RankTower: A Synergistic Framework for Enhancing Two-Tower Pre-Ranking Model

YaChen Yan
yachen.yan@creditkarma.com
Credit Karma
San Francisco, California, USA

Liubo Li
liubo.li@creditkarma.com
Credit Karma
San Francisco, California, USA

ABSTRACT

In large-scale ranking systems, cascading architectures have been widely adopted to achieve a balance between efficiency and effectiveness. The pre-ranking module selects candidates for the subsequent ranking module, while maintaining efficiency and accuracy under online latency constraints. In this paper, we propose a novel neural network architecture called RankTower, which is designed to efficiently capture user-item interactions while following the user-item decoupling paradigm to ensure online inference efficiency. The proposed approach employs a hybrid training objective that learns from samples obtained from the full stage of the cascade ranking system, optimizing different objectives for varying sample spaces. This strategy enhances the pre-ranking model's ranking capability and alignment with the existing cascade ranking system. Experimental results conducted on public datasets demonstrate that RankTower significantly outperforms state-of-the-art pre-ranking models.

CCS CONCEPTS

• **Computing methodologies**; • **Machine learning**; • **Machine learning approaches**; • **Neural networks**;

KEYWORDS

Recommender Systems, Pre-Ranking, Learning to Rank, Differentiable Sorting

ACM Reference Format:
YaChen Yan and Liubo Li. 2023. RankTower: A Synergistic Framework for Enhancing Two-Tower Pre-Ranking Model. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In industrial information services, such as recommender systems, search engines, and advertisement systems, the cascading architecture ranking system has been widely used to achieve a balance between efficiency and effectiveness. A typical cascade ranking system, as illustrated in Figure 1, consists of multiple sequential stages, including recall, pre-ranking, ranking, and re-ranking stages. Pre-ranking is commonly regarded as a lightweight ranking module characterized by a simpler network architecture and a reduced set of features. Compared to ranking models, pre-ranking models are required to score a larger number of candidate items for each user and demonstrate higher inference efficiency. Given the emphasis on efficiency, pre-ranking typically employs a straightforward vector-product-based model.

Figure 1: The Architecture of Cascade Ranking System

We propose a novel pre-ranking framework called RankTower to address these challenges. The primary contributions are as follows:

- We introduce the RankTower architecture, which comprises three key components: Multi-Head Gated Network, Gated Cross-Attention Network, and Maximum Similarity Layer. This architecture efficiently captures user-item interactions while following the user-item decoupling paradigm to ensure online inference efficiency.
- We employ a full-stage sampling strategy by drawing the training samples from different stages of the cascade ranking system. Tightly coupled with this sampling approach, we strategically integrate a hybrid loss function that combines distillation and learning-to-rank losses. This synergistic approach facilitates comprehensive learning of the ordering dynamics underlying user interactions while aligning with the inherent patterns of the cascade ranking system.
- Experiments on public datasets demonstrate that RankTower significantly outperforms state-of-the-art pre-ranking models in terms of prediction accuracy and inference efficiency.

2 MODEL ARCHITECTURE

The RankTower architecture, as shown in Figure 2, introduces three main modules: Multi-Head Gated Network for computing diversified user and item representations, Gated Cross-Attention Network for modeling bi-directional user-item interactions, and Maximum Similarity Layer for efficiently capturing user-item interactions to compute the final prediction.

RankTower follows the user-item decoupling paradigm, enabling efficient online serving by pre-computing and caching user and item embeddings. During online serving, only the gated cross-attention

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

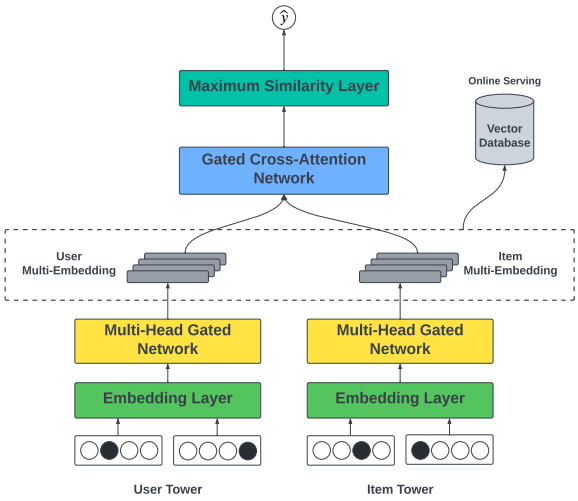

Figure 2: The Architecture of RankTower

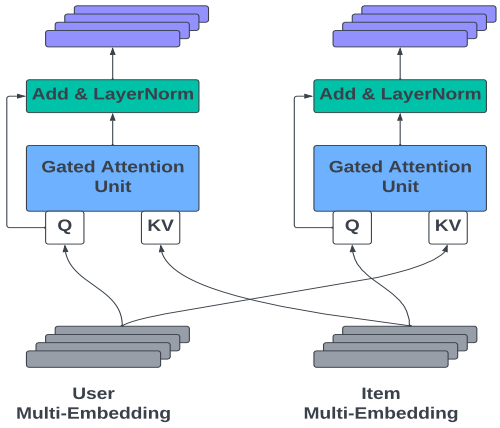

Figure 3: The Architecture of Gated Cross-Attention Network

layers require forward propagation, while other operations remain parameter-free, optimizing computational efficiency.

2.1 Preliminary

The dataset for building the pre-ranking model consists of instances (x_u, x_i, y, p), where x_u and x_i are the user feature and item feature respectively, X_U and X_I are the user and item input embeddings obtained by concatenating respective feature embedding vectors, $y \in \{0, 1\}$ indicates the user-item binary feedback label, p is the logged ranking model prediction that for knowledge distillation. z and $\hat{y}$ denote the pre-ranking model's logit and prediction.

2.2 Multi-Head Gated Network

The Multi-Head Gated Network is an enhanced MLP augmented with a gating mechanism for extracting diverse user and item representations. The MLP output is multiplied by an instance-aware gating vector, modeled by a two-layer MLP. The input embedding does not receive gradients from the gating network during training for stability. For example, given an user input embedding X_U, the h-th sub-space e_u^h of the user multi-embedding is:

$$\begin{aligned} e_u^h &= MLP_u(X_U)^h \circ \sigma(gMLP_u(X_U))^h \\ &\in \mathbb{R}^{B\times k}, \qquad h = 1, \cdots, H_u \end{aligned} \tag{1}$$

where $\circ$ denotes the Hadamard product, σ denotes the activation function of the gating network: Sigmoid(x), MLP_u denotes the MLP layer for modeling the user input embedding, $gMLP_u$ denotes the gating MLP for facilitating selective attention, B is the batch size and k is the embedding size of each sub-space.

Similarly, for item input embedding X_I, the h-th sub-space e_i^h of the item multi-embedding is:

$$\begin{aligned} e_i^h &= MLP_i(X_I)^h \circ \sigma(gMLP_i(X_I))^h \\ &\in \mathbb{R}^{B\times k}, \qquad h = 1, \cdots, H_i \end{aligned} \tag{2}$$

In the offline processing stage, we will periodically batch inference and store all the user/item's embeddings e_u^h and e_i^h into the vector database for online serving usage.

2.3 Gated Cross-Attention Network

The Gated Cross-Attention Network employs the cross-attention mechanism to effectively model the interaction between user embedding and item embedding. It utilizes the Gated Attention Unit as the main building block, along with residual connections and layer normalization for training stability.

2.3.1 Cross Attention Mechanism. The Bi-Directional Gated Cross-Attention Network interchangeably utilizes user and item embedding as queries and keys-values for bi-directional attention. Specifically, with the user multi-embedding $E_u = \text{Concat}(\text{e}_\text{u}^1, ..., e_u^{H_u})$ and item multi-embedding $E_i = \text{Concat}(\text{e}_\text{i}^1, ..., e_i^{H_i})$, the cross-attention compute the user attended embedding $\mathcal{E}_u$ and item attended embedding $\mathcal{E}_i$ as follows:

$$\mathcal{E}_u = \text{LN}(E_u + \text{GAU}(Q = E_u, K = E_i, V = E_i)) \in \mathbb{R}^{B\times H_u\times k} \tag{3}$$

$$\mathcal{E}_i = \text{LN}(E_i + \text{GAU}(Q = E_i, K = E_u, V = E_u)) \in \mathbb{R}^{B\times H_i\times k} \tag{4}$$

The cross-attention mechanism with two parallel branches is designed to simultaneously attend to both user preferences and item characteristics. This bidirectional approach allows the model to capture user-item interactions more accurately. The overall structure of the Gated Cross-Attention Network is illustrated in Figure 3.

2.3.2 Gated Attention Unit. The Gated Attention Unit introduces a gating mechanism to facilitate selective attention for better learning the dependency between user embedding and item embedding. Specifically, the Gated Attention Unit effectively enables an attentive gating mechanism as follows:

$$\begin{aligned} Q = \phi(X_Q W_Q), K = \phi(X_K W_K) \\ V = \phi(X_V W_V), U = \sigma(X_Q W_U) \end{aligned} \tag{5}$$

where X_Q, X_K, X_V are the query, key, and value input, ϕ is the non-linear activation function for projection layer, σ is the sigmoid function for computing gating value. With the learned projection Q, K, V, and the gating value U, we compute the attention weights, followed by gating and a post-attention projection.

$$O = (U \odot AV)W_o \tag{6}$$

$$A = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{7}$$

where $A \in \mathbb{R}^{H_u \times H_i}$ contains user to item attention weights. This example assumes that we use user embedding as the query, and item embedding as key and value.

2.4 Maximum Similarity Layer

The Maximum Similarity Layer computes the final probability prediction based on the user and item attended embeddings. Specifically, each user sub-space computes the maximum cosine similarity with all item sub-spaces, and the scalar outputs are summed across user sub-spaces:

$$s = (\sum_{p=1}^{H_u} \underset{q \in \{1,\cdots,H_i\}}{\text{Max}} COSINE(\mathcal{E}_u^p, \mathcal{E}_i^q))/\tau \tag{8}$$

where p and q are the sub-space indexes of user-attended embedding and item-attended embedding, respectively, and τ is the learnable temperature scalar for re-scaling the cosine similarity. Note that the Maximum Similarity Layer does not have any parameters which is suitable for online serving.

3 PRE-RANKING MODEL OPTIMIZATION

The pre-ranking models trained exclusively on impression samples, same as ranking models, suffer from sample selection bias. The pre-ranking model, which operates on the outputs of recall models, aims to identify the most relevant candidates set for the ranking model. Consequently, aligning the item distribution between the training and serving phases is essential to mitigate this sample selection bias and improve model effectiveness.

As illustrated in Figure 4, we implemented full-stage sampling to draw training data from impression samples, candidate samples, and random samples to mitigate sample selection bias. Moreover, we strategically applied various distillation and learning-to-rank losses to different sample scopes to effectively learn the ordering of user behaviors and the sequencing of the sample stages.

3.1 Full-Stage Sampling

The RankTower model is trained using user-level listwise samples containing multiple positive items and multiple objectives. The training samples for each user are sourced from various stages of the cascade ranking system, as shown in Figure 1. Detailed definitions and relationships among these components are provided below:

3.1.1 Impression Samples. The items output by the ranking model and viewed by the user consist of both positive and negative samples. Positive samples are items that have received various types of positive user feedback, while negative samples are items that have been exposed to the user without receiving user feedback.

3.1.2 Candidate Samples. The item candidates in the ranking or pre-ranking stages that are not viewed by the user are categorized based on their progression through the cascade ranking pipeline. Ranking candidates, which have advanced to the ranking stage, are generally considered as hard negative samples due to their higher relevance and quality compared to the pre-ranking candidates. Pre-ranking candidates are regarded as relatively easy negative samples because they were filtered out before reaching the ranking stage.

3.1.3 Random Samples. Items that are randomly sampled from the item corpus to serve as negative samples. These random samples are considered the easiest negative samples but are included to further enhance the generalization capability of the pre-ranking model. The incorporation of random samples ensures that the model remains effective and adaptable when encountering previously unseen items during the serving phase, thereby improving its robustness and ability to handle diverse item distributions.

3.2 Label Aggregation

Our framework incorporates two types of labels: hard labels and soft labels. Hard labels represent various types of positive user feedback on impression samples, while soft labels are predictions made by the ranking models, used knowledge distillation. Both categories of labels require an aggregation function to consolidate the different user behaviors into a single scalar value for the pre-ranking model's learning.

3.2.1 Hard Labels. The aggregation of hard labels is highly dependent on the specific business problem, requiring that labels be aggregated according to their orders of importance.

For instance, in online advertising, eCPM can be utilized based on the pricing model of the platform. In an e-commerce context, one might establish a relative preference order based on the depth of user feedback, such as *Purchase* > *Add to Cart* > *Click*. For scenarios like feed ranking or video recommendations, user feedback signals can be aggregated using a weighted sum approach. Additionally, we incorporate a general impression label applicable across business scenarios, for learning the pattern of the cascade ranking system. The label assigned a value of 1 for impression samples and 0 otherwise.

The user feedback labels help the pre-ranking model in learning the revenue or engagement level associated with different user behaviors. The exposure label facilitates the pre-ranking model's ability to learn and replicate the ranking patterns in the downstream cascade ranking system.

3.2.2 Soft Labels. For soft labels, we use the ranking objective function as aggregation function. This approach ensures that the soft labels are seamlessly integrated into the training process, maintaining the consistency between the pre-ranking model and the ranking model.

3.3 Hybrid Loss Functions

The pre-ranking model focuses on achieving both consistency and ranking accuracy through the following techniques:

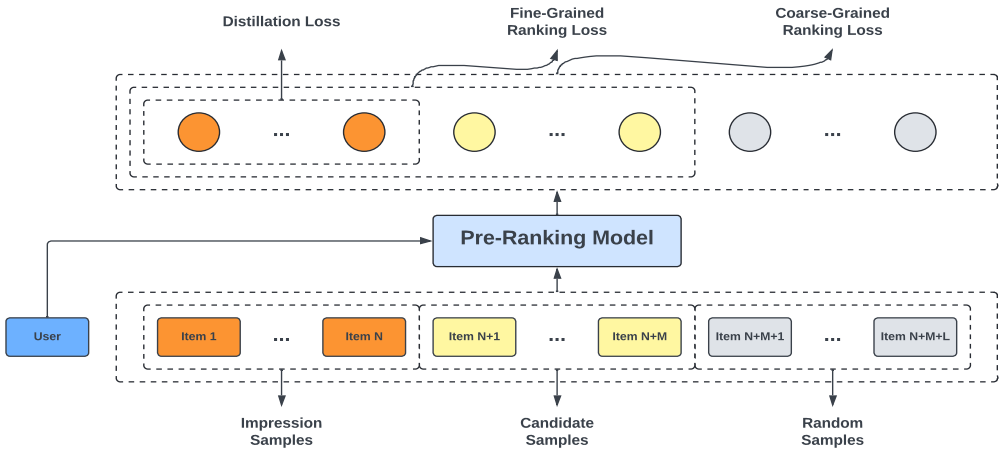

Figure 4: The Synergistic Framework for Learning User Behavior Ordering and Full-Stage Sample Ordering

- Knowledge Distillation: The ranking model's predictions are used as soft labels to transfer knowledge from the ranking model (teacher) to the pre-ranking model (student).
- Ranking Capability: Fine-grained and coarse-grained ranking losses are applied to improve ranking performance and retrieval capability, respectively.
- Diverse Training Samples: The model is trained on samples from different stages and varying easy/hard sample levels to achieve robust generalization and optimize hierarchical objectives.

Our synergistic framework is designed to learn both the hierarchy of user behaviors and the pattern of the cascade ranking system. For instance, in the context of online advertising, the model is expected to understand the following order of importance: converted items > clicked items > exposed items > candidate items and randomly sampled items.

3.3.1 Distillation Loss. As the main goal for the pre-ranking model is to output a high-quality item set for the ranking model, hence we used a listwise loss for distilling the knowledge from the ranking model as follows:

$$\mathcal{L}_{\text{Distillation}}(z, p) = -\sum_{i \in \mathcal{D}_I} p_i \log \frac{\exp(z_i)}{\sum_{j \in \mathcal{D}_I} \exp(z_j)} \tag{9}$$

where p is the prediction of the ranking model (soft label), z is the logit of the pre-ranking model, $\mathcal{D}_I$ is the impression samples set. Note the distillation process from the ranking model to the pre-ranking model is conducted exclusively on impression samples. As the ranking model is trained solely on these impression samples, its ability to generalize to candidate samples and random samples is inherently limited.

3.3.2 Fine-Grained Ranking Loss. The fine-grained ranking loss is applied to both impression and candidate samples, which directly correspond to the sample scope used in serving. We employ the **SoftSort**, a differentiable sorting loss, to learn user behavior and the patterns of the cascade ranking system. This loss function aims to precisely rank items according to the varying degrees of positive feedback they receive and effectively differentiate positives from impression samples and negatives from candidate samples.

Consider the **SoftSort** operator defined by metric function $\mathbf{d} = |\cdot|^p$ and temperature parameter τ for sorting n-dimensional real vectors $s \in \mathbb{R}^n$:

$$\textbf{SoftSort}_\tau^d(s) = \textbf{softmax}\left(\frac{-\mathbf{d}(sort(s)\mathbf{1}^T, \mathbf{1}s^T)}{\tau}\right) \tag{10}$$

The output of SoftSort operator is a permutation matrix of dimension n. The softmax operator is applied row-wise, thereby relaxing the permutation matrices into a set of unimodal row-stochastic matrices. In simple words: *the r-th row of the SoftSort operator is the* `softmax` *of the negative distances to the r-th largest element* [5]. We then employ the softmax cross entropy between the permutation matrices of label y and the permutation matrices of logit z. The **SoftSort** loss function is hereby defined as:

$$\mathcal{L}_{\text{Sorting}}(z, y) = -\mathbf{tr}\Big(\mathbf{J}_n\big(\textbf{SoftSort}_\tau^d(y) \circ \log \textbf{SoftSort}_\tau^d(z)\big)\Big) \tag{11}$$

where $\mathbf{J}_n$ is a $n \times n$ matrix of ones, $\mathbf{y} = (y_i)_{i \in \mathcal{D}_I \cup \mathcal{D}_C}$ is the hard label and $\mathbf{z} = (z_i)_{i \in \mathcal{D}_I \cup \mathcal{D}_C}$ is the logit of the pre-ranking model. We use the $\mathbf{tr}$ to compute the element sum of the matrix $\textbf{SoftSort}_\tau^d(y) \circ \log\big(\textbf{SoftSort}_\tau^d(z)\big)$.

3.3.3 Coarse-Grained Ranking Loss. The coarse-grained ranking loss, applied to all samples (impression, candidate, and random), aims to separate positive and negative samples while supporting ranking among positives by distinguishing varying degrees of relevance. We propose the Adaptive Margin Rankmax (AM-Rankmax) loss, an extension of the Rankmax loss [2] that introduces an adaptive margin based on the pair's nature and label distance, thereby

extending the Rankmax loss to address ranking problems with ordered or continuous positive labels.

Consider the Rankmax loss for ranking problems with binary labels only:

$$\mathcal{L}_{Rankmax}(z, y) = \sum_{j:y_j>0} \log \sum_{i=1}^{n} \left(z_i - z_j + 1\right)_+ \tag{12}$$

The Rankmax loss is reminiscent of pairwise losses. To extend the Rankmax loss to more general ranking problems involving multi-level positive labels, we introduce the adaptive margin with following enhancements:

- The loss is applied only when $y_i < y_j$, which is more suitable for multi-level positive label scenario.
- The margin adjusts based on whether y_i is positive or negative, to further enhancing the differentiality between positive and negative samples.
- The margin scales with the label distances between samples, reflecting varying degree of positive samples.

The adaptive margin function is:

$$m(i, j) = \alpha \cdot \mathbb{I}(y_i = 0) + \delta(y_i, y_j) \tag{13}$$

where α is a constant for adding additional margin between negative and positive items, $\mathbb{I}$ is the indicator function. The metric function δ can take various forms, for example $\delta(y_i, y_j) = 1$ or $\delta(y_i, y_j) = \beta|y_i - y_j|^p$. The adaptive margin Rankmax loss is then given by:

$$L_{\text{AM-Rankmax}}(z, y) = \sum_{j:y_j>0} \log \sum_{i:y_i<y_j} \left(z_i - z_j + m(i, j)\right)_+ \tag{14}$$

where $\mathbf{y} = (y_i)_{i\in\mathcal{D}_I\cup\mathcal{D}_C\cup\mathcal{D}_R}$ is the hard label from all the samples and $\mathbf{z} = (z_i)_{i\in\mathcal{D}_I\cup\mathcal{D}_C\cup\mathcal{D}_R}$ is the logit of the pre-ranking model.

The AM-Rankmax loss function can effectively adapt to scenarios with multiple positive labels of varying levels. This enhancement allows the model to handle different degrees of positive feedback, thereby improving its ability to generalize and accurately rank items in complex settings.

3.3.4 The Hybrid Ranking Loss. We design a hybrid ranking loss that integrates both distillation and ranking objectives. The hybrid ranking loss is the weighted sum of three losses:

$$\begin{aligned}\mathcal{L}_{\text{Hybrid}}(z, y) = \lambda_1 \mathcal{L}_{\text{Distillation}}(z, p) \\ + \lambda_2 \mathcal{L}_{\text{Sorting}}(z, y) \\ + \lambda_3 \mathcal{L}_{\text{AM-Rankmax}}(z, y)\end{aligned} \tag{15}$$

where λ_1, λ_2 and λ_3 are weights for each sub-objective. Balancing distillation and ranking losses is crucial for the pre-ranking model to inherit the ranking model's capabilities while generalizing to broader sample spaces. Weighting fine-grained and coarse-grained ranking losses ensures a balance between precise ranking and overall retrieval robustness.

4 EXPERIMENTS

We conduct experiments on three large-scale public datasets from online advertising, e-commerce, and short video recommendation domains to evaluate the effectiveness of RankTower. The experiments provide a comprehensive description of the evaluation metrics, and comparisons with state-of-the-art pre-ranking models. We aim to answer the following questions through our experiments:

- **Q1**: How does our proposed RankTower perform for pre-ranking task? Is it effective and efficient under extremely high-dimensional and sparse data settings?
- **Q2**: How do different settings on dataset sampling and training losses influence the performance of RankTower?

4.1 Experiment Setup

4.1.1 Datasets. We evaluate our model using real-world datasets: **Alimama**[1], **Taobao**[2], and **KuaiRand**[3]. For each dataset, we keep users with at least 100 impressions and 20 instances of positive feedback. The data is split into 70% for training, 10% for validation, and 20% for testing. As all labels in the datasets are binary, we aggregate them by summing the labels to form the hard label.

4.1.2 Evaluation Metrics. We consider Recall@K and NDCG@K for evaluating the performance of the models, and we set k to 100 for all experiment metrics.

Recall@K is the fraction of relevant retrieved within the top K recommendations. It's mainly used for measuring ranking system's capability on retrieving relevant items.

NDCG@K measures the quality of the ranking by considering both the relevance and the position of items within the top K recommendations. Items with higher relevance ranked at higher position contribute more to the metric.

4.1.3 Competing Models. We compare RankTower with the following pre-ranking models: LR [4], Two-Tower [1], DAT [8], COLD [7], IntTower [3] and ARF[6].

4.2 Model Performance Comparison (Q1)

Table 1: Performance Comparison of Different Algorithms on Alimama , Taobao and KuaiRand Dataset.

	Alimama		Taobao		KuaiRand	
Model	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K
LR	0.4802	0.3237	0.4792	0.2685	0.6713	0.5027
Two-Tower	0.5123	0.3428	0.5019	0.2921	0.6902	0.5258
DAT	0.5161	0.3472	0.5089	0.3013	0.6955	0.5312
COLD	0.5210	0.3518	0.5123	0.3070	0.7011	0.5349
IntTower	0.5215	0.3519	0.5101	0.3051	0.6960	0.5309
ARF	0.5318	0.3655	0.5215	0.3117	0.7096	0.5497
RankTower	**0.5462**	**0.3794**	**0.5301**	**0.3223**	**0.7182**	**0.5551**

The overall performance of different model architectures is listed in Table 1. We have the following observations for model effectiveness:

- LR exhibits the lowest performance compared to the other neural network-based models.
- Two-Tower brings the most significant relative improvement in performance, highlighting the importance of learning deep feature interactions.

[1] https://tianchi.aliyun.com/dataset/408
[2] https://tianchi.aliyun.com/dataset/649
[3] https://kuairand.com/

- COLD achieves strong performance among the competing models, indicating the significance of learning user-item feature interactions.
- ARF outperform other models without utilizing listwise ranking losses, highlighting the importance of using listwise ranking losses.
- RankTower achieves the best prediction performance, attributed to its effective modeling of bi-directional user-item feature interactions and the design of full-stage sampling and hybrid loss functions.

4.3 Model Study (Q2)

To gain deeper insights into the proposed model, we conduct experiments on the KuaiRand dataset and compare model performance on different settings, including: 1) the effect of full-stage data sampling; 2) the effect of listwise ranking losses; and 3) the effect of distillation from the ranking model.

4.3.1 Effect of Full-Stage Sampling. We conduct an ablation study to evaluate the impact of each sampling component on the model's performance. As shown in Table 2, the full-stage sampling strategy achieves the best overall performance. Training the pre-ranking model solely with impression samples hinders its ability to generalize to unexposed items, negatively affecting retrieval performance. We also observe that candidate samples are more important than random samples, as they significantly enhance the model's ability to discriminate between relevant and non-relevant items.

Table 2: Experiment Results for Different Sampling Strategies.

	Recall@K	NDCG@K
Full-Stage Sampling	0.7182	0.5551
w/o random samples	0.7125	0.5437
w/o candidate samples	0.7040	0.5401
w/o candidate & random samples	0.6981	0.5323

4.3.2 Effect of Listwise Ranking Losses. To better understand the properties of the proposed hybrid loss, we compare it with several widely used ranking losses in the industry. The experiment results, as shown in Table 3, indicate that the hybrid loss consistently outperforms other alternatives, surpassing both its individual components: the Sorting loss and the AM-Rankmax loss. Moreover, our proposed AM-Rankmax demonstrates superior performance compared to the original Rankmax loss and the Softmax loss.

Table 3: Experiment Results for Different Ranking Losses.

	Recall@K	NDCG@K
Hybrid Loss	0.7182	0.5551
Sorting	0.7128	0.5516
AM-Rankmax	0.7132	0.5507
Rankmax	0.7105	0.5492
Softmax	0.7109	0.5498
ApproxNDCG	0.7006	0.5436
RankNet	0.7072	0.5452

4.3.3 Effect of Distillation from Ranking Model. We conduct an ablation study on the distillation component and further compare Softmax loss with other alternatives.

The Table 4 demonstrate the efficacy of transferring knowledge distillation. Among various loss function experimented for distillation, the Softmax loss outperforms the other alternative losses. The Softmax loss, being a listwise ranking loss, proved more adept at distilling the ranking model's capabilities compared to the weighted logloss, which essentially is a pointwise approach and exhibited suboptimal performance in learning the relative ranking distribution. In contrast, the pairwise logloss, focusing solely on pairwise ordering of ranking model's predictions without considering the relative proximity of predictions, exhibited overfitting to the ranking model's outputs.

Table 4: Experiment Results for Different Distillation Losses.

	Recall@K	NDCG@K
Distillation (Softmax)	0.7182	0.5551
Distillation (Weighted Logloss)	0.7130	0.5519
Distillation (Pairwise Logloss)	0.7071	0.5432
No Distillation	0.7108	0.5495

5 CONCLUSION

This paper introduces the RankTower model, designed to enhance the performance of the two-tower model by effectively capturing bi-directional latent interactions between user and item. To ensure consistency with existing casecade ranking system, a hybrid loss function and full-stage sampling approach are integrated into the model's optimization framework. Comprehensive experiments demonstrate that RankTower significantly outperforms state-of-the-art pre-ranking models. In future work, we aim to study how to effectively and jointly optimize the cascade ranking system in an end-to-end fashion.

REFERENCES

[1] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[2] Weiwei Kong, Walid Krichene, Nicolas Mayoraz, Steffen Rendle, and Li Zhang. 2020. Rankmax: An adaptive projection alternative to the softmax function. *Advances in Neural Information Processing Systems* 33 (2020), 633–643.

[3] Xiangyang Li, Bo Chen, HuiFeng Guo, Jingjie Li, Chenxu Zhu, Xiang Long, Sujian Li, Yichao Wang, Wei Guo, Longxia Mao, et al. 2022. IntTower: the Next Generation of Two-Tower Model for Pre-Ranking System. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3292–3301.

[4] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1222–1230.

[5] Sebastian Prillo and Julian Eisenschlos. 2020. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*. PMLR, 7793–7802.

[6] Yunli Wang, Zhiqiang Wang, Jian Yang, Shiyang Wen, Dongying Kong, Han Li, and Kun Gai. 2023. Adaptive Neural Ranking Framework: Toward Maximized Business Goal for Cascade Ranking Systems. *arXiv preprint arXiv:2310.10462* (2023).

[7] Zhe Wang, Liqin Zhao, Biye Jiang, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2020. Cold: Towards the next generation of pre-ranking system. *arXiv preprint arXiv:2007.16122* (2020).

[8] Yantao Yu, Weipeng Wang, Zhoutian Feng, and Daiyue Xue. 2021. A Dual Augmented Two-tower Model for Online Large-scale Recommendation. (2021).