, Oldridge E. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In RecSys 2021

So… end of tutorial?

Items are not Tokens

Observation 1:

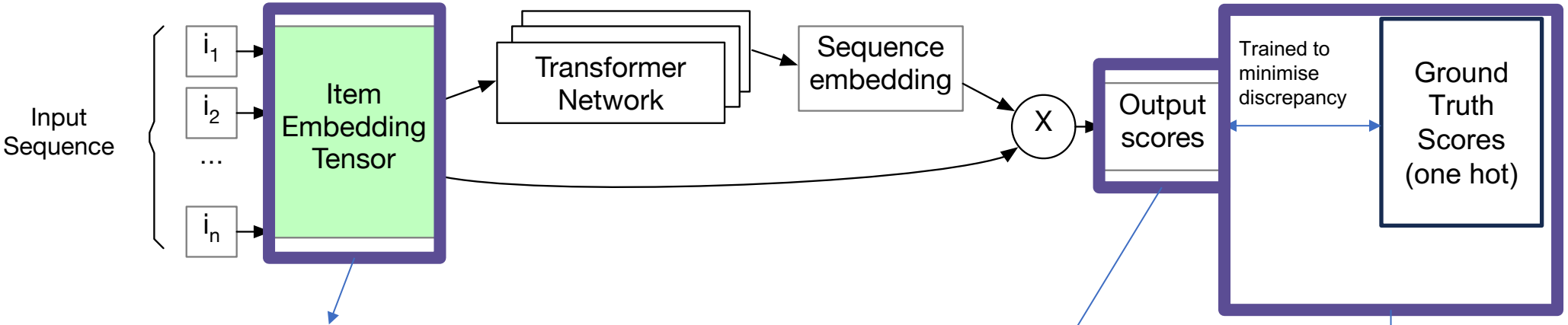
- Unlike in language modelling where tokens mean the same thing across corpora, items differ across recommendation datasets → No pre-training/fine-tuning.

Observation 2:

- The number of items in a recommender catalogue can be much larger compared to the number of tokens in language models.

ILO 1D. Transformer-based recommenders for large catalogues.

The main challenges with large-scale transformers



3. The item embedding tensor is too large.
(large catalogue)
↓
Embedding tensor compression is required

ILO 1D.3

2. Computing all scores is expensive during training
(large catalogue)
↓
Negative sampling is needed

ILO 1D.2

1. Training is slow.
(no pre-training)
↓
The training objective is important

ILO 1D.1

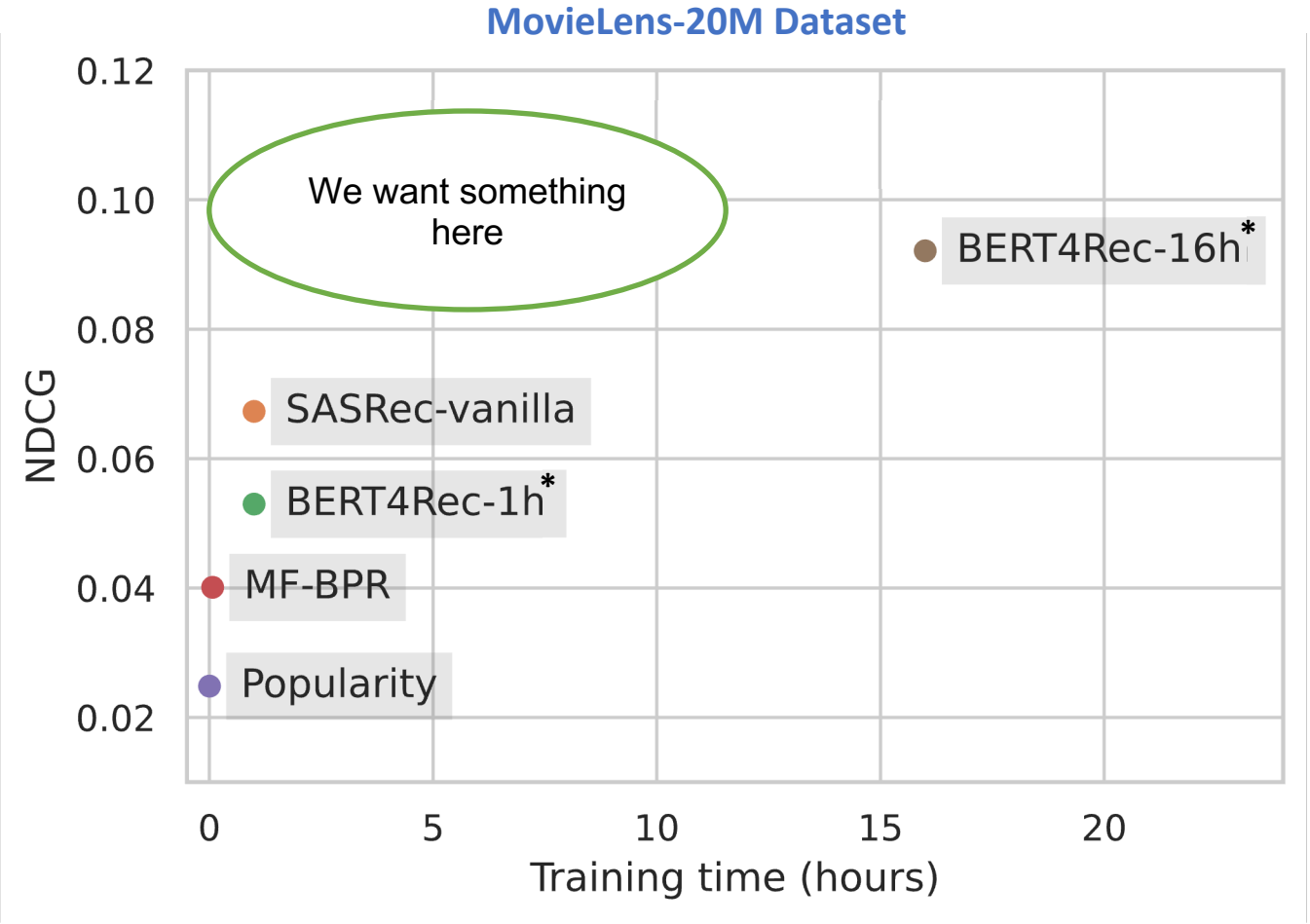
ILO 1D.1: Training objectives for Effective and Efficient training

RSS: Effective and Efficient Training for Sequential Recommendation using Recency Sampling

Aleksandr Petrov and Craig Macdonald

ACM RecSys 2022 (Best Student Paper Nominee); ACM Transaction on Recommender Systems (Highlights of RecSys '22)

Effectiveness-Efficiency Tradeoff



- Existing models are either effective (NDCG) or efficient (Training time)
- **Can we build a model, which is both effective AND efficient at the same time?**
- We observe that SASRec & BERT4Rec differ in their **training objectives**

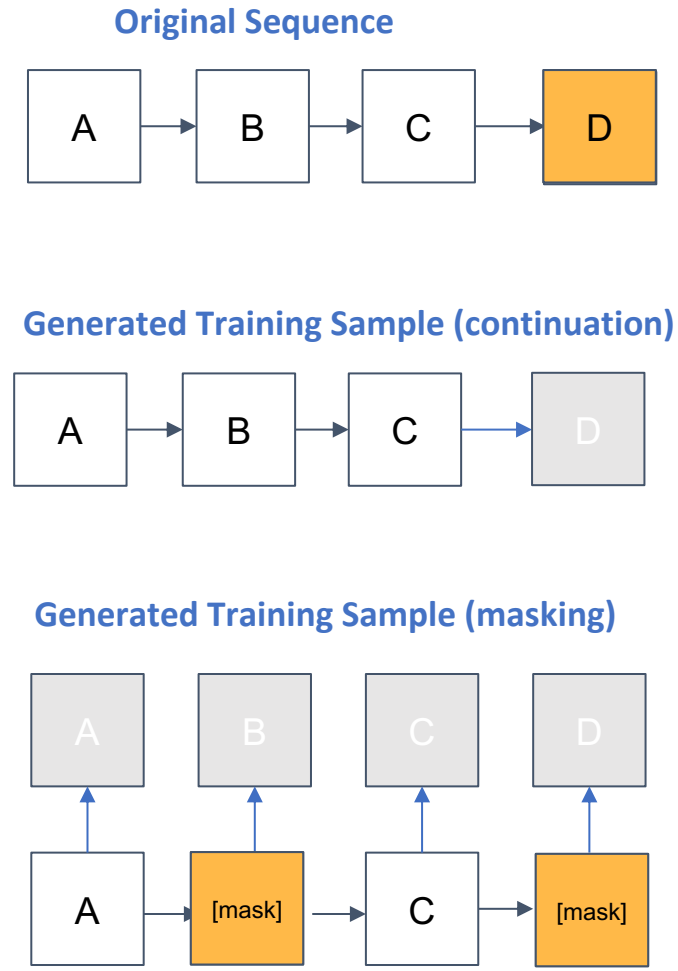
For more details on BERT4Rec's Effectiveness/Efficiency tradeoffs, see: *Petrov, A. and Macdonald, C., A systematic review and replicability study of BERT4Rec for sequential recommendation. *ACM RecSys* 2022.

Training Objectives

- There are two main classes of training objectives for Sequential Recommendation:

- **Sequence Shifting/Continuation** (GRU4Rec, Caser, SASRec)
  - Closely related to the recommendation goal → Fast to converge
  - Doesn't use training data efficiently – one sequence, one training instance

- **Item Masking** (BERT4Rec):
  - Uses training data efficiently – **multiple** positives per training sequence
  - Achieves SOTA
  - Only weakly related to the recommendation goal → Slow to converge



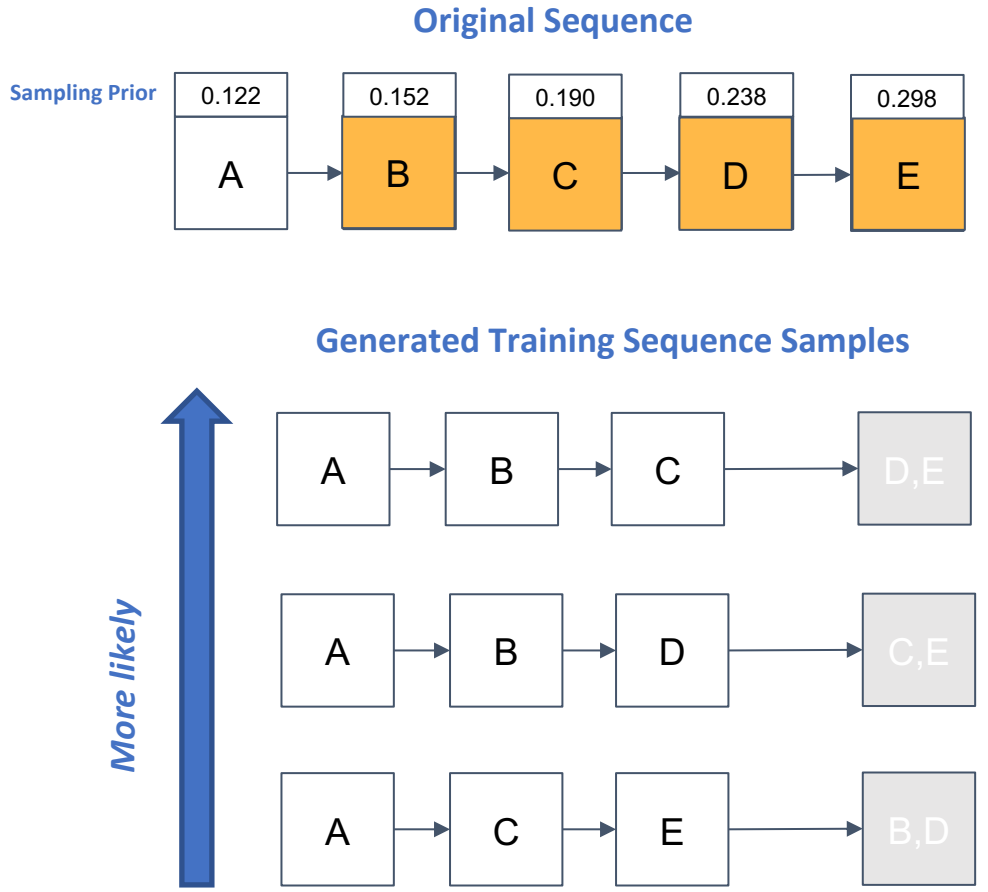
Recency Sampling of Sequences (RSS)

- We desire a new training objective that finds a balance between the desirable but opposing attributes of both sequence continuation and item masking:

  (a) Any item can be selected as positive (as per masking) → efficient use of training sequences
  (b) More recent interactions are used as positives (as per sequence continuation) → these are more realistic targets for learning sequence completion

- Our proposed **Recency Sampling of Sequences** *probabilistically* selects positives such that more recent items have increased chances of being selected

- The sampling probability is controlled by a *recency importance function* that depends on the item's position in the sequence (e.g., increases exponentially)



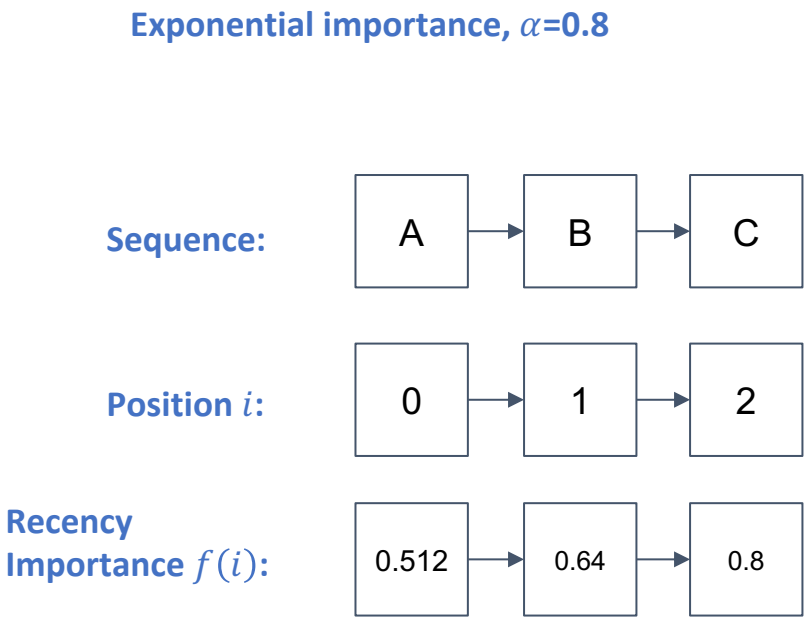
Recency Importance Function

- The prior probability of an item being sampled in a training sequence is defined by a **Recency Importance Function,** based on its position in the original sequence
- We use exponential importance:

$$f(i) = \alpha^{n-i}$$

where i is the position in the sequence, n is sequence length and α is a hyperparameter controlling recency importance

- Lower α places more sampling emphasis on later items



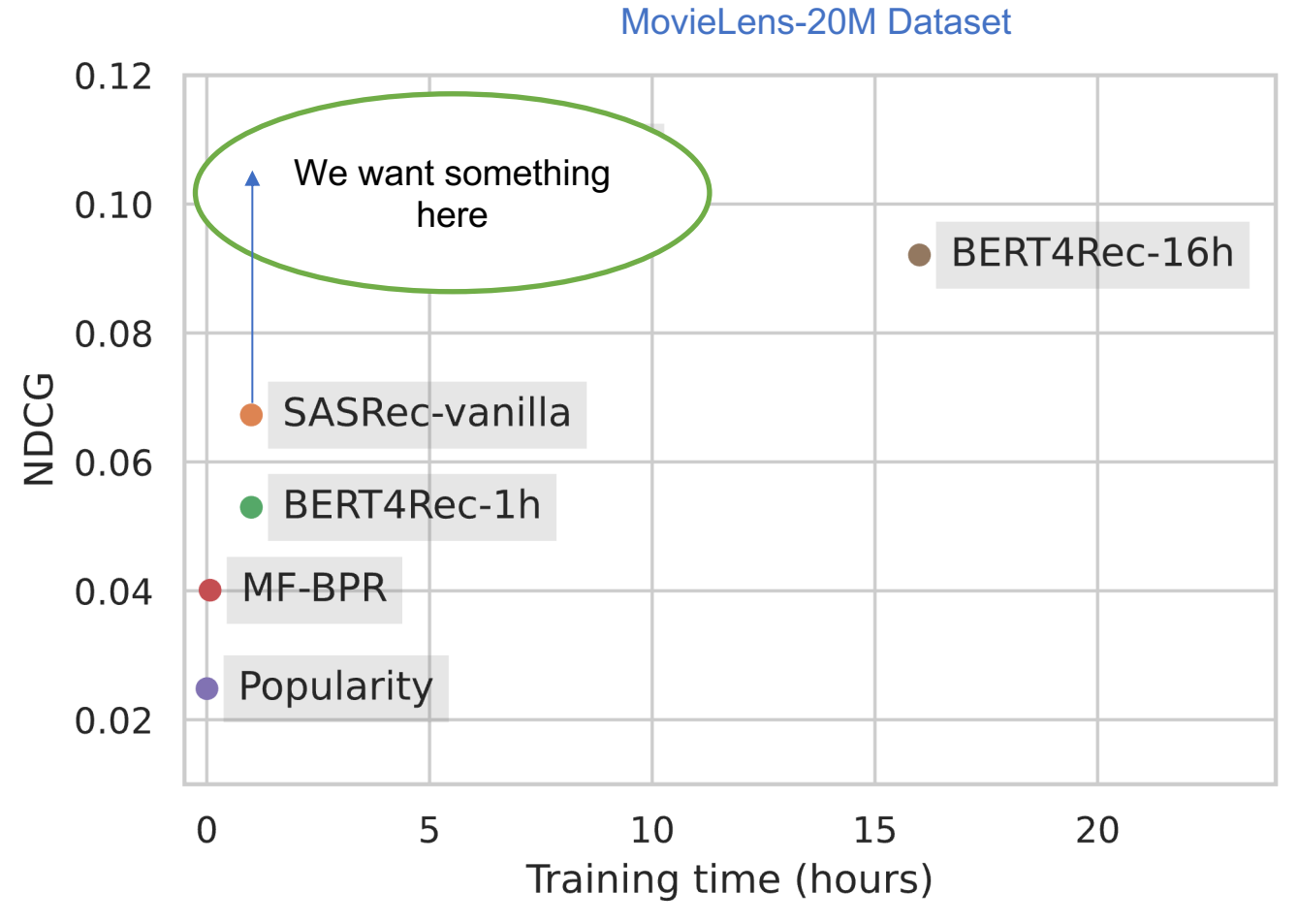
RSS Evaluation Results

NDCG@10; Leave-one-out strategy; 1 hour training time

		MovieLens-20M		Yelp		Gowalla		Booking.com	
Architecture	**Loss**	**Sequence Continuation**	**Recency Sampling**	**Sequence Continuation**	**Recency Sampling**	**Sequence Continuation**	**Recency Sampling**	**Sequence Continuation**	**Recency Sampling**
GRU4Rec	BCE	0.0115	**0.0183***	0.0035	**0.0049***	**0.0017***	0.0002	0.2829	**0.2899**
	λRank	0.0040	**0.0839***	0.0004	**0.0014***	0.0033	**0.0067***	**0.3132**	0.3093†
Caser	BCE	0.0784	**0.0995***	0.0021	**0.0049***	0.0039	**0.0040**	**0.3665***	0.3311†
	λRank	0.0177	**0.0814***	0.0003	**0.0007***	0.0055	**0.0100***	0.3181	**0.3226***
SASRec	BCE	0.0850	**0.1002***	0.0076	**0.0136***	0.0044	**0.0044**	**0.3633***	0.2966
	λRank	0.0579	**0.1073***	0.0021	**0.0025***	0.0478†	**0.0749***	**0.3623***	0.3122†

*** - Statistically significant difference, ($pvalue < 0.05$)**

Comparison with SOTA Models



- Our training task is both effective and efficient
- RSS-enhanced SASRec matches or outperforms the (SOTA) BERT4Rec in effectiveness
- Does not require more training time than original (vanilla) SASRec

Conclusions

- Recency Sampling of Sequences is an effective and efficient training objective for Sequential Recommendation that allows to train models with the quality close to SOTA models in limited time
- It improves results across a number of different datasets, model architectures and training losses
- However, RSS did not improve results when there are very strong sequential patterns (e.g. cities in a trip – as per Booking.com dataset)

ILO 1D.2: Effective negative sampling for large catalogues

gSASRec: Reducing Overconfidence in Sequential Recommendation Trained with Negative Sampling

Aleksandr Petrov and Craig Macdonald
ACM RecSys 2023 (Best Paper Award)

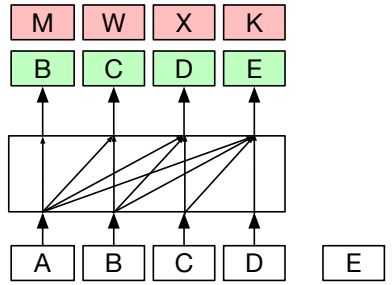
Large Catalogue & Negative sampling

Its challenging to train SOTA models (BERT4Rec; SASRec-RSS) on large catalogues*

800M videos on YouTube

(probably more)

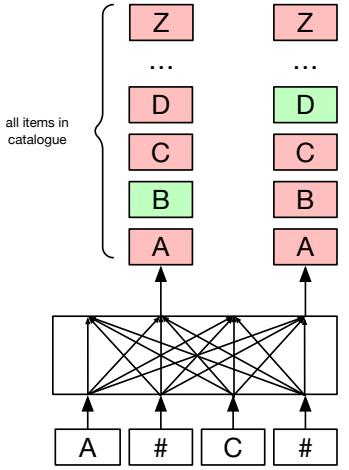
- **Not feasible** to score all items during training (e.g. of BERT4Rec) at this scale
- Typical solution – **negative sampling** (only use small proportion of negatives)
- **SASRec uses** negative sampling (1 negative per positive), but **BERT4Rec does not**
- Negative sampling is **inevitable** with large catalogues, but **it has problems**



SASRec:

One random negative per positive

Binary Cross-Entropy (BCE) loss



BERT4Rec:

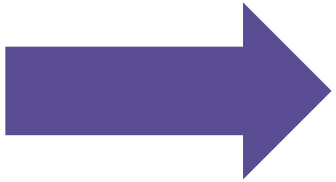
No negative sampling

Softmax Loss

*Petrov, A. and Macdonald, C., RSS: Effective and Efficient Training for Sequential Recommendation using Recency Sampling. RecSys, 2022; TORS 2023

Overconfidence

Negative sampling **increases the proportion of positives** in the training data



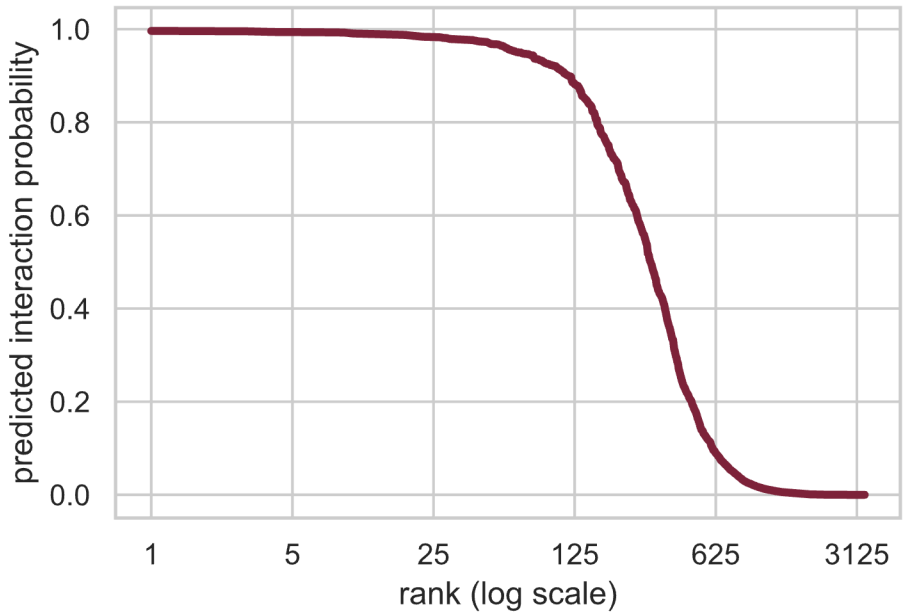
Models using negative sampling **overestimate** the probability of any item is relevant

Overconfidence

Negative sampling **increases the proportion of positives** in the data

Models using negative sampling **overestimate** the probability of any item is relevant

Example: SASRec's overconfidence*



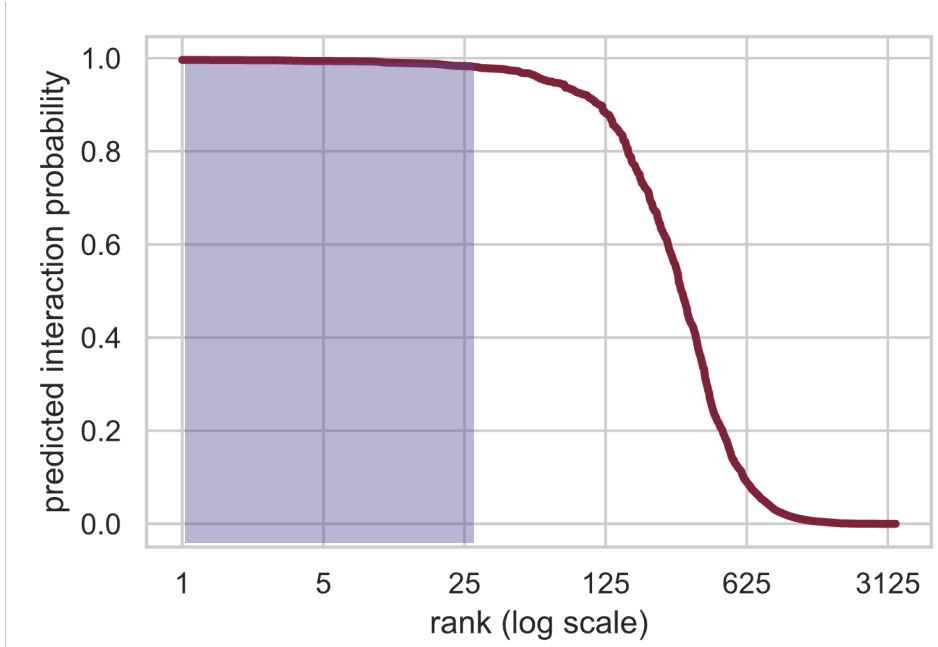
*User 963, MovieLens-1M dataset

Overconfidence

Negative sampling **increases the proportion of positives** in the data

Models using negative sampling **overestimate** the probability of any item is relevant

Example: SASRec's overconfidence*



*User 963, MovieLens-1M dataset

According to the model, the user will interact with all items ranked at positions 1..25 with a probability close to 1

Most of these predictions are **False Positives**

Normally the magnitude of retrieval scores doesn't matter, only their ordering matter (following the Probability Ranking Principle)

But all these False Positives are a problem for the loss functions used for training

False Positives vs. Binary Cross-Entropy

- SASRec (and many other models) uses Binary Cross-Entropy (BCE) Loss:

(I^- is the set of *sampled* negative items)

$$\mathcal{L}_{BCE} = -\frac{1}{|I^-| + 1}\left[\log\left(\sigma(s_{i^+})\right) + \sum_{i \in I^-} \log(1 - \sigma(s_i))\right]$$

This log(·) tends to $-\infty$ for false positives → large training gradients

- Overconfident models have many false positives
- Large gradients for those false positives → **unstable training**

Generalised Binary Cross-Entropy (gBCE)

To mitigate overconfidence, we propose gBCE:

$$\mathcal{L}_{gBCE}^{\beta} = -\frac{1}{|I^{-}|+1}\left[\log\left(\sigma^{\beta}(s_i^{+})\right) + \sum_{i \in I^{-}} \log(1 - \sigma(s_i))\right]$$

- Positive probability is raised to the power of $\boldsymbol{\beta}$ $(\alpha < \beta \leq 1)$
- α is the sampling rate (**number** of negative samples **as a proportion** of catalogue size)

Generalised Binary Cross-Entropy (gBCE)

To mitigate overconfidence, we propose gBCE:

$$\mathcal{L}_{gBCE}^{\beta} = -\frac{1}{|I^-|+1}\left[\log\left(\sigma^{\boldsymbol{\beta}}(s_i^+)\right) + \sum_{i \in I^-} \log(1 - \sigma(s_i))\right]$$

- Positive probability is raised to the power of $\boldsymbol{\beta}$ $(\alpha < \beta \leq 1)$
- α is the sampling rate (**number** of negative samples **as a proportion** of catalogue size)
- β is more easily controlled by a calibration parameter $0 \leq \boldsymbol{t} \leq 1$: $\beta = \alpha(\boldsymbol{t}\left(1 - \frac{1}{\alpha}\right) + \frac{1}{\alpha})$

Generalised Binary Cross-Entropy (gBCE)

To mitigate overconfidence, we propose gBCE:

$$\mathcal{L}_{gBCE}^{\beta} = -\frac{1}{|I^-|+1}\left[\log\left(\sigma^{\boldsymbol{\beta}}(s_i^+)\right) + \sum_{i \in I^-} \log(1 - \sigma(s_i))\right]$$

- Positive probability is raised to the power of $\boldsymbol{\beta}$ $(\alpha < \beta \leq 1)$
- α is the sampling rate (**number** of negative samples **as a proportion** of catalogue size)
- β is more easily controlled by a calibration parameter $0 \leq \boldsymbol{t} \leq 1$: $\beta = \alpha(\boldsymbol{t}\left(1 - \frac{1}{\alpha}\right) + \frac{1}{\alpha})$
- When t = 0, gBCE becomes regular Binary Cross-Entropy

Generalised Binary Cross-Entropy (gBCE)

To mitigate overconfidence, we propose gBCE:

$$\mathcal{L}_{gBCE}^{\beta} = -\frac{1}{|I^-|+1}\left[\log\left(\sigma^{\beta}(s_{i^+})\right) + \sum_{i \in I^-} \log(1 - \sigma(s_i))\right]$$

- Positive probability is raised to the power of $\boldsymbol{\beta}$ $(\alpha < \beta \leq 1)$
- α is the sampling rate (**number** of negative samples **as a proportion** of catalogue size)
- β is more easily controlled by a calibration parameter $0 \leq \boldsymbol{t} \leq 1$: $\beta = \alpha(\boldsymbol{t}\left(1 - \frac{1}{\alpha}\right) + \frac{1}{\alpha})$
- When t = 0, gBCE becomes regular Binary Cross-Entropy
- Higher values of t **decrease model's overconfidence**
- When t = 1, gBCE drives the model to predict **calibrated** interaction probabilities, but requires a lot of training samples to converge

Effect of calibration parameter t on score gradients

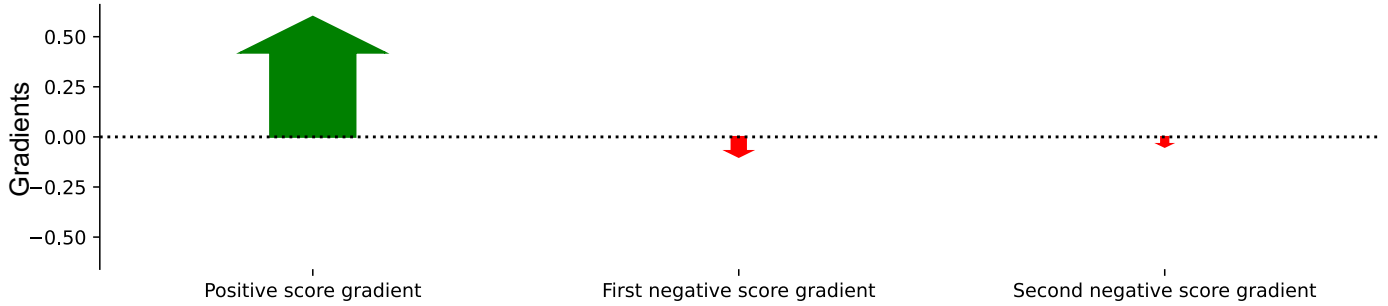
University of Glasgow | Terrier Team

Scenario:
Two negatives sampled out of 3415 (sampling rate $\alpha \approx 0.0006$)

Scores predicted by the model:

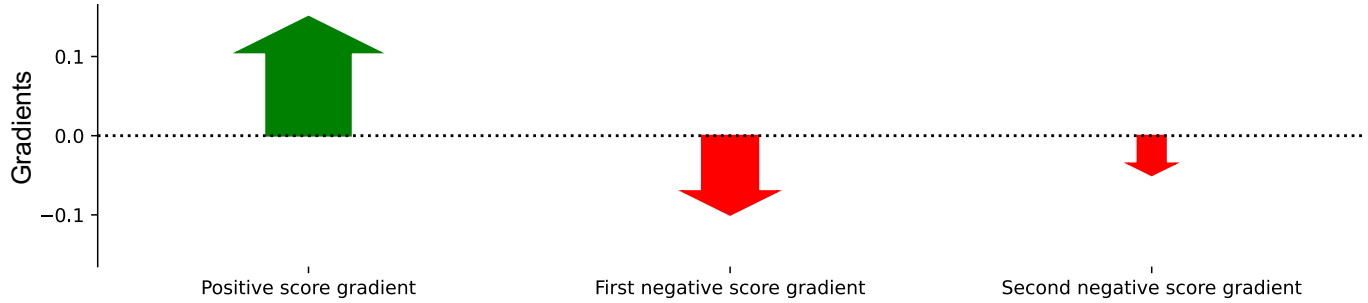
Positive	-0.4
Negative 1	-2.2
Negative 2	-2.9

(a) No calibration: $t = 0.0$. i.e. standard BCE



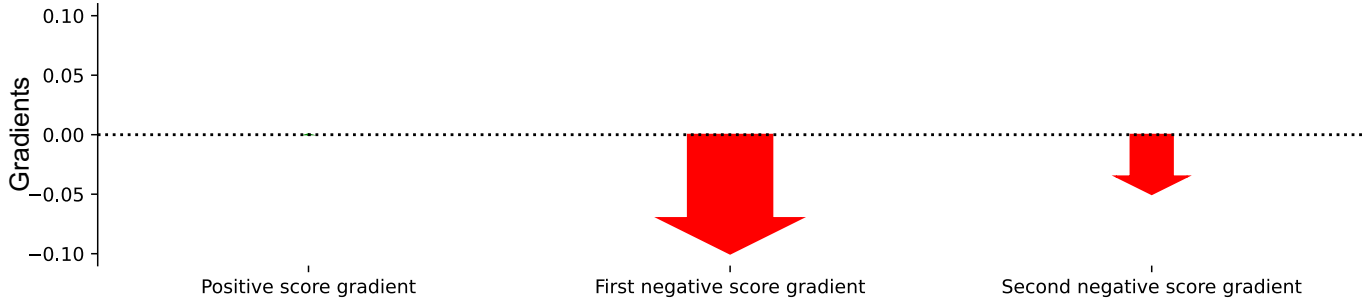
With BCE, the gradients from the positive item dominate, leading to overconfidence

(b) gBCE with recommended calibration: $t = 0.75$



With gBCE, there is balance between the positive and negative gradients

(c) gBCE with full calibration: $t = 1.0$

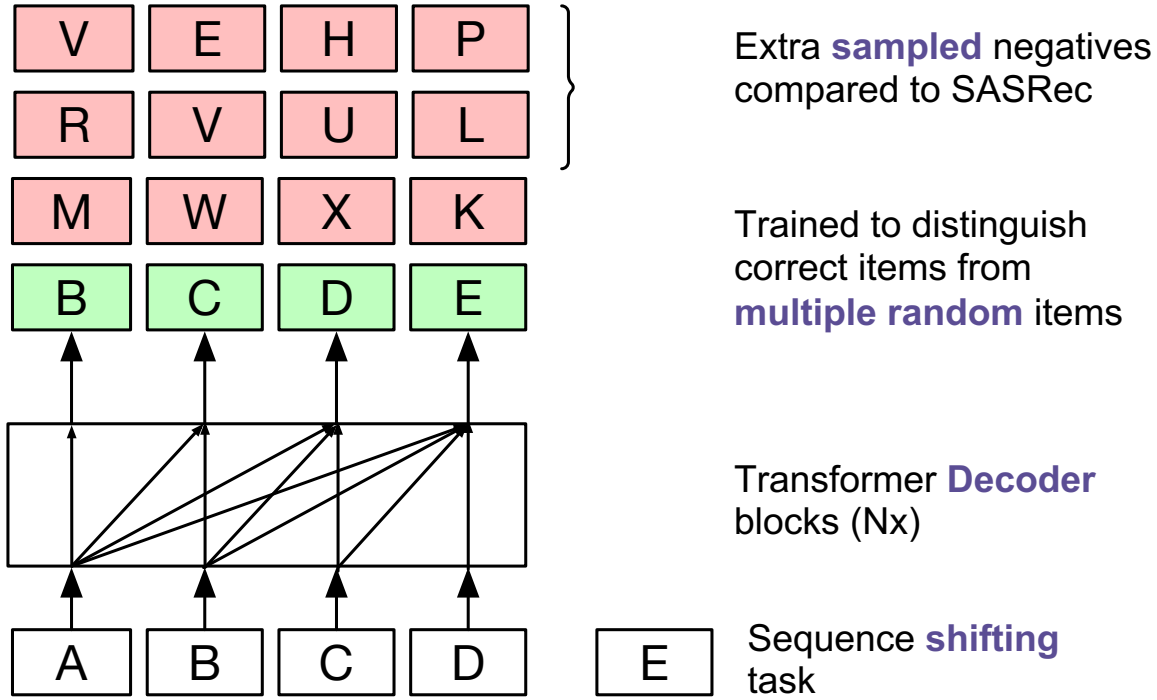


Full calibration of gBCE leads to focus on negative items' gradients instead (∴ slow training)

With higher *t* in gBCE, negative gradients dominate, reducing overconfidence.

gSASRec

- gSASRec builds upon SASRec – a Decoder-based transformer model
- gBCE loss (instead of BCE Loss)
  - reduces overconfidence, calibrates probabilities
- Configurable number of negatives (instead of one negative per positive in SASRec)
  - provide more training evidence



Does Negative Sampling affect performance?

NDCG, MovieLens-1M dataset

Architecture + task →	Transformer Decoder Sequence Shifting (as SASRec)	Transformer Encoder Item Masking (as BERT4Rec)	Architecture effect

Does Negative Sampling affect performance?

NDCG, MovieLens-1M dataset

Architecture + task → / Negative Sampling + loss ↓	Transformer Decoder Sequence Shifting (as SASRec)	Transformer Encoder Item Masking (as BERT4Rec)	Architecture effect
1 Negative per positive, Binary Cross-Entropy (as SASRec)			
No negative sampling SoftMax loss (as BERT4Rec)			
Negative Sampling effect			

Does Negative Sampling affect performance?

NDCG, MovieLens-1M dataset

Architecture + task → / Negative Sampling + loss ↓	Transformer Decoder Sequence Shifting (as SASRec)	Transformer Encoder Item Masking (as BERT4Rec)	Architecture effect
1 Negative per positive, Binary Cross-Entropy (as SASRec)	0.131	0.123	**-6.1%**
No negative sampling SoftMax loss (as BERT4Rec)	0.169	0.161	**-4.7%**
Negative Sampling effect	**+29%***	**+30.8%***	

*significant change (p value < 0.05)

BERT4Rec is more effective than SASRec because of the **absence of negative sampling** and **SoftMax loss… not because of the bidirectional Encoder architecture!**

Where do BERT4Rec gains really come from?

In the original BERT4Rec paper, Sun et al. attributed its improved effectiveness **to bi-directional** self-attention:

"

Question 1: *Do the gains come from the bidirectional self-attention model or from the Cloze objective?*

...

The results show that BERT4Rec with 1 mask significantly outperforms SASRec on all metrics. It demonstrates the importance of bidirectional representations for sequential recommendation.

"

Our analysis **contradicts** these findings.

The real cause of BERT4Rec performance gains is the **absence of negative sampling.**

When controlled for negative sampling, the effect of architecture **is small**, and in three out of 4 cases**, it is negative.**

In short:
- SASRec (decoder) vs BERT4Rec (encoder) architectures aren't the main difference
- negative sampling is desirable for training large models, but can damage effectiveness (due to overconfidence)…

💡 However, we need negative sampling for large catalogues. But negative sampling has the overconfidence problem, addressed by gBCE…

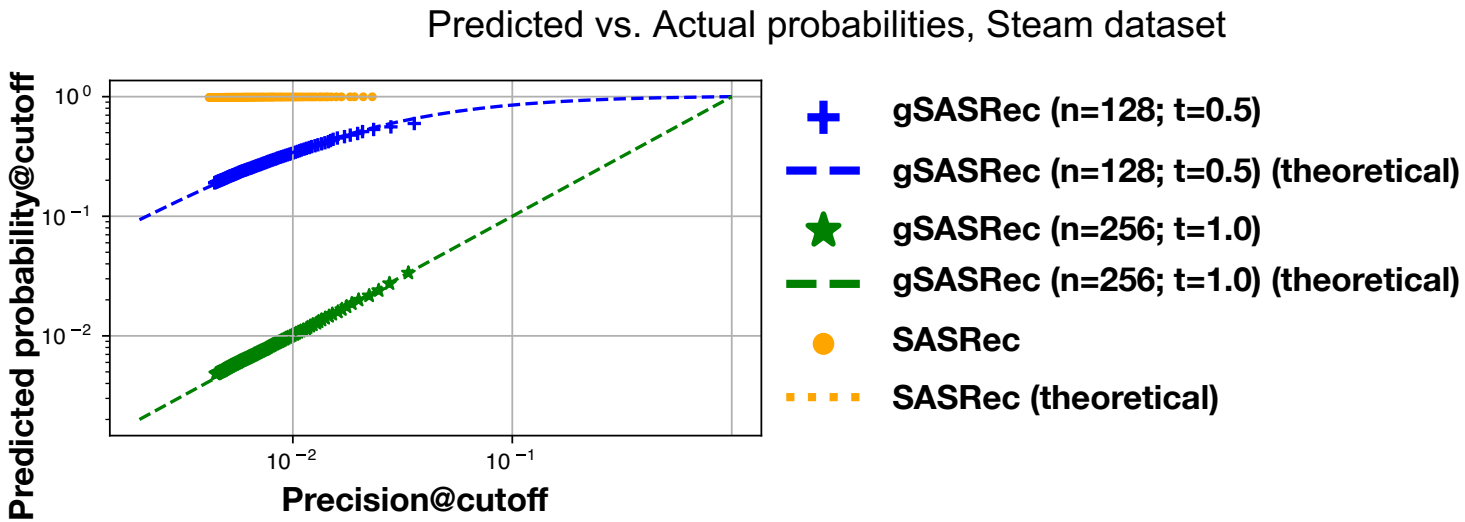
gBCE effect on probabilities calibration?

- If we fix overconfidence, then model outputs should approximate the probability of user interactions
- Ground truth: Precision@K can be seen as the actual probability of a user interacting with items ranked at positions less than K*

* Gordon V. Cormack, Ondrej Lhotak, and Christopher R. Palmer. 1999. Estimating Precision by Random Sampling.

gBCE effect on probabilities calibration?

- If we fix overconfidence, then model outputs should approximate the probability of user interactions
- Ground truth: Precision@K can be seen as the actual probability of a user interacting with items ranked at positions less than K*



- gBCE successfully mitigates the overconfidence problem
- Observed correlations are well-aligned with our theoretical analysis

* Gordon V. Cormack, Ondrej Lhotak, and Christopher R. Palmer. 1999. Estimating Precision by Random Sampling.

gSASRec performance vs. Baselines

Category	Model
Baselines	Popularity
	MF-BPR
Unsampled	BERT4Rec
	SASRec-Softmax
Sampled	SASRec
	gSASRec

gSASRec performance vs. Baselines

Category	Model	Datasets		
		MovieLens-1M	Steam	Gowalla
Baselines	Popularity			
	MF-BPR			
Unsampled	BERT4Rec			
	SASRec-Softmax			
Sampled	SASRec			
	gSASRec			

gSASRec performance vs. Baselines

Category	Model	Datasets					
		MovieLens-1M		Steam		Gowalla	
		NDCG@10	Recall@1	NDCG@10	Recall@1	NDCG@10	Recall@1
Baselines	Popularity						
	MF-BPR						
Unsampled	BERT4Rec						
	SASRec-Softmax						
Sampled	SASRec						
	gSASRec						

gSASRec performance vs. Baselines

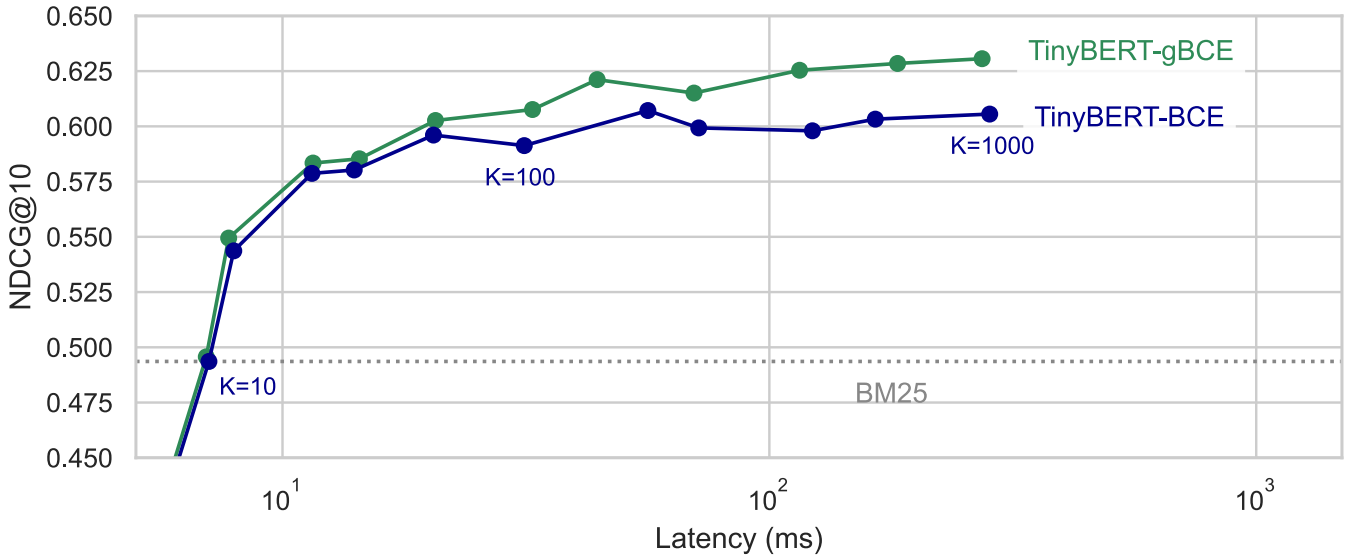
Category	Model	Datasets					
		MovieLens-1M		Steam		Gowalla	
		NDCG@10	Recall@1	NDCG@10	Recall@1	NDCG@10	Recall@1
Baselines	Popularity	0.017*	0.005*	0.0268*	0.0077*	0.0041*	0.0011*
	MF-BPR	0.037*	0.010*	0.0206*	0.0071*	0.0170*	0.0083*
Unsampled	BERT4Rec	0.161*	0.058*	**0.0746**	0.0281	Infeasible w/o negative sampling	
	SASRec-Softmax	0.169	0.073	0.0721*	0.028		
Sampled	SASRec	0.131*	0.046*	0.0581*	0.0193*	0.1097*	0.0505*
	gSASRec	**0.176**	**0.082**	0.0735	**0.0283**	**0.1616**	**0.0782**

*significant difference with the best model (p value < 0.05)

- gSASRec is always better than or similar to the unsampled models (e.g. BERT4Rec)
- gSASRec is always the best at Recall@1
- gSASRec exhibits big gains on the large Gowalla dataset over SASRec
- gBCE allows training effective model (i.e. better than original SASRec) while retaining negative sampling

gBCE outside of sequential recommendation

- gBCE applicable with a large class of tasks that requires negative sampling.
- E.g., gBCE can help to train small language models for l efficient document retrieval*:



* Petrov A, Macdonald C, MacAvaney S. Shallow Cross-Encoders for Low-Latency Retrieval. *In ECIR 2024*

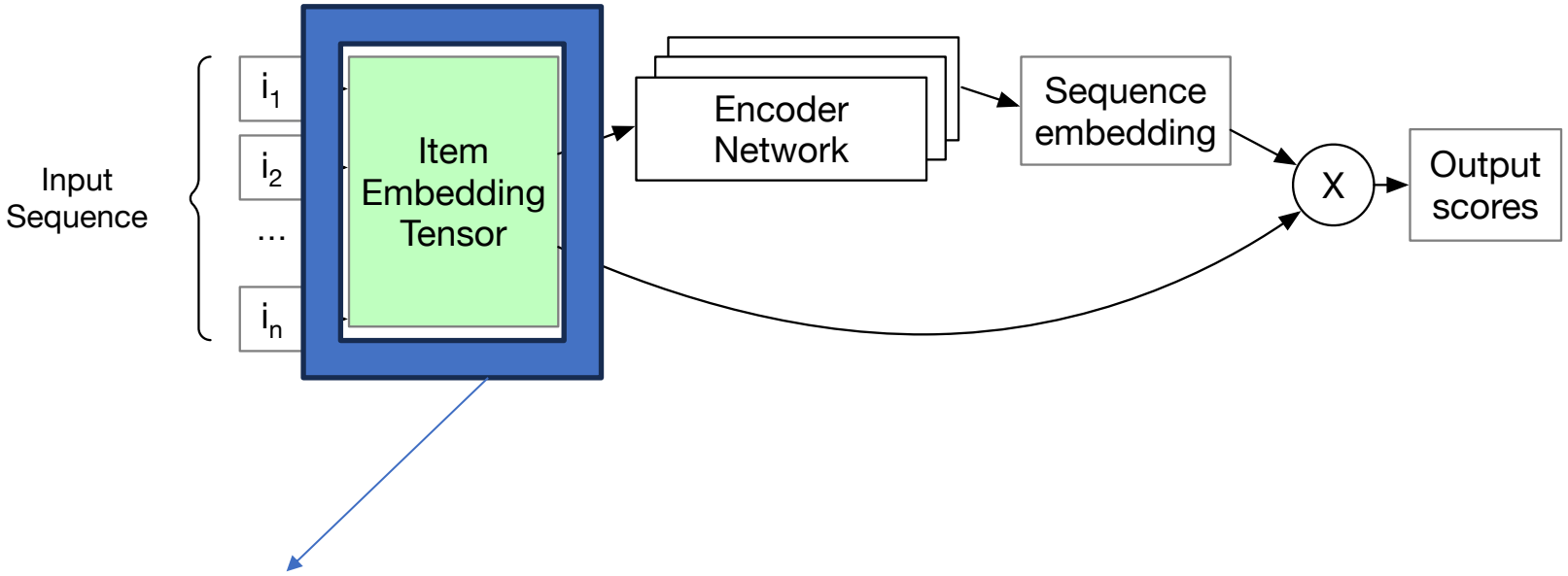
Conclusions

- Negative sampling is necessary for training recommenders on large catalogues
- Negative sampling and BCE loss are the key reason for SASRec's worse performance compared to BERT4Rec
- The root cause of worse performance of SASRec is overconfidence, caused by negative sampling and BCE loss
- gBCE loss successfully mitigates overconfidence while allows for negative sampling
- gSASRec, which uses gBCE loss achieves SOTA performance, while retaining negative sampling

ILO 1D.3: Compressing the Item Embeddings Tensor

RecJPQ: Training Large-Catalogue Sequential Recommenders
Aleksandr Petrov and Craig Macdonald
ACM WSDM 2024

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:

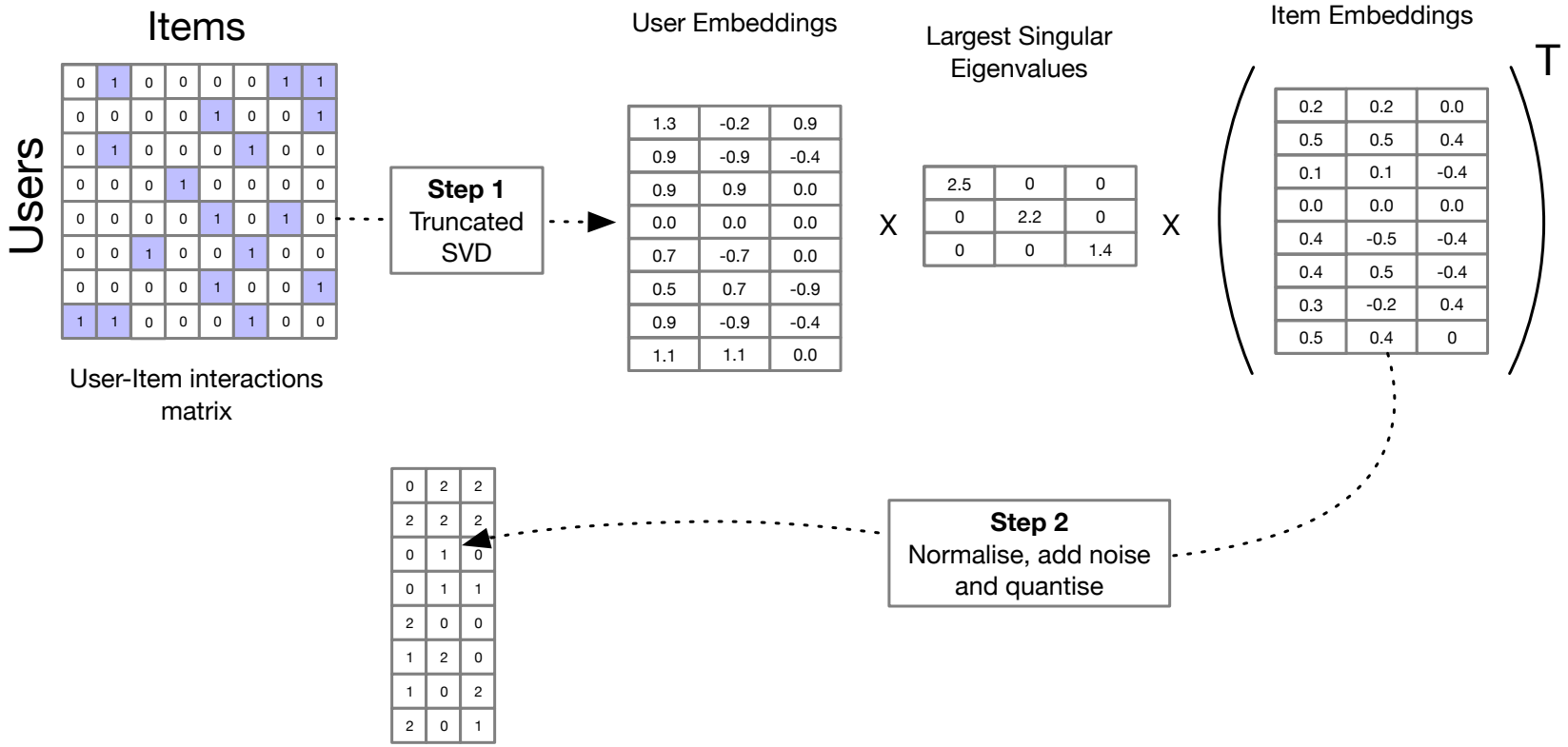
```
In [9]: tokenizer = AutoTokenizer.from_pretrained("gpt2")

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

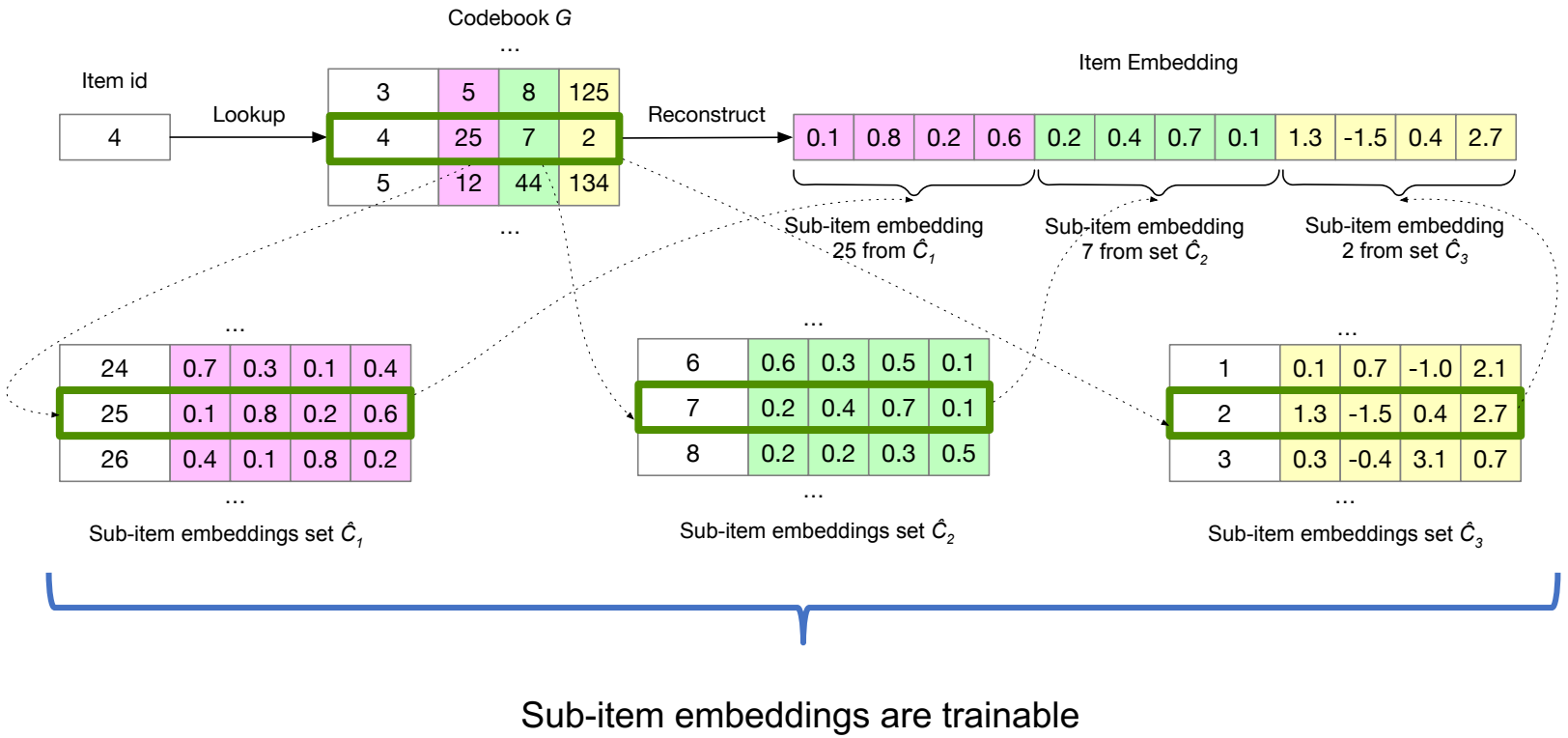


Codebooks of sub-item ids for each item

We call this “SVD tokenisation”

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

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) **

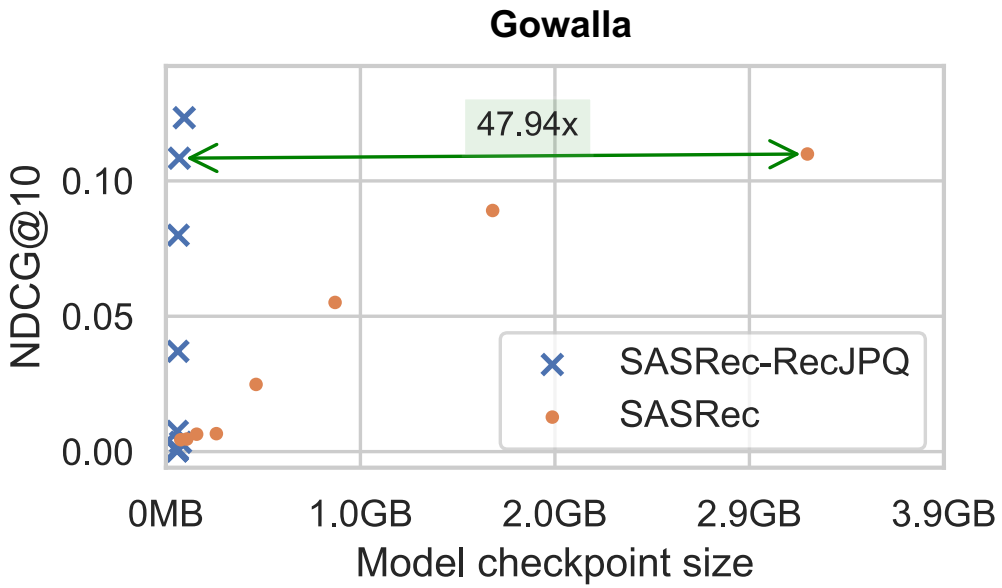
- Codes are assigned *before* full embeddings are built.
- Directly training full compressed model, moderate GPU memory requirements.
- Require code assignment strategy – in RecJPQ, we use Truncated SVD or BPR

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

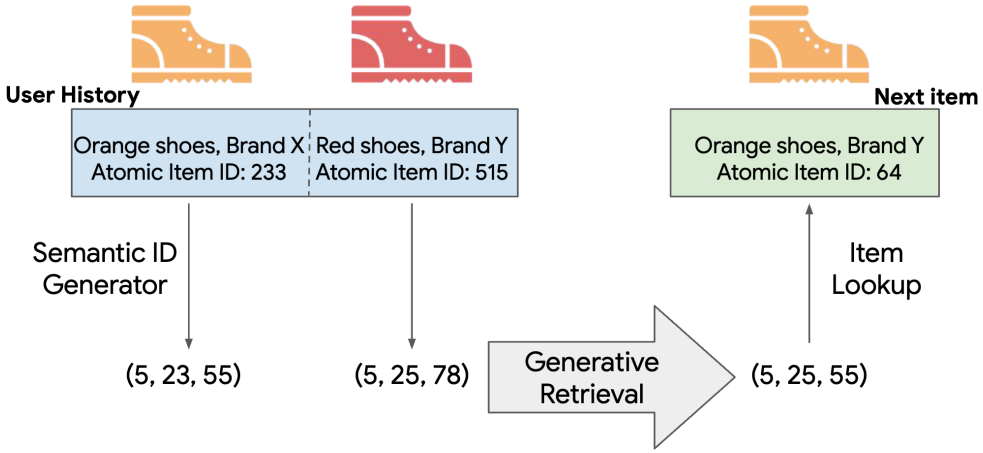
- **Efficient inference** (sub-id scores can be pre-computed).
- **Side information injection** (sub-ids can be based not only on SVD, but also use side information)
- **Generative RecSys** – same technique can be used for item tokenisation in Generative recommendation models.

Item tokenisation in Generative Recommender models

GPTRec*

- RecJPQ-style tokenisation using SVD
- Autoregressive next item generation

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

Part 2. Generative transformers for Sequential Recommendation

ILO 2A. Sequential recommendation as a generative task.

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.

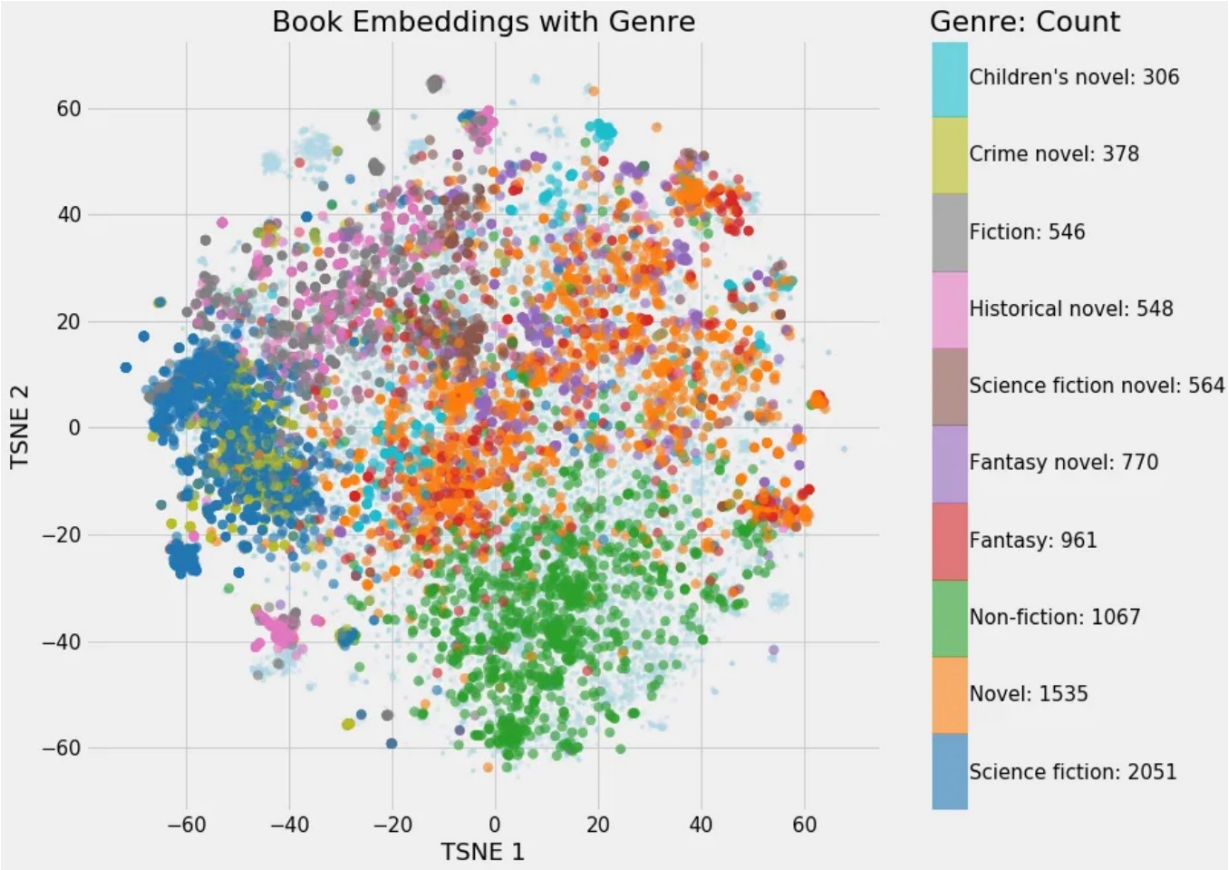
- Frequently, beyond-accuracy goals include components that are defined in terms of whole *lists* of items.

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)

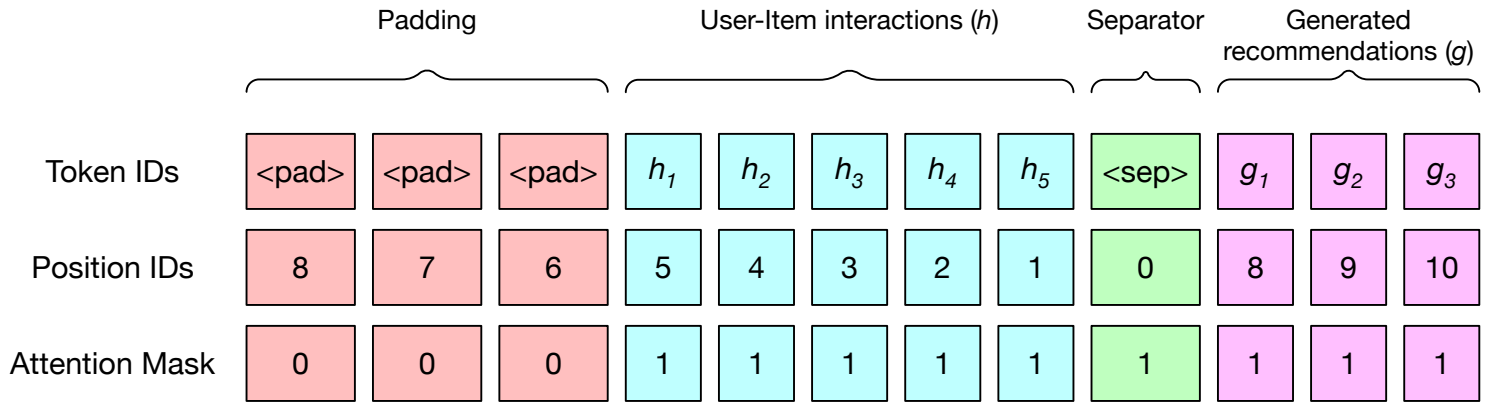
Given an input text (“Prompt”), generate an output text (model’s response) that satisfies complex human needs

Sequential Recommender Systems

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

GPTRec*

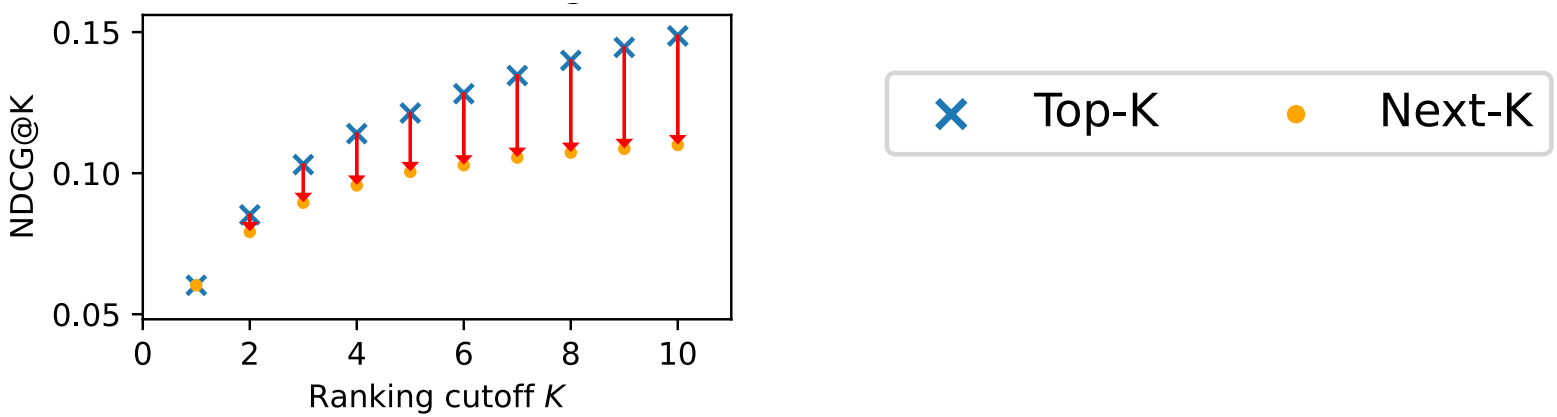
- A Transformer decoder-based recommendation model
  - Based on GPT-2 architecture (but not pre-trained weights)
  - Architecturally, similar to SASRec
  - Generates recommendations using the **Next-K strategy** (autoregressively, item-by-item) – ala `.generate()`
  - Input and output have the same structure:



* 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*

University of Glasgow | Terrier Team

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

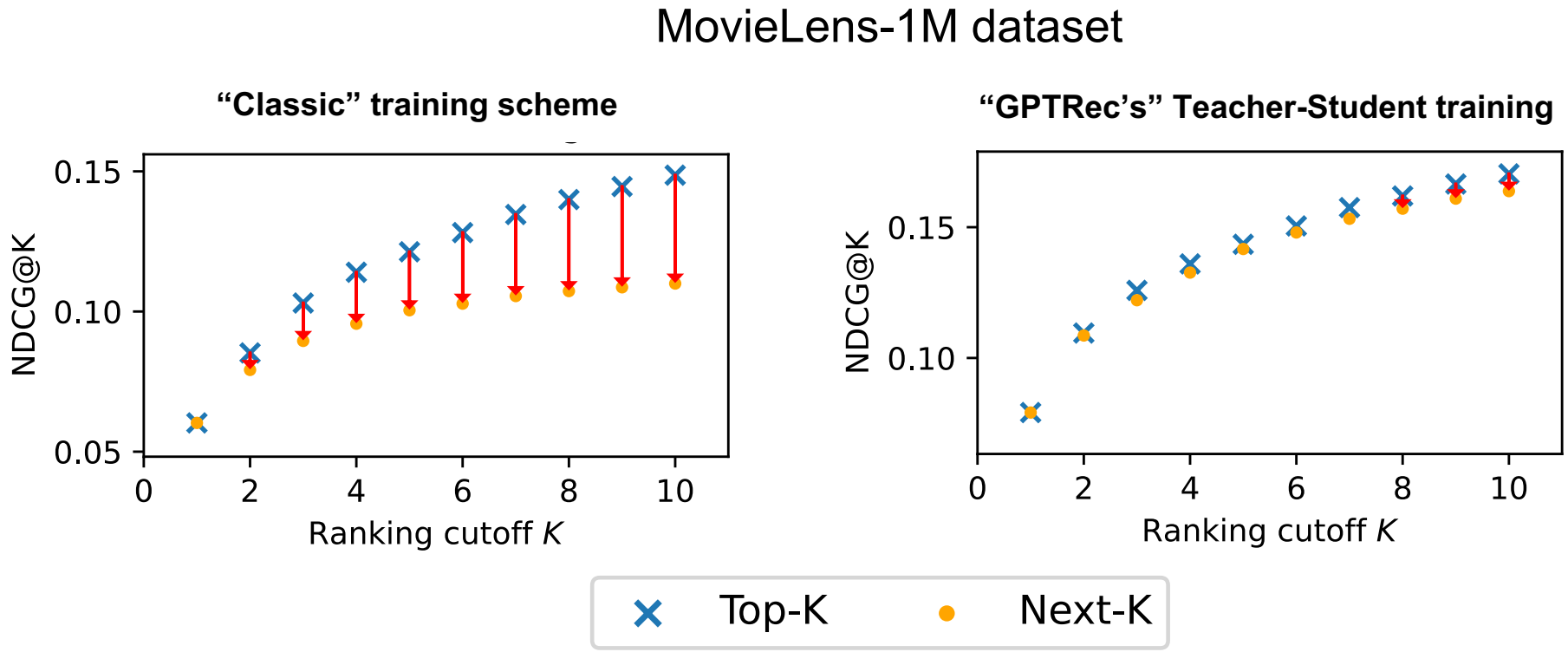
1. Train a “classic” model (e. g. BERT4Rec)

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

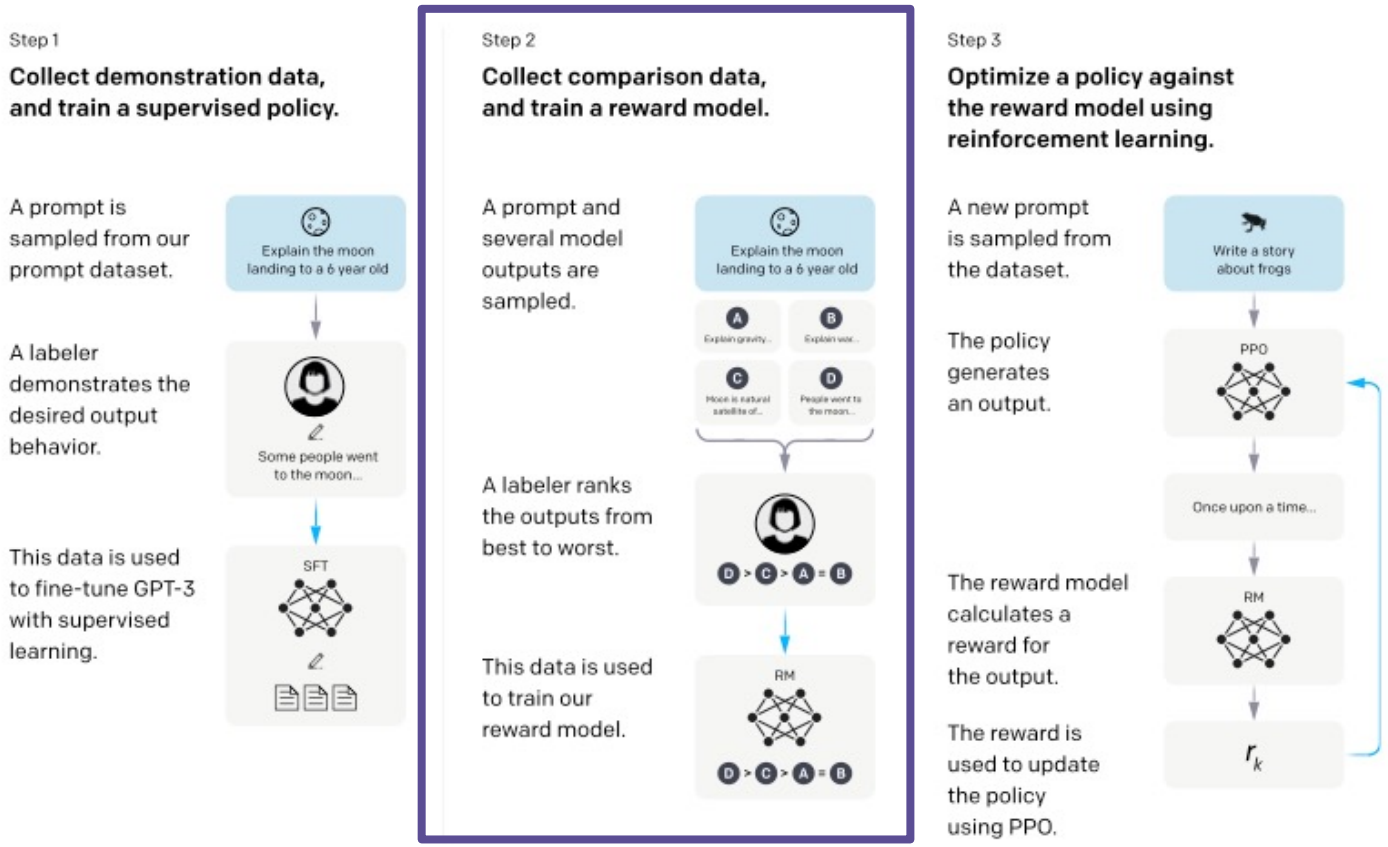
ILO 2B Generative Sequential Recommendation for Beyond-Accuracy goals

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

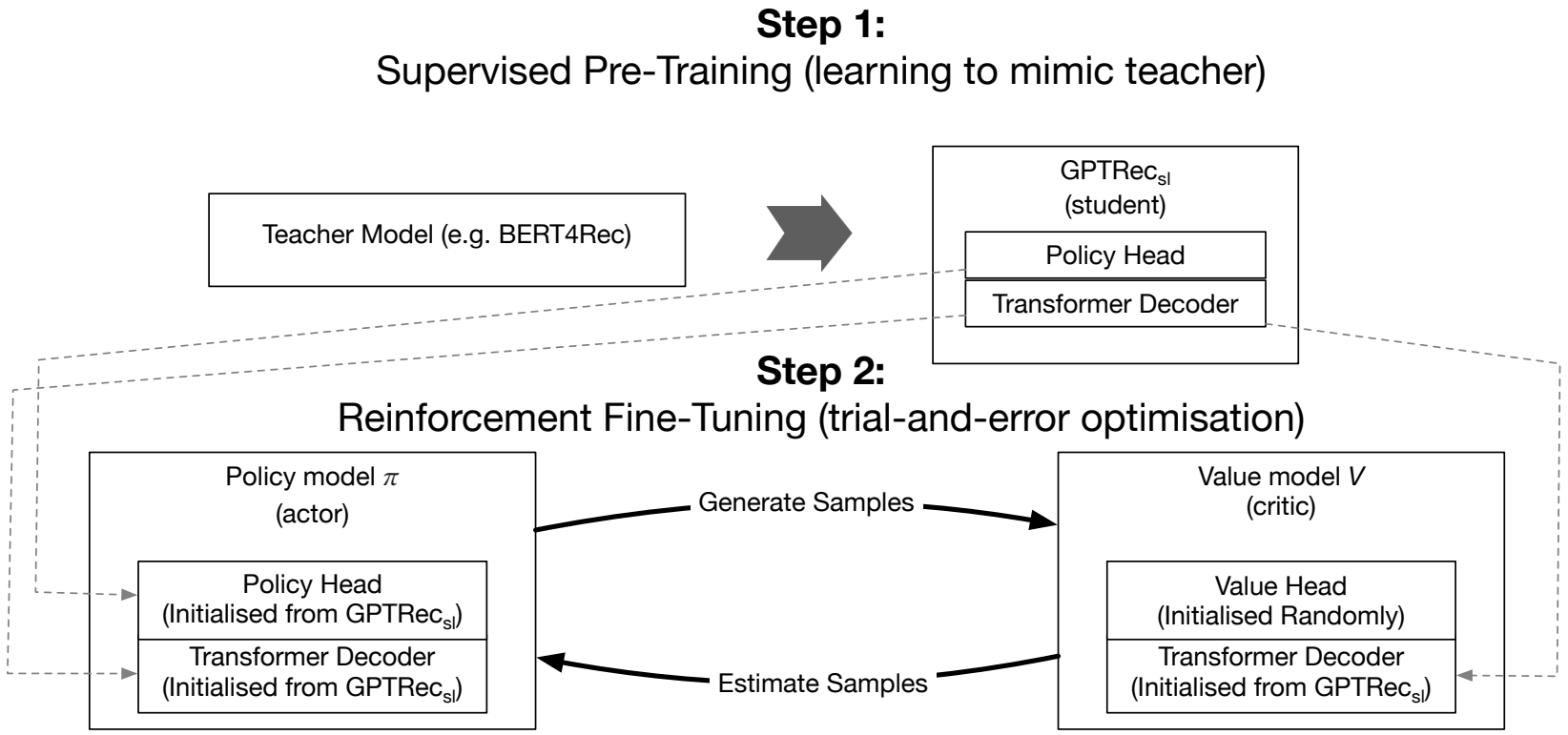
- Misalignment exists in language models.
- Typical Solution: Reinforcement Learning with Human Feedback (RLHF)*



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

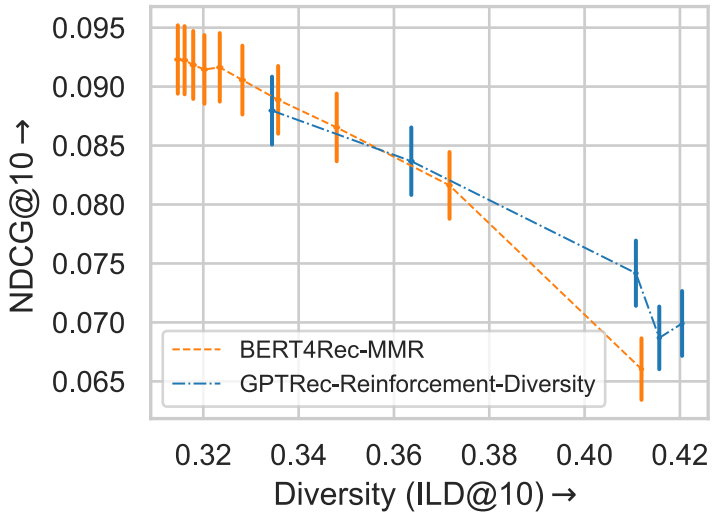


RLHF can be adapted to RecSys to align recommender models with beyond-accuracy goals

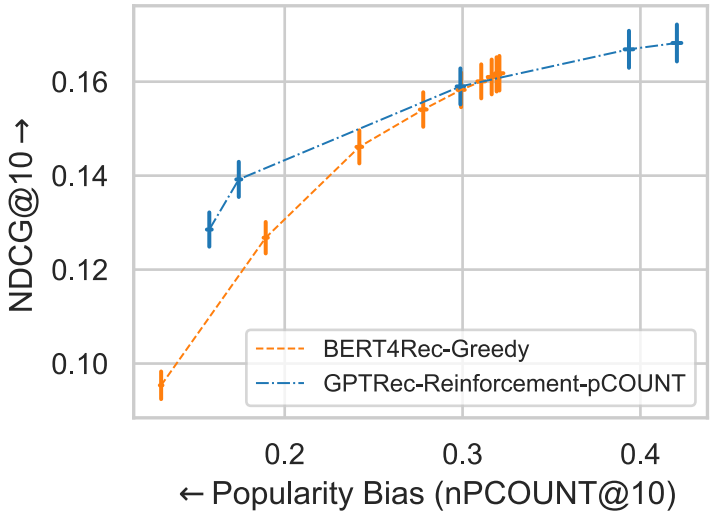
*Petrov, A. and Macdonald, C., 2024. Aligning GPTRec with Beyond-Accuracy Goals with Reinforcement Learning. Accepted by GenRec@WWW2024

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.

What other advancements in language models can be used for RecSys?

ILO 2C. The Role of LLMs in the future of Sequential Recommendation.

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

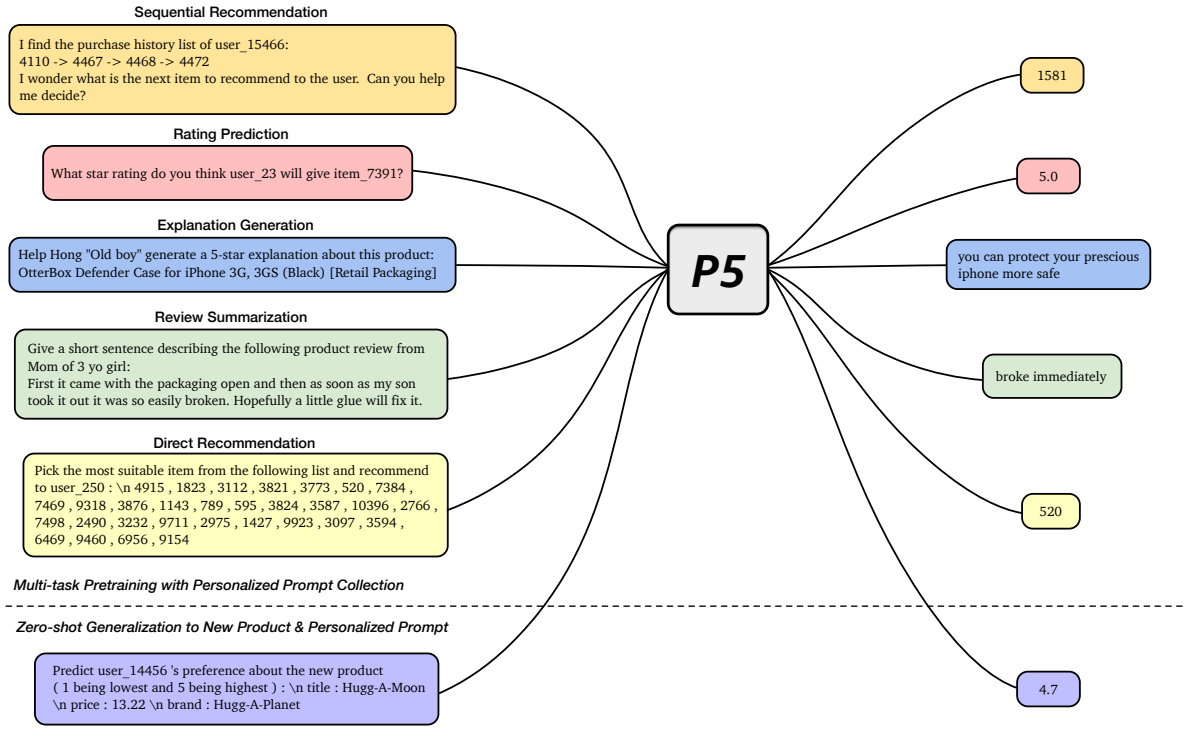
ChatGPT

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*

Tutorial key take-aways

- For many recommendation tasks, it is important to consider the order of interactions.
- Current SOTA models for sequential recommendation are based on Transformer architecture
- To scale transformers to large-scale catalogues, Transformers require a number of modifications, including:
    - Selecting an appropriate training objective
    - Selecting an appropriate negative sampling scheme and the loss function.
    - Applying item embedding compression techniques
- When the effectiveness measures of recommender systems include beyond-accuracy goals, generative (Next-K) recommendation may be more appropriate than traditional Top-K recommendation.
- Aligning generative models with beyond-accuracy goals is a hard task, but it can be solved with Reinforcement Learning.
- LLMs may have many potential use cases for Recommender systems, but we are still in the very early stage with these models.

Future work

- Many ideas presented in this tutorial can be expanded, for example:
  - RecJPQ (“tokenisation”) can be expanded for side information
  - Generative alignment can be integrated with real LLMs
  - For very long sequences we can consider new beyond-transformer architectures

Outlook

- Advances in language modelling will continue to be translated into sequence recommendation

- However, as we have shown, adaptions are necessary to address the specifities of the recommendation task

- There are other interesting crossovers:
  - In our ECIR 2024 paper, we showed that gBCE can be used in training cross-encoders for search re-ranking
  - Between generative IR and sequential recommendation (e.g. RecJPQ vs. non-atomic docids for generative IR)

University of Glasgow | Terrier Team

Q&A

@asash

@craig_macdonald

Our papers covered in the talk:

[1] Petrov, A. and Macdonald, C., 2022, September. A systematic review and replicability study of bert4rec for sequential recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems* (pp. 436-447).
We analysed many recent papers and found that many of them used an underfit version of BERT4Rec. We showed that when converged, BERT4Rec is still a SOTA method.

[2] Petrov, A. and Macdonald, C., 2022, September. Effective and efficient training for sequential recommendation using recency sampling. In *Proceedings of the 16th ACM Conference on Recommender Systems* (pp. 81-91); *extended version in ACM Transactions on Recommender Systems*
We proposed a novel RSS training objective and showed by using it ; we can improve the effectiveness of SASRec while retaining efficiency.

[3] Petrov, A. and Macdonald, C., 2023. Generative sequential recommendation with GPTRec. Presented at GenIR@SIGIR2023.
We showed the feasibility of the *generative* approach for solving such problems as the large embedding matrix and beyond accuracy goals.

[4] Petrov, A. and Macdonald, C., 2023, September. gSASRec: Reducing overconfidence in sequential recommendation trained with negative sampling. In *Proceedings of the 17th ACM Conference on Recommender Systems* (pp. 116-128).
We developed an effective and efficient training scheme for transformer-based models with negative sampling.

[5] Petrov, A. and Macdonald, C., 2024, March. RecJPQ: Training Large-Catalogue Sequential Recommenders. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (pp. 538-547).
We developed a methodology for compressing the item embedding tensor in recommender systems with large catalogues.

[6] Petrov, A. and Macdonald, C., 2024. Aligning GPTRec with Beyond-Accuracy Goals with Reinforcement Learning. To be presented at GenRec@WWW2024
We developed a reinforcement-learning-based methodology to train generative recommender systems for generating recommendations aligned with beyond-accuracy goals, such as diversity and novelty

Slides https://github.com/asash/transformers-for-recsys-tutorial

Other References:

Classic Matrix Factorisation:

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

Association Rule Mining:

- Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. *In SIGMOD 1993*
- McNicholas, P.D., Murphy, T.B. and O'Regan, M., 2008. Standardising the lift of an association rule. *In Computational Statistics & Data Analysis*, *52*(10), pp.4712-4721.

Markov Chains Based Methods:

- A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. *In UAI '01*
- Rendle et al., Factorizing Personalized Markov Chains for Next-Basket Recommendation, *in WWW 2010*

Probability Ranking Principle:

- Maron ME, Kuhns JL. On relevance, probabilistic indexing and information retrieval. *In Journal of the ACM (JACM). 1960*
- Robertson SE. The probability ranking principle in IR. *In Journal of documentation. 1977*

Key Deep Learning-based models for sequential recommendation:

- **GRU4Rec (RNN):** Hidasi, B., Karatzoglou, A., Baltrunas, L. and Tikk, D., 2015. Session-based recommendations with recurrent neural networks. *In ICLR 2016*
- **Caser (CNN):** Tang J, Wang K. Personalized top-n sequential recommendation via convolutional sequence embedding. *In WSDM 2018*
- **NextItNet (CNN)**: Yuan F, Karatzoglou A, Arapakis I, Jose JM, He X. A simple convolutional generative network for next item recommendation. *In WSDM 2019*
- **SASRec (Transformer):** Kang WC, McAuley J. Self-attentive sequential recommendation**.** *In ICDM 2018*
- **BERT4Rec (Transformer):** 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*

References:

Transformer Architecture:

- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. *In NeurIPS 2017*
- Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *In NAACL 2018*
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I., 2019. Language models are unsupervised multitask learners. *In OpenAI blog, 1(8), p.9.*

Contrastive Learning in Sequential Recommendation:

- Xie, X. et al., Contrastive learning for sequential recommendation. *In ICDE 2022*
- Qiu, R., Huang, Z., Yin, H. and Wang, Z., Contrastive learning for representation degeneration problem in sequential recommendation. *In WSDM 2022*
- Du, H., Shi, H., Zhao, P., Wang, D., Sheng, V.S., Liu, Y., Liu, G. and Zhao, L. Contrastive learning with bidirectional transformers for sequential recommendation. *In CIKM 2022.*

Side Information in Sequential Recommendation:

- Li J, Wang Y, McAuley J. Time interval aware self-attention for sequential recommendation. *In WSDM 2020*
- Li J, Zhao T, Li J, Chan J, Faloutsos C, Karypis G, Pantel SM, McAuley J. Coarse-to-fine sparse sequential recommendation. *In SIGIR 2020*
- Li J, Wang M, Li J, Fu J, Shen X, Shang J, McAuley J. Text is all you need: Learning language representations for sequential recommendation. *In KDD 2023*
- de Souza Pereira Moreira G, Rabhi S, Lee JM, Ak R, Oldridge E. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In RecSys 2021

References:

Reproducibility of Sequential Recommender Systems:

- Petrov A, Macdonald C. A systematic review and replicability study of bert4rec for sequential recommendation. *In RecSys 2022*
- Hidasi, B. and Czapp, Á.T., 2023, September. The effect of third party implementations on reproducibility. *In RecSys 2023*

Effective and efficient Transformers for Sequential Recommendation:

- Petrov A, Macdonald C. RSS: Effective and Efficient Training for Sequential Recommendation Using Recency Sampling*. In ACM Transactions on Recommender Systems, 2024*
- Petrov A, Macdonald C. gSASRec: Reducing overconfidence in sequential recommendation trained with negative sampling. In RecSys 2023
- Petrov A, Macdonald C. RecJPQ: Training Large-Catalogue Sequential Recommenders. *In WSDM 2024*

Product Quantisation for Embeddings Compression:

- 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).

Generative Sequential Recommendation for Beyond-Accuracy goals:

- Petrov, A, and Macdonald, C., 2023. Generative sequential recommendation with GPTRec. *In GenIR@SIGIR 2024*
- Petrov, A. and Macdonald, C., 2024. Aligning GPTRec with Beyond-Accuracy Goals with Reinforcement Learning. In GenRec@WWW 2024

References:

Reinforcement Learning with Human Feedback:

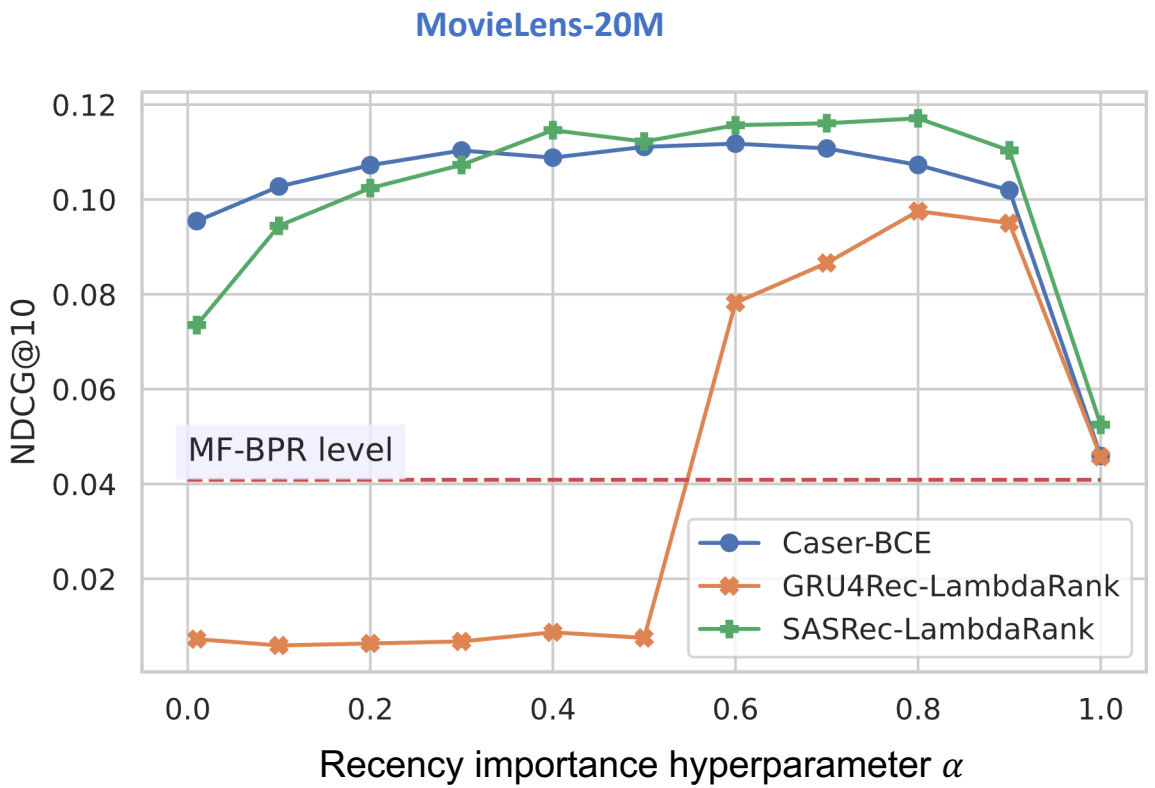
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A. and Schulman, J., 2022. Training language models to follow instructions with human feedback. *In NeurIPS 2022*

LLMs for recommender systems:

- Geng, S., Liu, S., Fu, Z., Ge, Y. and Zhang, Y. Recommendation as language processing (rlp): A unified Pretrain, Personalized Prompt & Predict Paradigm (p5). In RecSys 2022
- Hou Y, Zhang J, Lin Z, Lu H, Xie R, McAuley J, Zhao WX. Large language models are zero-shot rankers for recommender systems. *In ECIR 2024.*

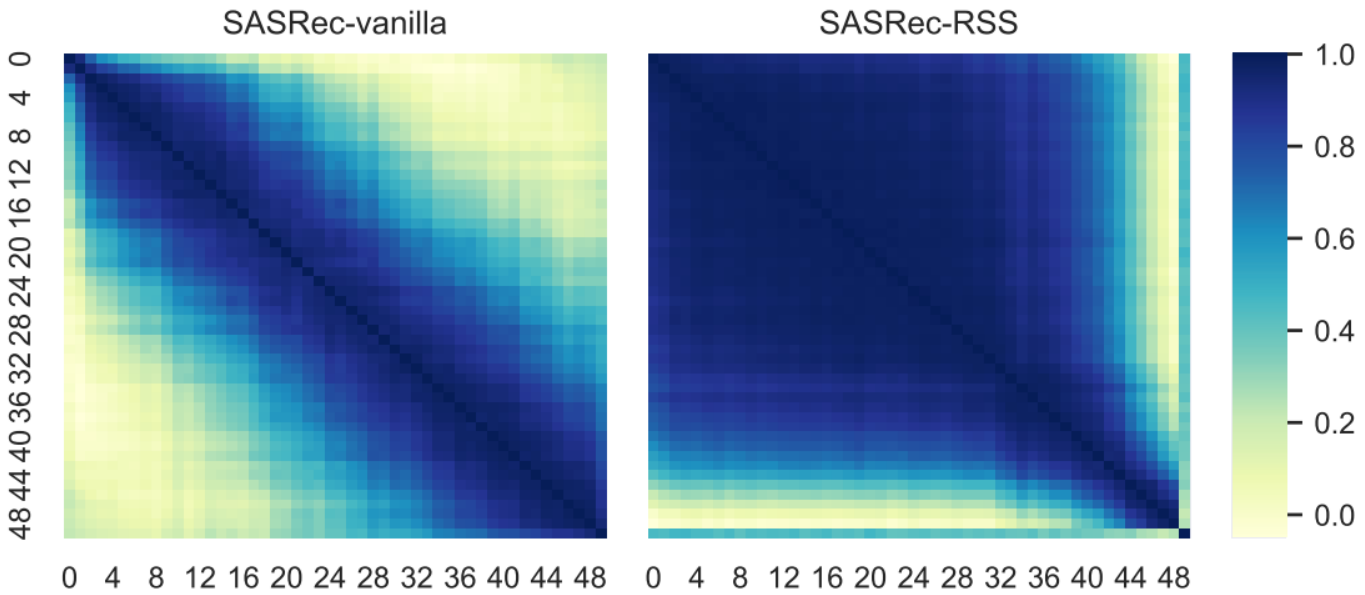
Impact of the Recency Importance Hyperparameter

We investigate impact of the recency importance hyperparameter α of the exponential recency importance function $f(i) = \alpha^{n-i}$



- Both Caser and SASRec exhibit good performance across a wide range of values for Recency Importance α
- When Recency Importance α equals 1, models do not rely on sequential order and performance drops close to Matrix Factorization levels

Position Embedding Similarities in RSS

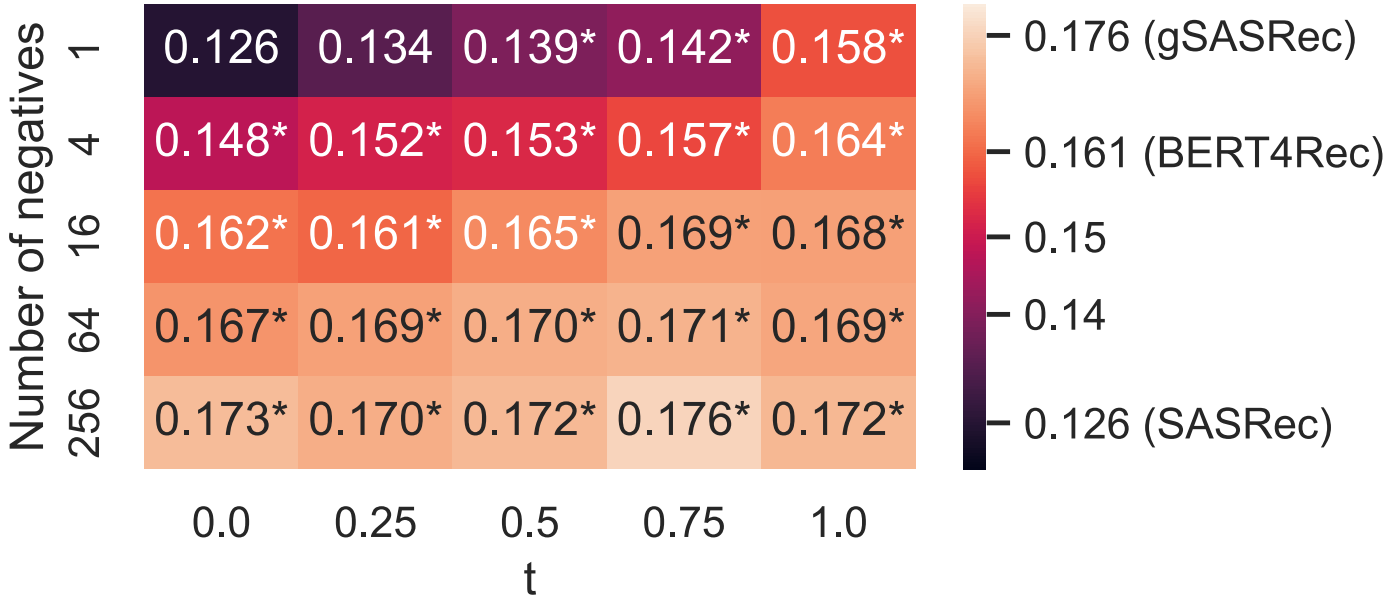


(a) MovieLens-20M position similarities

- Vanilla SASRec does not distinguish recent positions from early positions much.
- SASRec trained with RSS clearly distinguishes recent positions from older ones.

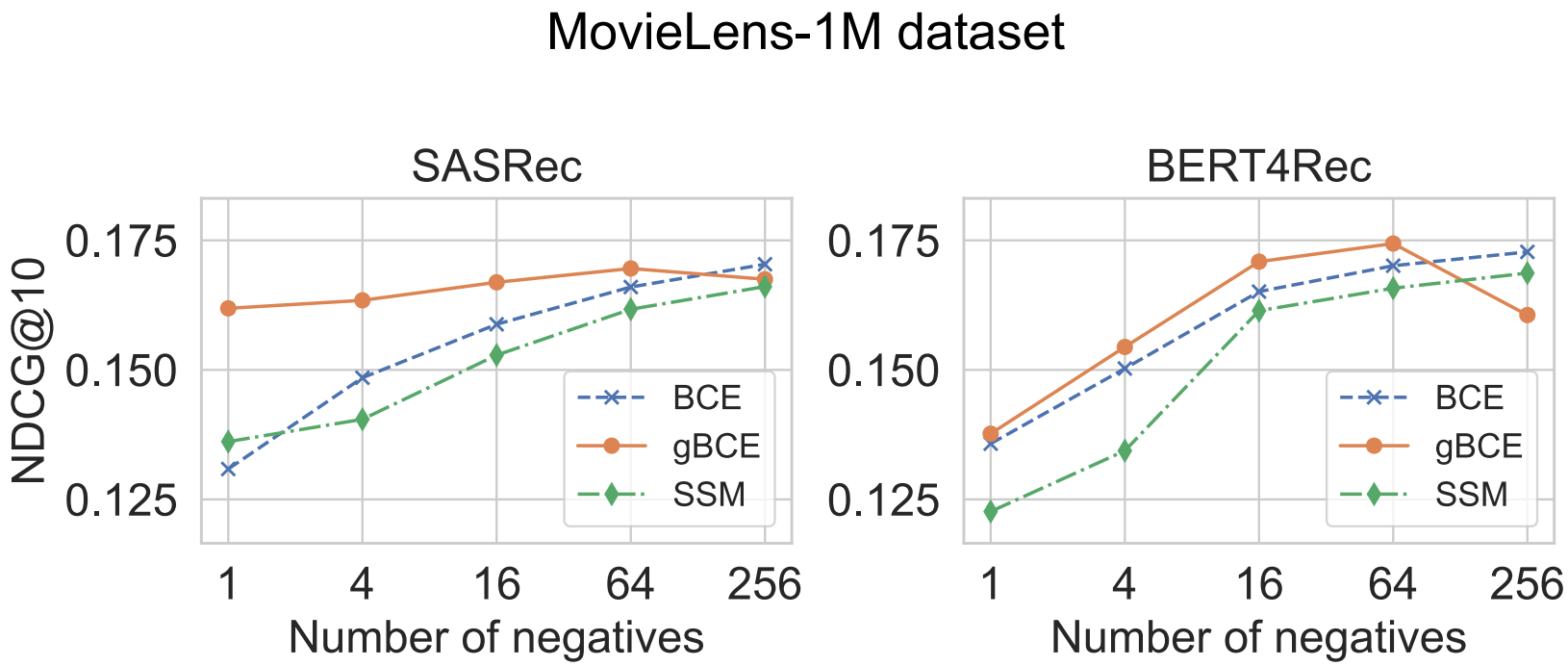
RQ3 Effects of the number of negatives and calibration parameter t?

Grid search results, NDCG@10, MovieLens-1M dataset



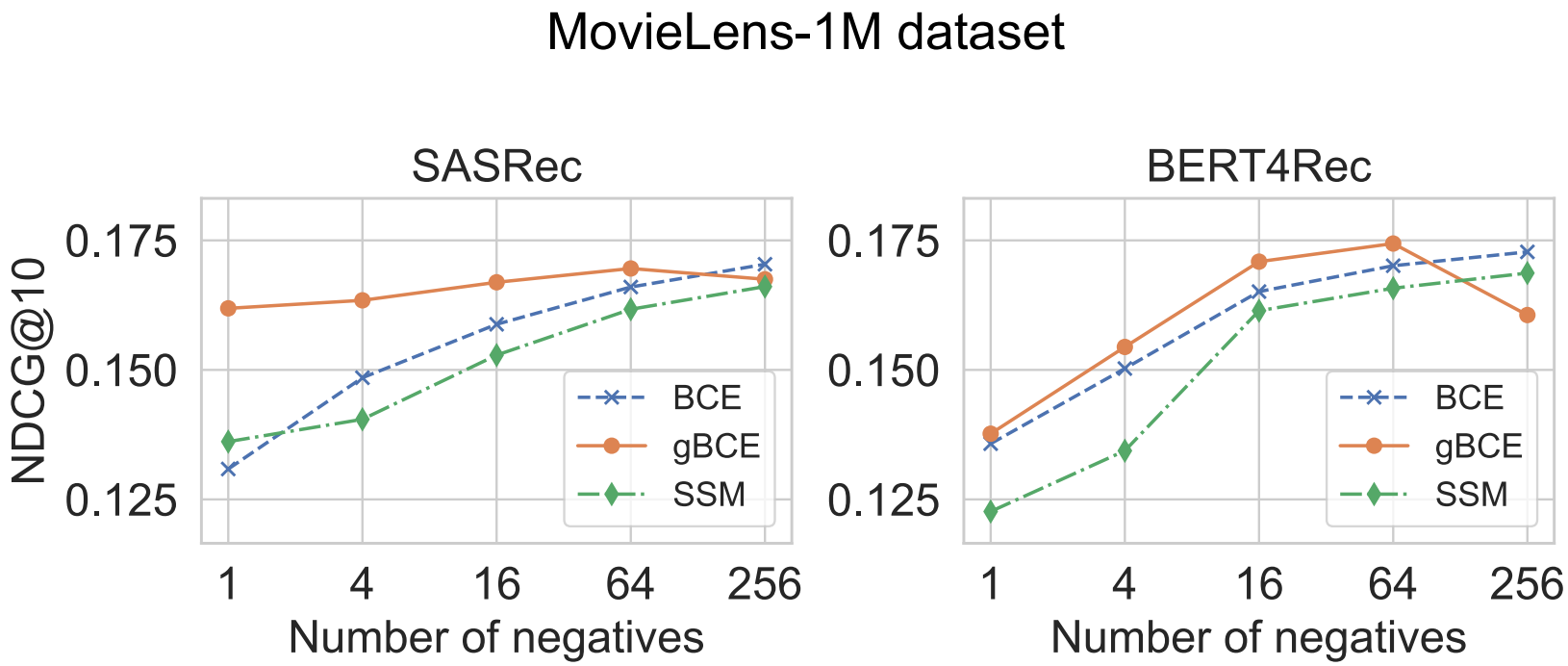
- Higher *t* increases performance
- More negatives increase performance

RQ4 gBCE effect on the performance of SASRec and BERT4Rec?



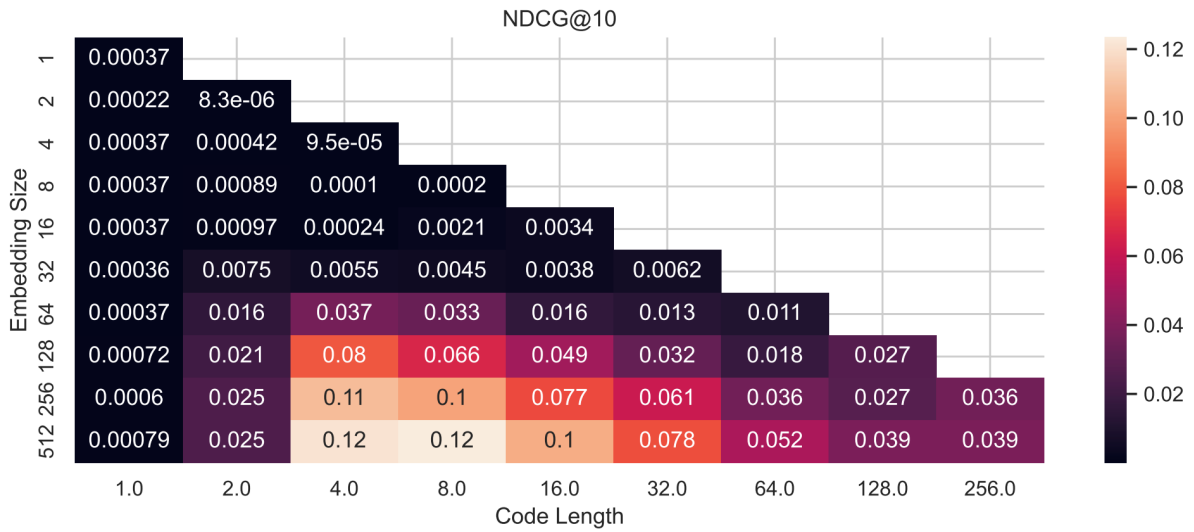
- gBCE outperforms both BCE and Sampled Softmax Loss
- The effect is larger with less negatives
- The effect is more pronounced with SASRec model

RQ4 gBCE effect on the performance of SASRec and BERT4Rec?



- gBCE outperforms both BCE and Sampled Softmax Loss
- The effect is larger with less negatives
- The effect is more pronounced with SASRec model

RecJPQ hyperparameters tuning



(b) Gowalla

Figure 3: RecJPQ performance while varying embedding size d and the number of centroids per item m.

1. Larger embeddings are always good.
2. More centroids per item is not always good.

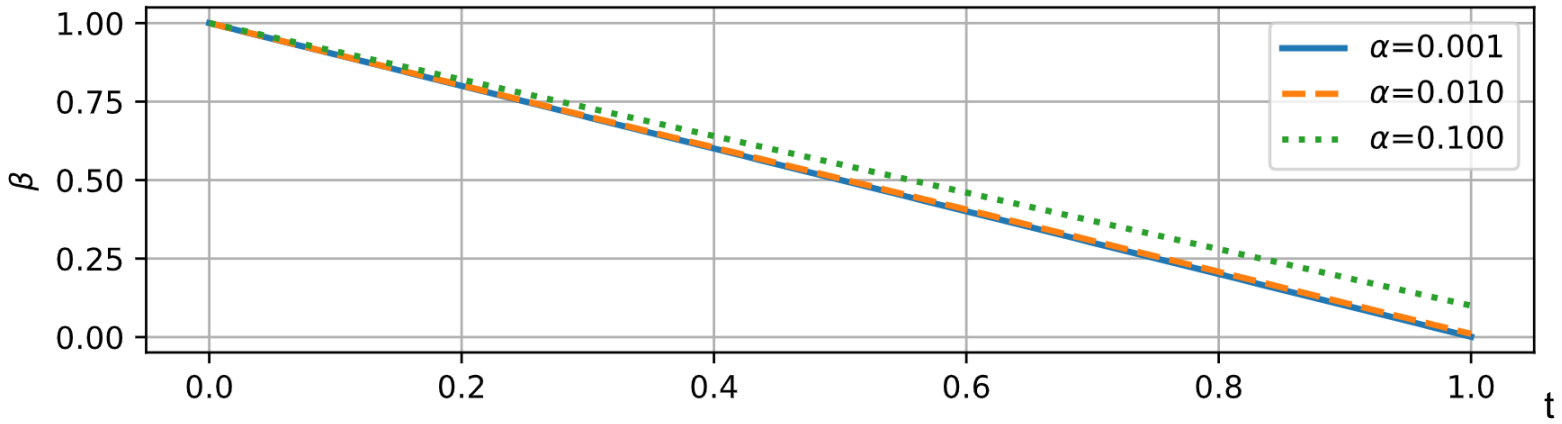


Fig. 2. Relation between calibration parameter t and power parameter β when using different sampling rates α.