SERank: Optimize Sequencewise Learning to Rank Using Squeeze-and-Excitation Network

Ruixin Wang
Zhihu Search
wangruixin@zhihu.com

Kuan Fang
Zhihu Search
fangkuan@zhihu.com

Rikang Zhou
Zhihu Search
zhourikang@zhihu.com

Zhan Shen
Zhihu Search
shenzhan@zhihu.com

Liwen Fan*
levyfan@163.com

ABSTRACT

Learning-to-rank (LTR) is a set of supervised machine learning algorithms that aim at generating optimal ranking order over a list of items. A lot of ranking models have been studied during the past decades. And most of them treat each query document pair independently during training and inference. Recently, there are a few methods have been proposed which focused on mining information across ranking candidates list for further improvements, such as learning *multivariant* scoring function or learning contextual embedding. However, these methods usually greatly increase computational cost during online inference, especially when with large candidates size in real-world web search systems. What's more, there are few studies that focus on novel design of model structure for leveraging information across ranking candidates. In this work, we propose an effective and efficient method named as SERank which is a Sequencewise Ranking model by using Squeeze-and-Excitation network to take advantage of cross-document information. Moreover, we examine our proposed methods on several public benchmark datasets, as well as click logs collected from a commercial Question Answering search engine, Zhihu. In addition, we also conduct online A/B testing at Zhihu search engine to further verify the proposed approach. Results on both offline datasets and online A/B testing demonstrate that our method contributes to a significant improvement.

CCS CONCEPTS

• **Information systems** → **Learning to rank**.

KEYWORDS

deep neural network, learning to rank, information retrieval, squeeze-and-excitation network

1 INTRODUCTION

In the past decades, a plenty of learning to rank (LTR) algorithms have been studied and applied in search engine systems, where the task of these methods is to provide a score for each document in a list for a given query, so that the documents ranked higher in the list are expected to have higher relevance. The majority of ranking methods take each document's feature as input and learn a scoring function by optimizing loss functions which could be categorized into *pointwise* [8], *pairwise* [6] and *listwise* [31]. In the last few years, benefiting from the powerful nonlinear representation ability of deep learning methods [9] as well as the huge amount of web data, deep ranking models have been widely proposed and deployed in many real-world scenarios [11].

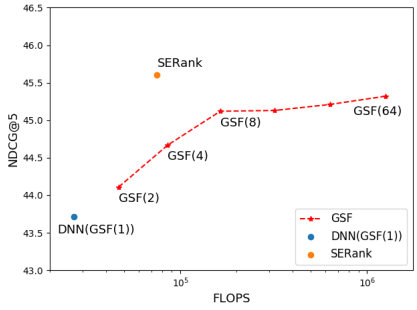


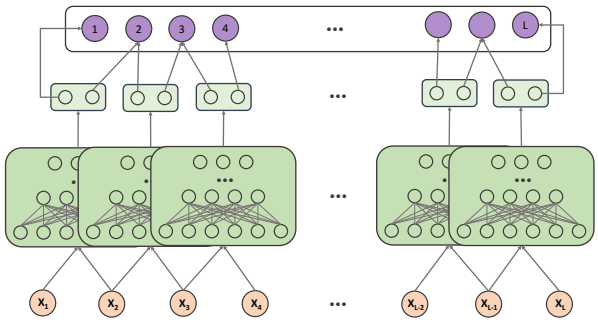
Figure 1: FLOPS vs NDCG@5 between different models on Web30K dataset. We use GSF(.) to represent the Groupwise Ranking model with different group size, and DNN(GSF(1)) model is the standard feed forward neural network with three fully-connected layers. The SERank model outperforms all of the GSF models on metric NDCG@5 with a speed of 16.7x faster than the GSF(64) model.

Recently, leveraging information across ranking documents becomes an emerging topic in the LTR domain. A number of works have proven that cross-document information could be mined to enhance final ranking performance [1] [2] [3]. Qingyao et al. [1] defines the *multivariant* scoring functions which named as GSF (Groupwise Scoring Function) by feeding concatenated features among a group into the DNN model, so that information across documents could be automatically learned. However, the GSF ranking with large group size increases model complexity and results in an huge expansion of computation cost, which makes it unappealing for real-world online services with sensitive responding time, while small group size often results in an insignificant gain of ranking quality.

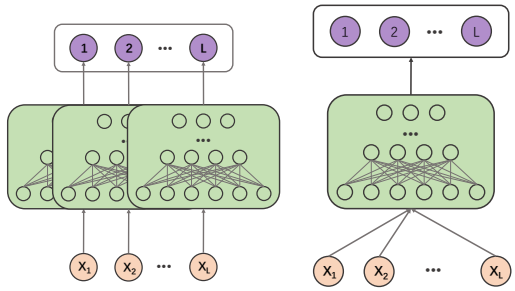
In this paper, to tackle the problem of utilizing cross-document information efficiently, we define a new Sequencewise ranking model named as SERank[1] which jointly scores and sorts a sequence of ranking candidates at once. As shown in Figure 2c, the proposed SERank takes a sequence of documents as input and scores them jointly, rather than predicts each document individually. Furthermore, feature importance, which is crucial for LTR settings, may

*The work was done when the author was with Zhihu Search

[1]Our source code will be released soon.



(a) GSF(2), groupwise model with group size = 2



(b) basic DNN model, equivalent (c) Sequencewise model to GSF(1)

Figure 2: Score mechanisms of Groupwise Ranking (a), basic DNN model (b) and Sequencewise Ranking (c). The Groupwise model accepts input features grouped by documents and output a group of relevance scores, then it aggregates each item's score among different groups to get the final score. The basic DNN model accepts each document's input feature and outputs single score corresponding to it. The proposed Sequencewise model takes a sequence of documents as input, and jointly outputs their final scores.

vary when the feature distribution of ranking candidates varies. Therefore, Squeeze-and-Excitation [12] is introduced to learn the feature importance information dynamically from a sequence of ranking candidates in each query. Figure 1 summarizes the efficiency and effectiveness of SERank with other models. Our SERank model surpasses all the baseline models on the NDCG metric [15]. On the efficiency aspect, the SERank runs 16.7x faster than GSF(64) model. Finally, different from the architecture proposed in [1], our proposed method does not require an initial ranking order over the candidates. The SERank model could accept any arbitrary permutations of input document sequence and jointly output score for each document.

In summary, our main contributions are listed as follows:

- We define a Sequencewise ranking model called SERank, which scores all ranking candidates in one episode and does not require an initial ranking order over the candidate documents.
- We introduce Squeeze-and-Excitation Network into the LTR settings for mining feature importance information across ranking candidates and propose the concrete implementation of the SERank model.
- Results on different benchmark datasets and online A/B testing illustrate our designed model obtains better ranking quality and requires little additional computations cost.

The rest of this paper is organized as follows. In Section 2, we review related works that are relevant to our proposed model. After that, we formally define the research problem and explain our proposed SERank model in Section 3 and Section 4. We will present experimental explorations on offline benchmark datasets as well as online A/B testing in Section 5. Finally, we discuss empirical results and conclude this work in Section 6.

2 RELATED WORK

Overall speaking, there are four main subjects of research that are related to the work in this paper: they are studies on learning to rank, nerual ranking models, nerual re-ranking models and Squeeze-and-Excitation Network.

2.1 Learning to Rank

Learning to rank [23] refers to methods that provide an order for a list of ranking candidates via machine learning approaches. Most research in LTR fields could be categorized by two aspects: the structure of scoring function and the type of loss function. Scoring functions can be parameterized by Gradient Boosting Trees [18], Support Vector Machine [16] [17], and Neural Networks [6]. Airbnb successfully deployed a feed-forward neural network as a replacement of decision trees in their search system [11]. While loss functions can be generally categorized into three types as *pointwise* [8], *pairwise* [6] and *listwise* [31]. The *pointwise* loss function treats LTR as a classification problem where each relevance grade corresponds to one class label. *Pairwise* loss functions learn document preference between each document pair in a query, and different weight are assigned to each pair according to their relevance label and ranking positions. *Listwise* methods put together all documents in a query and optimize ranking metrics directly.

Recently, Ai et al.[1] proposed a multi-variant scoring function that scores query documents by a learned *Groupwise* function, which takes cross-document interatcions into account. While the name *listwise* is a type of loss function, and *Groupwise* means a type of scoring function here, our Sequencewise model focuses on the improvement on the model structure.

2.2 Neural Ranking Models

In learning to rank circle, LambdaMART has been the long-standing state-of-the-art model for past decades. However, with the tremendous growing amount of data on the web, building a more effective ranking model with millions or tens of millions of training data becomes one challenging problem.

The neural network based models have the capability of learning from large scale data and high flexibility to the type of input features. For instance, in the scenario of Airbnb search [11], a neural network with single hidden layer and 32 fully connected ReLU activations obtains comparable result against tree based ranking models, and a deep NN model with 2 hidden layers and 10x larger training data

gains significant improvement over GBDT. Moreover, the NN based model has the advantage of learning representations of sparse id features. In the scenario of Gmail Search [25], they achieve substantial improvement over the baseline model after adding sparse token id features of each query and document, while the tree based model often fails to encode this kind of feature type. Therefore, in the condition of large scale training data and various kinds of feature types, the neural network based models have been proven as superior models against the traditional LambdaMART model.

Despite the fact that the NN model has prove to surpass GBDT in the scenarios described above, there are a lot of efforts[19] [20] [22] focused on combining GBDT and NN models together for further enhancement. These methods are designed to distil advantages of NN and GBDT respectively and then makes the combined model more powerful than each separate model. However, in this paper we focus on designing new model structure on NN aspect, and this model structure could be easily incorporated into a GBDT-NN combined framework.

2.3 Neural Re-Ranking Models

Recently, different from the traditional global ranking model that scores the entire ranking candidates purely based on each candidate's own features, a few works[1][2][3][26][28] have established that the final ranking performance of the top results could be further refined by utilizing a re-ranking stage based on the ranking order of a global ranking model. Most of the re-ranking methods focus on designing cross-document interactions which makes model captures more information about the whole ranking list. For instance, Ai et al. [1] designed a *listwise* re-ranking algorithm by using RNN to extract additional context-aware features for top candidates of a ranked list . Bello et al. [3] treats the re-ranking model through a sequence-to-sequence model so that the ranking order of the top candidates are given by the generation order of the decoder in the sequence-to-sequence model.

However, these methods require a strong initial ranking order of the input candidates (for example, utilizing LambdaMART to select top tens of documents before re-ranking in [2]), which introduce additional computation overhead, while our proposed method is served as a global ranking method which scores the entire ranking candidates and could be combined with other arbitrary re-ranking methods.

2.4 Squeeze-and-Excitation Network

In the computer vision circle, the Squeeze-and-Excitation network [12] has been proposed and applied successfully, where the main intuition is to learn inter-dependencies between different feature channels for an image. This kind of mechanism contributed a winning of first prize in the ILSVRC 2017 [24] classification task.

There are a few variants of the original Squeeze-and-Excitation network [21][7]. For example, Li et al. propose SKNet [21] which collect local information with multi branches of different receptive field size, and merge different branches by the learned attention weights. With a little increase in model complexity, this method outperforms many state-of-art architectures.

Recently, Huang et al. [13] successfully incorporate the Squeeze-and-Excitation network in the recommendation ranking model.

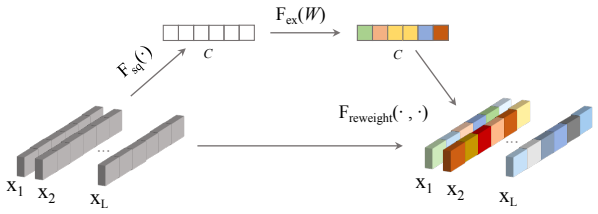


Figure 3: A Squeeze-and-Excitation block. $\{x_1, x_2, \ldots, x_L\}$ is the input documents for a given query. The squeeze operation $F_{sq}(.)$ gathers information across feature channels among documents and the excitation operation $F_{ex}(W)$ gain the feature importance. Finally, the raw inputs x_i are re-weighted by the output of $F_{ex}(W)$.

However, they use SENet to learn feature importance weights only by each recommendation item's feature embedding, while we focus on using SE block to design a Sequencewise ranking model.

3 PROBLEM FORMULATION

In the learning-to-rank settings, the training set could be represented as $\Psi = \{\boldsymbol{x}, \boldsymbol{y}\} \in X^n \times \mathbb{R}^n$, where $\boldsymbol{x}$ is a vector of n items $\boldsymbol{x_i}$, $1 \leq i \leq n$, $\boldsymbol{y}$ is also a vector of n real values that represents the relevance grade of each $\boldsymbol{x_i}$, and X is the space of all items. In this paper, we denote $\boldsymbol{x_i}$ as a vector of features represents for a query document pair. The main goal of a ranking model is to learn a scoring function, which maps the input feature vector of $\boldsymbol{x_i}$ to a real output value R_i, and the score function should minimize the empirical loss over the training set, which could be formally stated via a supervised machine learning framework

$$\mathcal{L}(f) = \frac{1}{\Psi} \sum_{\boldsymbol{x}, \boldsymbol{y} \in \Psi} l\{\boldsymbol{y}, f(\boldsymbol{x})\}, \tag{1}$$

where $l(.)$ is a loss function over training examples, and $f(.)$ is usually an *univariate* function which accepts single item $\boldsymbol{x_i}$ as input.

Most research focuses on optimizing loss functions or model structures that still rank each document independently. Recently a few works contributed on how to leverage cross-document information to further improve the ranking performance. The ranking model predicts document relevance with additional information either by using *multivariate* [2] score function which receives aggregated items $\{\boldsymbol{x_i}, \boldsymbol{x_j} \ldots \boldsymbol{x_k}\}$ as inputs or learning additional context features for each item $\boldsymbol{x_i}$ [2]. Besides these approaches, we propose SERank to leverage cross-document information. Generally, our methods could be represented as a learning process as follows

$$\mathcal{L}(F) = \frac{1}{\Psi} \sum_{\boldsymbol{x}, \boldsymbol{y} \in \Psi} l\{\boldsymbol{y}, F(\boldsymbol{x}, g(\sum_{\boldsymbol{x_i} \in \Psi_q} \boldsymbol{x_i}))\}. \tag{2}$$

We define a function $g(\sum_{\boldsymbol{x_i} \in \Psi_q} \boldsymbol{x_i})$ to capture feature importance through aggregating feature info across documents in each query, where Ψ_q is the documents corresponding to the given query q. Then result from $g(\sum_{\boldsymbol{x_i} \in \Psi_q} \boldsymbol{x_i})$ and raw input $\boldsymbol{x}$ are combined together to the final score function $F(.)$. We will describe the details in the next sections.

4 SEQUENCEWISE DEEP RANKING MODEL

In this section, we describe how our proposed Sequencewise model works in the learning-to-rank regime. Our main intuition is to let model learn feature importance automatically by gathering feature information across ranking documents in each query. Then we incorporate learned feature importance information into the ranking model.

4.1 Sequencewise Input Layer

Before describing model structure, we first explain the details of the input layer. In the LTR settings, the documents are grouped by queries, and for each query, the data can be organized as a set of documents $\{\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, ..., \boldsymbol{x}_{(L)}\}$ corresponding to it. Each document $\boldsymbol{x}_i$ could be represented as a feature vector $\{\boldsymbol{x}_{(i)}^1, \boldsymbol{x}_{(i)}^2, ..., \boldsymbol{x}_{(i)}^c\}$, where $\boldsymbol{x}_{(i)}^c$ is the c-th feature channel of document $\boldsymbol{x}_{(i)}$. Therefore, the ranking model accepts documents $\mathcal{D} \in \mathbb{R}^{L*C}$ and output scores $\mathcal{S} \in \mathbb{R}^L$ for all documents per query. In this paper, we use $\boldsymbol{C}$ to denote the size of feature channels of each query-document pair, and $\boldsymbol{L}$ to denote number of ranking candidates for each query.

4.2 The SE (Squeeze-and-Excitation) Block

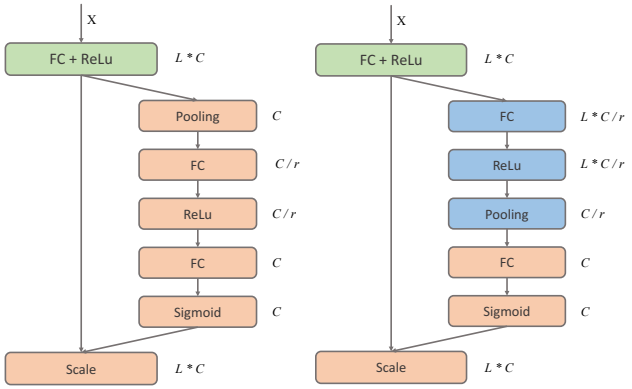
According to the motivation proposed in GSF[2], the relevance of each document depends on the distribution of the whole list. For instance, consider an ad-hoc document retrieval scenario where a user is searching for the name of an artist. If all the results returned by the query (e.g., âĂIJcalvin harrisâĂİ) are recent, the user may be interested in the latest news or tour information. If, on the other hand, most of the query results are older (e.g., âĂIJfrank sinatraâĂİ), it is more likely that the user seeks information on artist discography or biography. So in the ranking tasks, feature importance may vary when met with different candidate lists. Therefore, the main intuition of our method is to design a supplementary block for learning feature importance across ranking documents.

Inspired by SENet [12], we apply SE block in our proposed model for feature importance learning. As shown in Figure 3, the SE block computes feature weight through two main mechanisms, gather Sequencewise information by squeeze operation and gain feature importance by excitation operation.

4.2.1 Squeeze Operation. The squeeze operation is designed to compute each feature's statistics over different documents for a given query. To be specific, the squeeze operation process input $\mathbb{X} \in \mathbb{R}^{L*C}$ and then output collected feature statistics $\boldsymbol{U} \in \mathbb{R}^C$. The squeeze operation could be either max pooling or mean pooling over each feature among different documents.

4.2.2 Excitation Operation. After aggregating feature information over documents, the excitation operation aims to generate each feature's importance weight. Two fully-connected layers are used to learn the feature weights. In the first layer, we reduce the dimension with a shrinkage parameter r so that the output of first fully-connected layer is $\mathbb{R}^{\frac{C}{r}}$. Then we recover the dimension with the same r in the next fully-connected layer. Concretely, the final feature weight could be calculated as below

$$\boldsymbol{s} = \boldsymbol{F}_{ex}(\boldsymbol{U}) = \sigma 1(\boldsymbol{W}_2 \times \sigma 2(\boldsymbol{W}_1 \times \boldsymbol{U})), \tag{3}$$



(a) SE block structure **(b) SE-b block structure**

Figure 4: The original SE block structure (a) and the modified SE block structure (b)

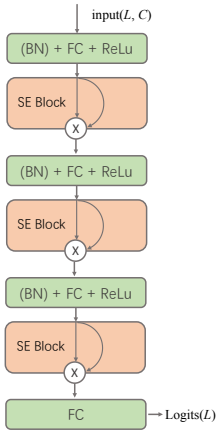


Figure 5: Overall SERank Model Structure. The BN is optional which is only used in Web30K and Web10K dataset.

where $\boldsymbol{s} \in \mathbb{R}^C$ is the learned feature weights, $\sigma 1$ and $\sigma 2$ are ReLu activation and $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are parameters of two fully-connected layers. After the excitation operation, the SE block use the $\boldsymbol{s}$ to re-weight the input layers, by element-wise multiplying the raw input $\mathbb{X} \in \mathbb{R}^{L*C}$ by the excitation output $\boldsymbol{s} \in \mathbb{R}^C$.

Finally, after the Squeeze and Excitation operations, the important feature channels are strengthened while uninformative feature channels are decreased.

4.3 Ranking with Squeeze-and-Excitation Block

Primarily, we use a multi-layer DNN model as the basic model structure for ranking, where all layers are fully-connected layer with different hidden units size. We incorporate SENet into our model by adding the SE block in each layer. The model structure is shown in Figure 5.

In this paper, different from the original SE block proposed in [12], we design a modified SE block version denoted as SE-b. As

shown in Figure 4b, SE-b first uses a fully-connected layer to reduce the dimension of each query-document pair, after that, the pooling is adopted for feature information gathering. The reason why we try this minor modification is to let the model gather global feature weight from a compressed input space, rather than from raw input of the last hidden layer directly. We conduct an experiment in the next section and the result shows that our modification leads to further improvement.

4.4 Loss Functions

Our proposed model could be trained through any arbitrary loss functions. To verify the generality of SERank, we train with both *pairwise* and *listwise* loss functions. Since in LTR the documents are grouped with queries, the loss functions listed here are described for query document pairs within a query.

The first one is the Pairwise Logistic Loss [6] which is one of the most classic loss functions in LTR field

$$\mathcal{L}(\boldsymbol{y}; \hat{\boldsymbol{y}}) = \sum_{i=1}^{n-1} \sum_{j=1, \hat{y}_j < \hat{y}_s}^{n} log(1 + exp^{-(\hat{y}_i - \hat{y}_j)}), \tag{4}$$

where the subscript i represents the i-th document in a query, y_i is the true label of document i and $\hat{y}_i$ is the predicted score. Note that the Pairwise Logistic Loss can be extended to a *listwise* loss function by multiplying λ-weight [30] on the document pairs. Therefore, we also examined Pairwise Logistic Loss with λ-weight in the experimental section.

The second loss function is the Softmax Cross-Entropy Loss [4], which is a *listwise* loss calculated as

$$\mathcal{L}(\boldsymbol{y}; \hat{\boldsymbol{y}}) = \sum_{i=1}^{n} \frac{y_i}{\sum_i y_i} log(\frac{exp(\hat{y}_i)}{\sum_i exp(\hat{y}_i)}). \tag{5}$$

These two loss functions are used in the experimentation section, and we leave analysis with other loss functions on our work to future study.

5 EXPERIMENTS

In this section, we first outline the datasets, baseline models and hyperparameters used in our experiments. We then compare our proposed model with baseline models on model effectiveness and complexity. Finally, we try to explain why our method works.

5.1 Datasets

The first dataset used in our experiments is MSLR-Web30K [29], which is a publicly available learning-to-rank dataset that contains more than 30000 queries. The relevance labels take 5 values from 0 (irrelevant) to 4 (perfectly relevant). We discard queries with no relevant documents. There are 136 dense features per query-document pair. The number of documents within one query is variable, which is on average 120. During training, we limit at most 200 documents per query, but when evaluate we use all. In this dataset, there are 5 folds containing the same data, and each fold randomly splits to train, validation and test set. Following [2] we use Fold1 in our experiments since results from other folds are similar. MSLR-Web10K [29] is another learning-to-rank dataset that is similar to MSLR-Web30K but contains fewer samples (10,000 queries). And we will also report the experimental results on this dataset.

The second dataset, Zhihu dataset, is a real-world learning-to-rank dataset created from Zhihu (a Chinese Question Answering community, www.zhihu.com) search log. We randomly sample user click logs from a week's traffic, and create a dataset with 2 million queries which is 60x larger than Web30K. For each query, there are on average 16 documents. The relevance label is obtained from the userâĂŹs click which is 1 when one document is clicked otherwise 0. Different from the Web30K dataset which mainly consists of dense features, the Zhihu dataset contains 129 dense features as well as sparse id features, like query and document title token ids. The dataset is randomly splited into three sets: train, validation and test set. The validation set and test set contain 5% of queries respectively. We report metrics in NDCG@1,5,10 [14] on test set for all experiments.

5.2 Models

We have compared our method with multiple existing learning-to-rank models including tree-based methods and DNN models. The tree-based model used in our experiments is implemented in lightGBM [18]. RankNet [6] and GSF [2] with different group size are used as DNN baseline methods.

For Web30K and Web10K dataset, the hyperparameters we used in LambdaMART (LightGBM) are consistent with previous work [5]. We train at most 1000 trees, then select the best number of trees by NDCG@5 on the validation set. For Zhihu dataset, we tune hyperparameters including learning rate, max number of leaves per tree and max number of trees to select the best model based on NDCG@5 of the validation set.

We implement RankNet and GSF models by TF Ranking [25]. RankNet is a multi-layer feed-forward fully-connected DNN model with Pairwise Logistic Loss (equation 4). For the Web30K and Web10K dataset, we implement GSFs with the same hyperparameters as previous work [2], which is a three layers (64,32,16) DNN model with batch normalization. For the Zhihu dataset, we use 7 dense layers (layer dims from 1024 to 16) with ReLU activation without batch normalization but we normalize the raw input features before feeding them into model. We transform sparse id features into fixed embedding and then apply average pooling on the sparse id emebdding features, then we concatenate dense input layer and embedding layer to obtain the final input layer. GSF models with various group sizes share the same model hyperparameters with RankNet except for the loss function which is softmax loss for Web30K dataset and λ-weighted Pairwise Logistic Loss for Zhihu dataset. For SERank we also use softmax loss on the Web30K dataset and λ-weighted Pairwise Logistic Loss for Zhihu dataset respectively.

For our proposed SERank model shown in Figure 4a, based on the multi-layer feed-forward fully-connected structure, we further add the SE block followed by every dense layers. The shrinkage rate is 2 by default. The batch size is set to 128 and Adagrad [10] optimizer is chosen with a learning rate of 0.5 for all DNN models. For the Web30K dataset, we train 30000 steps and select the best model on NDCG@5 of validation sets and then report predicted results on test set. And for the Zhihu dataset, we train 10 epoches for every model and evaluate on the test set using the last checkpoint.

Table 1: Comparison of test NDCG with baseline models on Web30K and Web10K dataset. For GSF and SERank models, we use softmax loss function to train the model.

Dataset	Model	NDCG@1	NDCG@5	NDCG@10
Web30K	LambdaMART (LightGBM, best reported [5])	**50.33**	**49.2**	**51.05**
	RankNet	40.74(±0.11)	42.1(±0.05)	44.73(±0.06)
	GSF(1) (best reported [2])	-	43.14	-
	GSF(1) (fine-tuned)	43.33(±0.13)	43.70(±0.07)	46.08(±0.05)
	GSF(64) (best reported [2])	44.21	44.46	46.77
	GSF(64) (fine-tuned)	45.01(±0.09)	45.32(±0.10)	47.67(±0.08)
	SERank	44.38(±0.12)	44.50(±0.07)	46.83(±0.06)
	SERank-b	**45.14**(±0.13)	**45.60**(±0.11)	**47.80**(±0.09)
Web10K	LambdaMART (LightGBM)	**46.20**(±0.13))	**46.23**(±0.09))	**48.33**(±0.11))
	RankNet	39.71(±0.09)	40.94(±0.07)	43.37(±0.05))
	GSF(1)	41.52(±0.12)	42.12(±0.11)	44.79(±0.09)
	GSF(64)	42.75(±0.14)	42.65(±0.12)	44.87(±0.13)
	SERank	40.39(±0.13)	41.94(±0.11)	44.46(±0.09)
	SERank-b	**43.09**(±0.14)	**43.35**(±0.12)	**45.77**(±0.10)

Table 2: Comparison of test NDCG with baseline models on Zhihu dataset. For GSF and SERank models, we use pairwise logistic loss with λ-weight to train the model.

Model	NDCG@1	NDCG@5	NDCG@10
LambdaMART (LightGBM)	51.16	60.89	66.63
GSF(1)	53.23	63.28	69.24
GSF(2)	53.15	63.21	69.10
GSF(64)	51.91	62.35	68.39
SERank-b	**53.41**	**63.40**	**69.33**

5.3 Comparison with Baseline Models

In Table 1, we compare our proposed method with other existing methods including tree-based models and DNN models on the Web30K and Web10K dataset. For GSF(1) and GSF(64), we cite the metrics reported in [2]. In addition, we further fine-tuned these two models and report our results which show 95% bootstrapped confidence intervals. For Web30K dataset, our proposed method (SERank-b) achieved the best result of DNN models which significantly outperform RankNet and GSF(1) by 3.5% and 1.9% on NDCG@5 (measured from 1 to 100) respectively, measured by paired t-test with p-value threshold of 0.05. And it also slightly surpass GSF(64) by 0.28%. Because of the advantage of dense features and rather small data volume, the tree-based model LambdaMART (LightGBM) performs best on Web30K. This phenomenon is also observed in the related works [2][26], where the evaluation score of their methods are lower than LambdaMART on Web30K dataset but performs opposite on the industrial dataset which is much larger than Web30K. Besides this, we can observe that our improved version SERank-b (with SE-b block) outperforms the original SERank (with SE block) by a large margin. In the following part, we adopt the improved SERank-b by default. On the Web10K dataset, the results show a similar tendency as Web30K which indicating the strong robustness of our proposed approach.

In Table 2, we compare the performance of SERank with other methods on the Zhihu dataset. Similar to the scenario in Airbnb[11] and Gmail Search[2], we also experiment SERank in the industrial dataset, which is 60 times larger than Web30K. This magnitude of dataset makes the DNN model more advantageous than the tree-based LambdaMART. As the result shows, our proposed method outperforms all other methods. In terms of the results of GSF, we try different group sizes for GSF and find that as group size increases the performance becomes even worse and these similar phenomenons are also reported in [27].

Table 3: Comparison of normalized GFLOPs and rank metrics between SERank and GSFs on Web30K dataset

Model	Δ NDCG@5	Δ FLOPs
GSF(1)	-	-
GSF(64)	3.68%	45.01 times
SERank-b	4.32%	1.75 times

5.4 Model Efficiency

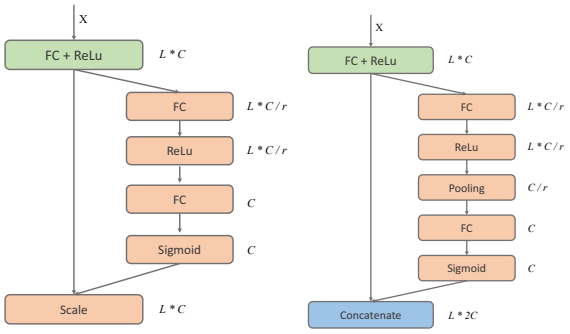
Approaches that try to take advantage of cross-document interactions usually lead to huge increasing on computation cost, which is sensitive in online serving. We use FLOPs as the computation cost metric to evaluate model efficiency between SERank and other baseline models. The result of FLOPs is related to the input shape and model complexity. Therefore, we compute FLOPs in a forward pass of a document sequence for one query in Web30K dataset, which has an input shape of [document_size, feature_size], where document size is 200 and feature_size is 136 here. In Table 3, the FLOPs and test set NDCG@5 on the Web30K in dataset. Compared with GSF(1), the GSF(64) and the SERank improve relative 3.68% and 4.32% on NDCG@5 respectively, but GSF(64) cost 45.01 times FLOPs while SERank is only 1.75 times. In other words, the increasing computational cost of SERank is little over GSF(1) compared to the GSF(64) model, while the performance improvement is more significant. Combined with Figure 1, we can draw the conclusion that our proposed model is more effective and efficient on leveraging the Sequencewise information.

Table 4: Comparison of rank metrics between scoring with full docs and scoring with remaining docs

Model	NDCG@1	NDCG@5	NDCG@10
SERank-b-base	44.85(±0.06)	47.83(±0.04)	51.53(±0.05)
SERank-b-remain-doc	45.01(±0.05)	47.84(±0.03)	51.48(±0.03)

Table 5: Effect of squeeze and excitation on Web30K dataset

Model	NDCG@1	NDCG@5	NDCG@10
SERank-b	**45.14**	**45.60**	**47.81**
SERank-b-W/O-Squeeze	42.84	44.02	46.45
SERank-b-W/O-Excitation	44.87	44.9	47.2



(a) SE block without squeeze **(b) SE block without excitation**

Figure 6: Ablation study on Squeeze-And-Excitation operation

5.5 Model Stability

For ranking models utilizing cross-document information, it is important to study their ranking stability, i.e, the remaining documents should be ranked as the same order if some documents are removed from the candidate set. To examine the SERank's stability, we randomly mask out 50% of documents for each query in Web30K's test set and predict scores of the remaining ones. The NDCG metric is denoted as SERank-b-remain-doc in Table 4. On the contrary, we compare NDCG metric of the remained documents with their scores under no-missing circumstance (denoted as SERank-b-base). As Table 4 shows, the NDCG is very close between these two conditions, which demonstrates that missing documents have little impact on the ranking order of other documents. Since this experiment has verified the robustness of our proposed work, we leave the stability test of GSF in future work.

5.6 Ablation Study

In the previous section, we draw the conclusion that our proposed SERank is both effective and efficient. And we believe that the SE block which consists of squeeze structure and excitation structure plays the key role. In this section, we conduct an ablation study on the Web30K dataset to gain a better understanding of their roles.

Table 6: Online clicked search ratios at position 1,3,5

Metric	Click@1	Click@3	Click@5
Increase	0.95%	0.4%	0.2%
P-value	0.001	0.002	0.004

5.6.1 Effect of Squeeze. Squeeze structure is a key component of the SE block. By pooling operation, the SE block can capture Sequencewise information to figure out the feature importance under current context. To verify our hypothesis, we experiment with a variant of the SE block which has no squeeze operation as Figure 6a shows. Unlike the standard version, because the pooling operation is removed, the excitation outputs of the variant version are independent for different documents thus one document can not gain information from the others. As shown in Table 5, the NDCG@5 of the SERank model without squeeze operation is significantly lower than the standard one, which proves the effectiveness of the squeeze operation.

5.6.2 Effect of Excitation. The excitation operation helps model learn the feature importance by applying the sigmoid function on the output of the dense layer and multiply with the original input of the SE block. By replacing the multiply operation with concatenate operation (concatenate the sigmoid out with the SE blockâĂŹs inputs) as shown in Figure 6b, we can study the importance of excitation structure. Although this variant model can also benefit from Sequencewise information, it loses the physical meaning that the sigmoid outputs of excitation are the signal of the feature importance. In Table 5 we compare the ranking metrics of the SERank-b-W/O-Excitation with the standard one. As we can see, the performance of the SERank-b-W/O-Excitation model deteriorates, which is good proof of the effectiveness of excitation operation.

5.7 Online A/B Testing

Besides evaluating SERank on the benchmark datasets, we further validate its performance by deploying it at the search engine of Zhihu, which is one of the largest Question Answering communities in China. We train SERank as well as the basic DNN model with nearly 20 million queries collected from one day's real search traffic. The baseline DNN model has three hidden layers with {128,64,32} hidden units with ReLU activation functions respectively, and neither dropout nor batch normalization were taken into training in both SERank and baseline DNN model. To eliminate position bias from click data, we adopt the method proposed by Zhao et al. [32], which de-bias click data by adding an independent position aware tower in the model.

We compared results of SERank with baseline model in term of the clicked search ratio at different positions, which is the percentage of clicked search sessions at top 1,3,5 among all search sessions. The higher of the clicked search ratio means the model performs better. To make online A/B testing reliable, we use p-value to test the significance of the two ranking models. Finally, to remove randomness between different traffic and ensure the results are comparable between traffic groups, we set two groups of study and two groups of control with equal amount of traffic.

As shown in Table 6, the SERank model significantly outperform baseline DNN model at clicked search ratios. The clicked search ratios at position 1,3,5 are enhanced by relatively 0.9%, 0.4% and 0.2% respectively. The p-value of all the metrics are smaller than 0.01 which indicates obvious significance of SERank. Therefore, our proposed SERank indeed improved ranking quality over the baseline DNN model.

6 CONCLUSIONS

In this paper, we propose a learning-to-rank approach denoted as SERank which aims at leveraging Sequencewise information to enhance the ranking metrics. We conduct a series of experiments comparing our proposed model with the existing methods on multiple datasets. The experimental results show that the SERank, which is not only effective but also efficient, outperforms the existing method in a statistically significant manner both on publicly available datasets and the large scale real-world dataset. Finally, the online A/B test shows that the SERank can significantly improve user experience in a large-scale industrial search engine.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 135–144.

[2] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 85–92.

[3] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019* (2018).

[4] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 75–78.

[5] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. 2019. Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1241–1244.

[6] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[7] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. 2019. GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond. *arXiv preprint arXiv:1904.11492* (2019).

[8] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*. 315–323.

[9] Balázs Csanád Csáji et al. 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* 24, 48 (2001), 7.

[10] Farideh Fazayeli. 2014. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. (2014).

[11] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C Turnbull, Brendan M Collins, et al. 2019. Applying deep learning to Airbnb search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1927–1935.

[12] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[13] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction. *arXiv preprint arXiv:1905.09433* (2019).

[14] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[15] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 243–250.

[16] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 217–226.

[17] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 781–789.

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3146–3154.

[19] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. 2019. DeepGBM: A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 384–394.

[20] Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining Decision Trees and Neural Networks for Learning-to-Rank in Personal Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2032–2040.

[21] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. 2019. Selective Kernel Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 510–519.

[22] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 689–698.

[23] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[24] E Park, W Liu, O Russakovsky, J Deng, L Fei-Fei, and A Berg. 2017. ILSVRC-2017. *URL http://www. image-net. org/challenges/LSVRC/2017* (2017).

[25] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2970–2978.

[26] Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. Self-Attentive Document Interaction Networks for Permutation Equivariant Ranking. *arXiv preprint arXiv:1910.09676* (2019).

[27] Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. Self-Attentive Document Interaction Networks for Permutation Equivariant Ranking. *arXiv preprint arXiv:1910.09676* (2019).

[28] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 3–11.

[29] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[30] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1313–1322.

[31] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*. ACM, 1192–1199.

[32] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.