IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 2, 2019, accepted February 19, 2019, date of publication April 26, 2019, date of current version May 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902640

Deep Neural Network Regularization for Feature Selection in Learning-to-Rank

ASHWINI RAHANGDALE AND SHITAL RAUT
Department of Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur 440010, India
Corresponding author: Ashwini Rahangdale (rahangdaleashwini@gmail.com)

ABSTRACT Learning-to-rank is an emerging area of research for a wide range of applications. Many algorithms are devised to tackle the problem of learning-to-rank. However, very few existing algorithms deal with deep learning. Previous research depicts that deep learning makes significant improvements in a variety of applications. The proposed model makes use of the deep neural network for learning-to-rank for document retrieval. It employs a regularization technique particularly suited for the deep neural network to improve the results significantly. The main aim of regularization is optimizing the weight of neural network, selecting the relevant features with active neurons at the input layer, and pruning of the network by selecting only active neurons at hidden layer while learning. Specifically, we use group ℓ_1 regularization in order to induce the group level sparsity on the network's connections. Set of outgoing weights from each hidden layer represents the group here. The sparsity of network is measured by the sparsity ratio and it is compared with learning-to-rank models, which adopt the embedded method for feature selection. An extensive experimental evaluation considers the performance of the extended ℓ_1 regularization technique against classical regularization techniques. The empirical results confirm that sparse group ℓ_1 regularization is able to achieve competitive performance while simultaneously making the network compact with less number of input features. The model is analyzed with respect to evaluating measures, such as prediction accuracy, *NDCG@n*, *MAP*, and *Precision* on benchmark datasets, which demonstrate improved results over other state-of-the-art methods.

INDEX TERMS Deep neural network, feature selection, information retrieval, learning-to-rank, regularization.

I. INTRODUCTION

Since the last decade, machine learning has apparently been used in a wide range of applications, including ranking. Machine learning framework used for ranking is known as learning-to-rank. Learning-to-rank framework has first originated for information retrieval (IR) system yet these days it is used for wide range of applications like recommendation system, collaborative filtering, spam detection, bioinformatics, etc. Learning-to-rank framework specifically uses the supervised machine learning algorithms and discovers the best order of a list according to their preferences, score or rank [1]. Learning-to-rank does not look at the absolute score for each item however it is involved with one item being ranked above or below the another. Algorithms devised under learning-to-rank designed with discriminative training are categorized into three main approaches, e.g. pointwise [2]–[5], pairwise [6], [9], [10], [12], [16] and listwise [11], [13]–[15] approach. These approaches have different input space, output space, hypothesis and loss function. Learning-to-rank provides optimal way for learning by combining the items associated with features through discriminative training [20].

In recent trends, number of ranking models are build using deep neural network. Deep neural network [21] has the ability of automatic learning by using the available features. It has achieved great success for many machine learning applications such as computer vision, natural language processing, speech recognition, etc. New direction of the neural network for ranking applications uses deep learning to tackle the problem of learning-to-rank by using the extracted features of objects. The concept of deep learning for learning-to-rank was first proposed by Cherif *et al.* [22] for preference learning. This model is based on the SortNet algorithm [12] using the CmpNN (comparative neural network) model. Pang *et al.* [23] proposed DeepRank, a new architecture for

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa Rahimi Azghadi.

relevance ranking to the IR. They consider automatically retrieved features from raw data to learn the model. Song [24] proposed the ConvRankNet that combines a Siamese convolutional neural network encoder and RankNet [9] ranking model which could be trained in end-to-end fashion. There are also some other pioneer works like, *DSSM* [26] and *CDSSM* [27] that use the semantic information for document retrieval. They implement deep neural network model based on the semantic features of query-document pair and compute the score based on their cosine-similarity. However, it ignores relevance ranking between the documents. This issue of relevance ranking is solved by another deep learning architecture *DRMM* [28] that considers both ranking criteria and relevance ranking between the documents.

The above learning-to-rank models make learning feasible using the deep neural network. It leads to remarkable accuracies for high dimensional datasets. Deep neural network directly encodes the feature vectors of the retrieved documents to learn context embedding and use it to improve the learning-to-rank systems. However, it involves the large number of weights while learning. The majority of weights in most deep network are not necessary to its accuracy. The model can learn on the small percentage of weights without compromising its accuracy. There are number of methods proposed to optimize the weights (number of weights). Most of these either require strong assumptions on the connectivity or they require multiple, separate training steps. Pruning method can also be adopted by setting the weights to zero based on a fixed threshold, and then fine-tuning the remaining connections with a second training step. Feature selection is an additional problem in deep learning when considering high dimensional datasets. The learning model with large number features may leads to low accuracy, high computational complexity and overfitting.

In deep neural network, feature selection and network pruning are interrelated problems [39]. The learning model with high dimensional features generally results in need of growing the network capacity in terms of number of neurons and it may add up-to the last hidden layer. In neural network, input layer is considered as an additional layer, having only outgoing connection to the first hidden layer. However, it does not have the incoming connections. Thus, pruning of neurons from initial layer are same as deleting the corresponding irrelevant feature from input layer.

Considerable work has been done on feature selection for learning-to-rank. The number of learning-to-rank models are proposed with different feature selection methods, viz. filter method, wrapper method, embedded method. The filter method is adopted by [29]–[32]. This method is not based on machine learning algorithms, and use general techniques such as correlation of features for prediction. In the wrapper method [33], [34], feature space is partitioned into subsets and then combines these subsets of feature to maximize ranking metric. These methods use a trained model and test the samples in order to derive feature importance from performances regardless the type of machine learning algorithm used. In embedded approach, feature selection process integrates with the learning algorithms. These algorithms are also known as the feature sparse algorithm for feature selection [35]–[37]. There are very few ranking models that implement the embedded method for feature selection. In the proposed model, we aim at improving the performances of deep neural network based ranking model by reducing number of feature, having a better understanding of the underlying distribution and avoid overfitting of network.

Sparse regularized network for learning-to-rank is one of the embedded method for feature selection. It exploits correlations among the features to enforce network to be sparse. At the same time, it also removes feature redundancies to fully utilize the network capacity [40]. Principal way to achieve this objective is ℓ_1 regularization wherein loss function is optimized with a penalty term obtained by aggregating absolute value of weights during training [42]. It is an indirect way to solve the problem of network pruning with feature selection. Here, low-rank (a low absolute value for weight) neuron with all its incoming and outgoing connection are set to zero and it does not participate in further learning. However, this is a highly sub-optimal solution to an equally sparse network. Hence, we prefer group level sparsity that gives more structured level sparsity and keeps smaller number of neurons per layer. The group level sparsity imposes sparsity such that all variables within a group are either simultaneously *zero* or none. Thus, we achieve our objectives of optimizing weight of neural network and selecting the active neurons at input layer by using ℓ_1 regularization technique.

PAPER CONTRIBUTION

We have adopted deep neural network architecture for learning-to-rank. Further, network architecture optimizes with different regularization techniques. We have put more emphasis on ℓ_1 regularization technique for feature selection. The main aim of regularization is to induce sparsity and to avoid overfitting of deep neural network [38]. As an additional variety, we have adopted *group* ℓ_1 and *sparse group* ℓ_1 regularization to speed up the learning and improve result significantly. The *sparse group* ℓ_1 is mainly used to induce sparsity on non-sparse group. The group of features can be formed based on all outgoing connections of the neuron. In this way, optimization of deep neural network is forced to remove the low-rank neurons at learning time.

In this paper, *sparse group* ℓ_1 regularization is used as a comprehensive tool to impose more substantial network with subset of most relevant features. The empirical analysis on the benchmark datasets shows better results with sparse group weight penalty term. It also gives better performance in comparison to other baseline algorithms against IR evaluating measures such as *NDCG@n*, *P@n* and *MAP*. The sparsity of the network has evaluated in terms of sparsity ratio and compared with baselines methods which have been published with respect to feature selection for learning-to-rank.

In short, the major contribution of paper are as follows:

1) We implement the dense neural network architecture for application of learning-to-rank. It automatically selects highly informative features while learning. This model adopts the embedded method of feature selection which also avoids overfitting.
2) The listwise approach is considered to build the learning-to-rank system using the deep neural network. The ListNet algorithm is followed for implementation. Regularized network with ListNet, first applies the gradient on loss function and then it optimizes the regularization term while performing projection of it to the solution space.
3) Experimental comparisons and analysis show that the best results are obtained with the sparse group ℓ_1 regularization where we can obtain comparable accuracies to ℓ_2-regularized and ℓ_1-regularized networks.

ORGANIZATION OF PAPER

This paper is organized as follows: Section **II** summaries the problem formulation and different notions of proposed model. Section **III** describes the supporting background for architecture and regularization techniques used to optimize the learning model. We formulate the optimization problem in Section **IV**. It briefly discusses the novel *sparse group* ℓ_1 weight penalty terms to show the meaning of group in this context. Section **V** fully describes about datasets, software tools, evaluating measures and experimental set-up. In Section **VI**, we first analyze the ability of our approach to induce sparsity into the learning model. The results are also compared with other baselines in terms of IR evaluating measures *NDCG@n* and *MAP*. We conclude our paper with final remarks in Section **VII**.

II. BACKGROUND

A. PROBLEM FORMULATION AND NOTATION

We have considered learning-to-rank problem in the context of IR in which ranking of documents associated with different queries must be optimized. Let $Q = \{q_i\}_{i=1,2...,m}$ be the total number of queries and $D = \{d_j^i\}_{j=1,2...,N}$ be the total number of documents associated with each query q_i. Furthermore, every query-document pair has a label $y_j^{(i)}$ that represents the relevance judgment of document j towards query q_i. Let $L = \{l_1, l_2, \ldots l_k\}$ be the set of all possible degree of labels and k is the finite number of relevance judgment. There exists order between the relevance judgment such as $\{l_1 \succ l_2 \succ \cdots \succ l_k\}$ where $\succ$ represents the preference relationship between labels of object. In learning-to-rank model, every query-document pair is categorized by the feature vector $\psi(q_i, d_j^i) \in \mathbb{R}^{\mathbb{F}}$ where $\mathbb{F}$ is total number of features considered for learning. $W = \{w_i\}_{i=1,2,...\mathbb{F}}$ be weight of the i^{th} corresponding feature. Specifically, learning algorithms are used to adjust the weights of the features in each iteration and these weighted features are used to predict the score for query-document pair. All symbols and notations used in this paper are explained in Table 1.

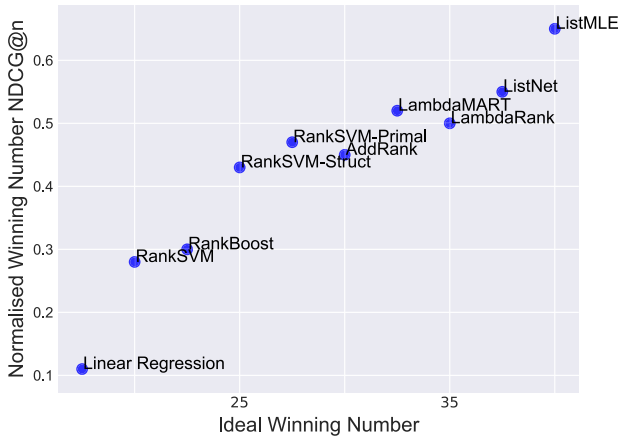
TABLE 1. Notations and definitions.

Notations	Explanation
$Q = \{q_i\}_{i=1}^{m}$	query set
$D = \{d_j\}_{j=1}^{N}$	document set associated with q_i
m	total number of query
N	total number of documents associated with query q_i
y_j	relevance label associated with document d_j
$L = \{l\}_{i=1}^{k}$	set of all possible degree of labels
$\succ, \prec$	preference relationship between label of objects
$l_1 \prec l_2 \prec \cdots \prec l_k$	label order
$\psi(q_i, d_j^i)$	feature vector for query document pair
W	weight matrix
w	weight vector
s_j	score of document d_j
$f(x, w)$	hypothesis ranking function
$P(y_i)$	probability distribution for ground truth label
$P(f(x_i, w))$	probability distribution for predicted rank list
Π	permuted list of the documents
$\mathcal{L}(.)$	loss function
$\mathcal{H}$	total number of hidden layer
$\sigma(.)$	activation function
b	bias
λ	regularization factor
$\mathcal{G}_{in}$	sparse group at input layer
$\mathcal{G}_h$	sparse group at hidden layer
$\mathcal{G}$	total number of sparse groups

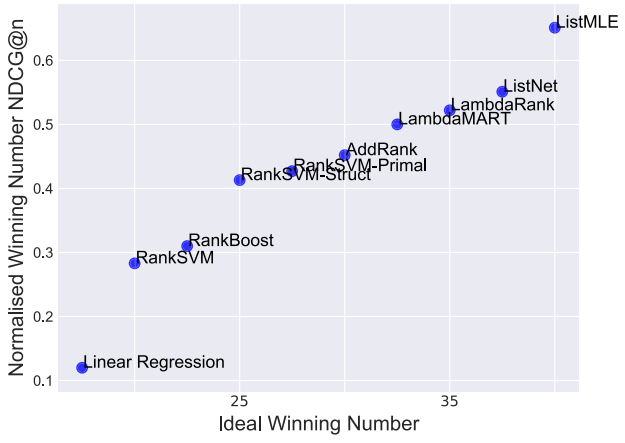
B. LEARNING TO RANK MODELS

Learning-to-rank model refers the group of machine learning techniques such as linear regression [2], SVM [6]–[8], boosting [16]–[18], neural network [9]–[15], genetic programming [19] that attempt to perform ranking with using feature representation of query-document pairs. On comparing the performance between these algorithms, it is proved that listwise approach is a better approach for solving the ranking problem. Figure 1 depicts the achievements of listwise approach using neural network methods over other machine learning methods used for application of learning-to-rank [43]. The neural network based learning models are the RankNet [9], LambdaRank [11], ListNet [13], ListMLE [14], SortNet [12], SoftRank [15], etc. These algorithms give effective performance on each benchmark datasets which is measured in terms of winning number for different evaluating measures. ListNet [13] algorithm wins for maximum number of times and gives better performance as compared to other state-of-the-arts.

Inspired by the success for various applications, deep learning has been applied to learning-to-rank, where learning is performed using effective ranking features. The first deep learning-to-rank model [22] is build on CmpNN [12]. Further, Wang and Klabjan [46] propose deep neural network model which incorporates different embedding of the queries and search results using an attention-based mechanism. A convolutional neural network architecture for re-ranking pairs of short text uses the intrinsic feature extraction technique for optimal representation of a pair and map them in a similarity function using the supervised learning [25]. Furthermore, couple of deep model have been proposed for IR which considered the semantic features for a query-document pair. It includes *DSSM* [26], *CDSSM* [27] and *DRMM* [28]. *DSSM* [26] is the deep structured semantic model trained by maximizing the conditional likelihood



(a)



(b)

FIGURE 1. Winning Number for baseline learning-to-Rank. *Winning Number* is count that gives how many times an algorithm beasts over the set of data sets. A normalized version of the *Winning Number* metric is used to compare algorithms based on a sparse set of evaluation measurements. (a) Winning Number for NDCG@n. (b) Winning Number for MAP.

of documents for given query through click-through data. It implements the word hashing technique to handle the large vocabularies for large scale web data. *CDSSM* [27] uses the convolutional neural network instead of regular deep neural network to preserve the local order information with respect to semantic information from the query-document pair. It uses the max-pooling layer to extract the salient features from global feature vector. *DSSM* and *CDSSM* consider only the semantic features for IR, however it ignore the importance of intrinsic relevance ranking. Guo *et al.* [28] proposed a new deep learning architecture *DRMM*. This model shows the important characteristics of IR such as exact matching signals, query terms importance, diverse matching requirement, etc. Recently, Pang *et al.* demonstrated a new deep learning architecture for IR, named as DeepRank [23] to simulate the human judgment process which is used to get the local relevance label. The deep neural network is also used to optimizing the IR measures [60] which is referred direct optimization of loss function [15], [18]. One of the deep neural network models performs the ranking using weak supervision signal like BM25. It trains the ranking model effectively based on feed-forward neural networks and investigates its effectiveness on the different learning approaches (pointwise, pairwise) using different input representations [61]. Recently, Ai *et al.* [62] propose to use the inherent feature distributions of the top results to learn a Deep Listwise Context Model that helps to fine tune initial ranked list. It employs the recurrent neural network to sequentially encode the top results using a feature vector, learn the local context model and use it to rerank top results.

The primary objective of above deep neural network model is to apply deep learning to achieve the task of ranking. The inputs to these models are hand-crafted features provide in benchmark datasets [52], [53]. The output of these models is relevance ranking. These models attempt to learn the mapping from provided features to the relevance level. Thus, it involves the large number of complex interaction between the features and difficulty in interpretation for high dimensional features. On the other hand, large number of weights are involved in learning even if all are not necessary to the learning accuracy. It takes a lot of efforts to devise and manage new strategies to significantly boost the training process for ranking. The deep learning model with high dimensional learning parameters reduces the accuracy, leads to overfitting and increase the computational complexity. Thus, training deep neural networks is known to be a difficult task.

Network embedded with the classical regularization techniques such as ℓ_1, ℓ_2 regularization are mainly used to reduce the overfitting and reducing the complexity of deep neural network. The ℓ_1 regularization is also performs the network pruning, however it not an efficient solution for the compact network. Thus, we apply strategies that are best suited for deep neural network to learning-to-rank. The main objective of this paper is to implement the ranking model with sparse regularized deep neural network with significantly less number of feature variables than the original dense network and obtain the better performance to original model.

III. DEEP LEARNING WITH ListNet

A. ARCHITECTURE

The traditional model for deep learning for the application of IR considers the dense network where the number of connections of each node is close to the maximum number of nodes. Traditional dense architecture for deep neural network for relevance ranking is demonstrated in Figure 3. The main objective of this paper is to reduce the density and complexity of network. The input to these models are feature of object and output is their relevance ranking. We first implement the deep neural network architecture for application of learning-to-rank. Modern neural network consist $\mathcal{H}$ number of hidden layers, where each hidden layer $p \in \{1, 2 \ldots \mathcal{H}\}$ has

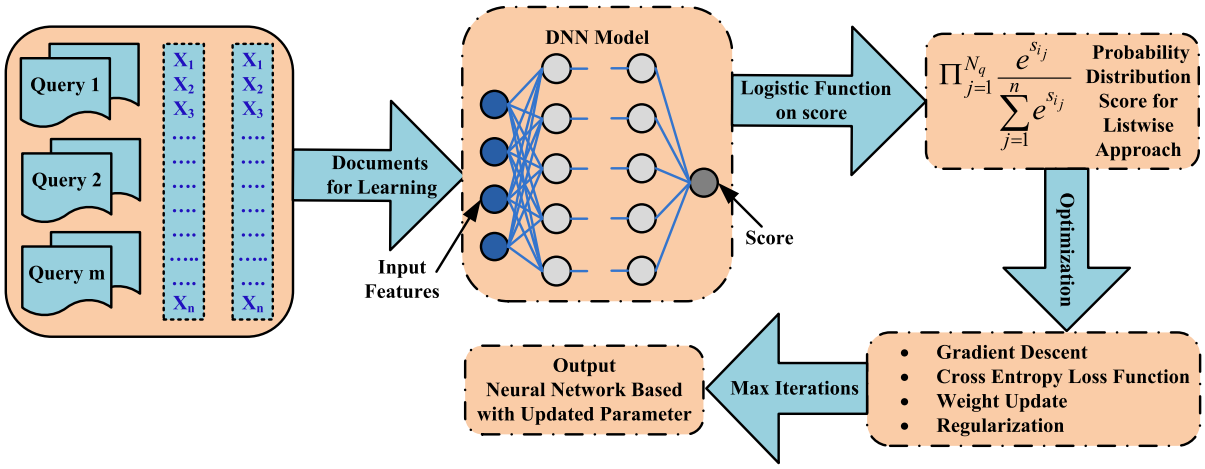


FIGURE 2. Demonstration of steps of proposed model using deep neural network for application of learning-to-rank.

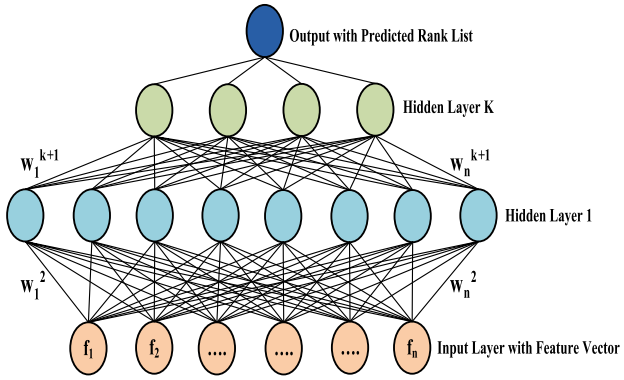


FIGURE 3. Dense neural network architecture for learning-to-rank.

decreasing number of neurons in the subsequent hidden layers. w_k^i denotes weight vector for the k^{th} node of the i^{th} layer. For the input layer, value of i is set to 1. The bias (b) is associated with each layer of the network.

The generic neural network model $y = f(x, w)$ takes the input vector $x \in \mathbb{R}^{\mathbb{F}}$ and gives output in the form of the scoring value of the object after propagating through $\mathcal{H}$ hidden layers. At input layer, weight vector $w \in \mathbb{R}^{\mathbb{F}}$ is used as shorthand for the feature-vector concatenation for adaptable objects. Each hidden layer takes the input from the previous hidden layer and generates the output using activation function:

$$h_p = \sigma\big(\sum W_p h_{p-1} + b_P\big), \tag{1}$$

where σ is properly chosen activation function applied to the network and W is the weight matrix or (tensor) at each layer. The learning method follows the major steps of ListNet algorithm [13] that includes the optimization of cross entropy loss function using the gradient descent. In this learning algorithm, network is trained with the stochastic gradient and determines rank using the probability distribution. This ranking model estimates the score based on the number of feature value associated with each document. This score is used to predict the rank of the documents for a query q_i. Moreover, loss function is defined using the probability distribution of predicted rank list and ground truth ranked list. This probability distribution for a pair of object x_i and x_j is given as:

$$P_{ij} = P(x_i \succ x_j) = \frac{exp(s_i - s_j)}{1 + exp(s_i - s_j)}, \tag{2}$$

where P_{ij} is the probability of the item x_i preferred over the item x_j, s_i and s_j representing respective scores of objects. The probability distribution for listwise approach is given as:

$$P(x_{i_1}, x_{i_2}, x_{i_3} \ldots, x_{i_n}) = \prod_{j=1}^{N_q} \frac{exp(s_{i_j})}{\sum_{j=1}^{n} exp(s_{i_j})}. \tag{3}$$

We optimize this loss function for deep neural network by using regularization techniques. The preferred architecture for deep neural network for application of learning-to-rank is illustrated in Figure 2.

For the generic training set on n examples given by $\{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$. The network is trained by minimizing the loss function defined for learning-to-rank:

$$\mathcal{L}(w) = argmin_w \left\{ \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f(x_i, w)) \right\}. \tag{4}$$

The loss function $\mathcal{L}(.)$ at every instance is given as:

$$\mathcal{L}(y_i, f(x_i, w)) = -\sum_{i=1}^{Q} P_{y_i}(\Pi) log P_{f(x_i,w)}(\Pi), \tag{5}$$

where P_{y_i} is the probability distribution with respect to ground truth whereas $P_{f(x_i,w)}$ is probability distribution with respect to predicted rank list Π. From (3) and (4),

we get:

$$\mathcal{L}(y_i, f(x_i, w)) = -\sum_{i=1}^{N_q}\prod_{j=1}^{N_q}\frac{exp(y_{i_j})}{\sum_{j=1}^{n_i} exp(y_{i_j})} \times log\prod_{j=1}^{N_q}\frac{exp(f(x_{i_j}), w)}{\sum_{j=1}^{n_i} exp(f(x_{i_j}, w))}. \quad (6)$$

This neural network employs gradient descendant to perform optimization:

$$\Delta(w) = \frac{\delta\mathcal{L}(y_i, f(x_i, w))}{\delta w}, \quad where \quad (7)$$

$$\delta\mathcal{L}(y_i, f(x_i, w)) = -P\sum_{j=1}^{N_q} P_{y_i}\frac{\delta f(x_i, w)}{\delta w} + \frac{1}{\sum_{j=1}^{N_q} exp(f(x_i, w))} \times \sum_{j=1}^{N_q} exp(f(x_i, w))\frac{\delta f(x_i)}{\delta w}. \quad (8)$$

Algorithm 1 shows that network is learned via backpropagation to determine the parameter values which optimizes the loss function. However, this dense network penalizes the algorithm with large computational cost and the model is overfitted because of the large number of parameters involved in learning. Hence, we adopt manifold learning to sparsify the deep neural network. It helps to reduce the computational cost and gives better performance. The standard methods used for regularizing deep neural network are listed in following subsections.

Algorithm 1 ListNet Using Deep Neural Network

Input: Training Data in the form of a query-docuemnt pair $(x^1, y^1), (x^2, y^2) \ldots (x^n, y^n)$, where x represents feature vector of document and y is the relevance label.
Output: Deep Neural Network model with w and b.
Initialization:random initialization of weight,
Batchsize $= m$, learning rate $= \eta$, iterations $= T$;
for $i \leftarrow 1$ ***to*** T **do**
  Samples of a mini batch size containing from $(x^i, y^i)_{i=1}^m$;
  $(y^i, h) \leftarrow$ FeedForwardNetwork (x^i, y^i, w, b);
  Compute the score s_i with respect to output of deep neural network;
  Compute the gradient using (8);
  $w^k = w^k - \eta\Delta w$ # *update weights*;
end

B. REGULARIZATION

Regularization in neural network makes a modification in learning algorithms to reduce the generalization (testing) error however to not training error. Basically, it is intended for linear learning model like linear regression, that permits simple, clear and convincing regularization methods. With regard to deep learning, the favored regularization methodology is based on regularizing estimators. The effective regularizer is one that gives a profitable trade, reducing error considerably by not excessively increasing the bias [47]. The optimal loss using regularization is given by:

$$\mathcal{L}(f(x, W), y) + \lambda\Omega(W), \quad (9)$$

where λ is the regularization factor that reveals the relative contribution of penalty term Ω to optimize the standard objective loss function corresponding to regularization norms. In this paper, we note the regularization behavior of different norms for deep neural network, particularly for the task of ranking.

In deep neural network, it is advisable to use separate penalty with different regularization factor λ for each layer, but it may take efforts to search for a correct value of multiple regularization factor [47]. Thus, it is desirable to use the same regularization factor at each layer.

The suitable loss function (4) imposes the weight level regularization and regularization factor. In learning-to-rank, choice of loss function depends on the different approaches like pointwise, pairwise and listwise. Generally, regression based learning-to-rank algorithms use MSE (mean square error) as loss function, pairwise approach considers cross entropy or hinge loss whereas listwise approach focuses on metric based loss function. The classical regularization techniques use for optimizing deep neural network are discussed below:

1) ℓ_2-REGULARIZATION

The ℓ_2-regularization is the most preferred choice in machine learning due to its ability to simplify the solutions. However, ℓ_2-regularization avoids overfitting by penalizing the squared magnitude of weight parameters to the loss function. The term λw_i is added to the original loss function for each weight vector w_i in the neural network architecture. ℓ_2 regularization is also known as weight decay in neural network [41], because in a gradient descent approach, it prevents the weight growing to large value and reduce it by factor proportional to their magnitude at every iteration. The ℓ_2 regularization penalty is given is as:

$$\Omega(W^{(p)})_{l_2} \equiv \| W \|_2^2. \quad (10)$$

2) ℓ_1-REGULARIZATION

ℓ_1 regularization is another option to penalize the size of model parameters:

$$\Omega(W^{(p)})_{\ell_1} = \| w \| = \sum_{k=1}^{m} | w_k | . \quad (11)$$

This regularization is the sum of absolute values of the individual feature added term $\lambda|w|$ to an objective loss function. As compared to ℓ_2 weight decay, ℓ_1-norm controls the

strength of regularization by scaling the penalty Ω using positive high-parameter λ. Moreover, ℓ_1-regularization has fascinating property to lead the weight vectors to become sparse during optimization. It is a powerful technique for the feature selection in the learning process. Input layer neurons with ℓ_1 regularization propagate a sparse subset of most relevant features and become nearly invariant to the noisy features. The ℓ_1 regularizer is particularly developed for convex loss function however for non-smooth loss function, ℓ_1-regularization is customized to:

$$\Omega(w)_{\ell_1} = \sum_{k=1}^{m} \sqrt{w_k^2 + \beta}, \tag{12}$$

where β is a small scalar factor to obtain a smooth problem. Sometimes mixture of ℓ_1 and ℓ_2 are used for regularization denoted as elastic penalization [45]. Exclusive sparsity regularization method (based on ℓ_1 and ℓ_2) also promotes for features selection based on their different weights and hence enforcing them to fit to disjoint sets of features [40].

Built-in property of feature selection is a useful property of the ℓ_1-norm, which the ℓ_2-norm lacks [48]. This result of the ℓ_1-norm tends to produce sparse values for irrelevant features. If the model has 100 features and only 20 of them have non-zero coefficients, then it is effective to remove the 80 features with zero coefficient. These features are not useful in predicting the target values and unnecessarily increase computational overhead. The ℓ_2-norm produces only non-sparse coefficients, so it does not owe property of feature selection. Hence, we consider the ℓ_1-norm that gives more sparsity and helps to reduce features in ranking benchmarks.

In neural network model, value of weights rarely become zero. Thus, low-rank weight is known as sparse weight or sparse value. Sparsity of network connections can be seen as aggregation of sparsity of weights because it is equivalent to a fully connected network with zero weights in most places. Sparsity of connections reduces the computational cost by explicitly multiplying each input by zero and adding up all those zeros. This sparsity regularization is often learned in order to get the important feature at that instance.

IV. PROPOSED APPROACH OF SPARSITY CONTROL FOR DEEP NEURAL NETWORK WEIGHTS

Training of deep neural network is inherently challenging due to multiple hidden layers. The ℓ_1 regularization in (11) and ℓ_2 regularization in (10) are best to prevent the overfitting of the neural network, but they are not efficient solutions for obtaining compact neural network. This complication can be aggravated when the whole features are employed as input pattern. For the compact network, dead units can be removed only if all its connection have been zeroed out during training. However, this objective is hard to achieve while minimizing the cost function (4). For many local minima, some weight might be equivalent to its accuracy, corresponding to more compact and efficient network. Due to the difficulty of unstructured sparsity computation, the average percentage of zero weight is used as measure for group formation and then group ℓ_1 is applied as a structured sparsity regularization.

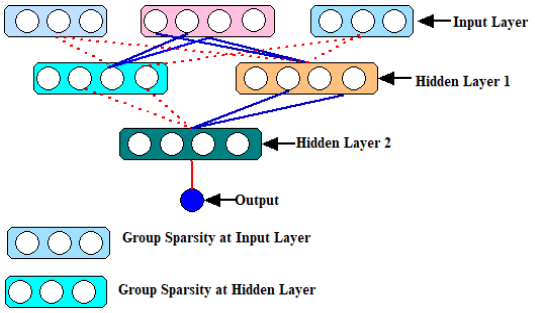


FIGURE 4. Group sparsity produced for deep neural network. The group is formed at each layer of deep neural network based on the sparse values. The group with light blue color represents the sparse group at initial layer and light green color represents the sparse group at hidden layers. These sparse groups do not propagate at the subsequent layer.

The concept of group sparsity is considered to explicitly control the degree of sparsity of the weights for each layer of the network. This regularization makes the parameters zero in some groups, while parameters in other groups are all non-zero. The group sparsity enforces all outgoing weights from single neuron to be either simultaneously zero or not [50]. In our approach, we not only use group ℓ_1 for network sparsity, but also introduces the exclusive ℓ_1 on each group at each layer. It claims that this exclusive ℓ_1 forces each group to use different inputs, thus the hidden layers will learn on the more meaningful features.

The group sparse regularization follow the procedure given by the Yuan and Lin [44] for selecting the grouped variables for accurate prediction. It is an extension of lasso for feature selection. In reducing the structured sparsity, weight tensor at each layer $W^{(p)}$ divided into the different groups. Each group represents the weight of vector greater than some absolute value of weight and applies group ℓ_1 to these groups as a regularization term in the loss function. The group ℓ_1 forces some groups of weights all zero, while other groups are all non-zero, thus the network is regularized to have structured sparsity. The group ℓ_1 regularization is given as:

$$\Omega(W^{(p)}) = \sum_{g} \parallel W_g^{(p)} \parallel_1 = \sum_{g} \sqrt{\sum_{i} |w|_{g,i}}, \tag{13}$$

where $g \in G$ is a weight group, $W^{(p)}$ is the weight a matrix or tensor for group g that is defined on layer p and $w_{g,i}$ is a weight at index i, for group g. For given weight tensor at each layer $\{W^{(1)}, W^{(2)}, \ldots W^{(\mathcal{H})}\}$, this group ℓ_1 regularization is defined as the solution to:

$$\frac{1}{2} \parallel \mathcal{L}(W, f(x)) \parallel^2 + \lambda \sum_{g} \parallel W_g^{(p)} \parallel_1 = \frac{1}{2} \parallel \mathcal{L}(W, f(x)) \parallel^2 + \lambda \sum_{g} \sqrt{\sum_{i} |w|_{g,i}} \tag{14}$$

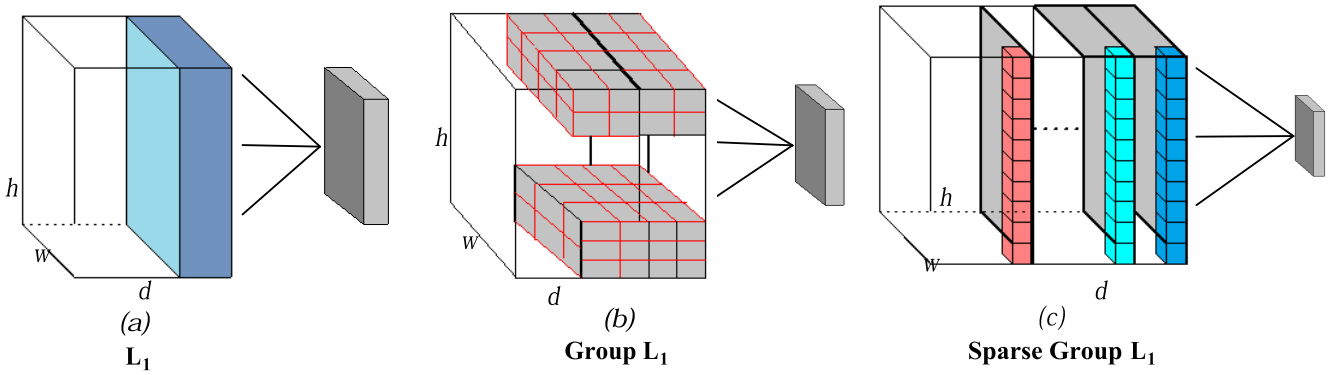


FIGURE 5. Illustration of effect of each regularization technique. (a) represents the regularization of deep neural network with ℓ_1; (b) group ℓ_1 regularization, more compact network obtained by removing group in each layer; (c) sparse-group level sparsity that gives the sparse value within-group and hence obtained more compact the network as compare to the group ℓ_1.

where λ is tuning parameter. In deep neural network, sparsity is produced at each layer. Hence, structure sparsity is constructed at each layer using sparse group. The corresponding groups at each layer are as:

- ***Input layer*** $\mathcal{G}_{in}$: $g_i \in \mathcal{G}_{in}$, where $i = 1, 2 \ldots d$, represents a vector of all outgoing connection from the i^{th} input layer neuron of the network. It is the transpose of the first row of matrix $W^{(1)}$.
- ***Hidden layer:*** In this case, individual element of $g \in \mathcal{G}_n$ represents weight vector of all outgoing connection from one of the neuron h_k of hidden layer of network. It is corresponds to the one row transposed of the weight matrix $W^{(k)}$, where $k > 1$. There are total $\sum_{k=2}^{\mathcal{H}+1}$ groups corresponding to the neuron of each hidden layer to the output layer.

We have intentionally ignored the sparse group sparsity at the bias level. The biases normally require fewer data than weight to fit accurately. Hence, we penalize only weights with suitable affine transformation at each layer and biases remain unregulated. Thus, we have total $\mathcal{G} = \sum_{k=1}^{\mathcal{H}+1} N_k$ groups corresponding to specific effects on resulting network. If weight of features of input layer group is zero (or less than the threshold), the corresponding feature cannot be selected during prediction. Moreover, if weight of variable in the hidden layer group is zero, the corresponding neurons can be removed, thereby achieving thinning of the hidden layer. In this way, *sparse group* ℓ_1 regularization effectively follows the feature selection procedure. Thus, total number of sparse groups obtained in model are:

$$\mathcal{G} = \mathcal{G}_{in} \cup \mathcal{G}_h. \tag{15}$$

The adopted sparse group regularization technique is demonstrated in Figure 4. It is a sparse network with one input layer and two hidden layers. Single output node is shown in dark blue at bottom of figure. Whereas neurons with light blue background are sparse features at the input layer and neurons with light green background are group of sparse weights at hidden layer. Sparse group regularization can be deduced by [44]:

$$\Omega(w_g) = \sum_{g \in \mathcal{G}} \sqrt{|g|} \parallel g \parallel_1, \tag{16}$$

where $|g|$ represents the group dimension. It is necessary that each group weighted uniformly.

For a single group (weight matrix with single dimension), the expression in (16) simplifies to the standard ℓ_1 regularization. Equation (16) might still be a sub-optimal solution for pruning of network, because there may be a possibility of sparsity at the group level even after removing some of sparse groups. This forces us to consider the following composite sparse group ℓ_1 (*SGL*) penalty i.e sparsity within the group [51] :

$$\Omega(w)_{SGL} = \Omega(w^g)_{(l)} + \Omega(w)_{\ell_1}. \tag{17}$$

SGL constitutes same properties as of the ℓ_1 and group ℓ_1 norms i.e convex but non-smooth and non-differentiable. Nevertheless, *SGL* outperform standard regularization techniques for the optimal value of regularization factor λ. Figure 5 illustrates the effect of ℓ_1, *group* ℓ_1 and *SGL* regularization on the network dimensions. In Figure 5(a), all dead unit are not removed from each layer thus network size is moderately reduced. In figure 5(b), only non-zero groups (the non-zero groups represented by grey color) are taking part in learning thus it reduces network size but not satisfactory. However, in 5(c) only non-dead units from each layer takes part in learning and hence sparse group ℓ_1 regularization effectively reduces the network size. Red, blue and green boxes in 5(c) represents the active units in their respective layers which are propagated for learning. The group sparse regularization force to remove the insignificant neurons from each layer and speed up the learning process by simplifying network.

A. NUMERICAL OPTIMIZATION

Regularized network can be achieved using proximal gradient descent, which is used for optimizing objectives formed as

a combination of both smooth and non-smooth terms [49]. First, it obtains the intermediate solution by applying gradient on loss function, $\mathcal{L}(.)$, and then it optimizes the regularization term while performing projection of it to the solution space.

$$\begin{aligned}\Delta w &= argmin_{\Omega(w)}\left\{\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y, f(x_i, w)) + \lambda\eta\Omega(w)_{SGL}\right\} \\ &= argmin_{\Omega(w)}\left\{\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y, f(x_i, w))\right) \\ &\quad + \lambda\eta\Omega(w^g)_{(l)} + \lambda\eta\Omega(w)_{\ell_1}\right\} \qquad (18)\end{aligned}$$

The combined regularizer can optimize each gradient step, after updating the variable with the loss based on gradient. Algorithm 2 describes the proximal gradient algorithm for optimizing our regularized objective.

Algorithm 2 Stochastic Proximal Gradient With Group Sparse Regularization

Input: Training Data in the form of a query-document pair $(x^1, y^1), (x^2, y^2) \ldots (x^n, y^n)$, where x represents feature vector of document and y is the label.
Output: Deep Neural Network model with w and b
Initialization: random initialization of weight, *Batchsize*, learning rate η, iterations T;
for $i \leftarrow 1$ ***to*** T **do**
  Sample a mini batch size containing from $(x^i, y^i)_{i=1}^m$;
  $(y^i, h) \leftarrow$ FeedForwardNetwork (x^i, y^i, w, b) ;
  compute the score s_i with respect to output of deep neural network;
  compute the gradient for each layer using (8);
  compute the loss with regularization (18) ;
  $w^k = w^k - \eta\Delta w$ # *update weights* ;
end

V. EXPERIMENTAL FRAMEWORK

In the ensuring sections, the number of computational experiments are conducted on different benchmark datasets dedicated to learning-to-rank to analyze the performance of proposed model. This section gives a full description of the datasets, software tool, different evaluating measures and experimental setup. The performance of proposed model is deeply analyzed and presented in subsequent section.

A. DATASET

In this paper, we focus on learning-to-rank with context to information retrieval. TERC[1] (Text Retrieval Conference) keeps the track of all scope of advancements in the IR and supports the research within the IR community. It also provides database required for large-scale evaluation of different text retrieval techniques. On their TREC web track, corresponding benchmark datasets are provided by the Qin *et al.* [53] and Liu *et al.* [52]. These are used to analyze the performance of state-of-the-arts designed under learning-to-rank.

LETOR[2] is the complete package that contains standard features, relevance judgment, partitioned data for cross validation and different evaluation tools. We conduct the experiments on these datasets. It includes total 10 datasets. Description and characteristic of all these datasets is summarized in Table 2. Each dataset is partitioned into five fold cross validation and every fold consists of the training, testing and validation sets. We use pyltr python[3] library to load data for experimentation. The MSLR-WEB10K and MSLR-WEB30K data is normalized using sklearn while LETOR datasets are already available in normalized form. In normalized data, every feature value is kept in range of [0, 1] with an affine transformation.

TABLE 2. Learning-to-rank datasets description.

Dataset	Benchmark	#Queries	Documents	#Relevance	#Features
TD2003	LETOR2.0	50	50k	2	44
TD2004	LETOR2.0	75	75k	2	44
TD2003	LETOR3.0	50	50k	2	64
TD2004	LETOR3.0	75	75k	2	64
NP2003	LETOR3.0	150	150k	2	64
NP2004	LETOR3.0	75	75k	2	64
HP2003	LETOR3.0	150	150k	2	64
HP2004	LETOR3.0	75	75k	2	64
MQ2007	LETOR4.0	1692	70k	3	46
MQ2008	LETOR4.0	784	15k	3	46
MSLR-WEB30K	Microsoft	31531	3775k	5	136
MSLR-WEB10K	Microsoft	10000	1200k	5	136

B. SOFTWARE TOOLS

In order to implement the deep neural network for learning-to-rank and conduct a number of experiments, we have exploited the following libraries:

LASANGE

Lasange framework is built on the top of Theano library [58]. It is light-weight library particularly used to build and train the neural network. This Python library allows us to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

TensorFlow

TensorFlow [63] is an open-source library mainly used to deal with deep neural network. It excels at a numerical computing that are terribly advanced for deep learning.

C. EVALUATING MEASURES

Normalized discounted cumulative gain *(NDCG)* [54] and Mean Average Precision *(MAP)* [55] are the two most popular evaluating metrics used to measure the performance of

[1]https://trec.nist.gov/data/webmain.html

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

[3]https://github.com/jma127/pyltr

learning-to-rank model. However, we evaluate performance of our ranking models using *precision@k (P@k)*, *MAP*, *NDCG@k* and sparsity ratio. The former three evaluating measures are mainly used to evaluate the performance of retrieval system while sparsity ratio is used to evaluate the feature selection method. The detail description of these evaluating measures are given below:

MEAN AVERAGE PRECISION

MAP [55] is a standard evaluating measure used for IR. It is better to use for binary relevance judgment. The mean average precision is computed using the precision@k ($P@k$). The $P@k$ is the fraction of relevant documents at position k of ranking list associated with query q_i.

$$P@k_{q_i} = \frac{\#\text{ relevant documents out of top-k documents}}{k}.$$

Average $P@k$ is given as:

$$AP_{q_i} = \frac{\sum_{i=1}^{k} P@i}{\#\text{ total relevant documents for query q}}.$$

Thus, *MAP* value on all queries is given as:

$$MAP = \frac{\sum_{n=1}^{N_q} P@n * rel(n)}{N_q}.$$

NORMALIZED DISCOUNTED CUMULATIVE GAIN

NDCG measure is specially designed for non-binary notions of relevance. It utilizes the explicit rating of the documents in the list. The $DCG@k$ value at position k is computed as :

$$DCG_q@k = \sum_{i=1}^{k} \frac{2^{l_i} - 1}{log_2(1+i)},$$

where l_i represents the label for i^{th} document. $DCG@k$ gives value greater than k. $NDCG@k$ normalizes the value $DCG@k$ in range of $0-1$. Thus, *NDCG* is computed as:

$$NDCG = \frac{1}{Z_n} DCG_q@k,$$

where Z_n is normalization factor.

SPARSITY RATIO

We promote our research for feature selection and it has been achieved by using sparsity ratio. The sparsity ratio is fraction of remaining features in the model after network regularization. The sparsity ratio is given as:

$$SR = \frac{\#\text{remaning feature after learning model}}{\#\text{total number of features}}$$

The features with zero weight are considered as less important features for all queries. Thus, total number of features consider for predicting score of the document could be smaller than the number of features given in Table 2.

We have analyzed the sparsity ratio obtained by using different penalty norms and compared them with the algorithms, FenchelRank [36], FSMRank [35] and RankSVM-NC [37], those that specifically follow the procedure embedded method of feature selection.

D. EXPERIMENTAL SETUP

We begin with evaluate our proposed model with a number of experiments on LETOR datasets. We use modern deep neural network architecture which is specified in Table 3. We run the optimization algorithm for 100 number of epochs with variable batch size (mini-batch size = number of documents per query). The performance is analyzed in subsequent experiments by specifying the number of hidden neurons in each layer. Experiments have been carried out using neural networks with a decreasing number of hidden neurons from one layer to the other.

TABLE 3. Experimental architecture for deep neural network.

# NH	NI	# NHL
1	No. of Features in Dataset	128
2	No. of Features in Dataset	128<-64
3	No. of Features in Dataset	128<-64<-64

Configuration of the different deep neural network, consisting of 1, 2 and 3 hidden layers, used for the subsequent experiments. The number of neurons at input layer depends on the number of features. #NH= Number of hidden layers; #NI= Number of neurons at input layer; # NHL= number of neuron at each hidden layer.

The weights of the network are initialized using the weight initialization method given in [57]. Furthermore, network is trained with the popular 'Adam' method for optimization [56]. This stochastic gradient optimization has an adaptive step size and momentum. In overall experimentation, 'Adam' is applied with its default setting defined in [56]. The default values of parameters of Adam are $\beta 1 = 0.9, \beta 2 = 0.009, \alpha = 0.002$. With these default values of different parameters, we initialize learning rate as $= \frac{\alpha}{(1-\beta)}$. Then, at each iteration learning rate adopts the value automatically. For each case, network is implemented with 'ReLU' activation function at each hidden layer with standard one-hot encoding whereas output layer uses sigmoid activation function. The value of the regularization factor λ is kept fixed to $\lambda = 10^{-3}$ for every layer. For each dataset, the value of λ is computed using the best *NDCG@n*, *MAP* or *P@n* on the validation set. The model trained with λ is used to predict output on test set. The results of experiments include average training accuracy, testing accuracy, sparsity of network and size of hidden layers. The noted results of experiments on available datasets are obtained on Intel core $i5-6200U$ @ 240 GHz with 8GB RAM. For large scale data, GPU is preferred with CUDA in background.

VI. RESULTS AND DISCUSSION

In this section, we first compare modern regularized network with dense network without regularization against the IR evaluating measures. Secondly, we show the effect of the regularization parameter on the evaluating measures for different regularization technique. The value of λ that leads to the

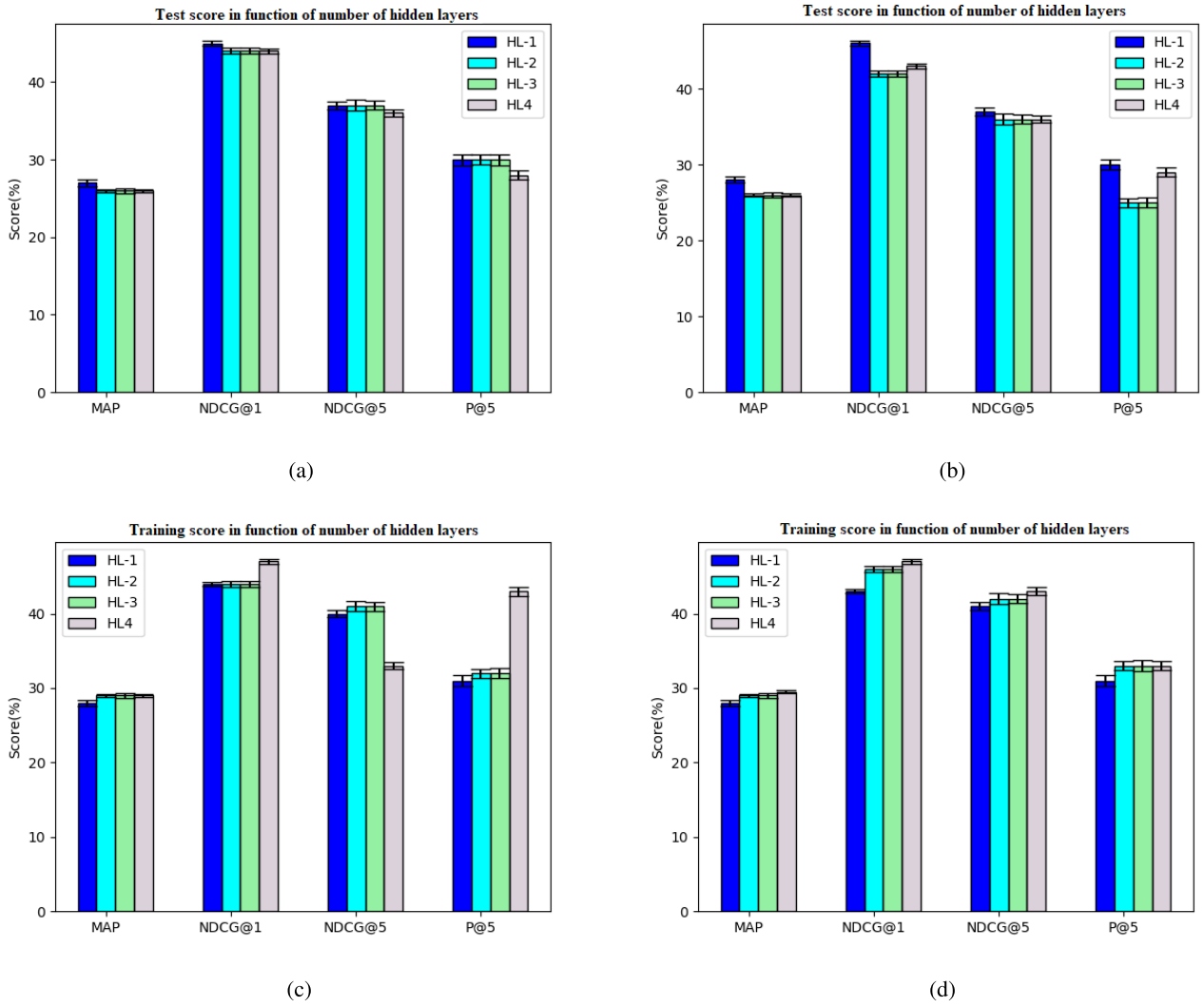


FIGURE 6. Comparison of performances (MAP, NDCG@1) on training and testing set for deep neural network with 1, 2, 3 and 4 hidden layers with and without regularization. (a) Training set. (b) Testing set. (c) Training set. (d) Testing set.

best NDCG@10 on performance of validation set is chosen. The effect of different regularizations on training accuracy, testing accuracy and sparsity are briefly presented. Finally, we confront the sparsity ratio and the performance in terms of IR measures to demonstrate that group regularizations that are truly competitive to state-of-the-art approaches.

A. RESULTS

1) EFFECT OF NUMBER OF HIDDEN LAYER

Experiments with increasing number of hidden layers show that there is almost no impact on the performance. Figure 6 summarizes the performance scores with respect to number of hidden layers. As per the results shown in Figure 6(a) and Figure 6(b), no improvements can be found by adding more hidden layers. The results are even slightly worse. Moreover, dense network increases the computational complexity of the learning model. This issue is minimized by adding regularization term (weight penalty) at each layer to increase the performance and to avoid overfitting of network. Figure 6(c) and Figure 6(d) present results of the deep neural network with regularization parameters.

2) EFFECT OF REGULARIZATION PARAMETER

The deep neural network is optimized with different regularization techniques primarily reduces the complexity of network and makes it more simple for learning. We have observed the performance of learning-to-rank model with respect to these regularization techniques. It includes ℓ_1, ℓ_2, *group* ℓ_1, and *SGL*. The main aim of this preliminary test is to evaluate optimization of loss function with different penalties when varying the regularization factor λ. With these regards, each case is run with the different exponential values of λ ranges from $[10^{-5} - 10^{-3}]$. Results of this experiment are given in Figure 7. There are various key observations made from this results. To begin with, the overall behavior has measured in terms of evaluation measure *NDCG@n* on three sets (training, testing and validation sets) with respect to the four penalties. For each case, it is observed that

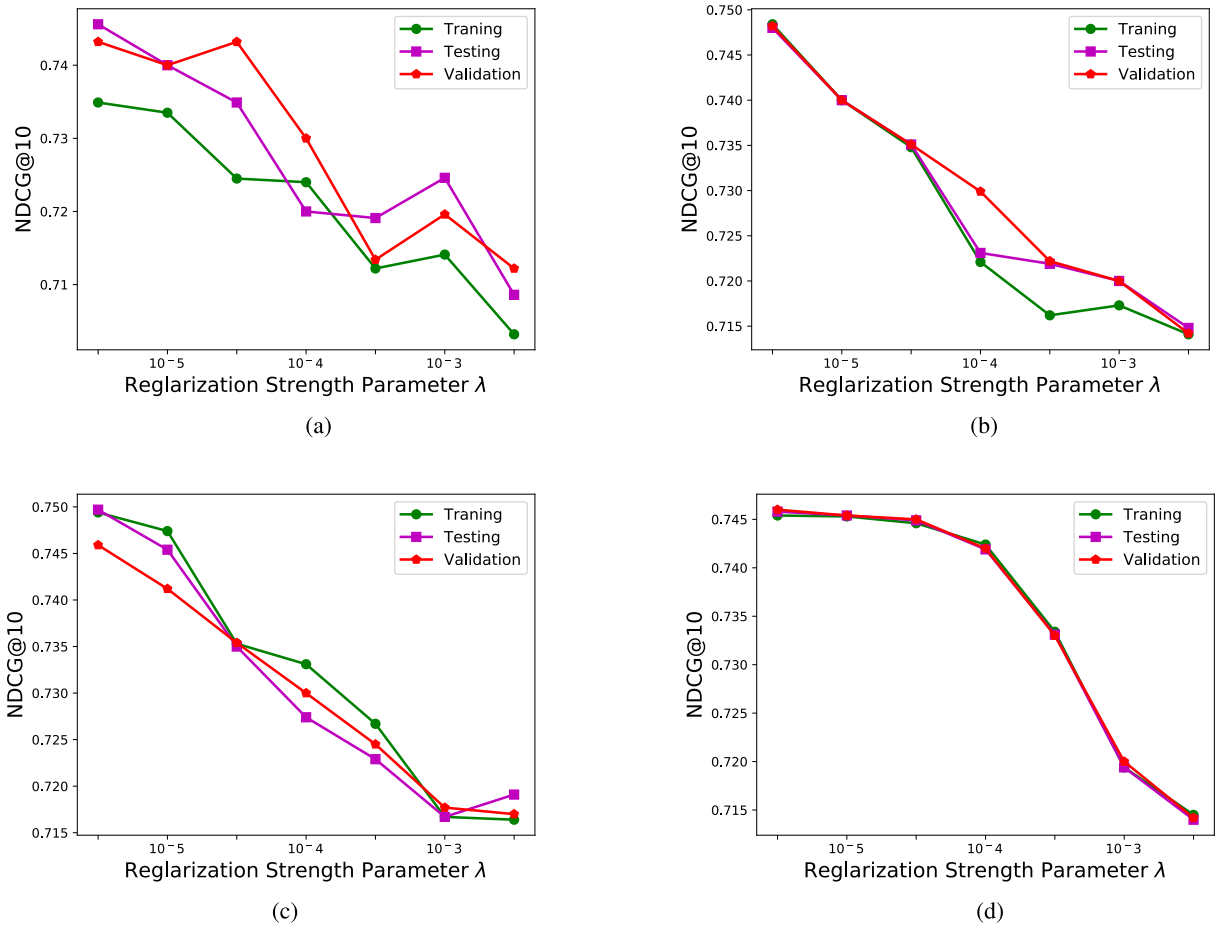


FIGURE 7. NDCG@10 score for the training, validation, and test set on a 3-hidden layered deep neural network in function of the ℓ_2-regularization, ℓ_1-regularization, group ℓ_1-regularization and sparse group ℓ_1-regularization with strength parameter λ. (a) ℓ_2 Regularization. (b) ℓ_1 Regularization. (c) Group ℓ_1 Regularization. (d) Sparse Group ℓ_1 Regularization.

TABLE 4. Comparison for sparsity ratio.

Dataset	FenchelRank	FSMRank	L2	L1	Group L1	Sparse L1
MQ2008	0.3	0.42	0.21	0.188	0.176	**0.19**
MQ2007	0.58	0.64	0.69	0.228	0.38	**0.214**
HP2004	0.19	0.268	0.34	0.281	**0.265**	0.286
NP2004	0.27	0.37	0.6	0.346	0.391	**0.344**
TD2004	0.46	0.67	0.154	0.106	0.126	**0.1047**
HP2003	0.27	0.48	0.36	0.339	**0.211**	0.362
NP2003	**0.23**	0.44	0.6	0.294	0.89	0.286
TD2003	0.53	0.76	0.42	0.554	0.694	**0.373**

evaluation measures rapidly converge to the optimal value for sufficiently small regularization factors λ. These results are basically indistinguishable from 10^{-3} onward. Figure (7d), shows that *SGL* rapidly converges to optimal accuracy for small regularization factor and gives better results in terms of evaluating measures. These regularization techniques can also affect the other performance measures such as training accuracy, testing accuracy, sparsity, number of active neurons, MAP, etc.

3) LEARNING ACCURACY WITH SPARSITY

For each fold of dataset, λ value that leads to the best *NDCG* performance on the validation set is chosen and it is used for the prediction in the test set. We use the same regularization factor for all cases, as it gives the best results in terms of accuracy and sparsity of the network. In Table 5, we stress our focus is on comparing the different penalties. Similar results can be obtained for different choices of network architecture and regularization factors. Thus, results in terms of test accuracy are comparable with negligible loss. However, difference observed in terms of sparsity. The sparsity leads to thinning of the network which is performed using the removal of insignificant connections. During training, all absolute values of weights under 10^{-4} are considered as sparse or low rank weight and it is set to zero. *SGL* neural network regularization results in network which is extremely sparse and more compact as compared to other competitors. Let us consider the example of LETOR3.0 (HP2003 and TD2003) datasets. In this case, algorithm removes almost 36% (HP2003) and 38% (TD2003) features in an average from the input vector as compared to approximately 33% (HP2003) and 35% (TD2003) features using ℓ_1 regularization. Resulting network also has large sparse group at each hidden layer as compared to ℓ_1 regularization and group ℓ_1 regularization. Thus, even though training accuracy of algorithms has

TABLE 5. Average results on benchmark datasets.

	Dataset	ℓ_2	ℓ_1	Group ℓ_1	Spasre Group ℓ_1
Training accuracy (%)	MQ2008	0.7996	0.8537	0.8728	0.7495
Testing accuracy (%)		0.7978	0.8537	0.8737	0.7455
Number of neurons		[64,128,64]	[64, 128, 64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.076,0.0042,0.0485]	[0.188, 0.01, 0.0537]	[0.176, 0.054, 0.0502]	[0.19, 0.01, 0.0529]
Training accuracy (%)	MQ2007	0.7125	0.7125	0.7125	0.7125
Testing accuracy (%)		0.7432	0.7432	0.7432	0.7432
Number of neurons		[64,128,64]	[64,128,64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.069, 0.005, 0.147]	[0.228, 0.254, 0.2006]	[0.08, 0.075, 0.166]	[0.214, 0.271, 0.1936]
Training accuracy (%)	HP2004	0.9988	0.9988	0.9988	0.9988
Testing accuracy (%)		0.995	0.995	0.995	0.991
Number of neurons		[64,128,64]	[64, 128, 64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.034, 0.075, 0.126]	[0.281, 0.283, 0.1717]	[0.065, 0.088, 0.1402]	[0.286, 0.296, 0.1658]
Training accuracy (%)	HP2003	0.999	0.999	0.999	0.999
Testing accuracy (%)		0.9991	0.9991	0.9991	0.9991
Number of neurons		[64,128,64]	[64,128,64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.036, 0.042, 0.2253]	[0.339, 0.371, 0.3244]	[0.081, 0.108, 0.2703]	[0.362, 0.379, 0.33]
Training accuracy (%)	TD2003	0.9953	0.9953	0.9953	0.9953
Testing accuracy (%)		0.9883	0.9883	0.9883	0.9883
Number of neurons		[64,128,64]	[64,128,64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.0042, 0.0067, 0.1632]	[0.354, 0.337, 0.2121]	[0.094, 0.092, 0.1872]	[0.373, 0.317, 0.2169]
Training accuracy (%)	TD2004	0.9843	0.9843	0.9843	0.9843
Testing accuracy (%)		0.986	0.986	0.986	0.986
Number of neurons		[64,128,64]	[64,128,64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.0154, 0.0096, 0.5317]	[0.106, 0.863, 0.7499]	[0.026, 0.25, 0.7493]	[0.1047, 0.933, 0.7941]
Training accuracy (%)	NP2003	0.999	0.999	0.999	0.999
Testing accuracy (%)		0.9987	0.9987	0.9987	0.9987
Number of neurons		[64,128,64]	[64,128,64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.06, 0.0104, 0.1611]	[0.294, 0.283, 0.2105]	[0.089, 0.125, 0.1876]	[0.286, 0.288, 0.2054]
Training accuracy (%)	NP2004	0.9986	0.9986	0.9986	0.9986
Testing accuracy (%)		0.999	0.999	0.999	0.999
Number of neurons		[64,128,64]	[64,128,64]	[64,128,64]	[64,128,64]
Sparsity (%)		[0.06, 0.0058, 0.1892]	[0.346, 0.346, 0.2593]	[0.091, 0.104, 0.2158]	[0.344, 0.333, 0.2307]

a negligible difference for regularized penalties, much difference observed in terms of sparsity. We repeat each experiment at least 100 times in order to get the statistical variation.

EMPIRICAL ANALYSIS ON BENCHMARK DATASETS

1) SPARSITY RATIO

As mentioned in the introduction, feature selection is an important task for machine learning. Hence, we propose highly effective learning model that can be learned with high accuracy and automatic selection of few numbers of highly relevant features while learning. In deep neural network based ranking model, main aim of regularization is to sharply reduce the irrelevant features and to make it more compact. In this section, we compare sparsity (%) (number of zero weights with respect to total number of features) that are generated by each technique. If a model has more sparsity (%) with high training accuracy, model is more robust with less features. We have analyzed the sparsity ratio obtained for different weight penalties and compared them with algorithms that are built with an embedded method of feature selection for application to learning-to-rank.

Table 4 presents the sparsity ratio obtained for different embedded algorithms of feature selection designed under learning-to-rank. It includes, FSMRank [35], FenchelRank [36], sparse SVM-NC [37] and ℓ_2, ℓ_1, *group* ℓ_1 and *SGL* regularization. Sparsity ratio represents the number of features removed. When we consider the sparse values, ℓ_1 regularization gives more sparse results. As we have discussed, in the deep neural network, we prefer group level sparsity (group ℓ_1 specifically for weights in hidden layer) to speed up the network and reduce the learning complexity. A group with sparse value is not considered in subsequent layer while learning. However, if any one of the value within group is not zero then that group will not be considered as the sparse group. Thus, when we consider sparsity ratio, the regularization with *SGL* is more beneficial.

From the survey on embedded methods for feature selection, it is deduced that the convex learned model selects the average of half number of features for learning. Whereas our proposed model with *SGL* regularization selects up-to $4-5$ times less feature as compared to the convex regularization used in FenchelRank and FSMRank and $5-7$ times less features than sparse SVM-NC. The sparse group penalty is particularly effective for high dimensional feature variables.

In Table 5, there are differences observed in sparsity produced among LETOR datasets due to different characteristic features. All features of *HP*, *NP* and *TD* datasets are similar however these are not related to similar notions of retrieval tasks. Hence, the number of relevant features are different for different datasets. The differences in performance in terms of sparsity ratio for the datasets is not the drawback of the algorithm, but it is because of specificity of datasets.

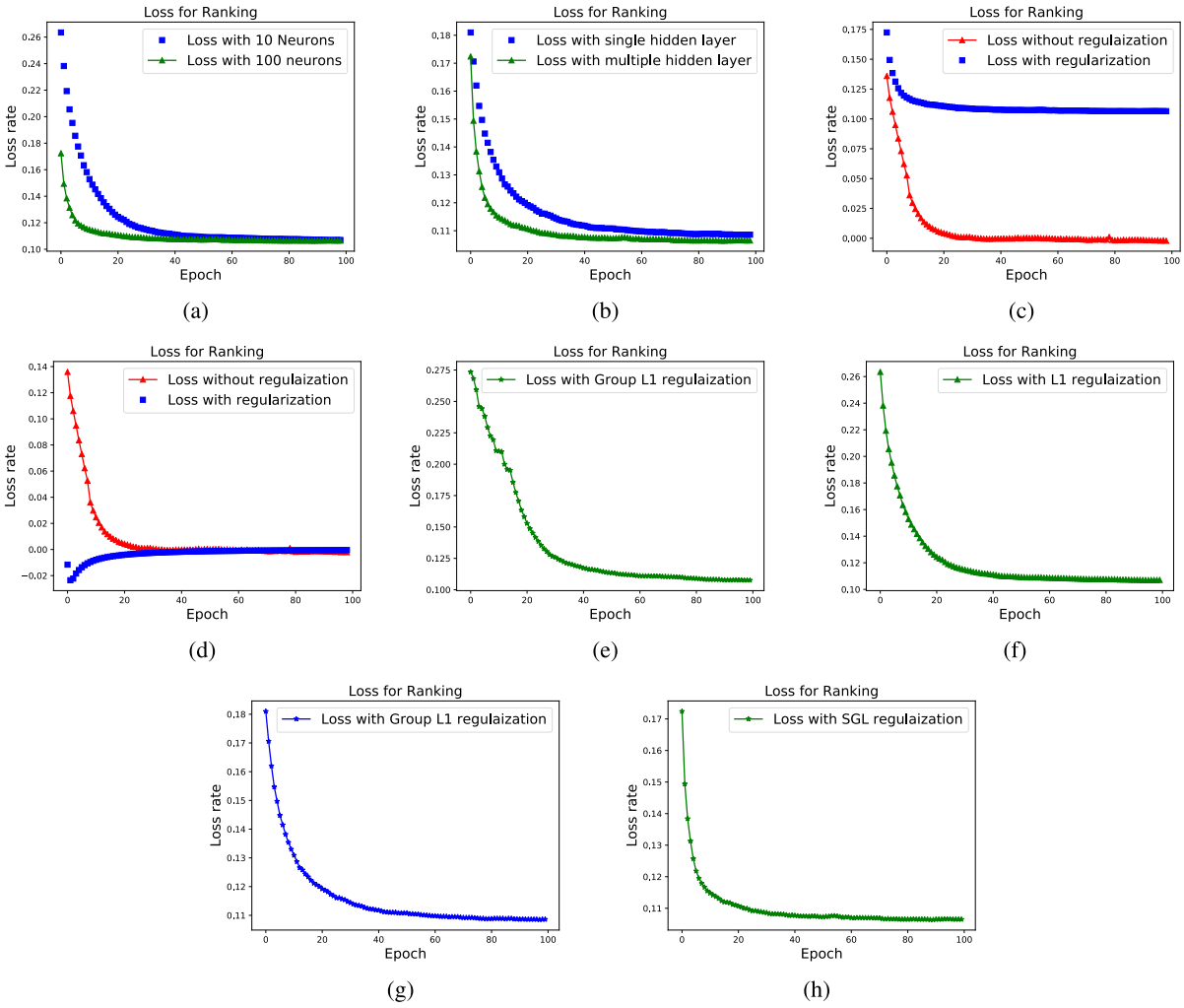


FIGURE 8. **(a) Training loss with 10 and 100 neurons single layered neural network model; (b) training loss with single hidden layer and multiple hidden layers; (c) training loss on single layer neural architecture with and without regularization; (d) training loss on multi layers neural architecture with and without regularization; (e) training loss with ℓ_2-regularization;(f) training loss with ℓ_1-regularization; (g) training loss with group ℓ_1-regularization; (h) training loss with *SGL*-regularization.**

2) TRAINING ACCURACY

We have evaluated deep neural network model on benchmark datasets described in Table 2. The accuracy of training algorithm is measured in terms of loss occurred while learning. Model is considered better, if the loss is less. In proposed model, cross entropy loss function is used as loss function and our main aim of learning is to optimize this loss. Figure 8 shows the effect on the accuracy of model (in terms of loss) for dense and sparse network. Figure 8(a), (b) and (c) show that performance for single layer neural network for the application of learning-to-rank. Figure 8(b) shows that loss with dense network is less as compared to single layer architecture. However, Figure 8(c) shows that loss with regularized network does not give the consistent performance with certain properties of network mapping. The method of regularization to generalize learning mainly uses a network that is just large enough to provide the adequate fit. The small network does not have enough power to overfit the data. Thus, a single layer regularized network have no effects or adverse effect on loss function. A more dense network reduces the training loss in each iteration however might degrade the value of evaluating measures supported predictions. The increasing number of layers do not provide satisfactory improvement. On the other hand, it increases the computational cost in terms of time and memory. It also gives better accuracy for trained data however performance is average for test data. Figure 8 presents the training loss occurred in each epoch while learning.

3) PERFORMANCE IN TERMS OF IR MEASURE

Rather than training and testing accuracy, the learning model for ranking puts more emphasis on the evaluating measures used for ranking. Hence, proposed model is analyzed with the evaluating measures *NDCG@n*, *P@n* and *MAP*. Subsequent experiments provide results in terms of *NDCG@n* and *MAP* on several benchmark datasets. A distinguishable difference is observed between MQ2007 and MQ2008. For these

TABLE 6. NDCG@n value of the sparse regularized deep neural network for different dataset where n={1,3,5,7,10}.

Dataset	$n=1$	$n=3$	$n=5$	$n=7$	$n=10$
MQ2007	0.19012	0.38993	0.56280	0.76897	0.79975
MQ2008	0.2042	0.4351	0.5902	0.7293	0.8239
TD2003	0.19832	0.21385	0.26607	0.32631	0.43621
TD2004	0.08864	0.19058	0.324959	0.32206	0.31021
NP2003	0.11398	0.24267	0.32914	0.40963	0.47682
NP2004	0.08841	0.21999	0.32914	0.40963	0.38022
HP2003	0.08682	0.18323	0.25064	0.31031	0.38429
HP2004	0.04228	0.18414	0.25322	0.30785	0.37345
MSLRWEB10K	0.055431	0.10569	0.143260	0.189552	0.24104
MSLRWEB30K	0.05313	0.11366	0.15264	0.18526	0.24239

datsets, we have noticed the largest variations for *MAP* and *NDCG@n*. Furthermore, MSLR dataset gives significant results in terms of performance measures. We have presented the results for MSLR dataset only for *NDCG@n* because of scalability issue.

TABLE 7. Comparison of *MAP* score between different state-of-the-art algorithms on MQ2008 and TD2003 datasets.

Model	MQ2008	TD2003
RankBoost	0.4775	0.2274
RankSVM	0.4946	0.2628
RankNet	0.441	0.1932
Frank	0.4211	0.2031
LambdaRank	0.4996	0.2283
ListNet	0.4431	0.2753
FenchelRank	0.4785	0.2780
FSMRank	0.4771	0.2560
Non-convex Regularization with Sparse SVM	0.435	0.2670
DNN-LTR with ℓ_2 Reg	0.5002	0.2121
DNN-LTR with ℓ_1 Reg	0.5142	0.2281
DNN-LTR with Group ℓ_1 Reg	0.5146	0.2132
DNN-LTR with *SGL* Reg	**0.5202**	0.2779

We compare the prediction of our proposed frameworks against those of the other state-of-the-art algorithms, such as RankBoost [16], RankSVM [6], RankNet [9], FRank [10], ListNet [13], LambdaRank [11]. Table 7 and Table 8 indicates that the proposed model algorithm gives the best value for *MAP*, *NDCG@n* and *P@n* respectively, where $n = 1, 3, 5, 7$ *and* 10. Proposed algorithm for each case is also compared with the traditional algorithms of learning-to-rank. For these evaluating measures, one can notice that some algorithms can not always perform better than other algorithms. Baseline algorithms tend to provide more or less same results in terms of *NDCG@n, P@n* and *MAP*. We have analyzed our results on TD2003 and MQ2008 datasets. There is a lot of difference in terms of performance which can be observed among the baseline algorithms. It gives significant variation for LETOR4.0 dataset. Among all traditional algorithms, LambdaRank and ListNet give the comparable results, thus we compare our model with it. The FenchelRank, FSMRank and sparse SVM-NC also show comparable results in terms of *NDCG*, *Precision* and *MAP* value.

a: NDCG@n ANALYSIS

There is not much difference between the baseline algorithms in terms of *NDCG@n*. However, our model gives significantly distinguishable results for LETOR datasets. The resultant value for *NDCG* on benchmarks datasets with default regularization setting are provided in Table 6. Highest *NDCG@10* value for the proposed model is 0.82 ± 04. Around 30% result is improved for *NDCG@10* than other baselines on the LETOR4.0 dataset whereas $3\% - 5\%$ on LETOR3.0. The value with regularized deep neural network produces non-significant and biased results because of the sparse features. The improvement of deep learning-to-rank with sparse group regularization against non-regularized network on LETOR4.0 is (2%). For a number of datasets, regularized network with ℓ_2 regularization shows more improvement however it could be biased because of more sparse features. The computational time for ℓ_2 regularization is also more than other regularization techniques. The results for NDCG@k also are compared for six baselines methods and three feature selection methods of learning-to-rank. The comparison results are shown in Table 8. Compared to baseline algorithms, proposed approach shows the significant ranking performance gain for MQ2008. The results for TD2003 are competitive.

b: MAP ANALYSIS

The *MAP* values give better results for the dataset with only two relevance judgments. We observe that the proposed model can lead to some degradation in terms of *MAP* for most of the datasets. Like *NDCG*, the value for *MAP* is higher for MQ2007 and MQ2008 in comparison to other datasets. Furthermore, it narrows down in decreasing manner, less than 1% for half of the datasets. Table 7 gives the comparison of the MAP values between all the baseline and proposed architecture.

Among all mentioned baselines, RankSVM-NC provides higher value of *MAP* on MQ2008 dataset. Our proximal approach with deep neural network can lead to the equivalent result. We observed that a fair improvement for the *MAP* value for models using regularization. The use of *SGL* regularization leads to (10%) upgradation for MQ2008 dataset. However, there is significant degradation observed for TD2003 dataset for dense network. On the other hand, the regularized network gives the results equivalent to the best algorithm. Finally, we notice that *SGL* penalty provide a good result as non-convex penalties for which *MAP* are highest.

B. DISCUSSION

Earlier, we have analyzed the performance of learning-to-rank models with deep neural network architecture on the benchmark datasets given in Table 2. It outperforms over all the traditional models and gives best results in terms of evaluating measures. The improvement of proposed model with sparse group regularization against without regularization on LETOR4.0 is 2.3% w.r.t. *NDCG@10*, 3.1% w.r.t. *P@10* and 3.8% w.r.t. *MAP*. The range of improvements may varies for different cut-off values. However, it gives the best

TABLE 8. Benchmark TD2003 and MQ2008: Comparison of the NDCG@n and P@n measures between the different state-of-the-art algorithms.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@7	NDCG@10	P@1	P@3	P@5	P@7	P@10
TD2003										
RankBoost	0.2212	0.3184	0.3312	0.349	0.3701	0.1	0.0667	0.12	0.21	0.18
RankSVM	0.3123	0.238	0.2119	0.1957	0.2167	0.42	0.34	0.26	0.23	0.21
RankNet	0.3112	0.2354	0.4077	0.3042	0.2929	0.2	0.1667	0.22	0.12	0.19
Frank	0.4421	0.37	0.33	0.33	0.34	0.44	0.32	0.23	0.21	0.19
LambdaMART	0.5667	0.5163	0.4378	0.4089	0.4027	0.2	0.233	0.2	0.15	0.13
ListNet	0.4456	0.3704	0.3826	0.3861	0.3945	0.46	0.36	0.29	0.26	0.22
Feature Selection Methods										
FSMRank	0.4394	0.4149	0.4013	0.3916	0.3824	0.3757	0.3706	0.3666	0.3634	0.3613
FenchelRank	0.4492	0.3908	0.369	0.3703	0.364	0.3549	0.3508	0.3496	0.3435	0.3447
Sparse SVM-NC	0.4547	0.5032	0.4794	0.4605	0.452	0.4426	0.4385	0.427	0.3951	0.402
Proposed Model										
DNN with ℓ_1 Reg	0.48	0.4231	0.421	0.4112	0.4126	0.45	0.32	0.31	0.28	0.24
DNN with ℓ_2 Reg	0.4672	0.3675	0.352	0.34	0.35	0.45	0.35	0.28	0.27	0.24
DNN with Group ℓ_1 Reg	0.4842	0.3845	0.36	0.36	0.36	0.46	0.36	0.31	0.28	0.24
DNN with Sparse Group ℓ_1 Reg	0.4732	0.4721	0.4612	0.4551	0.430	0.4231	0.34	0.31	0.26	0.23
MQ2008										
RankBoost	0.4147	0.4486	0.4609	0.4866	0.5017	0.4423	0.391	0.3551	0.2987	0.2662
RankSVM	0.4256	0.447	0.4529	0.4654	0.4794	0.4888	0.4394	0.4224	0.4043	0.3856
RankNet	0.288	0.3976	0.4397	0.4653	0.4856	0.391	0.3632	0.3372	0.2875	0.2653
Frank	0.3245	0.4256	0.4635	0.4872	0.392	0.3621	0.3372	0.2823	0.2823	0.2666
LambdaMART	0.368	0.4832	0.4559	0.493	0.5322	0.4137	0.391	0.3372	0.2903	0.263
ListNet	0.3148	0.3924	0.4276	0.4659	0.4861	0.3846	0.359	0.3423	0.283	0.2636
Feature Selection Methods										
FSMRank	0.3686	0.4399	0.4791	0.4994	0.4771	0.4602	0.4526	0.4231	0.3879	0.3856
Fenchelrank	0.357	0.425	0.456	0.485	0.511	0.44	0.4586	0.4126	0.3956	0.3752
Sparse SVM-NC	0.3712	0.4324	0.4671	0.5001	0.512	0.4401	0.46	0.4052	0.3721	0.3524
Proposed Model										
DNN-LTR Without Regularization	0.1944	0.4143	0.7054	0.7054	0.8064	0.3712	0.4295	0.4367	0.4132	0.3561
DNN-LTR with ℓ_1 Reg	0.2057	0.4358	0.5948	0.7324	0.8097	0.5124	0.4521	0.4132	0.4192	0.3924
DNN-LTR with ℓ_1 Reg	0.2001	0.4242	0.5929	0.7181	0.8064	0.5056	0.441	0.4032	0.395	0.3989
DNN-LTR With Group Reg	0.2036	0.3308	0.6009	0.7295	0.82	0.4932	0.4423	0.4093	0.396	0.4
DNN-LTR with *SGL* Reg	0.2042	0.4351	0.5902	0.7293	0.8239	0.5213	0.4555	0.4289	0.4199	0.4045

results over the best method of learning-to-rank (i.e. ListNet) on MQ2008. The gain of 26.22% w.r.t. *NDCG@10*, 34.83% w.r.t. *P@10* and 14.82% w.r.t *MAP* is obtained over ListNet. The less improvement observed for the TD2003 and it is 9.0% w.r.t. *NDCG@10*, 4.5% w.r.t *P@10* and around 1.0% w.r.t. *MAP*. On competitive analysis with LambdaRank, it shows around 54.5% improvements w.r.t. *NDCG@10*, 41.65% w.r.t. *P@10* and 39.9% w.r.t. *MAP* on MQ2008. For TD2003, it is 6.7% for *NDCG*, 7.6 w.r.t. *P@k* and 2.1% w.r.t. *MAP*. The results depict that significant performance can be obtained in learning-to-rank approach for IR applications with appropriately modeled relevance ranking model provided with deep neural network. The MSLR is bulky dataset available for learning-to-rank and particularly used to check the scalability of learning models. Hence, we have evaluated only *NDCG* value for this dataset. There is no competitive analysis done due to scalability issue.

The ability of our framework is to select few selective features for their performance prediction. Hence, we have shown results for considered penalties that are competitive in order to reduce the number of features used by the learned model. In deep neural network, regularization technique generates the sparsity within a network which can be counted by total number of active neurons. Let us consider the example for LETOR3.0 which has 64 features (64 neurons) at the input layer. Our proposed model removes the more than 30 features in the subsequent layer while ℓ_1 removes only 20 features. Figure 9 shows, total number of active neurons in our designed network. In our experiments, there are total 256 neurons available for learning however this number goes on decreasing in each iteration. The number of neurons is rapidly decreased with *SGL* penalty while it is gradually decreased for ℓ_1 and ℓ_2 penalties.

Moreover, the sparsity generated by each penalty has also been analyzed against the *NDCG* score for different regularization strength parameter λ. A small value of λ leads to convergence of optimal score. Table 5 shows the regularization, specifically ℓ_1 and *SGL* which are highly competitive network generalization methods both in terms of sparsity and accuracy of evaluating measures. It selects up-to 4-7 times fewer features without significant degradation of evaluation measures.

The SGL penalty is indeed faster to compute than both ℓ_1 and ℓ_2 penalties when experiments run on CPU whereas it takes time to compute results on GPU. Since, there is a need to compute two square root operations per group. This gap can be reduced by exploring various options for faster mathematical computations.

There is one more regularization technique available for optimization of the deep neural network which is called as 'dropout' [59]. It is mainly used to simplify the network during learning. Dropout removes a set of connections with

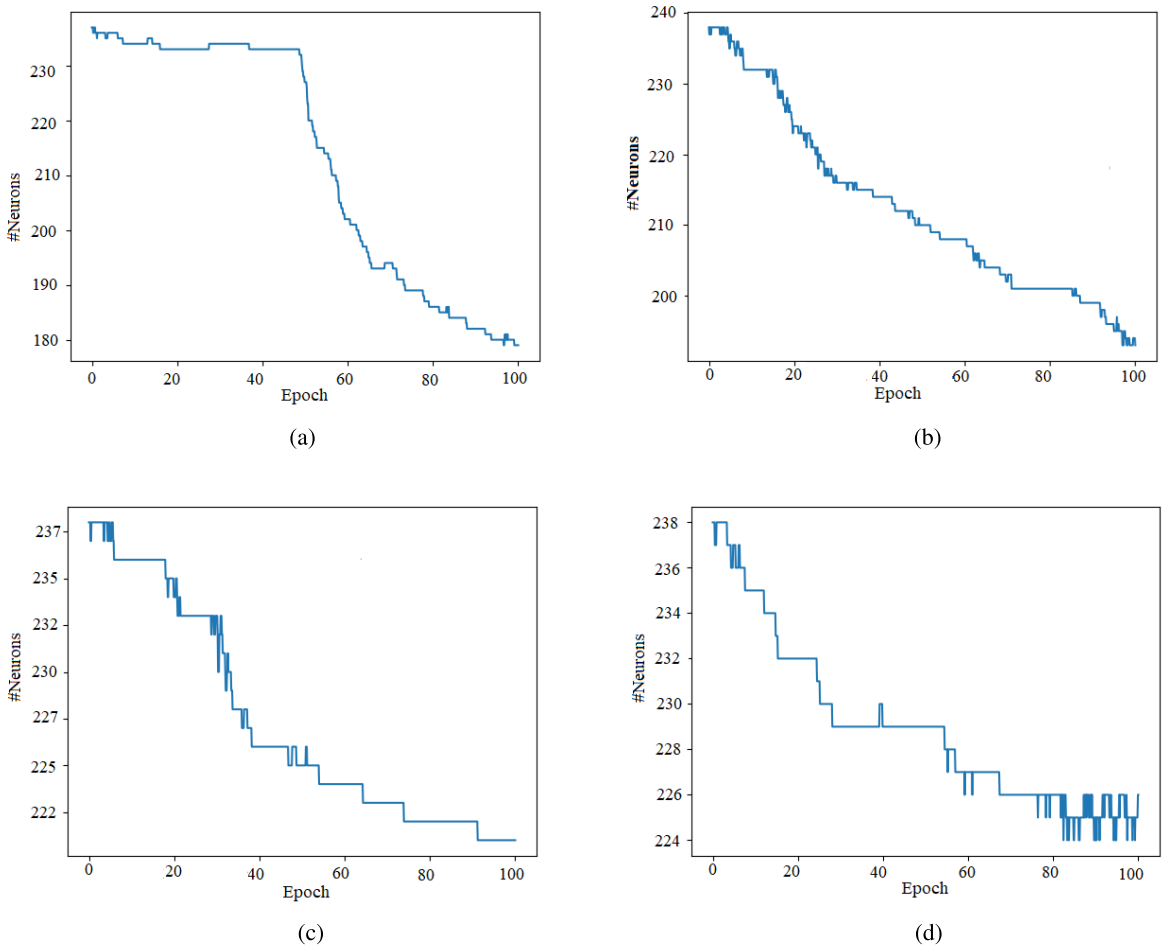


FIGURE 9. Total number of active neurons in each epoch. The total number of neurons is = 64 + 128 + 64 = 256. (a) number of active neurons with ℓ_2 regularized network; (b) number of active neurons with ℓ_1 regularized network; (c) number of active neurons with *group*-ℓ_1 regularized network; (d) number of active neurons with *SGL* regularized network.

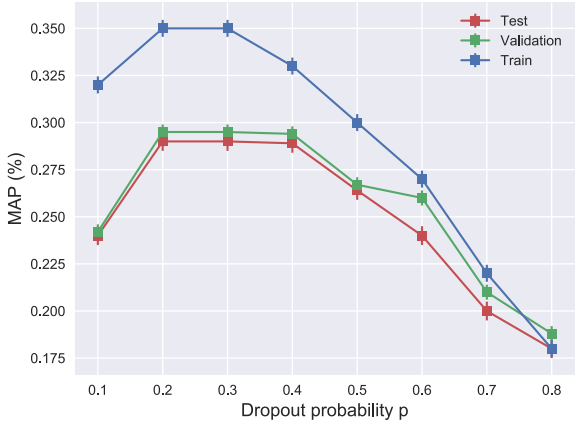


FIGURE 10. MAP score on the training, validation and test set for a 3-hidden layer neural network based learning model with a function of dropout probability *p*.

random probability p. It is only used to reduce the complexity of training phase and lower down the complexity while the entire network needs the prediction accuracy. Thus, it is only tangentially related to the proposed strategies.

The dropout technique is investigated on different sized neural networks. The parameter p is experimented with value $[0.1, 0.2, \ldots, 0.8]$. These are tuned through cross-validation for each network configuration. The results for dropout are shown in Figure 10. The performance of dropout regularization technique is competitive to the other generalization methods ℓ_1 and ℓ_2 however it prevents the feature co-adaptation which can be achieved by *SGL*. Thus, *SGL* performs better over the dropout. The reason behind better performance of sparse group regularization over the classical technique 'dropout' is not trivial. Figure 11 provides the explanation for it. On comparing, it is noticed that the activation value for the sparse group regularization are more equally scattered than dropout, especially for hidden layer 2 and 3. The equally diffused activation value shows that neurons are more robust. Generally, output does not depend on single dominating neuron. Thus, sparse regularization has resulted in better performance.

As a conclusion, the proposed framework provides better results in terms of quality prediction compared to state-of-the-arts with selecting a minimum number of features. Thus, it is a competitive method for learning-to-rank.

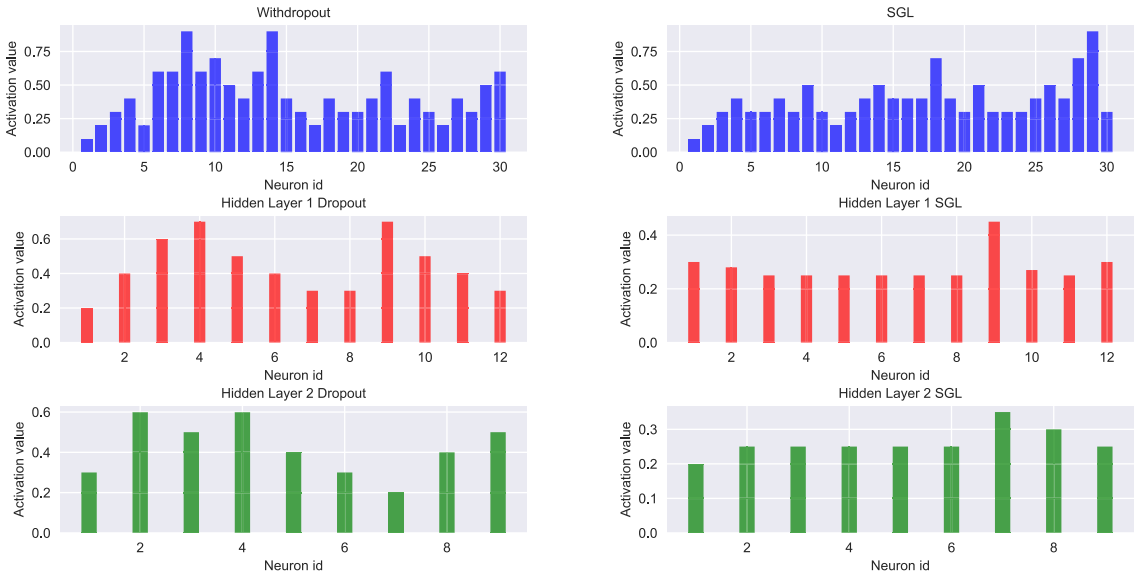


FIGURE 11. Average activation value of each neuron for their respective hidden layer. The results are displayed for a 3-hidden layered deep neural network model without and with dropout regularization.

VII. CONCLUSION

In this paper, we implement and analyze the performance of deep neural network architecture for learning-to-rank. The results greatly indicate thinning of the network by optimizing its weight, selecting only the relevant features from the input layer and removal of inactive neuron from the hidden layers. The performance of different regularization techniques was also analyzed over the benchmark datasets available for learning-to-rank. Our empirical analysis was helpful in observing that *SGL* has superior performance and gives highly compact networks, making more sparsified model along with high quality prediction in ranking. On the other hand, it can be easily implemented with very overhead as compared to standard ℓ_1 and ℓ_2 regularization. From the above findings, it is concluded that deep neural network architecture with suitable regularization technique significantly improves the result of relevance ranking model.

As a future direction, we would extend our proposed model for various applications of learning-to-rank like collaborative filtering, recommender system such as e-commerce, and bioinformatics for drug selection and prioritization.

REFERENCES

[1] H. Li, "A short introduction to learning to rank," *IEICE Trans. Inf. Syst.*, vol. 94, no. 10, pp. 1854–1862, 2011.

[2] K. Crammer and Y. Singer, "Pranking with ranking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 641–647.

[3] D. Cossock and T. Zhang, "Subset ranking using regression," in *Proc. 19th Annu. Conf. Learn. Theory*, 2006, pp. 605–619.

[4] P. Li, Q. Wu, and C. J. C. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 897–904.

[5] R. Herbrich, "Large margin rank boundaries for ordinal regression," in *Advances in Large Margin Classifiers*. Cambridge, MA, USA: MIT Press, 2000, pp. 115–132. doi: 10.1234/12345678.

[6] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, 2002, pp. 133–142.

[7] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with SVMs," *Inf. Retr.*, vol. 13, no. 3, pp. 201–215, Jun. 2010.

[8] T. Joachims, "Training linear SVMs in linear time," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, 2006, pp. 217–226.

[9] C. Burges *et al.*, "Learning to rank using gradient descent," in *Proc. 22nd Int. Conf. Mach. Learn.* New York, NY, USA: ACM, 2005, pp. 89–96.

[10] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma, "FRank: A ranking method with fidelity loss," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: ACM, 2007, pp. 383–390.

[11] C. J. C. Burges, R. Ragno, and Q. V. Le, "Learning to rank with non-smooth cost functions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 193–200.

[12] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, "Sortnet: Learning to rank by a neural-based sorting algorithm," in *Proc. SIGIR Workshop Learn. Rank Inf. Retrieval. (LRIR)*, 2008, vol. 42, no. 2, pp. 76–79.

[13] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proc. 24th Int. Conf. Mach. Learn.* New York, NY, USA: ACM, 2007, pp. 129–136.

[14] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: Theory and algorithm," in *Proc. 25th Int. Conf. Mach. Learn.* New York, NY, USA: ACM, 2008, pp. 1192–1199.

[15] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "Softrank: Optimizing non-smooth rank metrics," in *Proc. Int. Conf. Web Search Data Mining*. New York, NY, USA: ACM, 2008, pp. 77–86.

[16] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, Dec. 2003.

[17] Z. Zheng, K. Chen, G. Sun, and H. Zha, "A regression framework for learning ranking functions using relative relevance judgments," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: ACM, 2007, pp. 287–294.

[18] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, "Learning to rank by optimizing ndcg measure," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1883–1891.

[19] O. A. S. Ibrahim and D. Landa-Silva, "ES-Rank: Evolution strategy learning to rank approach," in *Proc. Symp. Appl. Comput. ACM*, 2017, pp. 944–950.

[20] T.-Y. Liu, *Learning to Rank for Information Retrieval*. New York, NY, USA: Springer, 2011.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[22] E. Nova, "Deep neural networks for learning to rank," Goeric Huybrechts, Univ. Catholique Louvain Benelearn, Ottignies-Louvain-la-Neuve, Belgium, Tech. Rep., 2016.

[23] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng, "Deeprank: A new deep architecture for relevance ranking in information retrieval," in *Proc. ACM Conf. Inf. Knowl. Manage.* New York, NY, USA: ACM, 2017, pp. 257–266.
[24] B. Song. (2018). "Deep neural network for learning to rank query-text pairs." [Online]. Available: https://arxiv.org/abs/1802.08988
[25] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: ACM, 2015, pp. 373–382.
[26] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.* New York, NY, USA: ACM, 2013, pp. 2333–2338.
[27] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for web search," in *Proc. 23rd Int. Conf. World Wide Web*. New York, NY, USA: ACM, 2014, pp. 373–374.
[28] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.* New York, NY, USA: ACM, 2016, pp. 55–64.
[29] X. Geng, T.-Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: ACM, 2007, pp. 407–414.
[30] A. Gigli, C. Lucchese, F. M. Nardini, and R. Perego, "Fast feature selection for learning to rank," in *Proc. ACM Int. Conf. Theory Inf. Retr.* New York, NY, USA: ACM, 2016, pp. 167–170.
[31] G. Hua, M. Zhang, Y. Liu, S. Ma, and L. Ru, "Hierarchical feature selection for ranking," in *Proc. 19th Int. Conf. World Wide Web*. New York, NY, USA: ACM, 2010, pp. 1113–1114.
[32] H. Yu, J. Oh, and W.-S. Han, "Efficient feature weighting methods for ranking," in *Proc. 18th ACM Conf. Inf. Knowl. Manage.* New York, NY, USA: ACM, 2009, pp. 1157–1166.
[33] V. Dang and B. Croft, "Feature selection for document ranking using best first search and coordinate ascent," in *Proc. Sigir Workshop Feature Gener. Selection Inf. Retr.*, 2010, pp. 1–5.
[34] C. Li, L. Shao, C. Xu, and H. Lu, "Feature selection under learning to rank model for multimedia retrieve," in *Proc. 2nd Int. Conf. Internet Multimedia Comput. Service*. New York, NY, USA: ACM, 2010, pp. 69–72.
[35] H.-J. Lai, Y. Pan, Y. Tang, and R. Yu, "FSMRank: Feature selection algorithm for learning to rank," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 940–952, Jun. 2013.
[36] H. Lai, Y. Pan, C. Liu, L. Lin, and J. Wu, "Sparse learning-to-rank via an efficient primal-dual algorithm," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1221–1233, Jun. 2013.
[37] L. Laporte, R. Flamary, S. Canu, S. Déjean, and J. Mothe, "Nonconvex regularizations for feature selection in ranking with sparse SVM," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1118–1130, Jun. 2014.
[38] "Sparse regularized deep neural networks for efficient embedded learning," in *Proc. ICLR*, 2018.
[39] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, Jun. 2017.
[40] J. Yoon and S. J. Hwang, "Combined group and exclusive sparsity for deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3958–3966.
[41] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, 1992, pp. 950–957.
[42] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc., Ser. B (Methodol.)*, vol. 58, no. 1, pp. 267–288, 1996.
[43] G. de Castro Mendes Gomes, V. C. de Oliveira, J. M. de Almeida, and M. A. Gonçalves. (2013). "Is learning to rank worth it? a statistical analysis of learning to rank methods." [Online]. Available: https://arxiv.org/abs/1303.2277
[44] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Statist. Soc., B (Statist. Methodol.)*, vol. 68, no. 1, pp. 49–67, 2006.
[45] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc., B (Stat. Methodol.)*, vol. 67, no. 2, pp. 301–320, 2005.
[46] B. Wang and D. Klabjan. (2017). "An Attention-Based Deep Net for Learning to Rank." [Online]. Available: https://arxiv.org/abs/1702.06106
[47] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.
[48] J. Z. Kolter and A. Y. Ng, "Regularization and feature selection in least-squares temporal difference learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.* New York, NY, USA: ACM, 2009, pp. 521–528.
[49] M. Schmidt, N. L. Roux, and F. R. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1458–1466.
[50] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.
[51] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *J. Comput. Graph. Statist.*, vol. 22, no. 2, pp. 231–245, May 2012.
[52] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li, "Letor: Benchmark dataset for research on learning to rank for information retrieval," in *Proc. SIGIR Workshop Learn. Rank Inf. Retr.*, vol. 310. Amsterdam, The Netherlands: ACM, 2007.
[53] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retr.*, vol. 13, no. 4, pp. 346–374, 2010.
[54] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.
[55] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, vol. 463. New York, NY, USA: ACM, 1999.
[56] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[57] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, May 2010, pp. 249–256.
[58] J. Bergstra *et al.*, "Theano: A CPU and GPU math compiler in Python," in *Proc. 9th Python Sci. Conf.*, vol. 1, Jun. 2010, pp. 3–10.
[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[60] Y. Song, A. Schwing, and R. Urtasun, "Training deep neural networks via direct loss minimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2169–2177.
[61] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft "Neural ranking models with weak supervision," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: ACM, 2017.
[62] Q. Ai, K. Bi, J. Guo, and W. B. Croft. (2018). "Learning a deep listwise context model for ranking refinement." [Online]. Available:https://arxiv.org/abs/1804.05936
[63] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, and G. S. Corrado. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software Available From Tensorflow. org*. [Online]. Available: https://www.tensorflow.org

ASHWINI RAHANGDALE received the bachelor's degree from the Shri Guru Gobind Singh Institute of Engineering and Technology, Swami Ramanand Teerth Marathwada University, Nanded, India, and the master's degree from the Shri Ramdeobaba College of Engineering and Management, Nagpur, India. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur. Her current research interests include machine learning, deep learning, information retrieval, and its applications.

SHITAL RAUT received the Ph.D. degree from the Visvesvaraya National Institute of Technology, Nagpur, India, where she has been an Assistant Professor with the Computer Science and Engineering Department, since 2008. She has teaching experience of 17 years. She has also visited many institutes/universities around the world for invited lectures and collaborative research. She has many research publications. Her research interests include data mining, machine learning, computational biology, and bioinformatics.