Integrated Learning of Features and Ranking Function in Information Retrieval

Yifan Nie
Université de Montréal
Montréal, Québec, Canada
yifan.nie@umontreal.ca

Jiyang Zhang
Beihang University
Beijing, China
zhangjiy97@buaa.edu.cn

Jian-Yun Nie
Université de Montréal
Montréal, Québec, Canada
nie@iro.umontreal.ca

ABSTRACT

Recent deep learning models for information retrieval typically aim to learn features either about the contents of the document and the query, or about the interactions between them. However, the existing literature shows that document ranking depends simultaneously on many factors, including both content and interaction features. The integration of both types of neural features has not been extensively studied. In addition, many studies have also shown that the deep neural features cannot replace completely the traditional features, but are complementary. It is thus reasonable to combine deep neural features with traditional features. In this paper, we propose an integrated end-to-end learning framework based on learning-to-rank (L2R) to learn both neural features and the L2R ranking function simultaneously. The framework also has the flexibility to integrate arbitrary traditional features. Our experiments on public datasets confirm that such an integrated learning strategy is better than separate learning of features and ranking function, and integrating traditional features can further improve the results.

CCS CONCEPTS

• **Information systems** → *Retrieval models and ranking*.

KEYWORDS

Information Retrieval, Neural Network, Ranking

ACM Reference Format:
Yifan Nie, Jiyang Zhang, and Jian-Yun Nie. 2019. Integrated Learning of Features and Ranking Function in Information Retrieval. In *The 2019 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '19), October 2–5, 2019, Santa Clara, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3341981.3344232

1 INTRODUCTION

Deep learning models have been recently used in ad-hoc information retrieval (IR) and shown competitive results. Those models can be categorized into either representation-based models or interaction-based models [5]. Representation-based models [6, 7, 21] focus on learning representations of query and document through a series of neural layers and at the end, a match is estimated between the representations of query and document. On the other hand, interaction-based models [5, 14, 16, 24] first observe local interactions between query and document terms and then learn the interaction patterns through several neural layers. Intuitively, representation-based models can create more abstract representations about the contents, thereby enable some generalization so that similar contents can be matched. They are more appropriate to cope with conceptual queries which require some generalization. For example, for queries like "small business statistics", the documents containing "small company statistics" or "micro corporate statistics" may also be relevant. On the other hand, interaction-based models first build local term-to-term interactions by applying a similarity measure and the learning process focuses on interaction patterns allowing to match the query and documents. Therefore, they are more able to deal with lexical queries which require exact term match between query and document. Queries that contain named entities such as "distance between Grand Canyon and Phoenix" fall into this category.

The above examples illustrate the respective strength of the two approaches, which can be combined in an integrated approach. However, such a combined approach has not been extensively studied. The Duet Model [12] is an exception which combines the two approaches by linearly adding the ranking scores of two separate models and learns the ranking function by maximizing the probability score of positive document over negative documents for a given query. Despite the fact that Duet improved the retrieval effectiveness, we believe that both approaches can be better integrated. In particular, the way the two components interact with each other should be learned, and a better ranking function such as learning-to-rank framework could be employed.

In this paper, we propose a different and general approach to combine representation- and interaction-based models through learning-to-rank (L2R): the representation- and interaction-based models will generate a set of features that are fed into a L2R layer, and the latter will learn an appropriate ranking function based on the features. Different from the traditional L2R methods, the features used are also learned at the same time as the ranking function. Therefore, the approach we propose integrates both feature learning and ranking function learning. Compared to separate feature learning and ranking function learning, an integrated learning framework has a clear advantage: the features can adapt depending on how they are used in the ranking function, and the ranking function also adapts depending on the features at hand. Both the features and the ranking function are learned to maximize the final objective of document ranking.

The utilization of representation and interaction features in traditional L2R is not new: Most L2R approaches utilize both categories

ICTIR '19, October 2–5, 2019, Santa Clara, CA, USA

ACM ISBN 978-1-4503-6881-0/19/10…$15.00
https://doi.org/10.1145/3341981.3344232

of features. It has been shown in many experiments that both types of feature are useful and they are complementary. Our approach follows the same principle, but within the framework of neural networks, and by incorporating feature learning as well rather than using manually fixed feature inputs.

To summarize, this paper proposes a method to combine the representation- and interaction-based neural approaches within the learning-to-rank framework, in which both feature learning and ranking-function learning are conducted end-to-end. We call it Integrated Learning Model (ILM). Our contribution is three-fold: (1) We combine representation-based and interaction-based neural approaches in a flexible learning-to-rank framework. (2) We integrate feature learning with L2R ranking function learning, which are trained end-to-end simultaneously. (3) We show that the proposed model can significantly outperform the existing neural models on Million Query datasets, and that integrating traditional features can further improve the results.

2 RELATED WORK

2.1 Neural IR Models

Depending on how the matching score $S(q, d)$ between query q and document d is produced, previous neural IR models could be categorized into representation-based models and interaction-based models [5]. In a representation-based model, the relevance score can be computed by Eqn. 1.

$$S_{rep}(q, d) = S(\phi(q), \phi(d)) \tag{1}$$

where ϕ is a complex feature function to map query or document text into meaningful semantic representations through several hidden layers. S is the matching function. For example, in DSSM [7], ϕ is a feed forward neural network and S is a simple cosine similarity. CDSSM [21] uses convolutional network to implement ϕ. In ARC-I [6], S is further replaced by an MLP to allow more complex matching than cosine similarity.

In interaction-based models, relevance scores are calculated in a different way as depicted in Eqn. 2.

$$S_{inter}(q, d) = S_n \circ S_{n-1} \circ ... \circ S_0(w(q), w(d)) \tag{2}$$

where the feature function w is often a simple embedding look-up function which maps the term into its word embedding vector, and the matching function is a composition of a series of neural layers $S_0, S_1, ..., S_n$. For example, in MatchPyramid [14] and ARC-II [6], the feature function w is the embedding look-up function, and the neural layers are convolutional layers aiming at learning interaction patterns. In DRMM [5], the interaction between query q and document d is captured by histograms of interaction intensities and stacked MLP layers are used to analyze the interaction patterns from the histograms. In Match-SRNN [24], Neural Tensor Network [22] is used to build local interaction tensors and Spatial RNN is used to analyze the interaction patterns.

To take advantage of the strength and focus of both matching mechanisms, the original Duet Model [12] combines both a representation-based model and an interaction-based model, which generate two independent relevance scores. These scores are then added to produce a final score:

$$R_{Duet}(q, d) = R_{rep}(q, d) + R_{inter}(q, d) \tag{3}$$

To provide more informative matching signals, a recent Updated Duet Model [11] produces matching pattern vectors of the two sub-models and combines them through MLPs instead of adding the matching scores. The extended Duet model goes in the same direction as our ILM model, but is done in parallel to our work.

Although Duet Model and its variants yielded better results than the representation-based and interaction-based models separately, we can see several limitations. First, in the local interaction model, only exact match function is employed, which does no allow matching similar terms. Second, the ranking function employed in Duet is a softmax probability over scores of positive document and negative documents for a given query. It is better to optimize the final ranking objective for ranking tasks as shown in [2, 23].

2.2 Learning-to-rank

Learning-to-rank (L2R) is a general ranking framework for IR which yielded state-of-the-art results [9]. According to how the ranking model is trained, L2R algorithms could be categorized into 3 types: point-wise approaches, pair-wise approaches and list-wise approaches [10]. In particular, pair-wise approaches care about the relative preference between two documents given a query. A popular and effective pair-wise L2R model is LambdaRank [2]. LambdaRank tries to push relevant documents upwards and non relevant documents downwards in a ranked list during training by optimizing the NDCG metric [26]. As the training objective function directly corresponds to the evaluation metric, LambdaRank can outperform similar ranking models trained with a pairwise hinge or cross-entropy loss [2]. In [2], an MLP is used to act as ranker. A L2R model works with a set of features extracted from a document-query pair such as document and query length, term frequencies, and different matching scores between them. It has been found that both the features relating to the query and to the documents (contents), as well as the features relating to their interactions (matching scores or patterns) are important. This observation motivates us to combine both representation-based and interaction-based neural features in an integrated framework.

L2R has been recently adapted to the neural model context. In [1], a Group Scoring Function Model (GSF) is proposed. For a given query q and a list of document $[d_i, ..., d_n]$, the model considers each possible group $(q, d_i, d_j), i, j \in [1, n]$, and builds parallel MLPs for each group in order to produce intermediate group relevance scores. The final ranking score for a given document d_k is calculated by accumulating its intermediate scores.

Other studies also extended the LambdaRank approach to general cases. LambdaLoss [25] provides a theoretical analysis of the effectiveness when directly optimizing an evaluation metric. It also generalizes LambdaRank to optimize other metrics such as Average Relevance Position (ARP) [25]. A toolkit for neural L2R [17] is also made available recently.

All the existing L2R approaches require to be provided with a set of features. In general, these features are extracted independently from ranking-function learning. It is possible that pre-trained features are not optimal for the ranker. It is more reasonable to learn features and ranking function simultaneously, so that they can influence each other. A deep neural model offers a flexible framework to implement such an integrated approach.

Based on the above observations, we propose an Integrated Learning Model (ILM) which incorporates the representation- and interaction-based features within the learning-to-rank framework. In this model, both the ranker and features are trained simultaneously in an end-to-end fashion by LambdaRank. We will describe the details of our model in the next section.

3 INTEGRATED LEARNING MODEL FOR IR

The general architecture of the Integrated Learning Model (ILM) is shown in Fig. 1. It integrates the learning of representation- and interaction-based features and the ranking function within a learning-to-rank framework.

3.1 Model Components

The proposed ILM is composed of several components. In the lower part, several modules aim at generating features. We incorporate two types of neural modules to generate representation and interaction features. In addition to neural features, we also incorporate traditional (non-neural) features. The features will be fed into a L2R layer in order to produce a ranking score (S), and the neural features will be tuned together with the ranking function rather than being served as fixed inputs.

This framework is general and flexible. In fact, any existing representation-based and interaction-based neural models can be used to play the role of the two feature modules in the framework, provided that they can be trained in an end-to-end fashion together with the ranking function.

Content Representation Module: Inspired by the CDSSM [21] representation learning module, we employ a similar 1D convolutional model to learn query and document representations. The query and document are represented by a set of word embedding vectors $q = [t_1^{(q)}, ..., t_n^{(q)}]$, and $d = [t_1^{(d)}, ..., t_m^{(d)}]$, where $t_i^{(q)}$ and $t_j^{(d)}$ represent the embedding vectors for query term i and document term j respectively, and n, m are the query length and document length respectively. Afterwards 1D convolution is applied to aggregate term embeddings inside a window of size $2k+1$ into phrase representations as follows.

$$C_{i,rep}^{q} = f(W^{q} * [t_{i-k}^{q}; ..; t_{i+k}^{q}] + b^{q}) \tag{4}$$

$$C_{i,rep}^{d} = f(W^{d} * [t_{i-k}^{d}; ..; t_{i+k}^{d}] + b^{d}) \tag{5}$$

where $C_{i,rep}^{q}$ and $C_{i,rep}^{d}$ represent the convolved representation for the i^{th} query and document term respectively; f is the activation function; W and b represent the weight and bias of the convolution; and $2k+1$ is the window size of the convolution.

Once the convolution is performed, a dimension-wise max-pooling is performed and the max-pooled query and document representations are fed into an output MLP layer for dimensionality reduction. The process is summarized as follows.

$$P_{rep}^{q} = \max_{j}(C_{i,rep}^{q}(j)) \;\; P_{rep}^{d} = \max_{j}(C_{i,rep}^{d}(j)) \tag{6}$$

$$Q_{rep} = f(W_o^{q} P_{rep}^{q} + b_o^{q}) \;\; D_{rep} = f(W_o^{d} P_{rep}^{d} + b_o^{d}) \tag{7}$$

where P_{rep}^{q} and P_{rep}^{d} are the max-pooled representations of the query and document; $C_{i,rep}(j)$ represents the j^{th} dimension of the convolved vector $C_{i,rep}$; Q_{rep} and D_{rep} are the query and document representations after the dimension-reduction output layer.

The representation of query Q_{rep} and document D_{rep} are then concatenated and fed into a representation encoding MLP to produce the representation feature vector for the L2R layer. The process is defined as follows:

$$E_{rep} = concat(Q_{rep}, D_{rep}) \tag{8}$$

$$H_{rep} = f(W_{rep} E_{rep} + b_{rep}) \tag{9}$$

where E_{rep} is the concatenation of q_{rep} and d_{rep}; f is the activation function; W_{rep} and b_{rep} are the weight and bias of the representation encoding MLP.

Notice that this last step represents an important difference with a standalone representation-based model: our module aims at producing a vector of neural representation about the document and the query, rather than a matching score. As we stated earlier, these features can interact in the L2R layer, together with the interaction features that we will describe in the following section.

Interaction Module: MatchPyramid is a popular interaction-based model which has shown promising performance in applications ranging from short text matching [15] to IR [14]. We build similar interaction module to learn interaction pattern from query-document pairs and employ the learned pattern as interaction feature.

First the local interaction matrix I is built by applying cosine similarity between each query term embedding $t_i^{(q)}$ and document term embedding $t_j^{(d)}$.

$$I_{ij} = cos(t_i^{(q)}, t_j^{(d)}) \tag{10}$$

Once the input interaction matrix is constructed, a series of 2D convolution and max-pooling layers are added on top in order to build more abstract interaction patterns.

$$C_{1,inter}^{k} = f(W_1^{k} * I + b_1^{k}),\ k = 1, .., K \tag{11}$$

$$P_{1,inter}^{k} = max\_pool(C_{1,inter}^{k}),\ k = 1, .., K \tag{12}$$

$$C_{i,inter}^{k} = f(W_i^{k} * P_{i-1,inter}^{k} + b_i^{k}),\ i = 2, .., L,\ k = 1, .., K \tag{13}$$

$$P_{i,inter}^{k} = max\_pool(C_{i,inter}^{k}),\ i = 2, .., L,\ k = 1, .., K \tag{14}$$

where $C_{i,inter}^{k}$ is the feature map k of the i^{th} convolved layer; I is the input interaction matrix; W_i^{k} and b_i^{k} are the kernel and bias of layer i for the feature map k; L is the number of convolution layers, and K is the number of feature maps; f is a non-linear mapping; and $*$ represents the convolution operator.

In order to extract the interaction pattern of q and d, following [14, 20], the last max-pooled layer is flattened into a vector and fed into an interaction pattern encoding MLP. The encoded vector is then utilized as interaction feature and fed into the L2R layer.

$$E_{inter} = flatten(P_{L,inter}) \tag{15}$$

$$H_{inter} = h(W_o E_{inter} + b_o) \tag{16}$$

Non-neural ranking features: We concatenate scalar feature values together and reshape it into $BS \times N_{feats}$ matrix H_{feats}, where BS is the batch size, N_{feats} is the number of non neural features. These features are fed into the L2R layer.

L2R Layer: The L2R layer aims at computing a ranking score S based on the representation features H_{rep}, interaction feature

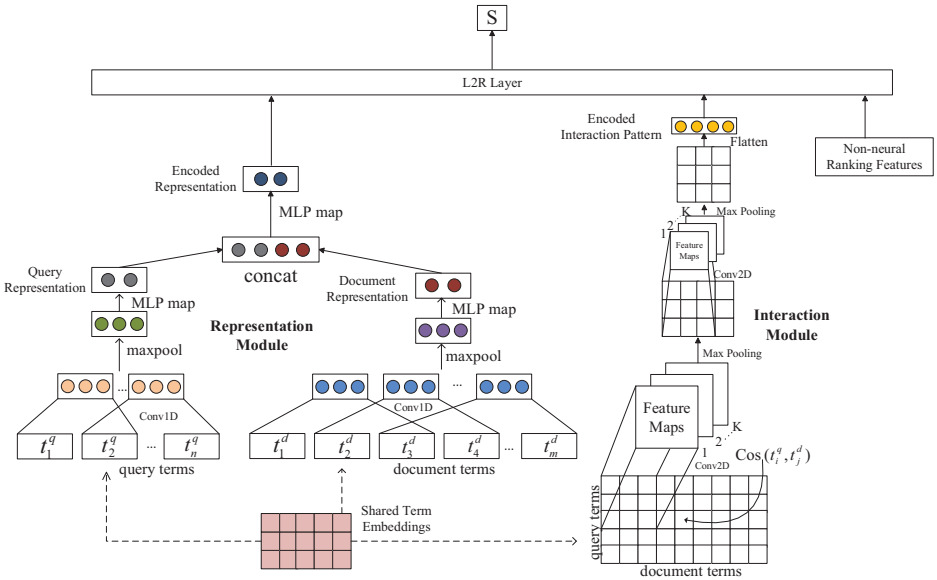

Figure 1: Integrated Learning Model

H_{inter} and non-neural ranking features H_{feats}:

$$S = f(W_s[H_{rep}; H_{inter}; H_{feats}] + b_s) \tag{17}$$

where W_s and b_s are the weight and bias and f represents the activation function. Different from the Duet model which adds the ranking scores of the two matching models, we combine the neural features of the two matching mechanisms to encourage interactions between them. The ranking function will be trained through LambdaRank. In the same training process, the neural features will also be adapted.

3.2 LambdaRank End-to-End Training

We use LambdaRank [2] learning to train the model in an end-to-end fashion. For a given query q, first, the probability that a document d_i is more relevant than another document d_j is modeled by the logistic function. Then, a cross entropy loss C_{ij} is employed to measure the discrepancy between the ground truth label probability $\overline{P}_{ij} = \frac{1}{2}(1 + S_{ij})$ and the predicted probability P_{ij}, where S_{ij} is the preference score of document d_i and d_j and takes values in $\{+1, 0, -1\}$.

$$P_{ij} = \frac{1}{1 + e^{-\sigma(s_i - s_j)}} \tag{18}$$

$$C_{ij} = -\overline{P}_{ij} log P_{ij} - (1 - \overline{P}_{ij}) log(1 - P_{ij}) \tag{19}$$

Afterwards the gradient $\frac{\partial C_{ij}}{\partial s_i}$ of loss function with respect to the predicted relevance score s_i is multiplied by the change of nDCG [26] values by swapping the positions of d_i and d_j in the ranked list as follows.

$$\lambda_{ij} = \frac{\partial C_{ij}}{\partial s_i} |\Delta nDCG(i, j)| \tag{20}$$

The gradient of the loss C_{ij} with respect to a trainable parameter w_k of the model could be derived by the chain rule.

$$\frac{\partial C_{ij}}{\partial w_k} = \lambda_{ij} \frac{\partial (s_i - s_j)}{\partial w_k} \tag{21}$$

where the second factor $\frac{\partial (s_i - s_j)}{\partial w_k}$ is the difference of the gradients of the predicted scores with respect to the parameter w_k which could be computed by back-propagation algorithm [4]. Note that for neural feature learning modules, we do not stop back-propagation at the L2R layer. We continue to back-propagate the loss signal to tune the neural feature learning layers and the word embeddings.

We train our model in a group-wise manner: for a given query q and its corresponding documents $D = \{d_1, d_2, .., d_n\}$, we consider all possible pairs (q, d_i, d_j) where $d_i, d_j \in D$. In a batch, there could be several query groups.

3.3 Alternative Configurations of ILM

To demonstrate the flexibility of ILM, we also test an alternative configuration, denoted as ILM-Hist, in which Deep Relevance Matching Model [5] (DRMM) is used as interaction feature module. The architecture is presented in Fig. 2

In the new interaction module, local interactions between query and document terms are mapped into histogram of B bins. Akin to [5], for each query term $t_i^{(q)}$, we calculate the interaction of this query term with all document terms $[t_1^{(d)}, ..., t_m^{(d)}]$ by cosine similarity and count the number of interactions falling in each bin. We use log-based counts as suggested by [5]:

$$I_i = [cos(t_i^{(q)}, t_1^{(d)}), ..., cos(t_i^{(q)}, t_n^{(d)})] \tag{22}$$

$$T_i = Hist(I_i, B) \tag{23}$$

$$T = concat([log(T_1), ..., log(T_m)]) \tag{24}$$

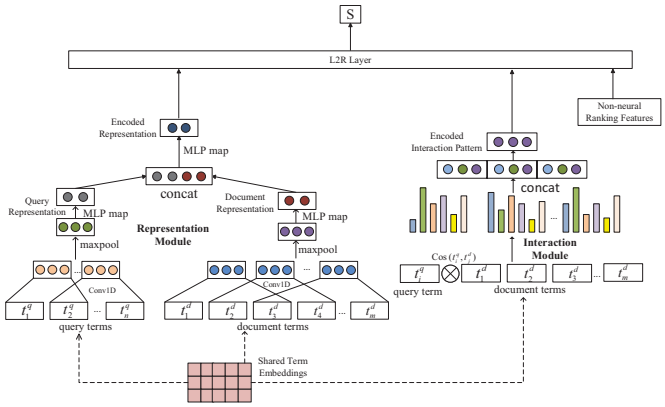

Figure 2: Alternative Configuration ILM-Hist

where I_i is the interaction vector of query term $t_i^{(q)}$ with all document terms, T_i is the histogram built based on the interaction vector I_i for the i^{th} query, and T is the concatenated histograms of all query terms. The concatenated histograms T are then mapped into the encoded interaction pattern H_{hist} by a feed-forward layer.

$$H_{hist} = f(W_{hist}T + b_{hist}) \tag{25}$$

where f is the activation function, W_{hist} and b_{hist} are the weight and bias of the model. This H_{hist} is fed into the L2R layer as interaction features.

4 EXPERIMENTAL STUDY

4.1 Dataset

Experimental study are conducted on the Letor dataset[1] which contains queries of Million Query Track 2007 and 2008, (denoted as MQ2007 and MQ2008) which contains documents from the GOV2 collection. These datasets are commonly used in a number of previous studies on ad-hoc retrieval task using deep neural models due to the fact that they contain a larger set of queries than other standard datasets. A large amount of training data is necessary for effectively training a neural model. The statistics of the two datasets are presented as follows.

Table 1: Statistics of the datasets in this study

	#queries	#docs	#rel_q	#rel_per_q
MQ2007	1,692	65,323	1,455	10.3
MQ2008	784	14,384	564	3.7

We perform 5-fold cross validation as in [3, 16] and directly rank the validation/test fold rather than reranking. The relevance judgments are integers ranging from 0 to 2. For a given training query, we consider all (q, d_i, d_j) pairs where the judgments are different. There are in average 12,886 and 2,799 training pairs over the 5 training folds of MQ2007 and MQ2008 respectively. During indexing and retrieval, we process queries and documents with Krovetz stemmer [8] and remove stop words according to the Indri standard stop list[2]. As stated in [3, 16], MQ2008 only contains 784 queries which is too small and will cause insufficient training problem in a deep model, following [3, 16] we merge the training set of MQ2007 into that of MQ2008 and keep the validation and test set unchanged.

4.2 Evaluation Metric

We evaluate the performance of baselines and our proposed models with MAP and nDCG [18].

4.3 Baselines and Alternative Configurations

Baselines: We implement the **BM25** [19] and Language Model with Dirichlet smoothing (**LM**) with Indri[1] as our traditional model baselines. Comparing with the traditional models is necessary because several previous experiments showed that neural models often have difficulty to match their performance [16]. We also compare the performance of our models with RankMLP-Letor and RankMLP-Letor+ which employ Letor features and Letor features plus BM25 and LM scores computed by Indri. We feed them into the L2R layer trained with LambdaRank framework. The latter two are equivalent to traditional L2R method.

Rep-MLP, Inter-MLP, HM-sum, HM-MLP: We first build models with only one matching mechanism. For representation-based model, we utilize the same representation-based module, denoted as **Rep-MLP** presented in Section 3.1 to learn representations. For interaction-based model, denoted as **Inter-MLP**, we utilize the same pyramid-based interaction module presented in Section 3.1.

To combine the representation and interaction module without non-neural ranking features, following the mechanism of Duet Model [12], **HM-sum** directly adds the matching scores of **Rep-MLP** and **Inter-MLP** to produce an aggregated relevance score. An alternative **HM-MLP** employs the similar mechanism depicted in 3.1, which feeds the representation features and interaction features into an MLP to produce an aggregated relevance score. The above models are trained with the hinge loss.

$$L(Q, D_+, D_-; \Theta) = max(0, 1 - (S(Q, D_+) - S(Q, D_-))) \tag{26}$$

ILM and its Variants: We also build the models within our proposed ILM framework trained with LambdaRank in an end-to-end manner. **ILM-Neu** is the ILM model presented in Fig. 1 with only neural (representation and interaction) modules. **ILM-BM25** and **ILM-LM** are the ILM-Neu model plus the baseline BM25 or LM score as non neural ranking feature. **ILM-Letor** is the ILM model presented in Fig. 1 with both representation, interaction and Letor[2] features as non neural ranking features. **ILM-Letor+** is the ILM-Letor model plus the baseline BM25 and LM scores as additional features.

To study the benefits of end-to-end learning of both the ranker and the neural feature learning modules, we also build 2 models with pre-trained fixed neural features output by Rep-MLP and Inter-MLP. **ILM-fix-Letor** is trained with fixed neural and letor features

[1] https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/

[2] http://www.lemurproject.org/stopwords/stoplist.dft

[1] https://www.lemurproject.org/indri/

[2] https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/

and **ILM-fix-Letor+** is trained with fixed neural and Letor features plus BM25 and LM scores.

To show that our framework is general and could be fit with other neural feature learning modules, we also build **ILM-Hist** which uses a histogram-based interaction module as presented in Section 3.3.

4.4 Experiment Settings

We employ the pretrained 300-dimensional GloVe.6B.300d embeddings[3] to initialize the embedding look-up matrix and fine-tune it during training. For 1-layered and 2-layered representation modules, following [13], we set the convolution window size to 3 and [3, 5], the dimension of convolved vector to 256. For the 2-layered pyramid interaction module, we set the shape of the 2D convolution filters to be (3, 3) and (5, 5), the number of filters to be [64, 32], and max-pooling shape to be (2, 2), based on a preliminary study. For the histogram interaction module, we set the number of bins B to 30 according to [5]. The dimension of the encoded representation H_{rep} and interaction pattern H_{inter} is set to 256, and the size of the L2R layer is set to 512. The vocabulary size is 400K. We set the max query length and document length to be $n = 15$, $m = 1000$, apply zero paddings [14] and omit OOV document terms. We employ Adam optimizer to optimizer the trainable parameters of our models and the initial learning rate is set to 1×10^{-3}.

4.5 Main Experimental Results

The main experimental results are presented in Table 2. We conduct paired t-test to compare ILM-BM25, ILM-LM, ILM-Letor and ILM-Letor+ with their respective counterparts BM25, LM, RankMLP-Letor and RankMLP-Letor+, respectively. The statistically significant results ($p < 0.05$) with respect to BM25, LM and RankMLP-Letor are marked with a, b, c respectively. For our proposed model ILM-Letor+, we also perform Bonferroni correction with respect to the set of all 4 baselines, and the statistically significant results after Bonferroni correction are marked with $*$.

We examine the following questions in the experiments:

(1) Are neural features useful? From Table 2, we first observe that our proposed ILM-Letor+ which combines neural representation and interaction features with non neural L2R features outperforms the set of baselines on all evaluation metrics on both MQ2007 and MQ2008. In most cases (except NDCG@1 on MQ2008), the difference is statistically significant. This confirms the effectiveness of our proposed ILM.

To answer the question more specifically, we can compare models (ILM-BM25, ILM-LM, ILM-Letor and ILM-Letor+) with neural features against their counterparts that do not contain neural features (BM25, LM, RankMLP-Letor and RankMLP-Letor+). In all the cases, we observe a large improvement on all the evaluation measures, and the differences are statistically significant. This result clearly demonstrates that the neural features on representation and interaction are useful, and they can help improve the effectiveness even when a set of traditional features are already included.

(2) Is integrated learning better than separate learning of the ranker and features? To investigate the benefits of integrated learning of the ranker and neural features, we build ILM-fix-Letor and ILM-fix-Letor+. In those models, we learn representation and interaction neural features with separate models, then input them as fixed features to the L2R layer and perform training by LambdaRank. Comparing them with the corresponding end-to-end versions (ILM-Letor and ILM-Letor+), from Table 3, we observe that the integrated learning is better than separate learning. This result confirms the advantage we expected with integrated learning. Although it is difficult to visualize the interactions between feature learning and ranking-function learning, we believe that the mutual influence between them reinforced both learning processes and this contributed to obtaining both better features and a better ranking function.

To further illustrate the benefits of integrated learning of the ranker and the neural feature modules in an end-to-end manner, we show a representative query "qid 7993: model railroads" from the test set and print the top 10 documents ranked by ILM-fix-Letor+ and ILM-Letor+ in Fig. 3. Note that those two models share the same neural components and non neural ranking features, but differ in whether the neural features are tuned together with the L2R ranking layer. ILM-fix-Letor+ employs pre-trained fixed representation and interaction features whereas ILM-Letor+ has the neural features trained in an end-to-end manner with the ranker.

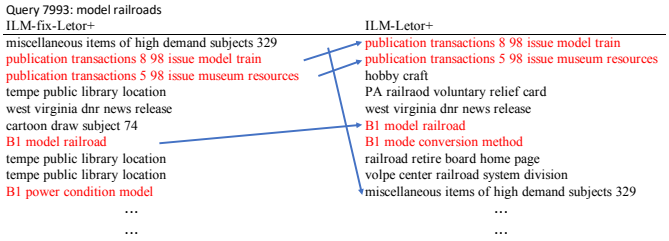

Figure 3: Rank List for Query 7993

The document titles of relevant documents (judgments ≥ 1) are marked in red, and non relevant documents (judgments $= 0$) are marked in black. By comparing the 2 ranking lists, we can observe that relevant documents are pushed upwards and non relevant documents are pushed downwards in the rank list produced by the model trained in an end-to-end manner. This shows that within the L2R framework, if we integrate the learning of the neural feature modules and the ranker in an end-to-end manner, relevant documents could be ranked further upwards, resulting in improved performance.

4.6 Discussion and Analysis

Focus of representations and interactions: To confirm the roles of representation- and interaction-based models, we extract some typical conceptual and lexical queries from Million Query and compare the performance of Rep-MLP, Inter-MLP and HM-MLP on NDCG@10 in Table 5. For conceptual queries (first part), which expect some degree of generalization or expansion from the term space, Rep-MLP outperforms Inter-MLP. For lexical queries (second part) about some people and places, which require exact term match, Inter-MLP outperforms Rep-MLP.

By combing the representations and interactions, the model HM-MLP outperforms both Rep-MLP and Inter-MLP for both lexical and

[3] https://nlp.stanford.edu/projects/glove/

Table 2: Experimental Results on MQ datasets. Statistical significance (p<0.05) with respect to BM25, LM and RankMLP-Letor is marked with a, b and c. * indicates statistical significance (p<0.05) with Bonferroni correction with respect to the 4 baselines.

	MQ2007					MQ2008				
Models	MAP	NDCG@1	NDCG@3	NDCG@10	NDCG@20	MAP	NDCG@1	NDCG@3	NDCG@10	NDCG@20
BM25	0.4584	0.4470	0.4489	0.5035	0.5765	0.4688	0.4853	0.5570	0.6832	0.7144
LM	0.4490	0.4216	0.4401	0.4819	0.5625	0.4569	0.4538	0.5352	0.6673	0.6997
RankMLP-Letor	0.4713	0.4825	0.4861	0.5298	0.5974	0.4789	0.5297	0.5781	0.7033	0.7307
RankMLP-Letor+	0.4748	0.4978	0.4911	0.5328	0.6013	0.4882	0.5589	0.5960	0.7131	0.7438
ILM-BM25	0.4918_a	0.5084_a	0.5077_a	0.5521_a	0.6208_a	0.5069_a	0.5713_a	0.6299_a	0.7312_a	0.7591_a
ILM-LM	0.4830_b	0.4812_b	0.4921_b	0.5374_b	0.6097_b	0.5022	0.5766_b	0.6164_b	0.7213_b	0.7506_b
ILM-Letor	0.4901_c	0.5338_c	0.5232_c	0.5538_c	0.6239_c	0.5112_c	$\mathbf{0.6035}_c$	$\mathbf{0.6400}_c$	0.7362_c	0.7645_c
ILM-Letor+	$\mathbf{0.4987}_*$	$\mathbf{0.5415}_*$	$\mathbf{0.5303}_*$	$\mathbf{0.5682}_*$	$\mathbf{0.6317}_*$	$\mathbf{0.5160}_*$	0.5984	0.6393_*	$\mathbf{0.7418}_*$	$\mathbf{0.7679}_*$

Table 3: Comparison of Integrated and Separate Learning of the Ranker and Features. Statistical significance (p<0.05) with respect to the ILM-fix-* counterparts is marked with e and f.

	MQ2007					MQ2008				
Models	MAP	NDCG@1	NDCG@3	NDCG@10	NDCG@20	MAP	NDCG@1	NDCG@3	NDCG@10	NDCG@20
ILM-fix-Letor	0.4659	0.4652	0.4761	0.5218	0.5916	0.4863	0.5531	0.5920	0.7103	0.7390
ILM-fix-Letor+	0.4694	0.4699	0.4785	0.5244	0.5914	0.4881	0.5548	0.6032	0.7162	0.7455
ILM-Letor	0.4901_e	0.5338_e	0.5232_e	0.5538_e	0.6239_e	0.5112_e	$\mathbf{0.6035}_e$	$\mathbf{0.6400}_e$	0.7362_e	0.7645_e
ILM-Letor+	$\mathbf{0.4987}_f$	$\mathbf{0.5415}_f$	$\mathbf{0.5303}_f$	$\mathbf{0.5682}_f$	$\mathbf{0.6317}_f$	$\mathbf{0.5160}_f$	0.5984	0.6393_f	$\mathbf{0.7418}_f$	$\mathbf{0.7679}_f$

Table 4: Experimental Results of Alternative Configurations. * indicates statistical significance (p<0.05) with Bonferroni correction with respect to the 4 baselines in Table 2.

	MQ2007					MQ2008				
Models	MAP	NDCG@1	NDCG@3	NDCG@10	NDCG@20	MAP	NDCG@1	NDCG@3	NDCG@10	NDCG@20
Rep-MLP	0.4199_*	0.3636_*	0.3865_*	0.4510_*	0.5321_*	0.4085_*	0.3804	0.4485_*	0.6093_*	0.6467
Inter-MLP	0.4156_*	0.3673_*	0.3785_*	0.4444_*	0.5287_*	0.4115_*	0.3886	0.4583_*	0.6095_*	0.6498_*
HM-sum	0.4169_*	0.3590_*	0.3789_*	0.4483_*	0.5308_*	0.4222_*	0.4275	0.4797_*	0.6258_*	0.6630_*
HM-MLP	0.4245_*	0.3832_*	0.3969_*	0.4608_*	0.5374_*	0.4280	0.4319	0.4977_*	0.6350_*	0.6707_*
ILM-Neu	0.4313	0.4057	0.4137_*	0.4652_*	0.5466_*	0.4593	0.5050	0.5537_*	0.6728_*	0.7075_*
ILM-Hist	$\mathbf{0.4884}_*$	$\mathbf{0.5218}_*$	$\mathbf{0.5167}_*$	$\mathbf{0.5527}_*$	$\mathbf{0.6185}_*$	$\mathbf{0.5192}_*$	$\mathbf{0.6037}_*$	$\mathbf{0.6511}_*$	**0.7481**	$\mathbf{0.7741}_*$

Table 5: NDCG@10 of Representative Queries

topic_num	Query	Rep-MLP	Inter-MLP	HM-MLP
9394	preventing alcoholism	0.75297	0.61683	**0.85124**
9963	small business statistics	0.11068	0.04793	**0.40252**
8023	voyager 2 Neptune	0.4807	0.65641	**0.73464**
8068	distance between grand canyon and phoenix	0.15508	0.20615	**0.22341**

conceptual queries. This result demonstrates the complementarity between the two types of features.

The general performance of the three models are presented in Table 4. We can see that the model using only one type of feature can yield equivalent performance. When both types of features are combined in HM-MLP, we obtain better results. This confirms again the two types of features are complementary.

Usefulness of traditional features: Several previous experiments have shown that neural models often have difficulty to match the performance of traditional models (e.g. [14]). Can a neural model combining both representation and interaction features be competitive to traditional models? To answer this question, we can compare ILM-Neu (which only has neural features) in Table 4 with BM25 and LM. We observe that ILM-Neu still cannot achieve competitive performance against BM25 and LM. Therefore, in the current context, it is useful to incorporate traditional features into a neural model. Traditional features may capture some relevance patterns which might be complementary to neural features.

Effectiveness of LambdRank vs. Hinge Loss: ILM-Neu and HM-MLP are two similar models that use the same neural features, but the former uses LambdaRank to optimize while the latter uses hinge loss. From Table 4, we can observe that ILM-Neu outperforms HM-MLP on all evaluation metrics on both datasets. This comparison demonstrates the benefits of employing LambdaRank

framework over the traditional hinge loss training approach. This confirms the result in [25].

Combining features vs. combining scores: One of our initial intuitions is that it is better to combine representation and interaction features than combining the scores they produce. This can be confirmed by comparing HM-MLP and HM-sum (equivalent to the Duet model) in Table 4: It is clear that combining features is better than combining scores. This suggests that MLP can indeed make better use of the features when they are presented together, and this allows them to have possible interactions.

Flexibility of ILM: Our ILM is also general, it is possible to replace the representation/interaction module with different ones. For interaction module, we replace the pyramid-based one with the histogram-based one depicted in Fig. 2 and build ILM-Hist. Experimental results in Table 4 show that it could still outperform traditional baselines and offer comparable performance with respect to the original ILM.

Learning curve: Finally, we plot the NDCG@10 curve of ILM-Letor+ on the validation set of fold 1 of MQ2007 in Fig. 4, together with BM25, LM for comparison. We can observe that at the beginning of the training, the NDCG value first increases. As training goes on, the performance reaches the maximum and begins to decrease. This could be possibly due to overfitting or to the lack of sufficient training data. The curve shows an area in the middle where our proposed model can outperform traditional models.

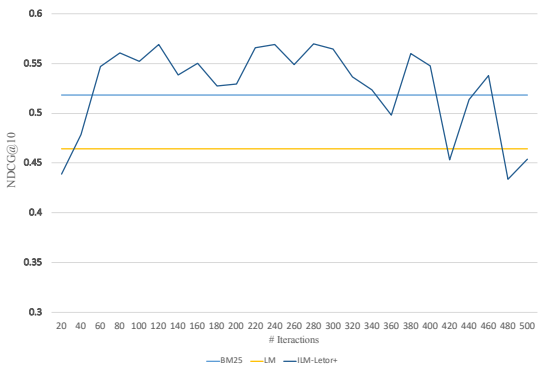

Figure 4: Learning curve of nDCG@10 on Validation data on MQ2007

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed an integrated learning framework to integrate both representation- and interaction-based features. In addition, we also integrate feature learning with ranking-function learning. Experiments on public datasets confirm the effectiveness of integrated learning of ranking features of different nature and ranking function.

In this study, the interactions between different matching mechanisms happen on the feature level. In future studies, we plan to explore their interactions on intermediate layers and test the ILM model on large-scale datasets such as MSMARCO.

REFERENCES

[1] Qingyao Ai, Xuanhui Wang, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. Learning Groupwise Scoring Functions Using Deep Neural Networks. *CoRR* abs/1811.04415 (2018). arXiv:1811.04415

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

[3] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling Diverse Relevance Patterns in Ad-hoc Retrieval. In *SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. 375–384.

[4] Jian Gu, Guang-Hua Yin, Pengfei Huang, Jinlu Guo, and Lijun Chen. 2017. An improved back propagation neural network prediction model for subsurface drip irrigation system. *Computers & Electrical Engineering* 60 (2017), 58–65.

[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. 55–64.

[6] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2042–2050.

[7] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 2333–2338.

[8] Robert Krovetz. 2000. Viewing morphology as an inference process. *Artif. Intell.* 118, 1-2 (2000), 277–294.

[9] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331. https://doi.org/10.1561/1500000016

[10] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer. https://doi.org/10.1007/978-3-642-14267-3

[11] Bhaskar Mitra and Nick Craswell. 2019. An Updated Duet Model for Passage Re-ranking. *CoRR* abs/1903.07666 (2019).

[12] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *WWW 2017, Perth, Australia, April 3-7, 2017*. 1291–1299.

[13] Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. 2018. Multi-level Abstraction Convolutional Model with Weak Supervision for Information Retrieval. In *SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. 985–988.

[14] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. *CoRR* abs/1606.04648 (2016).

[15] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *AAAI 2016, February 12-17, 2016, Phoenix, Arizona, USA*. 2793–2799.

[16] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *CIKM 2017, Singapore, November 06 - 10, 2017*. 257–266.

[17] Rama Kumar Pasumarthi, Xuanhui Wang, Cheng Li, Sebastian Bruch, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2018. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. *CoRR* abs/1812.00073 (2018). arXiv:1812.00073

[18] Stephen E. Robertson. 2000. Evaluation in Information Retrieval. In *ESSIR 2000, Varenna, Italy, September 11-15, 2000, Revised Lectures*. 81–92.

[19] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[20] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *SIGIR 2015, Santiago, Chile, August 9-13, 2015*. 373–382.

[21] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW'14, Seoul, Republic of Korea, April 7-11, 2014*. 373–374.

[22] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *NIPS 2013, Lake Tahoe, Nevada, United States*. 926–934.

[23] Ming Tan, Tian Xia, Lily Guo, and Shaojun Wang. 2013. Direct optimization of ranking measures for learning to rank models. In *SIGKDD 2013, Chicago, IL, USA, August 11-14, 2013*. 856–864.

[24] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-SRNN: Modeling the Recursive Matching Structure with Spatial RNN. In *IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 2922–2928.

[25] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *CIKM 2018, Torino, Italy, October 22-26, 2018*. 1313–1322.

[26] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A Theoretical Analysis of NDCG Type Ranking Measures. In *COLT 2013, Princeton University, NJ, USA*. 25–54.