Scalable representation learning and retrieval for online advertising

Olivier Koch
November 24, 2020

Introduction

Publications

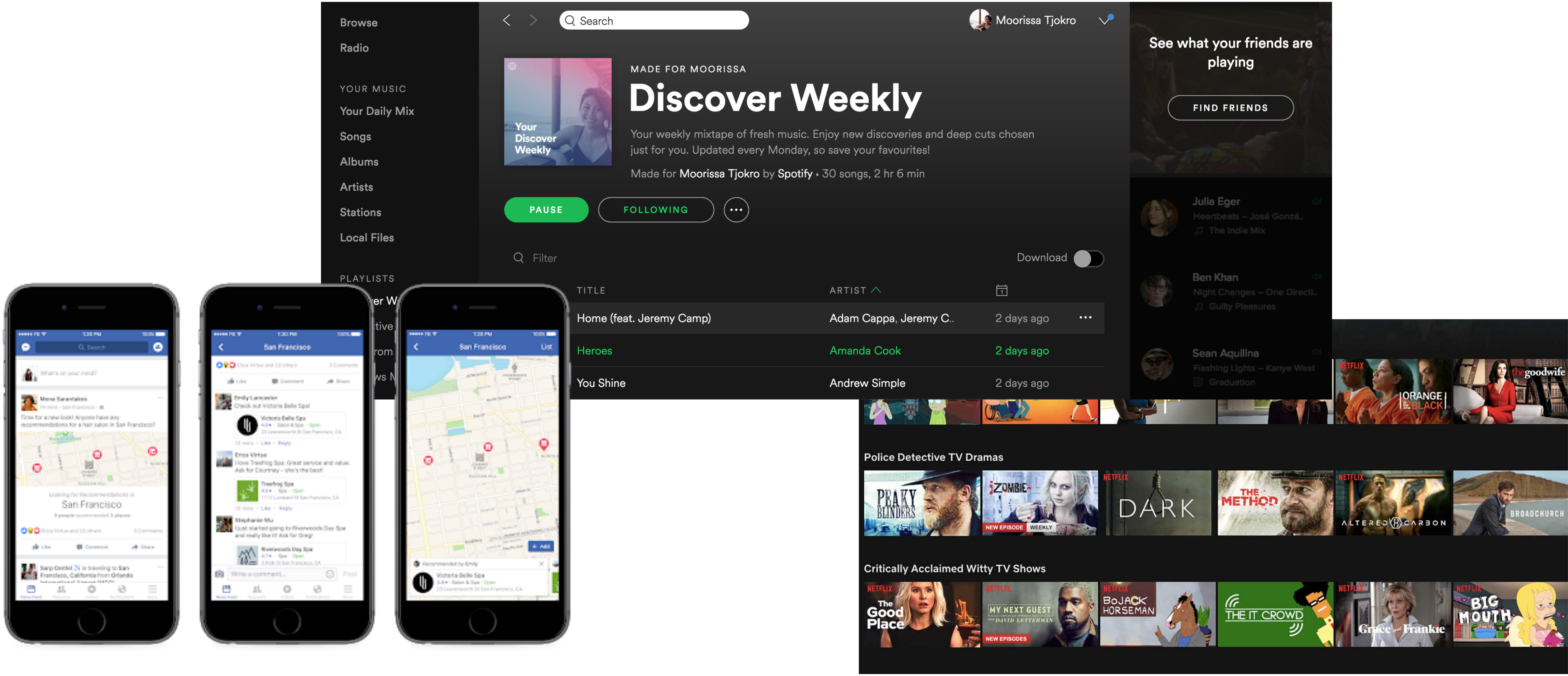
CVPR'07 Olivier Koch, Seth Teller, Wide-area egomotion estimation from known 3D structure [pdf]

IJFR'08 Leonard et al., A Perception Driven Autonomous Urban Robot [pdf]

ICCV'09 Olivier Koch, Seth Teller, Body-relative navigation using uncalibrated cameras [pdf]

ICRA'10 Olivier Koch, Matthew R. Walter, Albert S. Huang, and Seth Teller, Ground robot navigation using uncalibrated cameras [pdf]

Recommendation is everywhere



Problem statement

How to build *fast* and *scalable* machine learning algorithms in a context of representation learning?

Scalable representation learning and retrieval for online advertising, submitted to TheWebConf’21

Outline

The task

Efficient models

Experiments and results

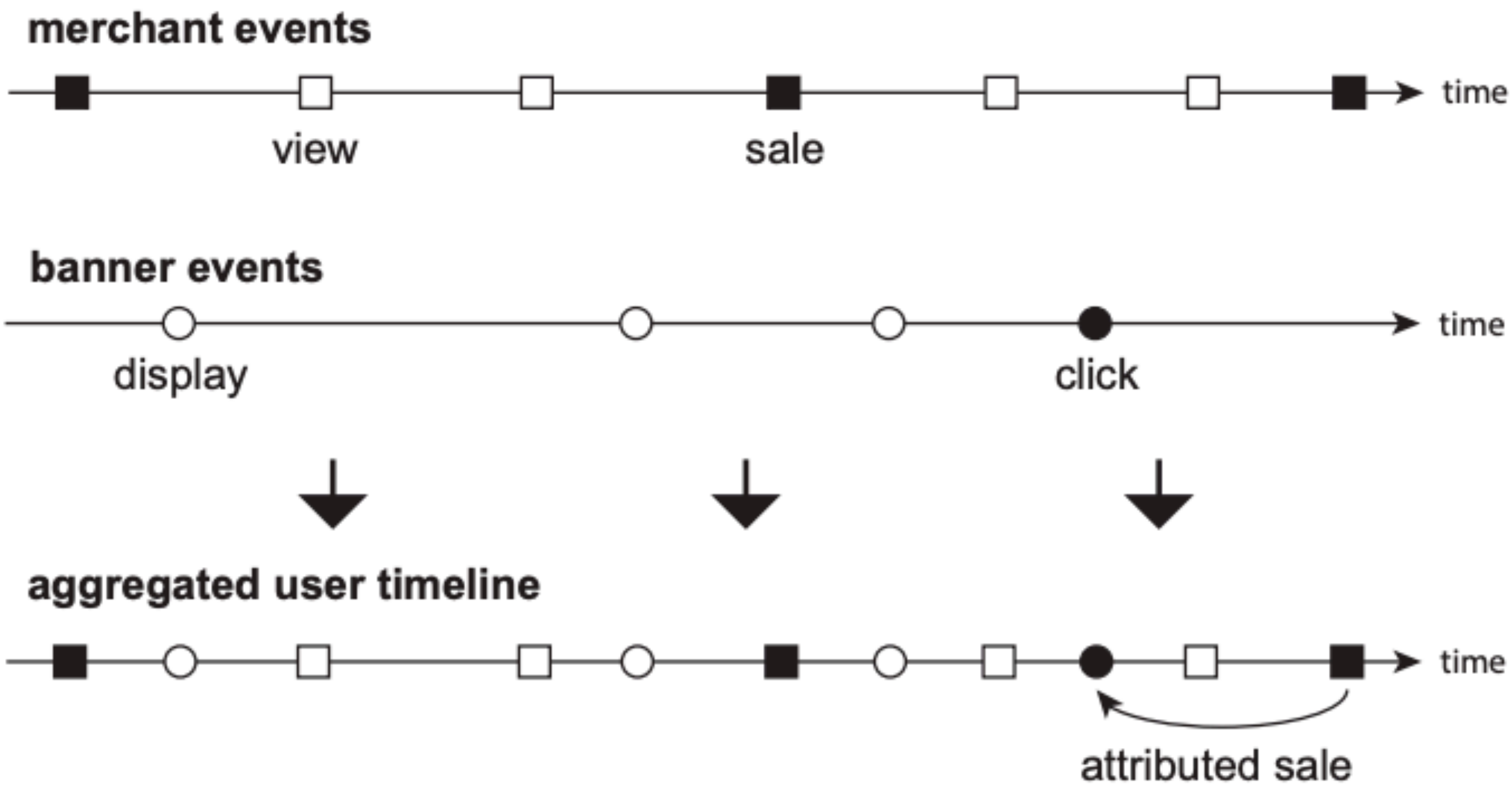
Real-time retrieval at scale

A/B testing results

Next steps

The recommendation task at Criteo

The recommendation task at Criteo



Challenges

1. Scale (billions of users, millions of items)
2. Latency (a few ms)
3. User churn
4. Multiple feedbacks (clicks, views)

Related work

Matrix factorization [4]

Variational auto-encoders (VAE) [1,2]

Graph networks [6,7]

Sparse linear methods [3]

Neural networks [8, 9]

Collaborative metric learning [5]

Representation learning

1. Build product embeddings from the data
2. Build user embeddings from product embeddings and the data
3. Find best products by searching for nearest neighbors around a user embedding

Our contributions

1. An efficient model (LED, for Lightweight Encoder-Decoder) reaching state-of-the-art performance with significant advantages of scale

2. A detailed architecture covering both offline training and real-time serving

3. Extensive experimentation demonstrating the efficiency of the system on real traffic over two months

Efficient models

1. Fast nearest-neighbor search

   Leverage scalable KNN methods

2. Amortized inference [13]

   Share the same procedure to compute user representations

3. Sampling-based losses

   Use a loss sub-linear in the number of items

4. Pre-training

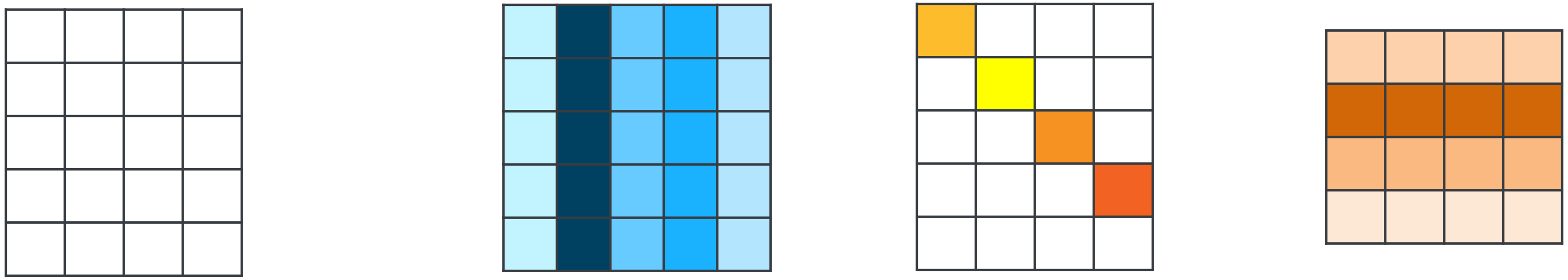
   Leverage high-volumes of data to pre-train on common events (views), then fine-tune on sparser events (clicks)

Fast nearest-neighbor search

For a given user u, the system ranks items with a scoring function expressed as an inner product $s(u,i) = \langle u, v_i \rangle$

Finding the k best items for user u is thus equivalent to finding the k nearest neighbors of u with maximum inner product.

Pretraining with large-scale SVD



$$A = U S V^T$$

m x n   m x m   m x n   n x n

Randomized SVD for large-scale factorization

Trick: Approximate A with a tall-and-skinny matrix Q

$$\boldsymbol{Q} \text{ has orthonormal columns and } \boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^{*}\boldsymbol{A}.$$

1. Form $\boldsymbol{B} = \boldsymbol{Q}^{*}\boldsymbol{A}$, which yields the low-rank factorization $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{B}$.
2. Compute an SVD of the small matrix: $\boldsymbol{B} = \widetilde{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^{*}$.
3. Set $\boldsymbol{U} = \boldsymbol{Q}\widetilde{\boldsymbol{U}}$.

How do we find Q?

1/ Generate random matrix $G \in R^{m \times (k+p)}$

with values drawn independently from gaussian distribution

where k - target approximation rank, p - oversampling

2/ Multiply by A several times: $\hat{Q} = (AA^T)^q AG$

3/ Orthogonolize after each iteration $AG = \hat{Q}R$

4/ Orthogonolize at the end $\hat{Q} = QR$

Approximation error bound

By Corollary 1.5 from [Witten, 2013]:

$$||A - QQ^T A|| \leq \sigma_{k+1} \left(\frac{\sqrt{n-k} + \sqrt{k}}{\sqrt{k+p} - \sqrt{k}} \right)^{1/(1+2q)}$$

With n=10^9, k=100, p=30, q=3:

$$||A - QQ^T A|| \leq 4.19 \times \sigma_{k+1}$$

Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp, Journal SIAM, May 2011

Fine-tuning SVD embeddings

Approach 1: train

Consider the SVD-initialized embeddings as trainable parameters

Approach 2: project

Learn a matrix $P \in \mathbb{R}^{d \times d}$ such that $\vec{v}_i = P \cdot \overrightarrow{v_i^{\text{SVD}}}$

Sampling-based losses

Unsampled loss (multinomial) uses a softmax:

$$\pi(i|u) \propto \exp\left(s(u,i)\right)$$

and involves a costly partition function:

$$Z(u) = \sum_{i=1}^{I} \exp\left(s(u,i)\right)$$

Sampling-based losses

Sampled loss 1: Complementarity Sum Sampling (CSS) [12]

$$\widehat{Z}_i = \exp\left(s(u,i)\right) + \frac{I-1}{N}\sum_{n=1}^{N}\exp(s(u,n)).$$

Sampling-based losses

Sampled loss 2: Bayesian Personalized Ranking (BPR) [11]

$$\sum_{n=1}^{N} \log \sigma(s(u, i) - s(u, n))$$

Sampled loss 3: Negative Sampling (NS) [10]

$$\log \sigma(s(u, i)) + \sum_{n=1}^{N} \log (1 - \sigma(s(u, n))) \, .$$

Amortized inference with LED

Encode user timeline with a simple average:

$$\vec{u} = \frac{1}{T}\sum_{t=1}^{T} \overrightarrow{v_{u_t}}$$

Decode the user representation to retrieve recommendation scores:

$$s(u, i) = \langle \vec{u}, \vec{v_i} \rangle + b_i$$

Evaluation: key take-aways

1. Training models using a sampling-based loss (Multi-CSS, BPR or NS) gives results close to the state-of-the-art, while enabling training at scale.
2. The number of sampled negatives should be carefully set depending on the computation budget and performance target.
3. The LED model achieves competitive performance compared to the state-of-the-art despite its simplicity.
4. Pre-training embeddings on view events and fine-tuning them using a projection matrix is an effective way to transfer knowledge to the click prediction task.

Experimental setup

Table 1: Dataset statistics. Density refers to the density of the item-item matrix.

	users	items	events	density %
ML20M	136K	20K	10M	2.47
Products	587M	5M	19B	0.076

Metrics

Recall @ 20, 50 (top-k retrieval task)

Click rank: normalized rank of a clicked item among other items in one banner sorted by score. Ranges from 0.5 (random system) to 0 (perfect system, clicked item has the highest score returned by the model)

Baselines

VAE: Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational Autoencoders for Collaborative Filtering, WWW’18. [pdf]

EASE: Harald Steck. Embarrassingly Shallow Autoencoders for Sparse Data, WWW’19 [pdf]

SLIM: Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for TopN Recommender Systems, ICDM’11 [pdf]

WMF: Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets, ICDM’08. [pdf]

CML: Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative Metric Learning, WWW’17 [pdf]

Making SOTA scalable with the sampled losses

Table 2: Comparison of VAE and LED models with and without sampling on ML20M. Percentages measure relative difference with the *Mult-VAE*. Only lines marked with a † are scalable. The LED model trained with BPR achieves results close to the Mult-VAE while enabling training at scale.

ML20M dataset			
Model	Loss	Recall@20	Recall@50
VAE [24]	Mult	**0.396**	**0.537**
VAE	Mult-CSS †	0.382 (-3.54%)	0.523 (-2.61%)
DAE [24]	Mult	0.387 (-2.27%)	0.524 (-2.42%)
LED	Mult	0.379 (-4.29%)	0.517 (-3.72%)
LED	Mult-CSS †	0.368 (-7.07%)	0.506 (-5.77%)
LED	BPR †	**0.375** (-5.30%)	**0.516** (-3.91%)
LED	NS †	0.375 (-5.30%)	0.514 (-4.28%)
EASE [42]		0.391 (-1.26%)	0.521 (-2.98%)
WMF [10]		0.360 (-9.09%)	0.498 (-7.26%)
SLIM [32]		0.370 (-6.57%)	0.495 (-7.82%)
CML [9]		-	0.466 (-13.22%)

More negatives is better, but with just 10 negatives, we lose only 5% in performance

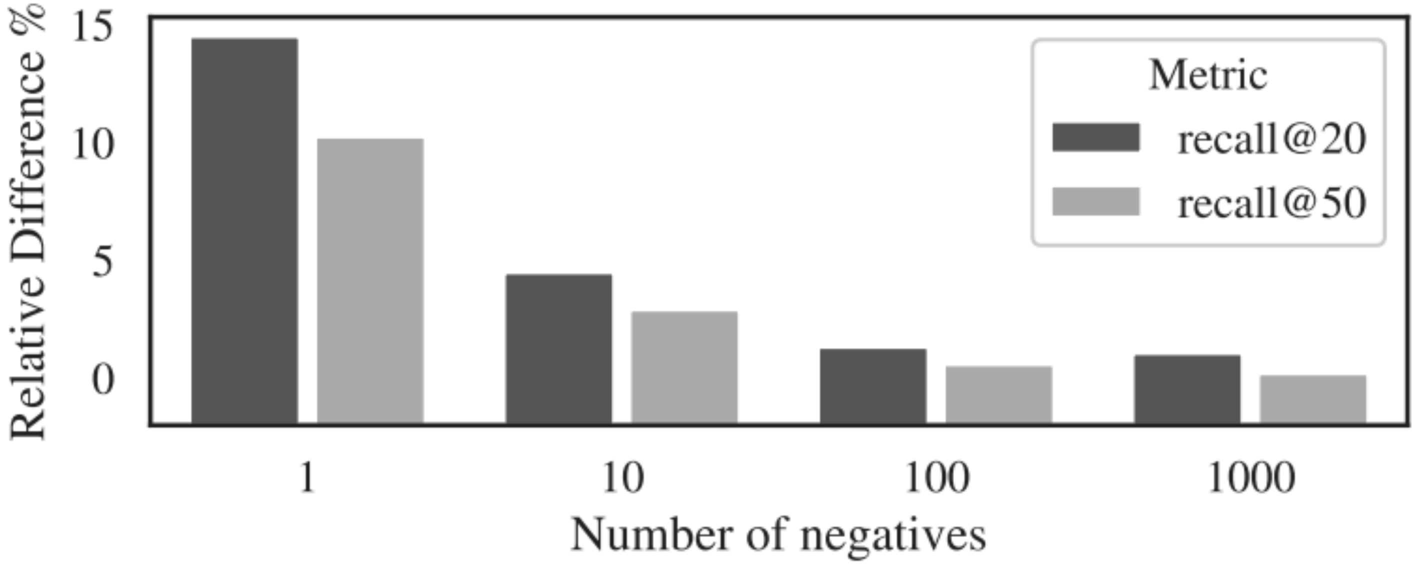


Figure 2: Relative performance drop of *LED* trained with BPR instead of multinomial likelihood (smaller is better) on ML20M. With only 10 negatives, the drop is less than 5 %.

Choosing the right init and fine-tuning method

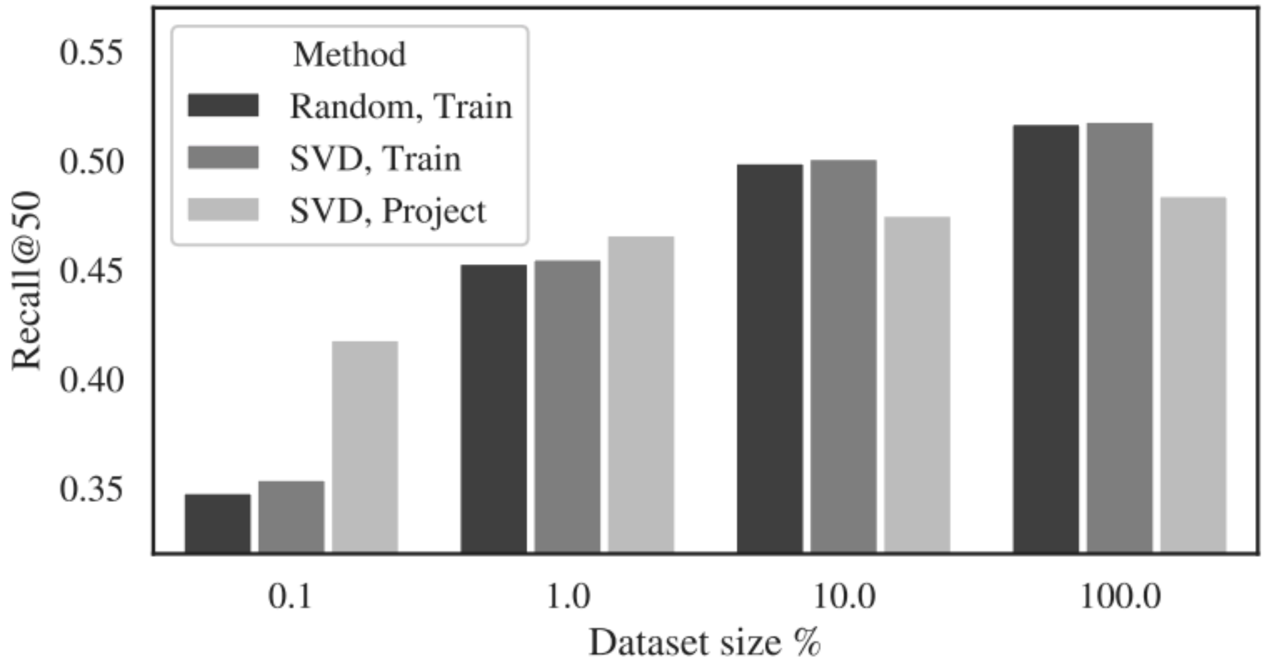


Figure 3: Recall@50 of *LED* for different initialization and fine-tuning methods on ML20M. The SVD embeddings are always pre-trained on the full training set. The *project* method outperforms both random initialization and classical fine-tuning for models trained on small fractions of the dataset.

Choosing the right init and fine-tuning method

Projection works better when model is trained on small subsets of the dataset.

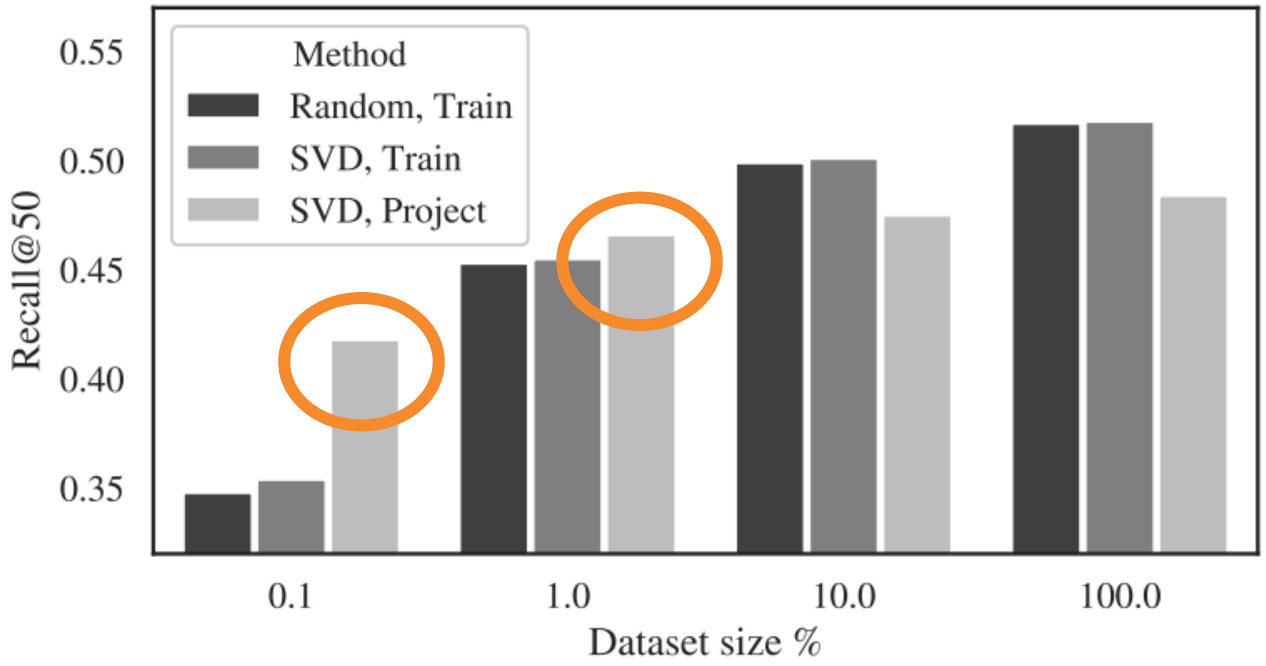


Figure 3: Recall@50 of *LED* for different initialization and fine-tuning methods on ML20M. The SVD embeddings are always pre-trained on the full training set. The *project* method outperforms both random initialization and classical fine-tuning for models trained on small fractions of the dataset.

Choosing the right init and fine-tuning method

Projection still performs decently, while it learns a *dxd* matrix instead of the full embedding matrix *lxd*

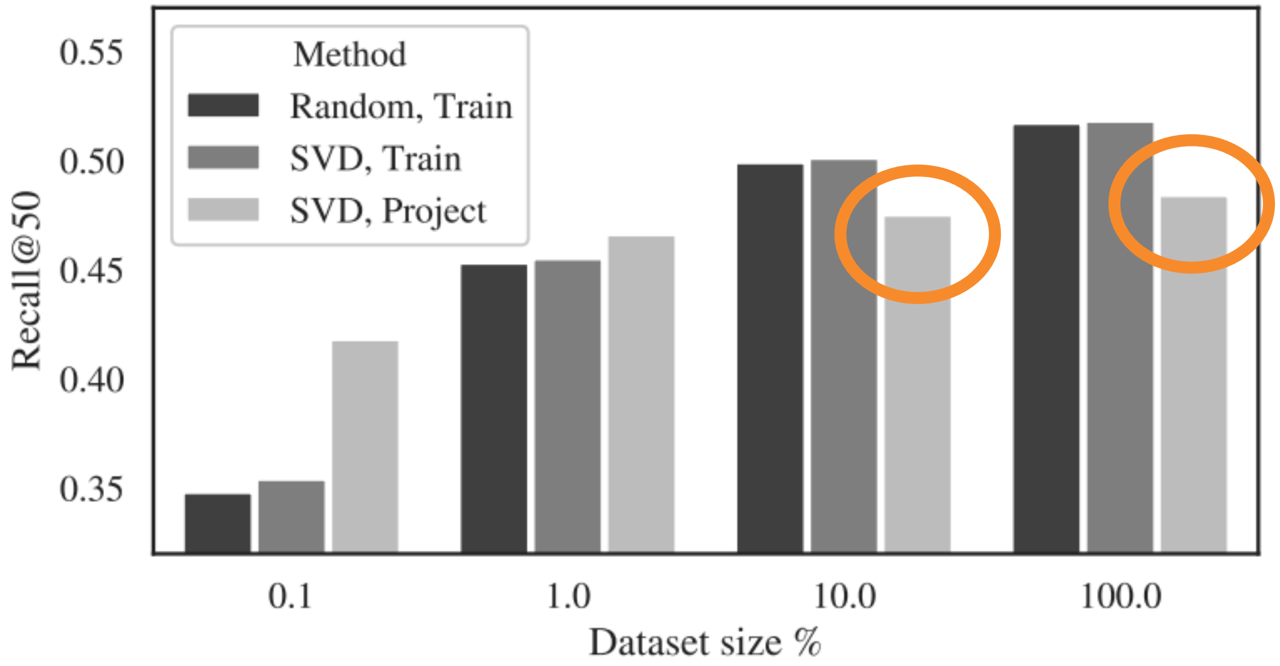


Figure 3: Recall@50 of *LED* for different initialization and fine-tuning methods on ML20M. The SVD embeddings are always pre-trained on the full training set. The *project* method outperforms both random initialization and classical fine-tuning for models trained on small fractions of the dataset.

LED performs better than VAE on the Products dataset

Table 3: Impact of pre-training and fine-tuning methods on the Products dataset. Despite its simplicity, LED outperforms the VAE. Pre-training is particularly effective, yielding better results than random initialization.

Products dataset				
Model	Init	Tuning	R@20	ClickRank
VAE	Random	Train	0.078	0.471
VAE	SVD	Train	0.083	0.457
VAE	SVD	Proj	**0.091**	**0.454**
LED	Random	Train	0.099	0.468
LED	SVD	Train	**0.109**	0.454
LED	SVD	Proj	0.104	**0.450**

Outline

The task

Efficient models

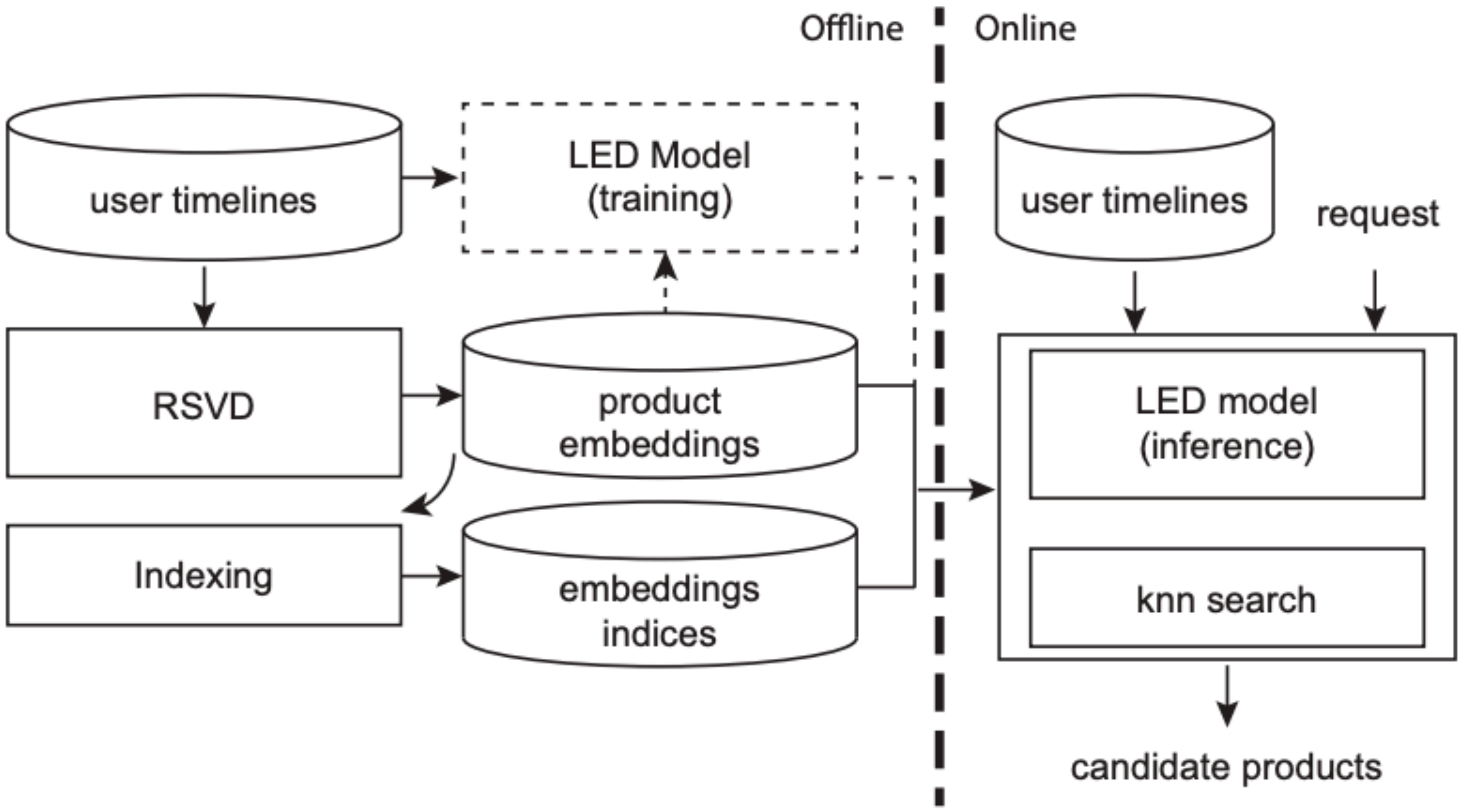
Experiments and results

Real-time retrieval at scale

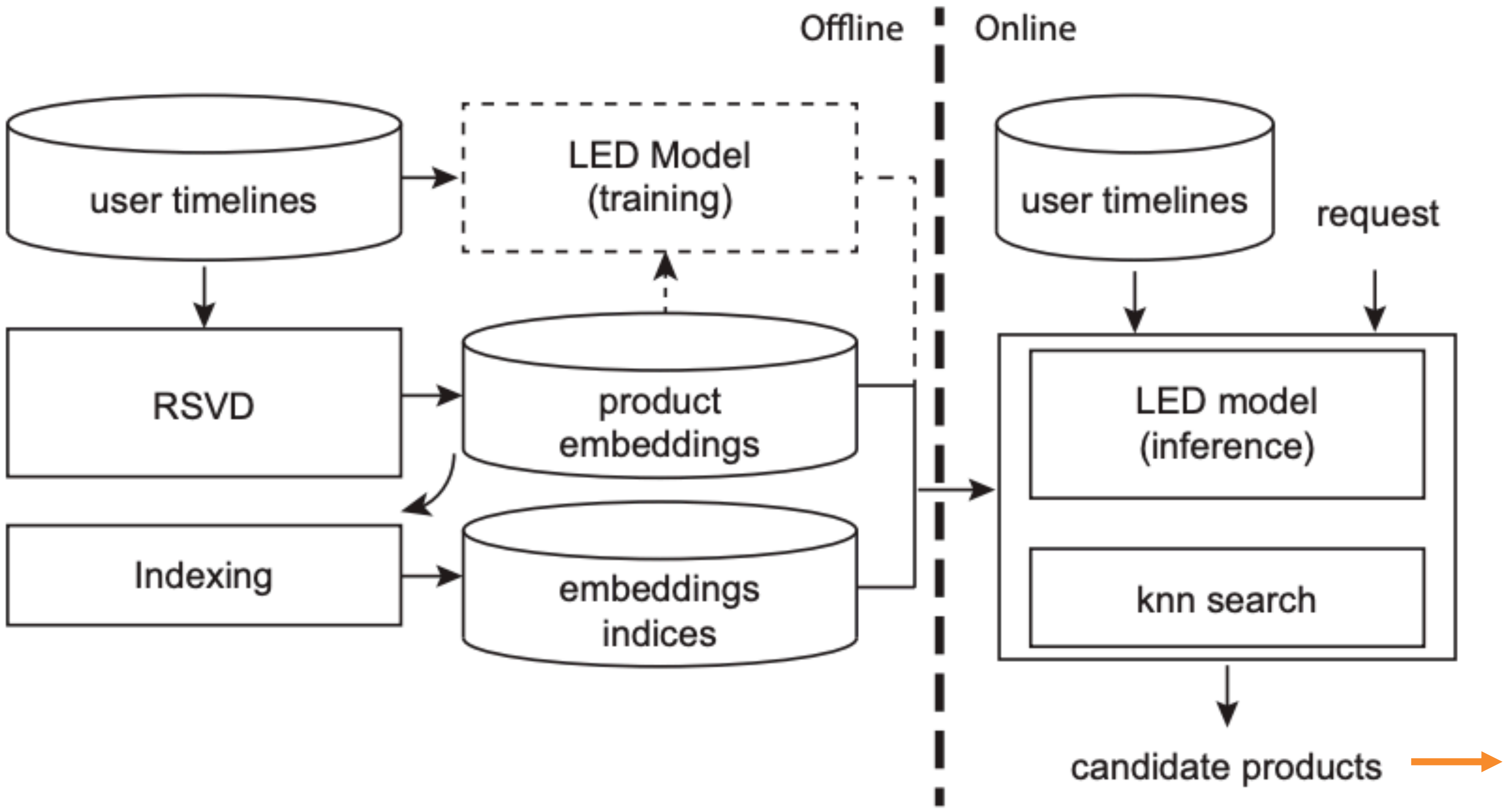
A/B testing results

Next steps

System architecture



System architecture



Ranking algorithm trained on user feedback

Real-time performance

Table 4: Real-time computing performance of our system with the LED model

Max Queries Per Second (QPS) per instance	3200
Latency @ 50th pct Latency @ 99th pct	$500\mu s$ 2ms
Latency of user embedding computation @ 50th pct Latency of user embedding computation @ 99th pct	$30\mu s$ $65\mu s$
Latency of KNN search @ 50th pct Latency of KNN search @ 99th pct	$160\mu s$ $450\mu s$
Instances used in production	200
Recommendations served per day	4B

A/B testing in the real-world

Why?

- To demonstrate the capability of our approach
- To validate positive results with real users

Algorithms

- GBO (global best-of = most popular products)
- CBO (cluster best-of = k-means + most popular products per cluster)
- LED

A/B setup

2 months, worldwide

2 billions displays, thousands of merchants

Populations:

- A: GBO
- B: GBO + CBO
- C: GBO + CBO + LED

All feed the same ranking algorithm trained on user feedback

LED brings significant uplift in business metrics

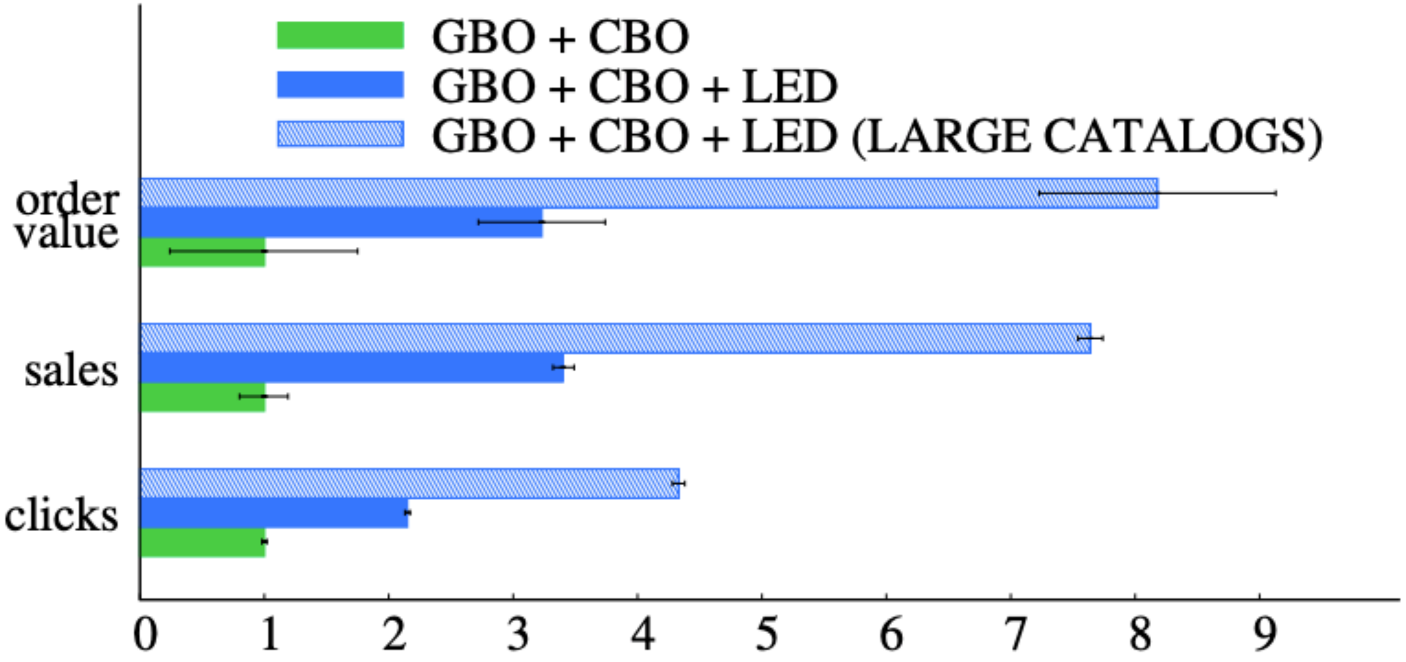


Figure 5: A/B test results: uplift of GBO + CBO and GBO + CBO + LED versus GBO. The uplift of GBO + CBO is scaled to 1. Error bars represent confidence intervals at 95%.

The ranking algorithm (i.e. user feedback) promotes LED

Table 5: Share of each algorithm in the displayed products after ranking. The ranking model predicts clicks and sales independently from the product origin.

A/B test population	GBO	CBO	LED
A	100%	0%	0%
B	61%	39%	0%
C	22%	11%	68%

LED shows less popular products (good for diversity)

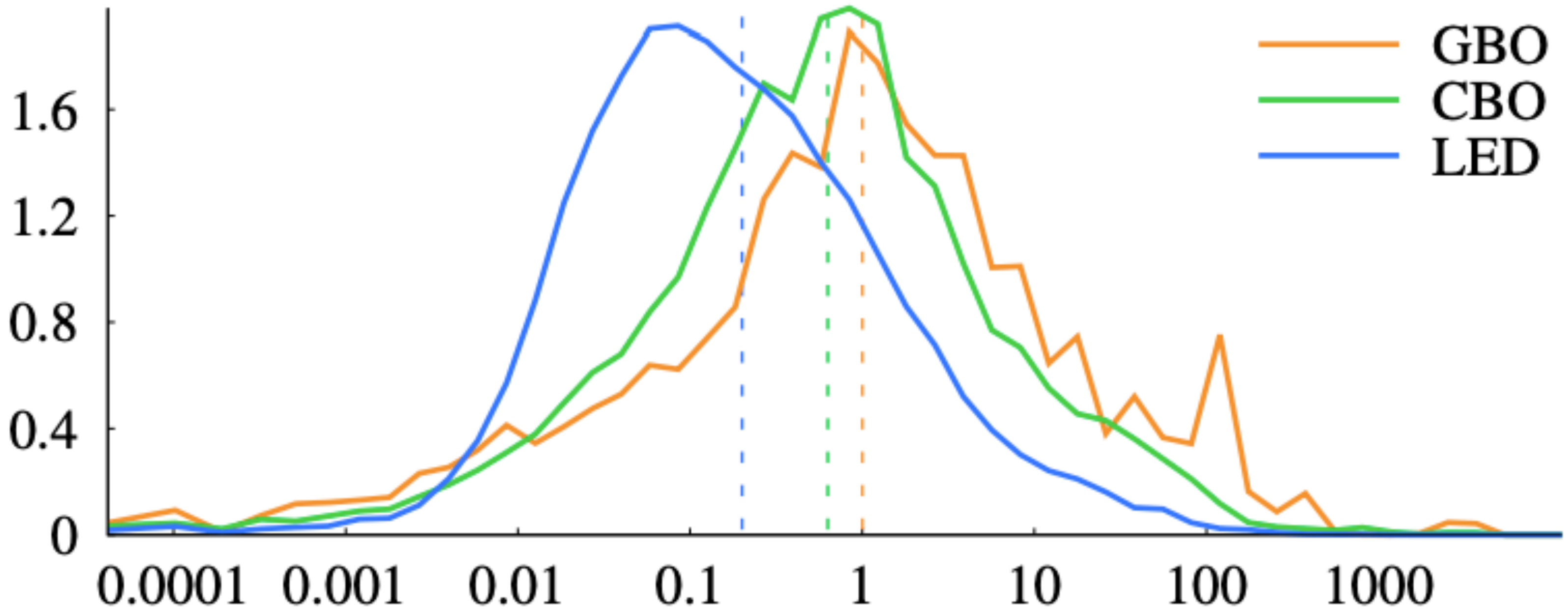
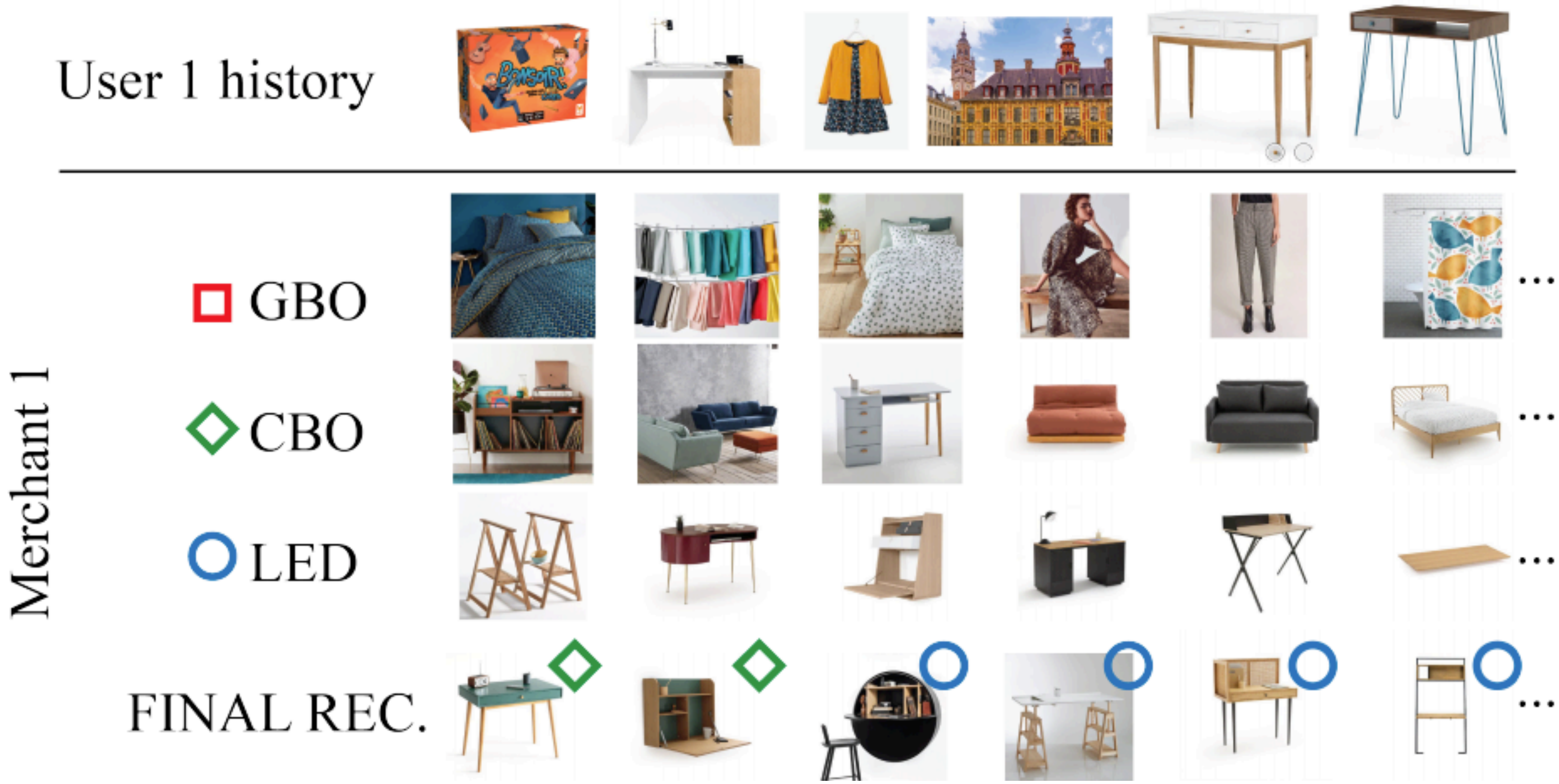


Figure 7: Distribution of product popularity per algorithm. x-axis: number of views per month on a log scale normalized to 1 for GBO. Average value in dotted line. Despite showing less popular products, LED generates more clicks and sales.

Qualitative results



A/B testing: conclusions

LED scales to billions of users, millions of items and ms latency

LED outperforms a fairly strong industrial baseline by a wide margin

Users (through the ranking algorithm) promote the LED algorithm

Next steps

1. Adding side-information
2. Diversity
3. Explainability & fairness

Conclusion: problem statement

How to build *fast* and *scalable* machine learning algorithms in a context of representation learning?

Our contributions

1. An efficient model (LED, for Lightweight Encoder-Decoder) reaching state-of-the-art performance with significant advantages of scale

2. A detailed architecture covering both offline training and real-time serving

3. Extensive experimentation demonstrating the efficiency of the system on real traffic over two months

References (1/2)

[1] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational Autoencoders for Collaborative Filtering, WWW'18.

[2] Harald Steck. Embarrassingly Shallow Autoencoders for Sparse Data, WWW'19

[3] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for TopN Recommender Systems, ICDM'11

[4] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets, ICDM'08.

[5] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative Metric Learning, WWW'17

[6] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems, KDD'18

References (2/2)

[7] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation, KDD’18

[8] Alexandros Karatzoglou and Balázs Hidasi. Deep Learning for Recommender Systems, RecSys’17

[9] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep Content-based Music Recommendation, NIPS’13

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality, NIPS’13

[11] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback, AUAI’09

[12] Aleksandar Botev, Bowen Zheng, and David Barber. Complementary Sum Sampling for Likelihood Approximation in Large Scale Classification, PMLR’17

[13] S. Gershman and Noah D. Goodman. Amortized Inference in Probabilistic Reasoning. Cognitive Science, 2014