A Domain Generalization Perspective on Listwise Context Modeling

Lin Zhu, Yihong Chen, Bowen He
Ctrip Travel Network Technology Co., Limited.
{zhulb, yihongchen, bwhe}@ctrip.com

Abstract

As one of the most popular techniques for solving the ranking problem in information retrieval, Learning-to-rank (LETOR) has received a lot of attention both in academia and industry due to its importance in a wide variety of data mining applications. However, most of existing LETOR approaches choose to learn a single global ranking function to handle all queries, and ignore the substantial differences that exist between queries. In this paper, we propose a domain generalization strategy to tackle this problem. We propose Query-Invariant Listwise Context Modeling (QILCM), a novel neural architecture which eliminates the detrimental influence of inter-query variability by learning *query-invariant* latent representations, such that the ranking system could generalize better to unseen queries. We evaluate our techniques on benchmark datasets, demonstrating that QILCM outperforms previous state-of-the-art approaches by a substantial margin.

Introduction

As an important learning paradigm for tackling the ranking problem in information retrieval, learning-to-rank (LETOR) has received a lot of attention both in academia and industry due to its importance in a wide variety of applications, such as document retrieval (Ai et al. 2018; Chapelle and Chang 2011), recommendation systems (Freno 2017; Volkovs and Zemel 2012), and E-commerce search (Zhuang, Ou, and Wang 2018; Karmaker Santu, Sondhi, and Zhai 2017).

The typical assumption behind LETOR methods is that for any given query, the relevance score of a candidate item is a parameterized *global* function of a set of features that describe the query, the item, and their interactions (Karmaker Santu, Sondhi, and Zhai 2017; Ai et al. 2018). The parameters of this function can be determined by fitting it to a training set that consists of query-item pairs and the associated relevance judgments. Once learned, the function is applied to handle any future query to rank the candidate items with respect the query. However, queries may vary significantly in the underlying users' intentions (Geng et al. 2008), in the distributions of relevant items (Ai et al. 2018), or in the features that are relevant to item ranking (Karmaker Santu, Sondhi, and Zhai 2017), and it is difficult to learn a single model that could encompass all these diversities. Moreover, given the substantial variations between queries, it is quite possible that a unseen test query have certain characteristics that are rarely encountered in the training set, leading to degradation in generalization ability of the ranking function (Karmaker Santu, Sondhi, and Zhai 2017).

So far, a number of approaches have been proposed to tackle the aforementioned problem. In particular, it has been demonstrated that the performance of a global ranking model can be greatly improved by incorporating the *local ranking context* information (Ai et al. 2018), which is generally extracted from the input data via feature engineering efforts or novel model architectures (Geng et al. 2008; Ai et al. 2018; Mottini and Acuna-Agost 2017; Zhuang, Ou, and Wang 2018). Although some promising results have been shown, these approaches still do not explicitly filter out the detrimental variations of queries, which may increase the risk of overfitting and hurt the performance.

At another extreme, we could train a ranking model for every query independently (Ai et al. 2018; Geng et al. 2008). While this strategy may avoid the negative influence of the heterogeneity of queries, it does not allow information to be shared and transfered between queries, which makes generalization to unseen queries even harder.

A better solution may lie in the middle of these two extremes: we still learn a common ranking model for all queries, and yet treat every query as a separate *domain*. All of the queries (domains) share the same learning task and the same input feature space, but have different distributions. In recognition of the variations between queries (domains), we would like the model to acquire knowledge from 'source' queries (domains) that are available as the training set, and adapt the learned knowledge to unknown 'target' queries (domains).

In this formulation, we notice that the LETOR problem is equivalent to a *domain generalization (DG)* problem (Blanchard, Lee, and Scott 2011), where the objective is precisely to infer a learning system that can take as input a set of training domains and will output a model that can achieve high accuracy for new domains. To achieve this goal, many existing DG methods propose to learn *domain-invariant* representations that are expected to remove the negative effects of distributional changes across domains (Muandet, Balduzzi, and Schölkopf 2013; Xie et al. 2017).

On the basis of the above considerations, in this paper we propose to solve the query heterogeneity problem via DG techniques. The main contributions of our work are summarized as follows:

- We introduce a DG formulation of the LETOR problem and discuss why a DG perspective is justified in this setting;
- We propose Query-Invariant Listwise Context Model (QILCM), a novel neural architecture for DG in this LETOR context;
- We perform comprehensive evaluations on three benchmark datasets, demonstrating that QILCM outperforms previous state-of-the-art approaches by a substantial margin.

Related Work

LETOR methods

According to how training losses are formulated, existing algorithms for LETOR can be broadly categorized into three categories: (1) The pointwise approaches (Li, Wu, and Burges 2008; Nallapati 2004), which reformulates LETOR as a classification or regression problem on single items;(2) The pairwise approaches (Freund et al. 2003; Burges et al. 2005), which formulate LETOR as a classification problem on the item pairs; (3) The listwise approaches (Taylor et al. 2008; Wu et al. 2010) which directly optimize the ranking metrics. As mentioned in the introduction section, queries could vary greatly in various aspects, and exhibit different 'standards' for ranking (Wu, Hsieh, and Sharpnack 2018). As a result, listwise and pairwise LETOR methods, which mainly focus on modeling the relative (instead of absolute) preferences between items for a query, generally perform better than pointwise approaches in practice (Karmaker Santu, Sondhi, and Zhai 2017; Cao et al. 2007; Chapelle and Chang 2011). Our proposed DG perspective for LETOR could be considered as a continuation of these methods, since it more explicitly enforces the model to filter out the 'global' differences between queries, and focus on the 'relative' differences between items for each given query. Meanwhile, most of existing listwise and pairwise LETOR methods still try to learn a global ranking function that assesses the relevance of each item individually. In contrast, listwise context modeling methods studied in this paper treat the candidate items as a whole, and can better model their mutual influences (Zhuang, Ou, and Wang 2018).

Neural Ranking Models

Deep neural network methods have been applied to numerous ranking applications, such as recommendation systems (Covington, Adams, and Sargin 2016), ad-hoc retrieval (Fan et al. 2018; Zamani et al. 2018), and context-aware ranking (Zamani et al. 2017). Similar to the traditional LETOR methods discussed above, these models also score each item independently and neglect the negative influence of distributional differences between queries. This limitation is clearly demonstrated in (Cohen et al. 2018), which shows that these models could overfit to the domains from which the training data is sampled, and generalize poorly to domains not encountered during training. To handle this problem, Cohen et al. also propose learning of domain invariant representations. However, they assume that the queries have already been categorized into a few broad domains and focus on avoidance of overfitting to these predetermined domains, while our method do not rely on such additional supervisory information, and instead focus directly on tackling the finer-grained variations between queries.

Context Aware Ranking

A significant amount of research has focused on leveraging contextual data to improve ranking. In particular, certain types of contextual information have been explored in depth, such as temporal dynamics of user behaviors (Xiang et al. 2010; Chen et al. 2018; Zhou et al. 2018) and the geological location data (Zamani et al. 2017; Manotumruksa, Macdonald, and Ounis 2018).

On the other hand, modeling of the local ranking context formed by candidate items is a less well studied problem, and state-of-the-art approaches are generally based on the Recurrent Neural Network (RNN) models (Mottini and Acuna-Agost 2017; Ai et al. 2018; Zhuang, Ou, and Wang 2018). For example, (Zhuang, Ou, and Wang 2018) reformulate the ranking task as a RNN-based sequence generation problem and use the beam search algorithm to generate the ranked item list, while Deep Choice Model (DCM) (Mottini and Acuna-Agost 2017) and Deep Listwise Context Model (DLCM) (Ai et al. 2018) adopt RNNs to generate a context encoding that summarizes the candidate items, which is then adopted to re-query the candidate items to generate the ranking scores. Although some promising results have been shown, a potential limitation of these methods is that RNNs are mainly designed for modeling sequential data, and yet the data encountered in ranking problems cannot always be organized as a natural and unique sequence. For example, the candidate items may be selected from a large repository using some initial retrieval methods (Covington, Adams, and Sargin 2016; Chapelle and Chang 2011), and are essentially orderless. For such problems, although one can still order the candidate items according to certain pre-specified rules, and in principle universal approximators such as RNNs should be robust to choices of such pre-ordering, previous studies nevertheless show that the input order enforces a implicit prior on how RNNs should encode the given data, and could have strong impact on the experimental performance (Vinyals, Bengio, and Kudlur 2016). Meanwhile, by adopting additional components such as positional encoding (Vaswani et al. 2017), models without recurrent structures can also incorporate useful order information to achieve good performance (Vaswani et al. 2017; Gehring et al. 2017; Zhou et al. 2018).

In addition, it is still necessary for these approaches to learn from the data to filter out the detrimental variations of queries, while preserving the necessary information for the ranking task. Better performance may be obtained by relieving such a learning burden using certain dedicated architectures that explicitly encode robustness to inter-query variations, such as the DG models reviewed later.

Domain Generalization (DG)

So far, a large number of formulations have been proposed to solve the DG problem, we refer the interested readers to (Li et al. 2018; Shankar et al. 2018) for up-to-date reviews of related techniques. Our method is directly inspired by the DG approaches which learn a high-level data representation that simultaneously minimizes domain dissimilarity and preserves the functional relationship with the learning target. To the best of our knowledge, this is the first time that such techniques are applied to model the local ranking context formed by candidate items.

Methodology

Problem Formulation

Let $\mathcal{Q}$ be the set of possible queries and $\mathcal{D}$ be the set of possible items. Assume we are given a set of queries $\mathcal{T} \subseteq \mathcal{Q}$ as the training data, where each query $q \in \mathcal{T}$ is provided with a set of n_q candidate items $\left\{d_{q,1}, d_{q,2}, \cdots, d_{q,n_q}\right\} \in \mathcal{D}$ to be ranked. Each pair of query-item $\left(q, d_{q,i}\right), 1 \leq i \leq n_q$ is described by the same set of categorical and numerical features, and assigned a score $y_{q,i}$ that quantifies the relevance of $d_{q,i}$ to q. The objective of LETOR is to infer a ranking model that can accurately assess the relevance of any given query-item pair.

Model Architecture

As discussed in the introduction, we propose to cast LETOR as a DG problem, and view every query as an individual domain. All of the domains/queries share the same learning task and the same input feature space, but have different distributions. To better account for such distributional differences, we parametrize the desired ranking model as a neural architecture that consists of five main components, as illustrated in Figure 1:

- An **item encoder** that transforms each query-item pair $(q, d_{q,i})$ into an encoding vector $\mathbf{h}_{q,i}$;
- An exchangeable **item pooling layer** that collapses the encoding vectors associated with each query q to a single query embedding vector $\mathbf{c}_q$, which is then combined with $\mathbf{h}_{q,i}$ to construct a refined item encoding vector $\widetilde{\mathbf{h}}_{q,i}$;
- A parameter-free **query normalization layer** that reduces the distributional differences between item encodings from different queries;
- A **ranking layer** that computes the ranking score using item representations learned from the previous layers;
- An **objective function** that simultaneously promotes the improvement of ranking accuracy and the semantic alignment of feature distributions among queries.

Item Encoder

The item encoder that we consider in this paper is similar to previous works (Mottini and Acuna-Agost 2017; Ai et al. 2018; Covington, Adams, and Sargin 2016). It accepts the features of query-item pairs as input. Given the sensitivity of neural networks to input scaling (LeCun et al. 2012; Covington, Adams, and Sargin 2016), numerical features are normalized to the interval $[0, 1]$, and categorical features are mapped to dense vectors via embeddings, which are learned jointly with all other model parameters through back-propagation.

Additionally, sometimes the input data may be supplemented with certain order information, such as the predictive results given by an initial LETOR algorithm (Ai et al. 2018). Different from previous works that model such information using RNNs, we simply incorporate it as an additional numerical feature. While more sophisticated schemes for encoding order information (Vaswani et al. 2017; Zhou et al. 2018; Gehring et al. 2017) are readily applicable, we found that such a simple option already performs well on the datasets that we tested.

For each pair of $(q, d_{q,i})$, all of the pre-processed features mentioned above are concatenated into an array $\mathbf{x}_{q,i}$. As noted in (Ai et al. 2018), direct usage of this feature representation may fail to fully leverage the expressive power of neural models. We thus follow (Ai et al. 2018), and pass $\mathbf{x}_{q,i}$ through 2 layers of fully connected exponential linear units (ELUs) (Clevert, Unterthiner, and Hochreiter 2015):

$$\begin{aligned} \mathbf{x}_{q,i}^{(1)} &= \mathrm{ELU}\left(\mathbf{W}^{(1)}\mathbf{x}_{q,i} + \mathbf{b}^{(1)}\right), \\ \mathbf{x}_{q,i}^{(2)} &= \mathrm{ELU}\left(\mathbf{W}^{(2)}\mathbf{x}_{q,i}^{(1)} + \mathbf{b}^{(2)}\right), \end{aligned} \tag{1}$$

the output of which is then concatenated with $\mathbf{x}_{q,i}$ to construct a new feature vector $\mathbf{h}_{q,i}$:

$$\mathbf{h}_{q,i} = \begin{bmatrix} \mathbf{x}_{q,i} \\ \mathbf{x}_{q,i}^{(2)} \end{bmatrix}. \tag{2}$$

Item Pooling Layer

In this work, different queries are treated as different domains, since we observe only a subset of the possible queries/domains during training, additional assumptions are needed to ensure that we can generalize to a new query/domain during testing. A useful technique in the DG literature to generalize to new domains is *domain embedding* (Shankar et al. 2018), which maps domains into the same semantic space. Such a technique captures the domain variations via continuous latent features, and thereby allows effective knowledge transfer between domains.

Along the same line, previous listwise context modeling works (Ai et al. 2018; Mottini and Acuna-Agost 2017) choose to construct ‘context encoding’ to capture contextual information using RNNs. Concretely, given the set of feature vectors $\{\mathbf{h}_{q,i}\}_{1\leq j\leq n_q}$ for query q, these works firstly feed the vectors sequentially into a RNN, and then create the context encoding $\mathbf{c}_q$ by using the final hidden state of the RNN. As $\mathbf{c}_q$ is then used to query the item vectors to generate the final ranking order, it plays a pivotal role in these models. However, the recurrent nature of RNN means that the creating process of $\mathbf{c}_q$ has no direct access to the candidate items except for the last one, and all relevant information has to be propagated by the RNN through the ordered items in an one-by-one manner, which imposes additional

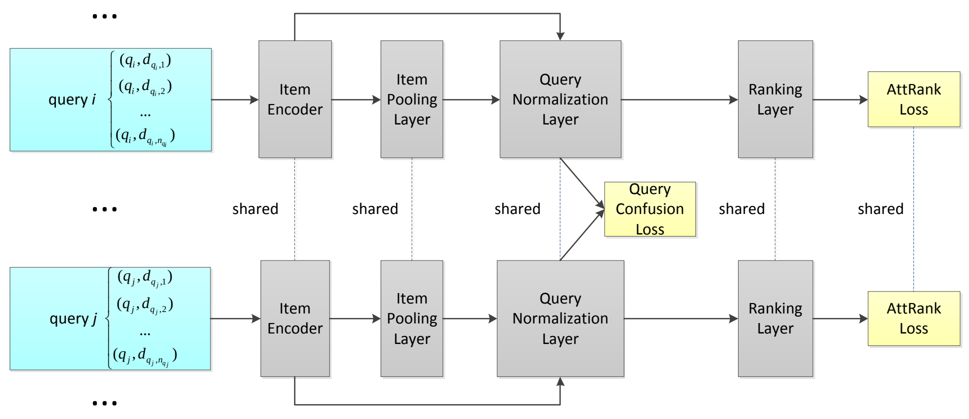

Figure 1: The overall architecture of Query-Invariant Listwise Context Model (QILCM).

memorization burden on the RNN and may incur unnecessary information loss.

Based on the above considerations, in the paper we adopt the self-attention mechanism (Lin et al. 2017; Vaswani et al. 2017) to create $\mathbf{c}_q$. Concretely, we first feed $\{\mathbf{h}_{q,i}\}_{1\le i\le n_q}$ through a multilayer perceptron MLP_{att} and then a softmax function to generate the attention distribution over the items of the list:

$$a_{q,i} = \frac{\exp\left(\text{MLP}_{\text{att}}\left(\mathbf{h}_{q,i}\right)\right)}{\sum_{k=1}^{n_q}\exp\left(\text{MLP}_{\text{att}}\left(\mathbf{h}_{q,k}\right)\right)}, 1 \le i \le n_q. \tag{3}$$

The generated positive weight $a_{q,i}$ in (3) can be interpreted as an estimation of the probability that item $d_{q,i}$ is the right place to focus on for downstream finer-grained ranking. Based on the attention distribution, we calculate $\mathbf{c}_q$ as the attention-weighted mean of the item vectors:

$$\mathbf{c}_q = \sum_{i=1}^{n_q} a_{q,i}\mathbf{h}_{q,i}. \tag{4}$$

Compared to previous models that simply adopt the last hidden state of RNN, attention layer defined in (3) and (4) may extract a more informative global context encoding $\mathbf{c}_q$ since higher weights are assigned to items that are estimated to be more relevant for context modeling. We then combine $\mathbf{c}_q$ with the original item representations to form the refined representation vectors as:

$$\widetilde{\mathbf{h}}_{q,i} = \begin{bmatrix}\mathbf{c}_{q,i} \odot \mathbf{h}_{q,i} \\ \mathbf{h}_{q,i}\end{bmatrix}, 1 \le i \le n_q, \tag{5}$$

where $\odot$ denotes the Hadamard product. $\widetilde{\mathbf{h}}_{q,i}$ is regarded as the refined representation since it encodes both the listwise contextual information and the item information that are relevant to the ranking task.

Remark 1. It is interesting to note that the attention pooling strategy adopted here is a variant of the orderless attention mechanism considered in (Raffel and Ellis 2015; Lin et al. 2017), where all items are independent instead of being sequentially dependent as in RNN-based models. Such a strategy allows the information contained in all item representations to be directly propagated to $\mathbf{c}_q$, which may prevent the long-term memorization problem in RNN-based models.

Query Normalization Layer

As mentioned earlier, the feature distributions of candidate items for different queries are often different, which makes the learning of an effective global ranking function difficult. This problem cannot be completely solved by adopting the context representation presented in the previous section, as the ranking model still needs to learn to disentangle the information about the 'global' differences between queries and the 'local' differences between items for each query, and then focus on the latter to perform effective ranking.

To address the distributional differences among queries, we borrow ideas from previous DG approaches (Ganin et al. 2016; Motiian et al. 2017; Muandet, Balduzzi, and Schölkopf 2013), and encourage the ranking system to learn *query-invariant* latent representations that have similar or even identical distributions across all queries, with the hope that the system may generalize better to unseen queries by eliminating the detrimental influence of inter-query variability.

Additionally, compared with general DG problems, LETOR has its own distinct characteristics that can be exploited to facilitate the learning of query-invariant representations. In particular, the candidate items for each query is available as a whole, which makes the problem of characterizing the underling distributions easier. For example, for any query q, we can directly compute the attention-weighted per-dimension mean and variance vectors of its associated feature distribution as:

$$\overline{\mathbf{c}}_q = \sum_{i=1}^{n_q} a_{q,i}\widetilde{\mathbf{h}}_{q,i}, \tag{6}$$

$$\overline{\boldsymbol{\sigma}}_q^2 = \sum_{i=1}^{n_q} a_{q,i}\left(\widetilde{\mathbf{h}}_{q,i} - \overline{\mathbf{c}}_q\right)^2. \tag{7}$$

As is the case with the context vector (4), the attention weights (3) are adopted here to emphasize items that are estimated to be more important. Based on (6) and (7), we can construct a query normalization (QN) layer that directly matches these two feature distribution statistics of any given query to zero and unit vectors, respectively:

$$\overline{\mathbf{h}}_{q,i} = \left(\widetilde{\mathbf{h}}_{q,i} - \overline{\mathbf{c}}_q\right) \odot \left(\overline{\boldsymbol{\sigma}}_q + \varepsilon\right)^{-1}, 1 \le i \le n_q, \quad (8)$$

where ε is a small positive constant to avoid division by zero.

The QN transform (8) can be viewed as a straightforward extension of the widely used batch normalization (BN) transform (Ioffe and Szegedy 2015), where the key difference is that the latter applies the normalization to a batch of training queries instead of to items in each single query.

Ranking Layer

The normalized hidden representations (8) are passed into another MLP with softmax function to infer the final ranking score:

$$s_{q,i} = \frac{\exp\left(\text{MLP}_{\text{rank}}\left(\overline{\mathbf{h}}_{q,i}\right)\right)}{\sum_{i=1}^{n_q} \exp\left(\text{MLP}_{\text{rank}}\left(\overline{\mathbf{h}}_{q,i}\right)\right)}, 1 \le i \le n_q. \quad (9)$$

Note that (3) and (9) have the same structure, thus the ranking layer is essentially the second attention layer in our model, and $\{s_{q,i}\}_{1 \le i \le n_q}$ are the calculated attention weights.

Objective Function

In general, for each training batch $\mathcal{B} \subseteq \mathcal{T}$, QILCM jointly minimizes the *ranking loss* and the *query confusion loss* with a weight parameter λ:

$$\mathcal{L}_{\text{QILCM}} = \mathcal{L}_{\text{rank}} + \lambda \mathcal{L}_{\text{conf}}. \quad (10)$$

In (10), the ranking loss $\mathcal{L}_{\text{rank}}$ can be any suitable loss function that measures the ranking accuracy. In this work, we specifically adopt the AttRank loss proposed in (Ai et al. 2018) due to its good performance. Let the relevance labels be normalized as:

$$\widetilde{y}_{q,i} = \frac{\psi\left(y_{q,i}\right)}{\sum_{j=1}^{n_q} \psi\left(y_{q,j}\right)}, 1 \le i \le n_q, \quad (11)$$

where $\psi(x)$ is the truncated exponential function that returns $\exp(x)$ if $x > 0$ and 0 otherwise. The ranking loss simply measures the cross entropy between the score (9) and the normalized relevancy labels:

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{B}|} \sum_{q \in \mathcal{B}} \left(-\frac{1}{n_q} \sum_{i=1}^{n_q} \widetilde{y}_{q,i} \log\left(s_{q,i}\right)\right), \quad (12)$$

where $|\cdot|$ denotes the cardinality of a set.

On the other hand, by using the QN layer, the latent feature distribution for any given query is enforced to have zero mean and unit variance in each dimension. However, other types of distributional differences, such as the differences of covariance patterns, can still be present between feature representations from two queries. The query confusion loss is therefore intended to further promote the alignment of distributions of latent features (8) mapped from different queries. In previous DG works, this goal is typically achieved either by minimizing a metric between distributions, or through domain adversarial learning (Ganin et al. 2016; Cohen et al. 2018) which updates the model parameters to fool a jointly learned domain discriminator that attempts to distinguish between samples from different domains. Due to the large number of domains/queries involved, it is difficult to train a domain discriminator in our setting, and thus we focus on the former approach and choose to minimize the following loss:

$$\mathcal{L}_{\text{conf}} = \frac{1}{|\mathcal{B}|^2} \sum_{q_1 \in \mathcal{B}} \sum_{q_2 \in \mathcal{B}} d_{\text{CH}}\left(q_1, q_2\right), \quad (13)$$

where d_{CH} is the Chamfer (pseudo)-distance (CD) (Fan, Su, and Guibas 2017) for measuring the distance between two point sets:

$$\begin{aligned} d_{\text{CH}}\left(q_1, q_2\right) = &\sum_{i=1}^{n_{q_1}} \min_{1 \le j \le n_{q_2}} \left(\left\|\overline{\mathbf{h}}_{q_1,i} - \overline{\mathbf{h}}_{q_2,j}\right\|_2^2\right) \\ &+ \sum_{j=1}^{n_{q_2}} \min_{1 \le i \le n_{q_1}} \left(\left\|\overline{\mathbf{h}}_{q_1,i} - \overline{\mathbf{h}}_{q_2,j}\right\|_2^2\right) \end{aligned} \quad (14)$$

Remark 2. As both the QN layer and the query confusion regularization are intended to reduce the distributional differences of queries, a naturally arising question is whether these techniques would lead to loss of useful query information and hurt the ranking performance. We will address this issue in the experiments.

Experiments

Baseline Methods

We compare the proposed QILCM with the following three benchmark methods whose implementations are publicly available:

DCM[1] (Mottini and Acuna-Agost 2017) and DLCM[2] (Ai et al. 2018) are two recently proposed listwise context modeling methods based on RNNs. Note that the initial formulation of DCM could only handle binary-valued relevancy labels, we resolve this problem by simply replacing the original softmax loss function of DCM with the AttRank loss (9) used in DLCM and our method.

LambdaMART (Wu et al. 2010) is one of most widely-used listwise LETOR method. We adopt the open-source implementation of this algorithm provided in (Capannini et al. 2016).

Ranking Tasks and Datasets

We evaluate various methods on two ranking tasks which listwise context modeling methods have been successfully applied to:

[1]https://www.dropbox.com/s/swghso88q0s3hp7/code.zip

[2]https://github.com/QingyaoAi/Deep-Listwise-Context-Model-for-Ranking-Refinement

Ranking Refinement. (Ai et al. 2018) show that the initial ranked list returned by a LETOR method can be greatly improved by using listwise context modeling methods to rerank the top-ranked results. For this task, we used two large-scale LETOR datasets: Istella-S[3] (Lucchese et al. 2016) and Microsoft Letor 30K[4] (Qin and Liu 2013). We followed the experimental protocols described in (Ai et al. 2018), and adopted LambdaMART to do the initial retrieval, items ranked among the top-100 positions were then re-ranked using various listwise context modeling methods. Note that item lists with less than 100 items were entirely re-ranked in our experiments.

User Choice Ranking. (Mottini and Acuna-Agost 2017) show that listwise context modeling methods can accurately predict the user's choice among alternative commodities. For this task, we used Airline Itinerary[5], which is an anonymized version of the dataset used in (Mottini and Acuna-Agost 2017), each record in the data corresponds to a query, and contains the candidate items presented to a customer. The items that the customers purchased are positive samples and others are negative samples.

Statistics of the used datasets are summarized in Table 1.

Dataset	Queries	Items	Rel.	Feats.
Airline Itinerary	33951	1,089k	2	17
Microsoft 30k	31531	3,771k	5	136
Istella-S	33018	3,302k	5	220

Table 1: Characteristics of the datasets used in the experiments: number of queries, items, relevance levels, and features.

Evaluation Metrics

For each task, we adopt the evaluation metrics used in prior work. For ranking refinement tasks, we follow (Ai et al. 2018) and use the standard discounted cumulative gain (NDCG) metrics that include NDCG@1, NDCG@3, NDCG@5, and NDCG@10. For Microsoft 30k dataset, an additional metric NDCG@50 was also evaluated. On the other hand, for user choice ranking tasks where relevance feedbacks are binary-valued, we follow (Mottini and Acuna-Agost 2017) and use top precision metrics that include P@1(Precision@1) and P@5.

Model Training

Each dataset is split in train, validation and test sets according to a 60%-20%-20% scheme. The validation set was used to select the optimal hyperparameters for all involved methods. We did not perform extensive hyperparameter search for the proposed model, and used virtually the same architecture throughout all the experiments and datasets. More specifically, the dimensions of the nonlinear transformations (1) in the Input Encoder were fixed as 100, while MLPs used in (3) and (9) consist of 2 hidden layers with either 256 or 128 ELUs. The models were trained with the Adam algorithm (Kingma and Ba 2014) with a learning rate of 0.001, batch size of 80. Training generally converged after less than 100 passes through the entire training dataset.

[3]http://quickrank.isti.cnr.it/istella-dataset/

[4]https://www.microsoft.com/en-us/research/project/mslr/

[5]https://www.dropbox.com/s/qzyt0xwn4u2a1ed/data_full_anonym.zip

Comparisons with Baseline Methods

The performance comparison results of various methods are reported in Table 2. To eliminate the influence of random initiations, all results are averaged over 20 runs. As is shown in this table, QILCM significantly outperforms all the baselines.

Ablation Studies

To elucidate the contributions of the main components of our system, in this section, we test several variants of the proposed model. The tested implementations include:

- **Variant 1**: Our model with the attention-weighted context encoding (4) replaced by simple average pooling of the item encoding vectors. Accordingly, the attention-weighted statistics (6) and (7) are replaced with standard mean and variance;
- **Variant 2**: Our model without the domain confusion loss;
- **Variant 3**: Our model without the domain confusion loss and QN layer.

The performance results of different implementations of the proposed methods are shown in Table 3, which shows that QILCM consistently achieves the best performance among all model variants.

Anomalous Query Analysis

To further investigate the performance of QILCM, in this section we conducted additional analysis of experimental results on the Microsoft 30k dataset. Concretely, we firstly followed (Geng et al. 2008), and constructed vector representation for each query by averaging over the feature values of the top k ranked items (k was set as 10 during the experiments). After that, we fitted Isolation Forest (Liu, Ting, and Zhou 2008) to the training queries, and then used the learned model to assign a 'anomaly score' to each query in the test set. Intuitively, this score quantifies how different a query is from the majority of training data, and we examine the performance difference between QILCM and the best-performing baseline (DLCM) for queries with different levels of anomaly. As shown in Figure 2, the performance gap between QILCM and DLCM is significantly widened for more anomalous queries. For example, the NDCG@1 gap between QILCM and DLCM is increased from 0.047 to 0.121, which clearly demonstrates the advantage of tackling heterogeneous queries using the proposed DG perspective.

Conclusion

In this paper, we introduce a DG formulation of the LETOR problem and propose a novel neural architecture for DG in this LETOR context. We evaluate our techniques on three benchmark datasets, demonstrating that the proposed approach outperforms previous state-of-the-art approaches by a substantial margin.

Dataset	Metrics	QILCM	DCM	DLCM	LambdaMART	Improv.	P-value
Airline Itinerary	P@1	*0.2833	$\underline{0.2618}$	0.2562	0.2327	8.21%	2.40×10^{-23}
	P@5	*0.6958	0.6586	$\underline{0.6651}$	0.6239	4.61%	1.22×10^{-20}
Microsoft 30k	NDCG@1	*0.5447	0.4938	$\underline{0.4973}$	0.4800	9.53%	3.83×10^{-19}
	NDCG@3	*0.5313	$\underline{0.4827}$	0.4811	0.4766	10.06%	7.58×10^{-16}
	NDCG@5	*0.5368	$\underline{0.4904}$	0.4892	0.4842	9.46%	1.60×10^{-16}
	NDCG@10	*0.5564	0.5093	$\underline{0.5135}$	0.5061	8.35%	2.27×10^{-15}
	NDCG@50	*0.6482	0.6127	$\underline{0.6146}$	0.6092	5.46%	4.51×10^{-15}
Istella-S	NDCG@1	*0.7023	0.6762	$\underline{0.6873}$	0.6644	2.18%	1.74×10^{-10}
	NDCG@3	*0.6696	$\underline{0.6552}$	0.6537	0.6378	2.20%	1.99×10^{-8}
	NDCG@5	*0.6953	$\underline{0.6846}$	0.6831	0.6741	1.56%	4.25×10^{-10}
	NDCG@10	*0.7645	0.7558	$\underline{0.7566}$	0.7456	1.04%	4.67×10^{-7}

Table 2: Performance comparison of various methods. The results are averaged over 20 random runs, and the best ones are marked with *. The last two columns show the improvement of QILCM over the best baseline algorithm (highlighted with underline), and the corresponding Student's t-test P-values.

Dataset	Metrics	QILCM	Variant 1	Variant 2	Variant 3
Airline Itinerary	P@1	*0.2833	0.2749	0.2762	0.2694
	P@5	*0.6958	0.6613	0.6803	0.6724
Microsoft 30k	NDCG@1	*0.5447	0.5287	0.5359	0.5139
	NDCG@3	*0.5313	0.5127	0.5294	0.4970
	NDCG@5	*0.5368	0.5230	0.5347	0.5030
	NDCG@10	*0.5564	0.5459	0.5538	0.5229
	NDCG@50	*0.6482	0.6363	0.6438	0.6331
Istella-S	NDCG@1	*0.7023	0.6966	0.6982	0.6837
	NDCG@3	*0.6696	0.6654	0.6672	0.6628
	NDCG@5	*0.6953	0.6936	0.6931	0.6923
	NDCG@10	*0.7645	0.7622	0.7637	0.7602

Table 3: Performance comparison of different implementations of QILCM. The results are averaged over 20 random runs, and the best ones are marked with *.

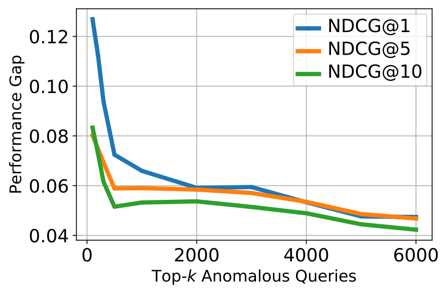

Figure 2: Performance gap between QILCM and the best-performing baseline (DLCM) on Microsoft 30k dataset.

References

Ai, Q.; Bi, K.; Guo, J.; and Croft, W. B. 2018. Learning a deep listwise context model for ranking refinement. In *SIGIR*, 135–144.

Blanchard, G.; Lee, G.; and Scott, C. 2011. Generalizing from several related classification tasks to a new unlabeled sample. In *NIPS*, 2178–2186.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, 89–96.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*, 129–136.

Capannini, G.; Lucchese, C.; Nardini, F. M.; Orlando, S.; Perego, R.; and Tonellotto, N. 2016. Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing and Management* 52(6):1161–1177.

Chapelle, O., and Chang, Y. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*, 1–24.

Chen, X.; Xu, H.; Zhang, Y.; Tang, J.; Cao, Y.; Qin, Z.; and Zha, H. 2018. Sequential recommendation with user memory networks. In *WSDM*, 108–116.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Cohen, D.; Mitra, B.; Hofmann, K.; and Croft, W. B. 2018. Cross domain regularization for neural ranking models using adversarial learning. In *SIGIR*, 1025–1028.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *RecSys*, 191–198.

Fan, Y.; Guo, J.; Lan, Y.; Xu, J.; Zhai, C.; and Cheng, X. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *SIGIR*, 375–384.

Fan, H.; Su, H.; and Guibas, L. 2017. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2463–2471.

Freno, A. 2017. Practical lessons from developing a large-scale recommender system at zalando. In *RecSys*, 251–259.

Freund, Y.; Iyer, R.; Schapire, R. E.; and Singer, Y. 2003. An efficient boosting algorithm for combining preferences. *JMLR* 4(Nov):933–969.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *JMLR* 17(1):2096–2030.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *ICML*, 1243–1252.

Geng, X.; Liu, T.-Y.; Qin, T.; Arnold, A.; Li, H.; and Shum, H.-Y. 2008. Query dependent ranking using k-nearest neighbor. In *SIGIR*, 115–122.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 448–456.

Karmaker Santu, S. K.; Sondhi, P.; and Zhai, C. 2017. On application of learning to rank for e-commerce search. In *SIGIR*, 475–484.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y. A.; Bottou, L.; Orr, G. B.; and Müller, K.-R. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer. 9–48.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2018. Learning to generalize: Meta-learning for domain generalization. In *AAAI*.

Li, P.; Wu, Q.; and Burges, C. J. 2008. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, 897–904.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *ICDM*, 413–422.

Lucchese, C.; Nardini, F. M.; Orlando, S.; Perego, R.; Silvestri, F.; and Trani, S. 2016. Post-learning optimization of tree ensembles for efficient ranking. In *SIGIR*, 949–952.

Manotumruksa, J.; Macdonald, C.; and Ounis, I. 2018. A contextual attention recurrent architecture for context-aware venue recommendation. In *SIGIR*, 555–564.

Motiian, S.; Piccirilli, M.; Adjeroh, D. A.; and Doretto, G. 2017. Unified deep supervised domain adaptation and generalization. In *ICCV*, 5716–5726.

Mottini, A., and Acuna-Agost, R. 2017. Deep choice model using pointer networks for airline itinerary prediction. In *SIGKDD*, 1575–1583.

Muandet, K.; Balduzzi, D.; and Schölkopf, B. 2013. Domain generalization via invariant feature representation. In *ICML*, 10–18.

Nallapati, R. 2004. Discriminative models for information retrieval. In *SIGIR*, 64–71.

Qin, T., and Liu, T.-Y. 2013. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*.

Raffel, C., and Ellis, D. P. 2015. Feed-forward networks with attention can solve some long-term memory problems. In *ICLR*.

Shankar, S.; Piratla, V.; Chakrabarti, S.; Chaudhuri, S.; Jyothi, P.; and Sarawagi, S. 2018. Generalizing across domains via cross-gradient training. In *ICLR*.

Taylor, M.; Guiver, J.; Robertson, S.; and Minka, T. 2008. Softrank: Optimizing non-smooth rank metrics. In *WSDM*, 77 – 85.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2016. Order matters: Sequence to sequence for sets. In *ICLR*.

Volkovs, M., and Zemel, R. S. 2012. Collaborative ranking with 17 parameters. In *NIPS*, 2294–2302.

Wu, Q.; Burges, C. J. C.; Svore, K. M.; and Gao, J. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13(3):254–270.

Wu, L.; Hsieh, C.-J.; and Sharpnack, J. 2018. SQL-rank: A listwise approach to collaborative ranking. In *ICML*, 5315–5324.

Xiang, B.; Jiang, D.; Pei, J.; Sun, X.; Chen, E.; and Li, H. 2010. Context-aware ranking in web search. In *SIGIR*, 451–458.

Xie, Q.; Dai, Z.; Du, Y.; Hovy, E.; and Neubig, G. 2017. Controllable invariance through adversarial feature learning. In *NIPS*, 585–596.

Zamani, H.; Bendersky, M.; Wang, X.; and Zhang, M. 2017. Situational context for ranking in personal search. In *WWW*, 1531–1540.

Zamani, H.; Mitra, B.; Song, X.; Craswell, N.; and Tiwary, S. 2018. Neural ranking models with multiple document fields. In *WSDM*, 700–708.

Zhou, C.; Bai, J.; Song, J.; Liu, X.; Zhao, Z.; Chen, X.; and Gao, J. 2018. Atrank: An attention-based user behavior modeling framework for recommendation. In *AAAI*, 4564–4571.

Zhuang, T.; Ou, W.; and Wang, Z. 2018. Globally optimized mutual influence aware ranking in e-commerce search. In *IJCAI*, 3725–3731.