GRU4Rec v2

Recurrent neural networks with top-k gains for session-based recommendations

Balázs Hidasi

@balazshidasi

GRAVITY
Research & Development

At a glance

- Improvements on GRU4Rec [Hidasi et. al, 2015]
  - Session-based recommendations with RNNs
- General
  - Negative sampling strategies
  - Loss function design
- Specific
  - Constrained embeddings
  - Implementation details
- Offline tests: up to 35% improvement
- Online tests & observations

GRU4Rec overview

Context: session-based recommendations

- User identification
  - Feasibility
  - Privacy
  - Regulations

- User intent
  - Disjoint sessions
    - Need
    - Situation (context)
    - „Irregularities"

- Session-based recommendations

- Permanent user cold-start

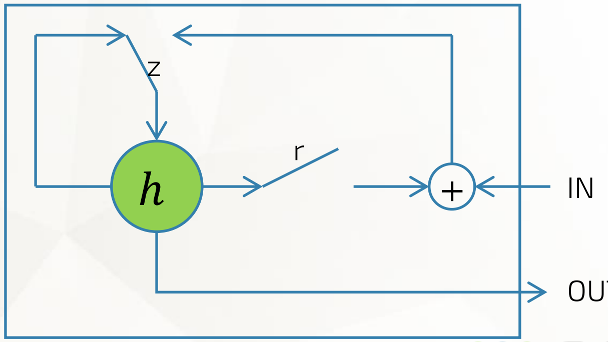
GRAVITY Research & Development

Preliminaries

- Next click prediction
- Top-N recommendation (ranking)
- Implicit feedback

Recurrent Neural Networks

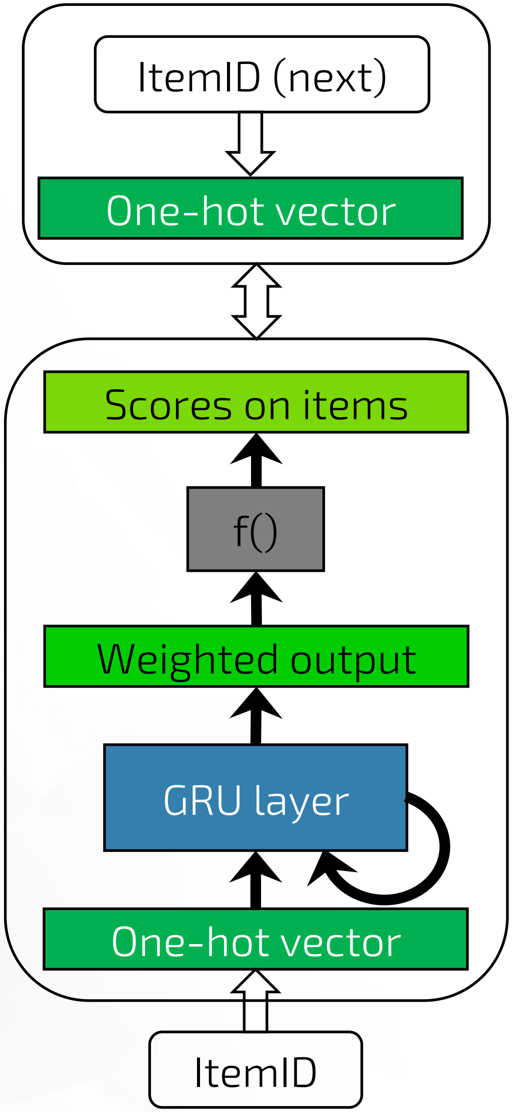
- Basics
  - Input: sequential information ($\{x_t\}_{t=1}^{T}$)
  - Hidden state (h_t):
    - representation of the sequence so far
    - influenced by every element of the sequence up to t
  - $h_t = f(Wx_t + Uh_{t-1} + b)$
- Gated RNNs (GRU, LSTM & others)
  - Basic RNN is subject to the exploding/vanishing gradient problem
  - Use $h_t = h_{t-1} + \Delta_t$ instead of rewriting the hidden state
  - Information flow is controlled by gates
- Gated Recurrent Unit (GRU)
  - Update gate (z)
  - Reset gate (r)
  - $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$
  - $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$
  - $\tilde{h}_t = \tanh(Wx_t + r_t \circ Uh_{t-1} + b)$
  - $h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$



GRAVITY Research & Development

GRU4Rec

- GRU trained on session data, adapted to the recommendation task
  - Input: current item ID
  - Hidden state: session representation
  - Output: likelihood of being the next item
- Session-parallel mini-batches
  - Mini-batch is defined over sessions
  - Update with one step BPTT
    - Lots of sessions are very short
    - 2D mini-batching, updating on longer sequences (with or without padding) didn't improve accuracy
- Output sampling
  - Computing scores for all items (100K – 1M) in every step is slow
  - One positive item (target) + several samples
  - Fast solution: scores on mini-batch targets
    - Items of the other mini-batch are negative samples for the current mini-batch
- Loss functions: cross-entropy, BPR, TOP1



GRAVITY
Research & Development

Negative sampling & loss function design

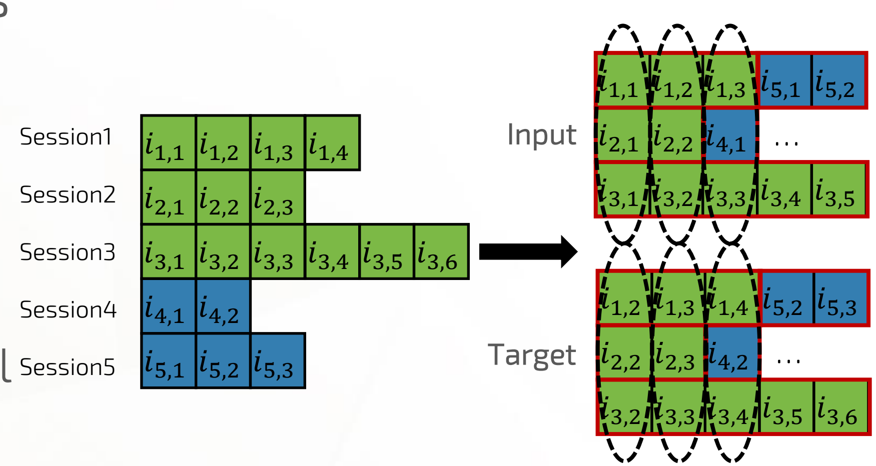
Negative sampling

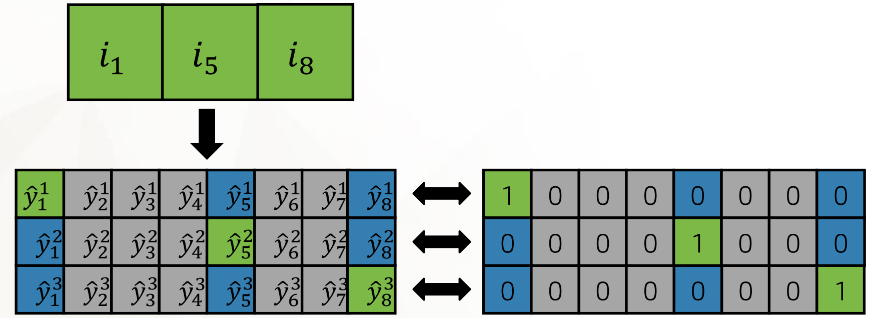
- Training step
  - Score all items
  - Push target items forward (modify model parameters)
- Many training steps & many items
  - Not scalable
  - $O(S_I N^+)$
- Sample negative examples instead → $O(KN^+)$
- Sampling probability
  - Must be quick to calculate
  - Two popular choices
    - Uniform → many unnecessary steps
    - Proportional to support → better in practice, fast start, some relations are not examined enough
  - Optimal choice
    - Data dependent
    - Changing during training might help

GRAVITY Research & Development

Mini-batch based negative sampling

- Target items of other examples from the mini-batch → as negative samples
- Pros
  - Efficient & simple implementation on GPU
  - Sampling probability proportional to support
- Cons
  - Number of samples is tied to the batch size
    - Mini-batch training: smaller batches
    - Negative sampling: larger batches
  - Sampling probability is always the same





GRAVITY
Research & Development

Solution: add more samples!

- Add extra samples
- Shared between mini-batches
- Sampling probability: $p_i \sim \mathrm{supp}^{\alpha}$
  - $\alpha = 0$ → uniform
  - $\alpha = 1$ → popularity based
- Implementation trick
  - Sampling interrupts GPU computations
  - More efficient in parallel
  - Sample store (cache)
    - Precompute 10-100M samples
    - Resample is we used them all

GRAVITY
Research & Development

Another look the loss functions

- Listwise losses on the target+negative samples
  - Cross-entropy + softmax
    - Cross-entropy in itself is pointwise
    - Target should have the maximal score
    - $XE = -\log(s_i)\,,\; s_i = \frac{e^{r_i}}{\sum_{j=1}^{K} e^{r_j}}$
    - Unstable in previous implementation
      - Rounding errors
      - Fixed
  - Average BPR-score
    - BPR in itself is pairwise
    - Target should have higher score than all negative samples
    - $BPR = -\frac{1}{K}\sum_{j=1}^{K} \log\left(\sigma\left(r_i - r_j\right)\right)$
  - TOP1 score
    - Heuristic loss, idea is similar to the average BPR
    - Score regularization part
    - $TOP1 = \frac{1}{K}\sum_{j=1}^{K} \left(\sigma\left(r_j - r_i\right) + \sigma\left(r_j^2\right)\right)$
- Earlier results
  - Similar performance
  - TOP1 is slightly better

GRAVITY Research & Development

Unexpected behaviour of pairwise losses

- Unexpected behaviour
  - Several negative samples → good results
  - Many negative samples → bad results
- Gradient (BPR wrt. target score)
  - $\frac{\partial L_i}{\partial r_i} = \frac{1}{K}\sum_{j=1}^{K}\left(1-\sigma\left(r_i-r_j\right)\right)$
- Irrelevant negative sample
  - Whose score is already lower than r_i
    - Changes during training
  - Doesn't contribute to optimization: $\left(1-\sigma\left(r_i-r_j\right)\right) \sim 0$
  - Number of irrelevant samples increases as training progresses
- Averaging with many negative samples → gradient vanishes
  - Target will be pushed up for a while
  - Slows down as approaches the top
  - More samples: slows down earlier

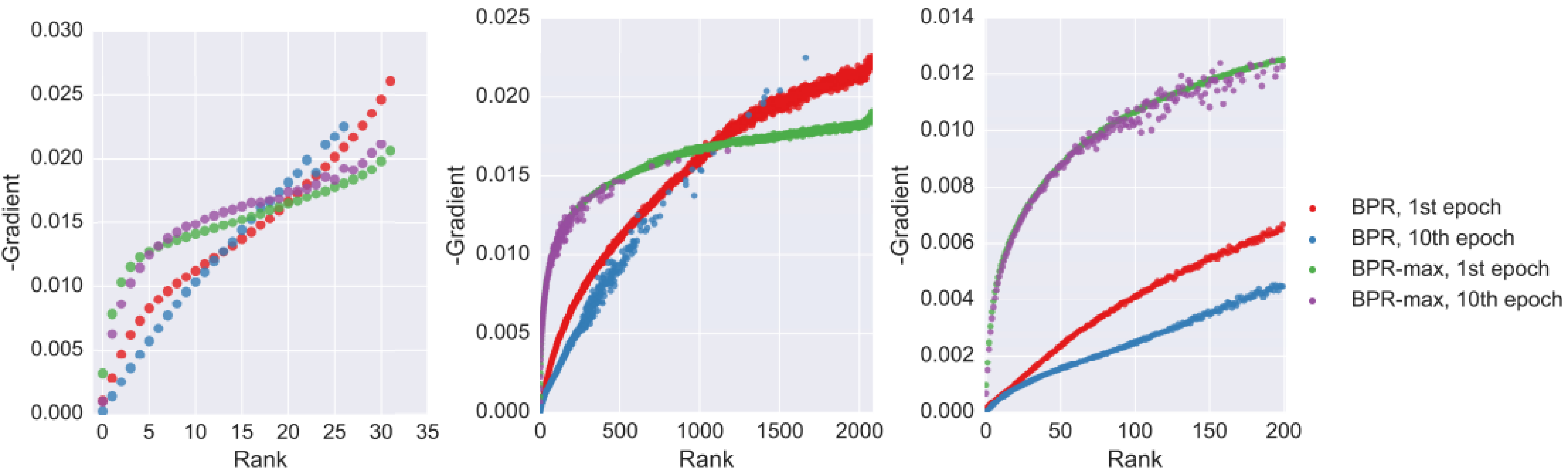
Pairwise-max loss functions

- The target score should be higher than the maximum score amongst the negative samples
- BPR-max
  - $P(r_i > r_{\mathrm{MAX}}|\theta) = \sum_{j=1}^{K} P\left(r_i > r_j \middle| r_j = r_{\mathrm{MAX}}, \theta\right) P\left(r_j = r_{\mathrm{MAX}}|\theta\right)$
  - Use continuous approximations
    - $P\left(r_i > r_j \middle| r_j = r_{\mathrm{MAX}}, \theta\right) = \sigma\left(r_i - r_j\right)$
    - $P\left(r_j = r_{\mathrm{MAX}}|\theta\right) = \mathrm{softmax}\left(\{r_k\}_{k=1}^{K}\right) = \frac{e^{r_j}}{\sum_{k=1}^{K} e^{r_k}} = s_j$
      - Softmax over negative samples only
  - Minimize negative log probability
  - Add ℓ_2 score regularization
- $L_i = -\log\left(\sum_{j=1}^{K} s_j \sigma\left(r_i - r_j\right)\right) + \lambda \sum_{j=1}^{K} r_j^2$

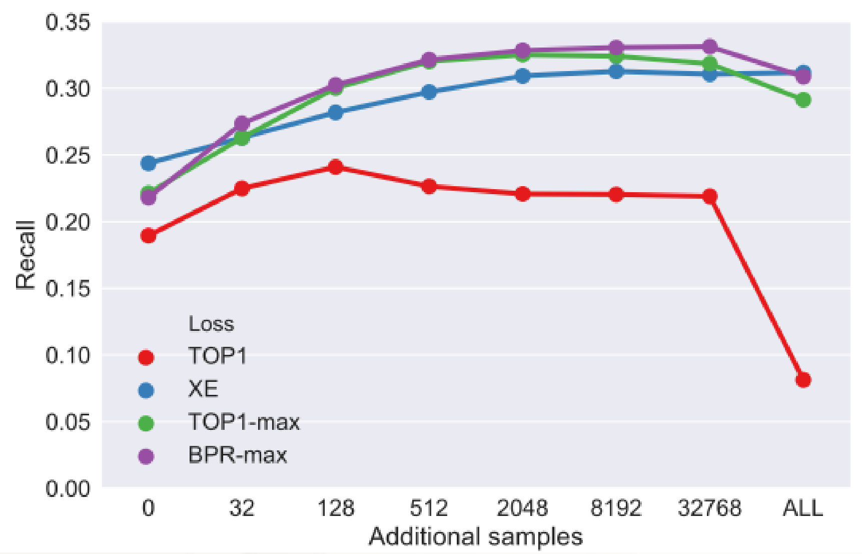
GRAVITY Research & Development

Gradient of pairwise max losses

- BPR-max (wrt. r_i)
  - $\frac{\partial L_i}{\partial r_i} = \frac{\sum_{j=1}^{K} s_j \sigma(r_i - r_j)\left(1 - \sigma(r_i - r_j)\right)}{\sum_{j=1}^{K} s_j \sigma(r_i - r_j)}$
  - Weighted average of BPR gradients
    - Relative importance of samples: $\frac{s_j \sigma(r_i - r_j)}{s_k \sigma(r_i - r_k)} = \frac{e^{r_j} + e^{r_j + r_k - r_i}}{e^{r_k} + e^{r_j + r_k - r_i}}$
    - Smoothed softmax
      - If $r_i \gg \max(r_j, r_k)$ → behaves like softmax
      - Stronger smoothing otherwise
      - Uniform → softmax as r_i is pushed to the top

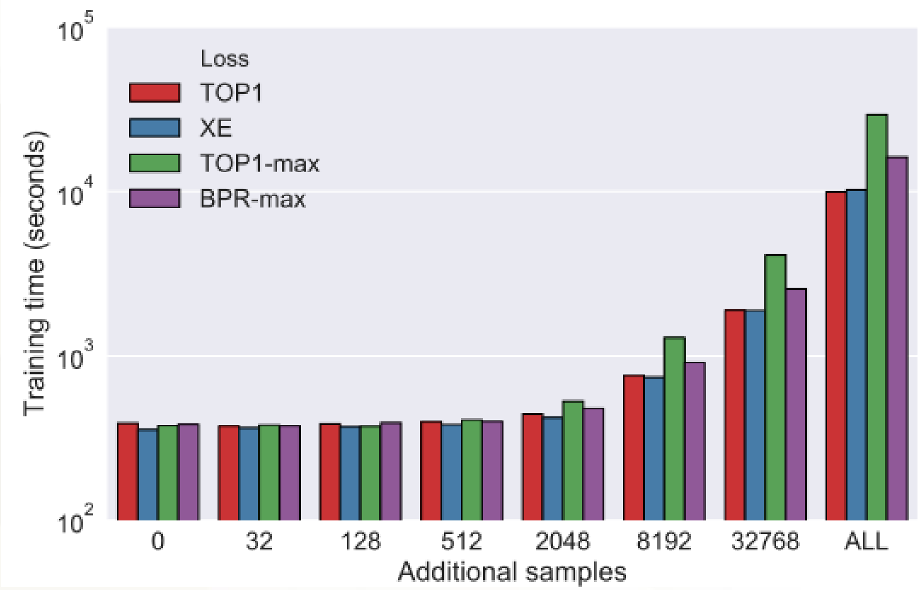


Number of samples: training times & performance



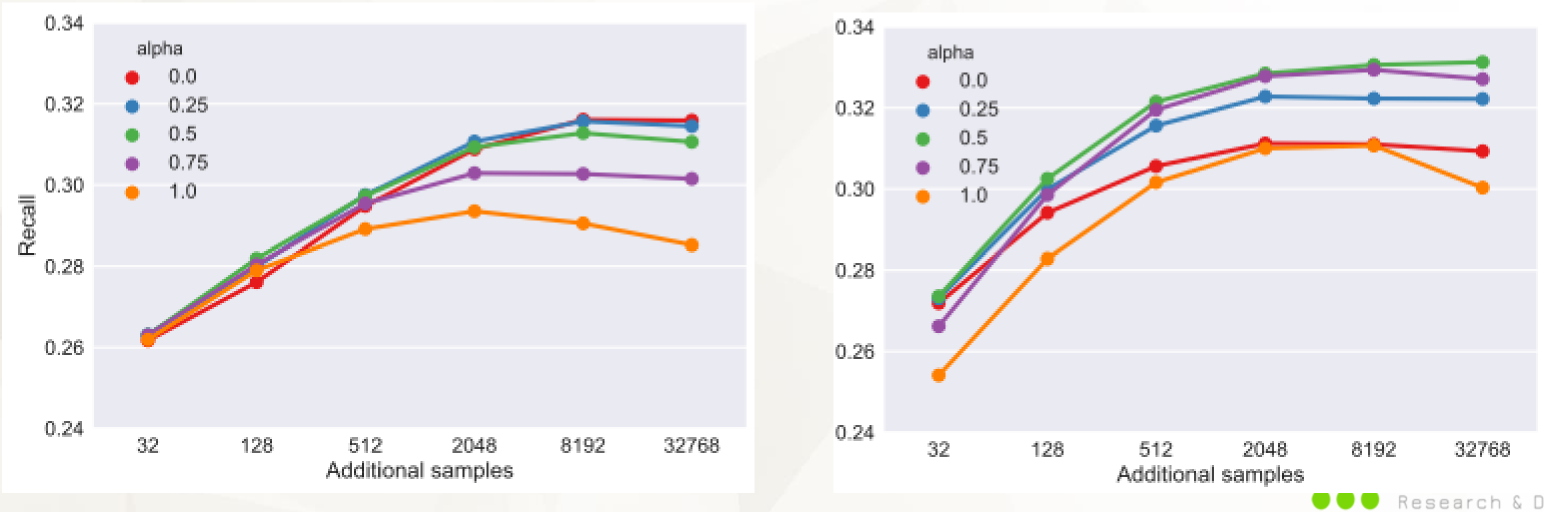
- Significant improvements up to a certain point
- Diminishing returns after

- Training times on GPU don't increase until the parallelization limit is reached
- Around the same place



The effect of the α parameter

- Data & loss dependent
  - Cross-entropy: favours lower values
  - BPR-max: 0.5 is usually a good choice (data dependent)
  - There is always a popularity based part of the samples
    - Original mini-batch examples
    - Removing these will result in higher optimal α
- CLASS dataset (cross-entropy, BPR-max)



Constrained embeddings & offline tests

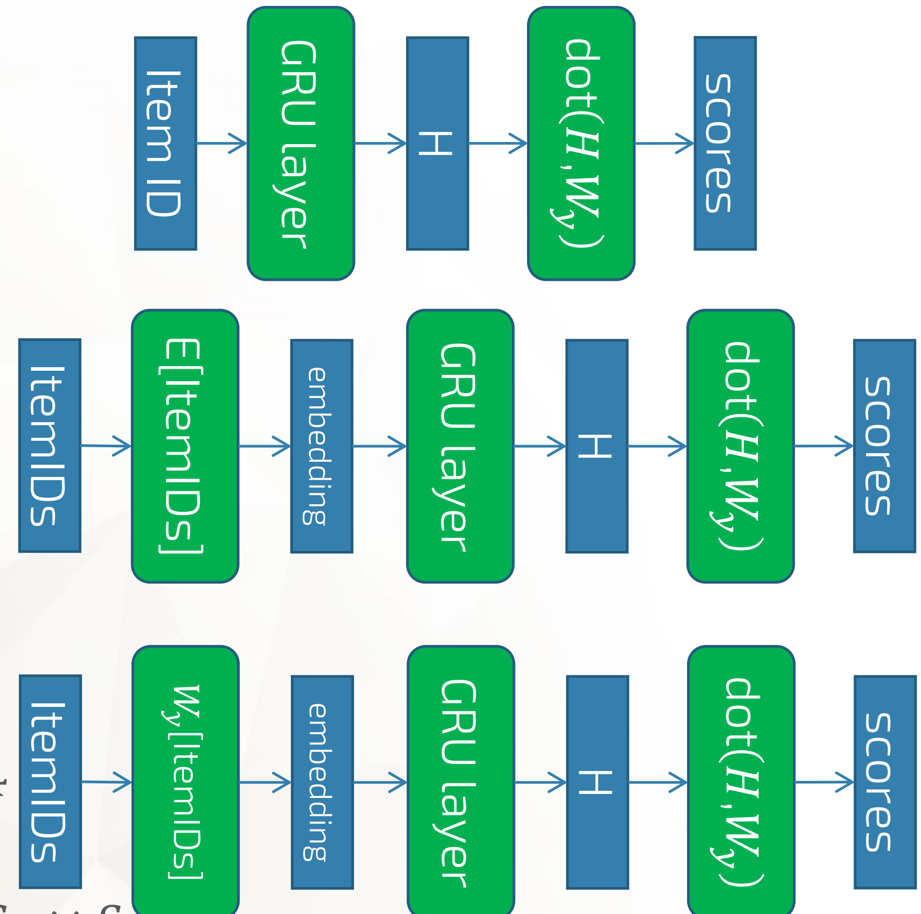
Unified item representations in GRU4Rec (1/2)

- Hidden state x W_y → scores
  - One vector for each item → „item feature matrix"
- Embedding
  - A vector for each item → another „item feature matrix"
  - Can be used as the input of the GRU layer instead of the one-hot vector
  - Slightly decreases offline metrics
- Constrained embeddings (unified item representations)
  - Use the same matrix for both input and output embedding
  - Unified representations → faster convergence
  - Embedding size tied to hidden the size of the last hidden state

GRAVITY Research & Development

Unified item representations in GRU4Rec (2/2)

- Model size: largest components scale with the items
  - One-hot input: X
    - $W_x^0, W_r^0, W_z^0, W_y \rightarrow S_I \times S_H$
    - $U_h^0, U_r^0, U_z^0 \rightarrow S_H \times S_H$
  - Embedding input: $X/2$
    - $E, W_y \rightarrow S_I \times S_H$
    - $W_x^0, W_r^0, W_z^0 \rightarrow S_E \times S_H$
    - $U_h^0, U_r^0, U_z^0 \rightarrow S_H \times S_H$
  - Constrained embedding: $X/4$
    - $W_y \rightarrow S_I \times S_H$
    - $W_x^0, W_r^0, W_z^0, U_h^0, U_r^0, U_z^0 \rightarrow S_H \times S_H$



GRAVITY
Research & Development

Offline results

- Over item-kNN
  - +25-52% in recall@20
  - +35-55% in MRR@20
- Over the original GRU4Rec
  - +18-35% in recall@20
  - +27-37% in MRR@20
- BPR-max vs. (fixed) cross-entropy
  - +2-6% improvement in 2 of 4 cases
  - No statistically significant difference in the other 2 case

Dataset	Item kNN	GRU4Rec original	GRU4Rec XE	GRU4Rec with additional samples TOP1	GRU4Rec with additional samples XE	GRU4Rec with additional samples TOP1-max	GRU4Rec with additional samples BPR-max
Recall@20							
RSC15	0.5065	0.5853	0.5781	0.6117 (+20.77%, +4.51%)	0.7208 (+42.31%, +23.15%)	0.7086 (+39.91%, +21.07%)	**0.7211 (+42.37%, +23.20%)**
VIDEO	0.5201	0.5051	0.5060	0.5325 (+2.40%, +5.43%)	0.6400 (+23.06%, +26.72%)	0.6421 (+23.46%, +27.12%)	**0.6517 (+25.31%, +29.03%)**
VIDXL	0.6263	0.6831	0.7046	0.6723 (+7.35%, -1.58%)	0.8028 (+28.19%, +17.53%)	0.7935 (+26.70%, +16.16%)	**0.8058 (+28.66%, +17.97%)**
CLASS	0.2201	0.2478	0.2545	0.2342 (+6.41%, -5.50%)	0.3137 (+42.54%, +26.61%)	0.3252 (+47.75%, +31.22%)	**0.3337 (+51.61%, +34.66%)**
MRR@20							
RSC15	0.2048	0.2305	0.2375	0.2367 (+15.61%, +2.69%)	0.3137 (+53.16%, +36.08%)	0.3045 (+48.70%, +32.08%)	**0.3170 (+54.78%, +37.52%)**
VIDEO	0.2257	0.2359	0.2609	0.2295 (+1.69%, -2.73%)	0.3079 (+36.42%, +30.52%)	0.2950 (+30.72%, +25.05%)	**0.3089 (+36.87%, +30.95%)**
VIDXL	0.3740	0.3847	0.4343	0.3608 (-3.53%, -6.21%)	0.5031 (+34.52%, +30.78%)	0.4939 (+32.05%, +28.39%)	**0.5066 (+35.45%, +31.68%)**
CLASS	0.0799	0.0949	0.0995	0.0870 (+8.83%, -8.36%)	0.1167 (+46.08%, +22.99%)	0.1198 (+49.93%, +26.25%)	**0.1202 (+50.40%, +26.63%)**

- Constrained embedding
  - Most cases: slightly worse MRR & better recall
  - Huge improvements on the CLASS datasset (+18.74% in recall, +29.44% in MRR)

GRAVITY
Research & Development

Online A/B tests

GRAVITY
Research & Development

A/B test - video service (1/3)

- Setup
  - Video page
  - Recommended videos on a strip
  - Autoplay functionality
  - Recommendations are NOT recomputed if the user clicks on any of the recommended videos or autoplay loads a new video
  - User based A/B split
- Algorithms
  - Original algorithm: previous recommendation logic
  - GRU4Rec next best: N guesses for the next item
  - GRU4Rec sequence: sequence of length N as the continuation of the current session
    - Greedy generation

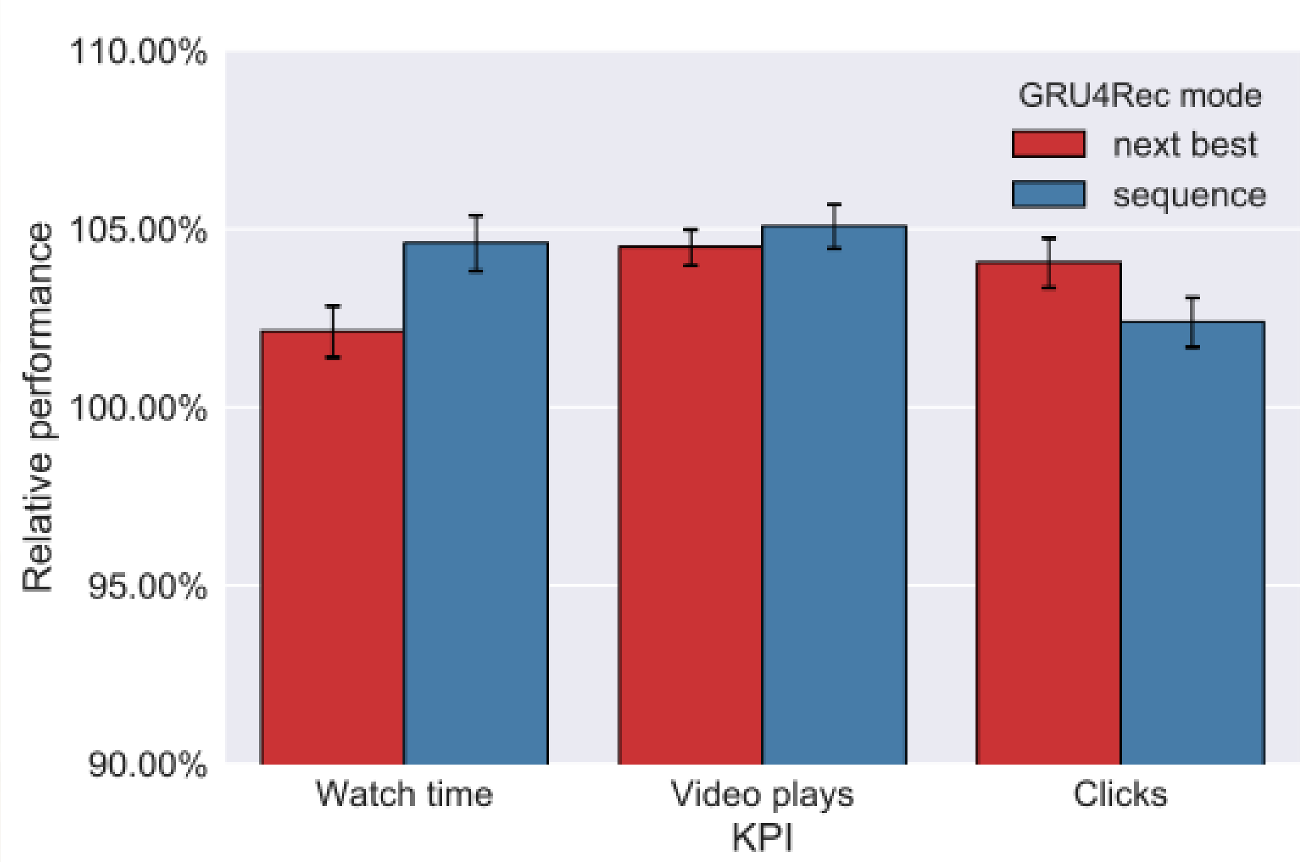
GRAVITY Research & Development

A/B test - video service (2/3)

- Technical details
  - User based A/B split
  - GRU4Rec serving from a single GPU using a single thread
    - Score computations for ~500K items in 1-2ms (next best)
  - Constant retraining
    - GRU4Rec: ~1.5 hours on ~30M events (including data collection and preprocessing)
    - Original logic: different parts with different frequency
  - GRU4Rec falls back to the original logic if it can't recommend or times out
- KPIs (relative to the number of recommendation requests)
  - Watch time
  - Videos played
  - Recommendations clicked
- Bots and power users are excluded from the KPI computations

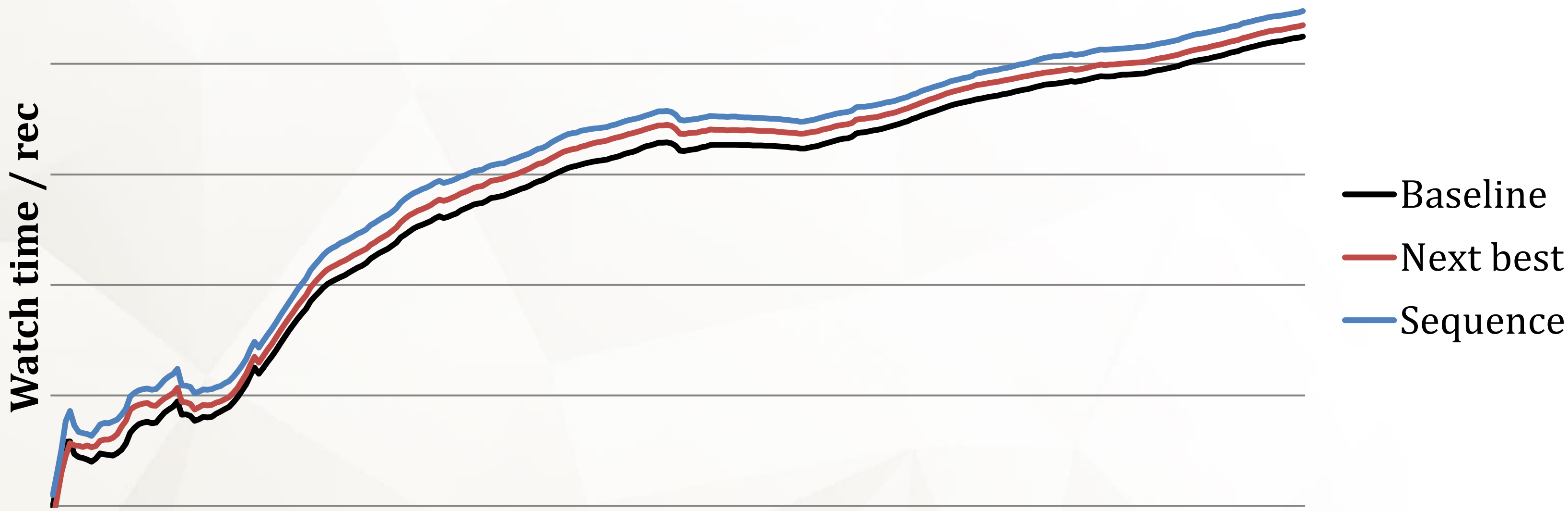
GRAVITY
Research & Development

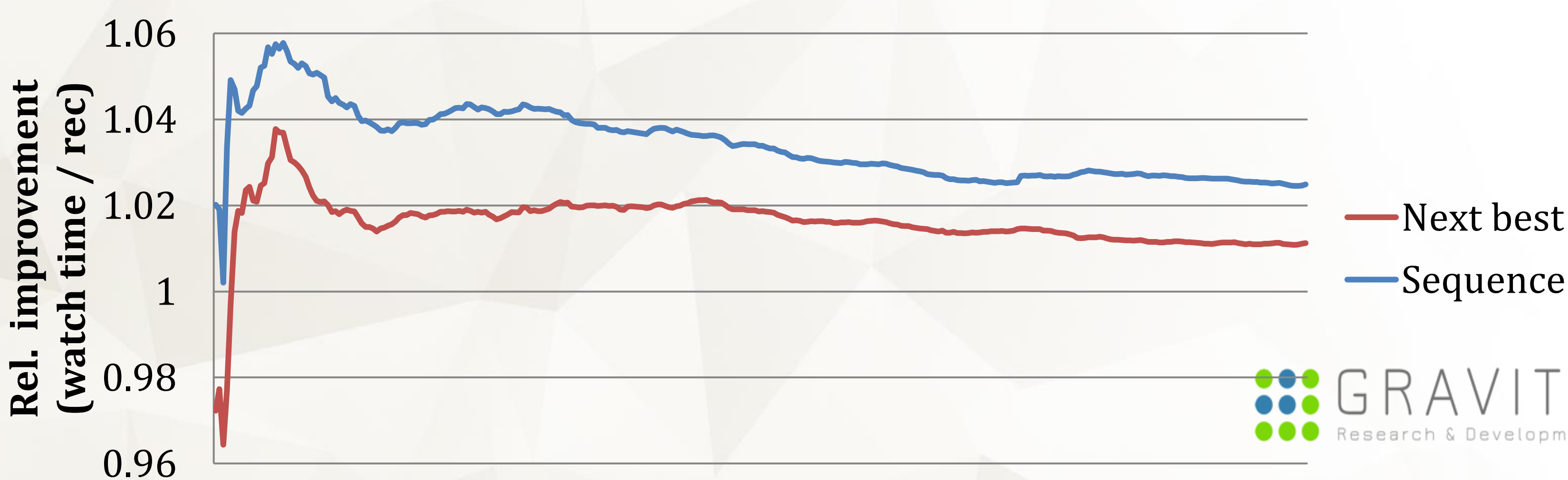
A/B test - video service (3/3)



A/B test – long term effects (1/3)

- Watch time

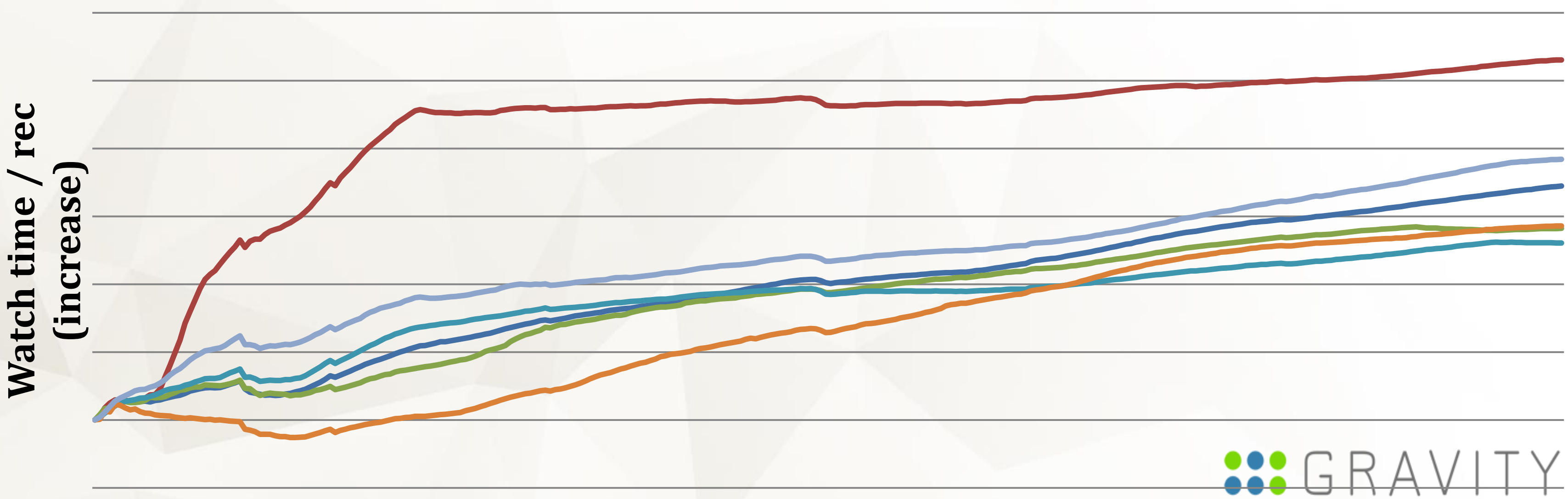




GRAVITY Research & Development

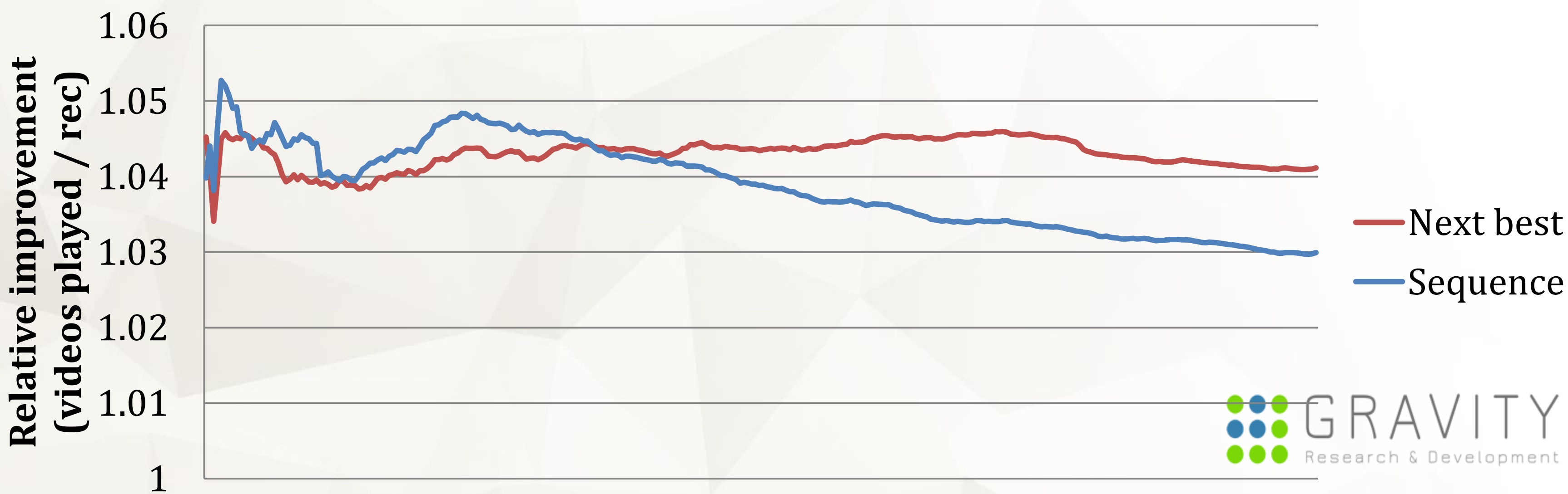
A/B test – long term effects (2/3)

- GRU4Rec: strong generalization capabilities
  - Finds hidden gems
  - Unlike counting based approaches
    - Not obvious in offline only testing
- Feedback loop
  - Baseline trains also sees the feedback generated for recommendations of other groups
  - Learns how to recommend hidden gems
- GRU4Rec maintains some lead
  - New items are constantly uploaded
- Comparison of different countries



A/B test – long term effects (3/3)

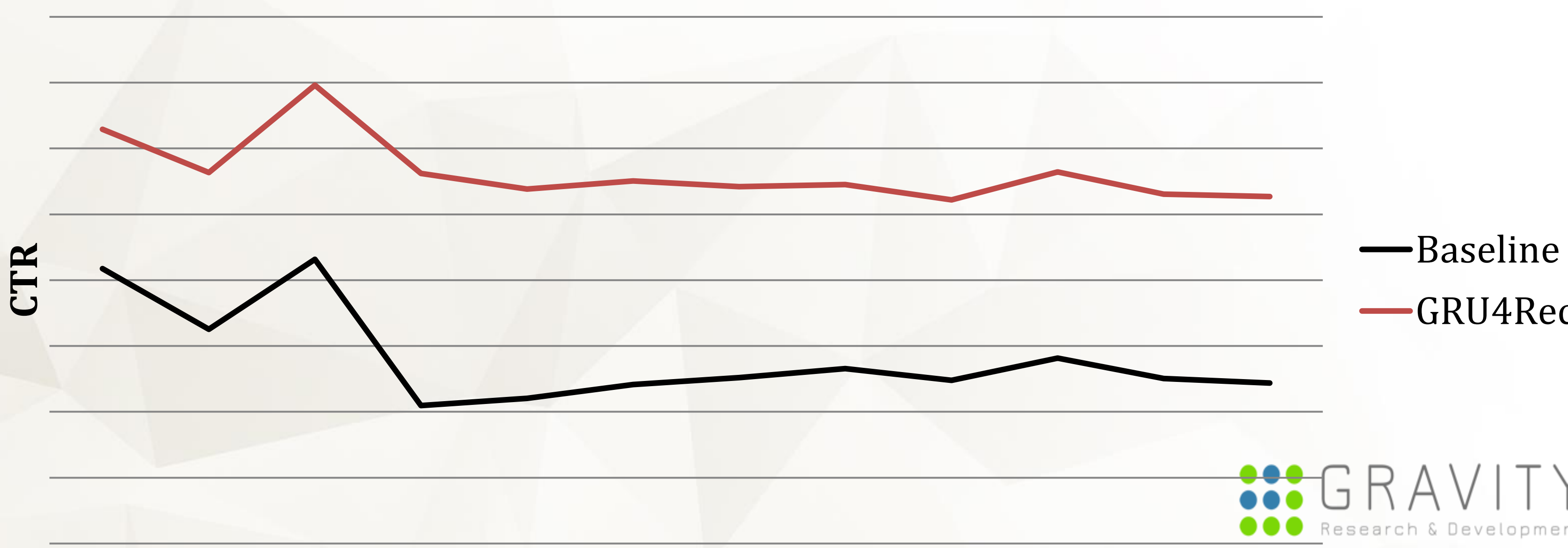
- Videos played
  - Next best and sequence switched places
  - Sequence mode: great for episodic content, can suffer otherwise
  - Next best mode: more diverse, better for non-episodic content
  - Feedback loop: next best learns some of the episodic recommendations



GRAVITY
Research & Development

A/B test - online marketplace

- Differences in setup
  - On home page
  - Next best mode only
  - KPI: CTR
- Items have limited lifespan
  - Will GRU4Rec keep its 19-20% lead?



GRAVITY
Research & Development

Thank you!
Q&A

Check out the code (free for research):
https://github.com/hidasib/GRU4Rec

Read the preprint: https://arxiv.org/abs/1706.03847

GRAVITY
Research & Development