TwERC: High Performance Ensembled Candidate Generation for Ads Recommendation at Twitter

Vanessa Cai, Pradeep Prabakar, Manuel Serrano Rebuelta, Lucas Rosen, Federico Monti, Katarzyna Janocha, Tomo Lazovich, Jeetu Raj, Yedendra Shrinivasan, Hao Li, Thomas Markovich

August 2, 2023

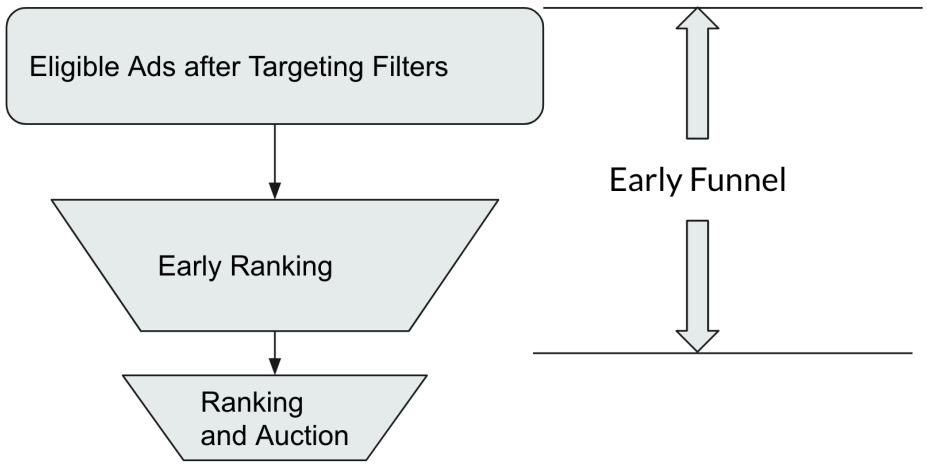
Twitter

Problem Statement

- Twitter is a microblogging site
- The logged in home page is a stream of micro-articles, called tweets
- Most users consume, rather than produce tweets
- Good user experience depends finding relevant and delightful tweets

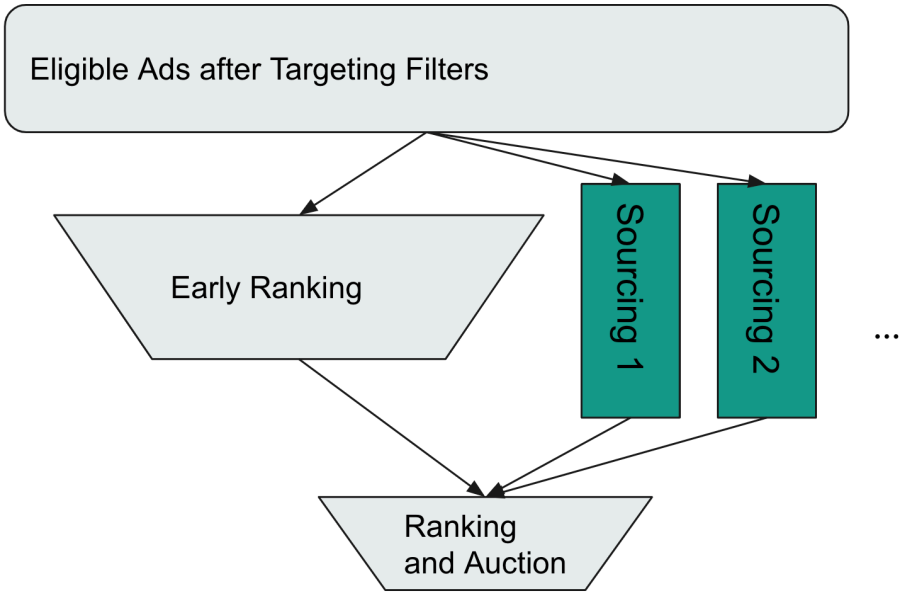
System Schematic

- First we apply hard and soft constraints to the active advertisements
- Then we pass them to a light ranker
- The topK ads are then sent to our heavy ranking system
- The heavy ranker outputs the rankscore, which is our best estimate of the overall value or utility of showing a particular ad



Candidate Sourcing

- Investigations found that the ad serving funnel had significant headroom
- We had already exceeded the compute envelope of the light ranker
- We explored a tail replacement strategy, where we replaced the bottom M% of the topK ads with methods that are complementary and efficient.
- In this work we present two such strategies – Rankscore Candidate Generation and Graph Candidate Sourcing

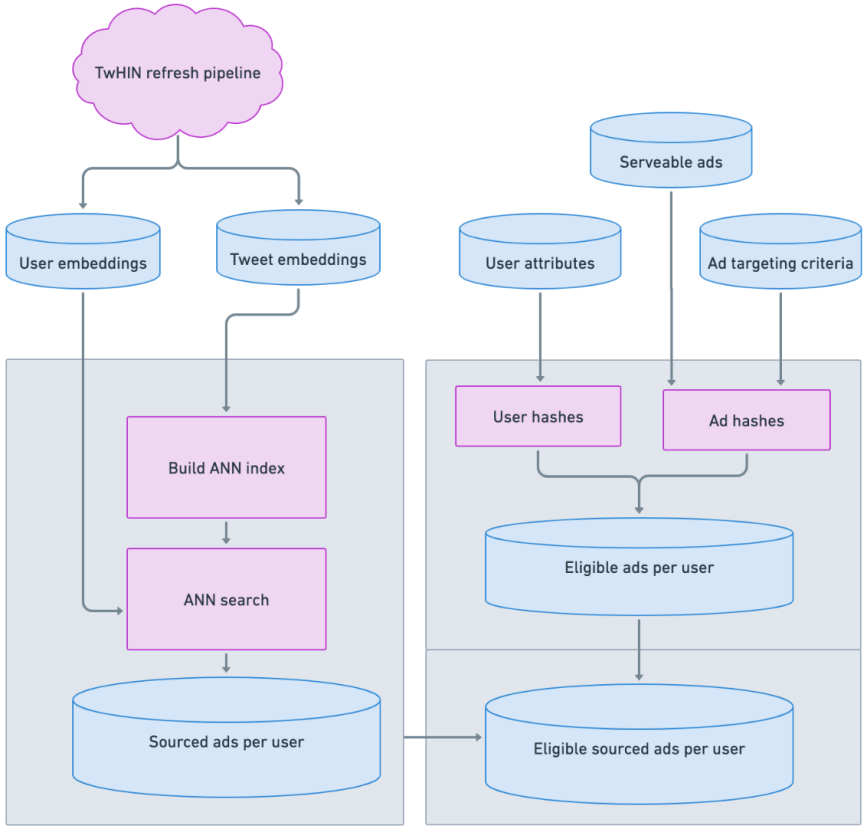


Rankscore Candidate Generation

- The unconstrained ad serving system scores all eligible ad candidates at the request level with a low sample rate
- We collect $rankscore(u, a, t)$ at time t from UAS with a 3-week lookback window
- This allows us to generate a fully counterfactual simulator
- We can generate user-ad quality scores as $q(u, a) = \frac{\sum rankscore(u, a, t) \cdot e^{t-t_0}}{\sum e^{t-t_0}}$
- For every UAS covered user, we generate the topK ads with the highest $q(u, a)$.

Graph Candidate Sourcing

- We start with a heterogeneous graph composed of user->ad engagements and embed it
- We update the user and item embeddings through a warm starting process
- We construct an ANN index with item embeddings
- We query the ANN index for all users, and scribe the topK active ads to a key-value store every 6 hours



Offline Experiments

Objective	RSCS	GBCS	TDGBCS
Obj 1	11.1%	1.1 %	-1.0%
Obj 2	2.3%	5.4%	7.9%
Obj 3	4.7%	2.3 %	10.2%
Obj 4	20.1%	4.5 %	7.2%
Obj 5	15.9%	2.0%	9.9%

Table 1: Offline experiment results from our counterfactual simulator. Here, RSCS is the rank score candidate sourcing strategy, GBCS is the graph based candidate sourcing strategy, and TDGBCS is the time dependent graph based candidate sourcing strategy. TDGBCS is generated by

$$\vec{u}_{i,t}^{user} = \frac{\sum_{j \in \mathcal{N}_i} e^{\lambda\left(t_j - t_0\right)} \vec{v}_j^{tweet}}{\sum_{j \in \mathcal{N}_i} e^{\lambda(t - t_0)}}$$

Online Experiments

Method	Revenue	Utility	Ads Value
RSSC	1.38%	4.71 %	1.05 %
GBCS	4.08%	6.42 %	1.26 %

Table 2: Results from the rankscore candidate sourcing online experiment. We observed statistically significant gains across reported metrics with adjusted p-values. We additionally analyzed this experiment in the context of advertiser inequality, and found a T1PS reduction of 1.2% which was a significant effect. $T1PS = \frac{\sum_{S_i \geq S_{(0.99|S|)}} S_i}{\sum_i S_i}$

Embedding Induction

Method	With	Without
Random ordering	6.5%	10.11%
Inducted graph embeddings	34.38%	42.07%
Graph embeddings	43.53%	-
Light Ranker	49.52%	55.61%

Table 3: Offline results on inducted embeddings for users that with and without a graph embedding (i.e. users that appear in the ad engagement graph). Inducted embeddings are done by:

$$x_i = \frac{1}{|N^+_{S,T}(i)|} \sum_{j \in N^+_{S,T}(i)} x_j \quad \forall i \in U_M.$$

Conclusions

- Unconstrained Ad Serving provides a counterfactual simulator
- Rankscore candidate sourcing provides an efficient lift in all product metrics
- Graph based candidate sourcing reduces inequality in our advertising ecosystem while improving revenue metrics

Questions?