

Compositional Feature Subset Based Ranking System (CFBRS) for Learning to Rank with User Feedback

¹Mohd Wazih Ahmad

*Research Scholar, Department of Computer Engineering,
Jamia Millia Islamia University, New Delhi, India.
Orcid: 0000-0001-5614-2591*

M.N.Doja², Tanvir Ahmad³

*^{2,3} Professor, Department of Computer Engineering,
Jamia Millia Islamia University, New Delhi, India.*

Abstract

This paper proposed a new method for learning to rank documents using a compositional feature subset based ranking system in the presence of implicit user feedback of various classes of the users. The objective of this research was to provide an alternative method for learning ranking functions using important compositions of the LETOR (Learning to Rank) features. The proposed model allows the search engine to dynamically utilize the implicit user feedback of the various classes of the users in learning ranking models repeatedly. The experiments performed on the LETOR MQ2008 dataset show that the proposed model gives better NDCG (Normalized Discounted Cumulative Gain) scores for the subsets of the users in the ensemble settings. Results also show that the variance of the predicted ranking can be controlled by controlling the hyper-parameters like the probability of the selection of a subset, feedback on the subsets and the weights of each subset used in training the low-level ranker. We have used cross-entropy pairwise learner RankNet and the maximum margin type svmRank as low-level rankers, but their objective functions are modified for the feature subsets. Results obtained by bagging and boosting based ensemble methods show that the proposed method is flexible enough to model a family of the feedback weights on the individual models and can be used to provide personalized rank learning functions to the selected subsets of the users.

Keywords: Learning to Rank, Subset Ranking, Feature-based Ranking, Ensemble Rank Learning.

INTRODUCTION

This research proposed a new technique for learning to rank documents using the compositions of the query-document features weighted by the user feedback on the subsets of features. Each subset of the features represents a class of users and a unique ranking model is learned for each class of the users. In traditional learning to rank methods, all the features are used in developing single rank learning function, which is ultimately used by all the classes of the users for prediction of the ranks of the new documents. Therefore, the traditional approach can be considered as a shared feature view of the relevance of the documents across the users in which every user is expected to see all the features before judging the performance of the search engine. But in practical situations,

most of the users see the subsets of the features of a document in deciding relevance and ranking of the documents. For example a user seeing a result set with the artistic nature focuses on the colors, fonts, calligraphy and other visible set of features to judge the relevance of the documents and in ranking them, while another user who is interested in the actual content of the documents may focus on the term frequency, word co-occurrences, sentence level features etc. Similarly there can be other classes of the users which can be identified using unique subsets of the features. The distribution of the ranks on the subset of the features may not be similar to the original distribution of the ranks on the full set of the features and therefore we need to learn each and every model in the feature subset space and also find the weight of each subset based model in the final ranking. In previous work, we have enumerated all the possible subsets of the feature set and used an ensemble of rankers to learn an average ranking for all the subsets of the features. This method has proven to be better than the shared feature ranking model as an ensemble of feature subset based models is able to represent most of the classes of the users with least variance between the feature subset based NDCG and the NDCG score of the aggregated ranking. But the main problem of the enumerative feature subset based ranking system is that due to the non-uniform distribution of the subsets in the power set, it was possible that the sampled subsets are biased towards a frequently occurring size of the subsets and hence particular class of the users is given more weight in training the ensemble models as compared to the less frequent classes of subsets.

In this paper, we tried to eliminate the problem of under-sampling of the subsets of features as well as to control the oversampling of the most frequent subsets of the features in rank learning. Learning to rank with the enumerative subsets of the features was proposed in [1]. A composition of an integer n is composed of the numbers summing to n . We hypothesized that using the compositions of the features as factorization mechanism we can generate all the possible ways in which various users can view the features. If there are n features, there are $2^{(n-1)}$ possible compositions for them. Each composition can be used to define a start and end index of the feature subsets coded by that composition. In this way, we have enumerated all the possible subsets of the features used in defining the unique compositions of the original feature set. These compositions

are used to train individual ranker and average ranking is computed by taking aggregate ranks of all the components of a composition. Each subset in a composition has its weight to be used in the ensemble, this weight is computed from the user feedback which is expected on that size of the feature subset. We hypothesized that the user feedback is a random variable composed of seven different factors including time to stay, number of clicks etc. User feedback is used to keep relevant documents on the higher ranks. We assumed that the subsets of the mean cardinality are most used features and therefore they get highest positive feedback, similarly extreme size subsets are least liked by the users and seldom a user see single or the full set of the features to judge the relevance of a document therefore, the probability of getting a positive feedback is least in those cases. On each cardinality, we define a normal distribution of the probability of getting positive feedback, the mean depends upon the number of subsets of that size and variance is selected as the parameter. For each round of the model training we sampled a composition, found the cardinality of each subset in the composition and used normal distributions of such cardinalities to generate the probability of getting positive feedback. The positive feedback probability is used as the weight of that subset in training the ensemble of rankers for that composition. This is repeated large number of times unless NDCG score of the aggregate ranks converges. For a small number of the features we can exhaustively use all the compositions and used a large number of user feedback iterations, but in the larger feature sets we need to search the composition space to get the representative ranking.

BACKGROUND

Learning to rank in the presence of list wise user feedback on the search results is defined as the problem of learning a hypothesis which maps from Q , the query features, D the document features, C a set of the feature compositions, α the given user feedback, Y the true labels of the documents in the list to a real score for each document in the list.

$$F: f(Q; D; C; \alpha; Y) \rightarrow R$$

As compared to our approach, the traditional shared feature learning to rank models like [2] [3] [4] are based on learning a mapping on whole feature set to a real score and therefore there is no way to utilize the implicit user feedback as well as no discrimination across the subsets of the users is possible in those models. The mapping function of the shared feature model is given by

$$F: f(Q; D; X; Y) \rightarrow R$$

In the proposed model, we used eight parameters for obtaining the average user feedback on all the documents of a list. These parameters contain information regarding the number of clicks received by the documents in a list, number of bookmarks, downloads, sharing and the proportion

of copying a document. The average feedback on a list is computed as a function ϕ of the implicitly observed user feedback parameters. Feedback value for each list i is computed as ai , it is normalized across the lists.

MOTIVATIONS

In case of the shared feature model of learning to rank, feature selection is performed in the literature as given in [5] and [6] but the objective of the feature selection was limited to finding more discriminating features out of the given features for all types of the users rather than the subsets of the users whose implicit feedback may give the different importance weights to active and inactive feature subsets. The objective of the shared feature model is to learn ranking model which minimizes the expected risk on the training and the test queries from the rank lists associated to the queries without considering the user preferences, feedback and the group behaviour. The main problems of the shared feature model which is a mostly used traditional model are that it fails to address the issue of the personalization of the search results and filtering of the documents based on the user feedback. Implicit user feedback in the form of the number of clicks, frequency of downloads, frequency of sharing, bookmarking, proportion of copying from a document and time to stay on a document can be used to quantify the importance of a document in the eyes of a user [7]. Feedback information can be used to identify the active and inactive subsets of the feature set, active feature subsets are those which are common in those documents which get consistently good user feedback while inactive feature subsets are common in the documents which get repeatedly negative user feedback. If the documents are ranked using all the features, it may not induce an optimum ranking for all the classes of the users, because different classes of the users judge documents on the different subsets of the original feature set. We consider each class of the users focuses on a particular subset of the features, and define a unique feature to label mapping.

RELATED WORK

In this section we discuss the representative work in learning to rank research reported in the recent past. Based on the design of the loss functions and the learned mapping, learning to rank algorithms can be divided into three categories named pointwise, pairwise and the listwise algorithms. We reviewed each category separately and discuss the comparative results and underlying methodologies in details.

Pointwise learning to rank methods

Pointwise learning to rank methods are those which consider the relevance score of the documents as a continuous dependent variable and features as the independent variables. Each instance of the query-document features in the sample is used separately in deciding the suitable parameters of the underlying model. These algorithms are designed to directly

compute the relevance scores and then sort these scores to rank the documents. Regression, classification and the ordinal regression are mainly used models in this class of learning to rank algorithms. Rankboost, PRank, SVM based ranking models are the real valued relevance score based models and belongs to the class of the pointwise learning to rank methods. The error bounds of these algorithms are derived from the union bound on individual errors, margin based error bounds in case of SVM, while stability and generalization bounds on the error of these algorithms can be derived using various theorems in probability and convergence. In the next subsection, we survey important penthouse models, their loss functions and corresponding optimization algorithms used. In [8] the problem of finding an optimal ranking of the top-k documents is considered in terms of the loss modelled as a discounted cumulative gain (DCG). Since optimization of the DCG criteria is a difficult problem both because of combinatorial nature, as well as the DCG is not a convex, smooth and continuously differentiable function over the ranks. Therefore the author has decided to transform the optimization of ranks into the equivalent optimization problem in two new formulations. In the first approximation, the author has used a surrogate regression loss of the DCG based scoring function and established the loose regret bounds on the ranking error. In another setting author converted the problem of ranking into a pairwise classification problem and computed the loss at the level of misclassified pairs, in this setting the zero-one classification loss is also minimized using a surrogate convex function. The regret bounds on the ranking error in terms of the classification error and the optimal Bayes ranking error is established. This method of optimization of the DCG driven ranking function is better than the simple regression based pointwise learning to rank technique because the number of documents is restricted to the top-k documents, all relevance labels are used in training, also the sample complexity of training the model is less. In case of regression based approximation, surrogate loss can be computed using ordinary least square loss function while in pairwise formulation the loss is computed by the convex surrogate of the weighted pairwise loss like hinges or logistic loss. Again, classification based optimization gives an opportunity to provide weights to the important pairs, or to use a predefined distribution over the preferences of the users and use in the training the models. Most of the bounds proposed in this work are loose in the sense that they do not quantify the asymptotically tight behavior of the approximations. Also the research fell short to discuss the sample complexity of the proposed models and feature based subset ranking. Upper bound on loss in case of regression based ranker is given by

$$DCG(r_f, S) - DCG(r_b, S) \leq \sqrt{2 \cdot \sum c_i^2} \cdot \sqrt{\sum (f(x_i, S) - f_b(x_i, S))}$$

Where $f(x_j; S) - f_b(x_j; S)$ represents the difference between the learned ranking function and the Bayesian score on pint x_j . This is achieved by optimizing the following loss function

$$argmin f \in F : 1/n \sum \left(\sum (f(x_j, S) - f_b(x_j, S))^2 \right)$$

This least square loss is a surrogate of the following DCG based loss function of ranking r over a subset S of the following type

$$DCG(r) = EsDCG(r, S)$$

Where

$$DCG(r, S) = \sum c^i E y_{ji}(x_{ji}, S) y_{ji}$$

Similarly, the pairwise classification based learner has the following bound on the ranking errors

$T(r_b, S) - T(r_f, S) \leq 4/\sqrt{m} \sqrt{\sum (f(x_j, S) - f_b(x_j, S))^2}$ This bound is achieved by optimizing following T measure as a surrogate loss

$$T(J, [y_j]) = 2/m(m-1) * \sum \sum (y_{ji} - y_{ji}^0)$$

In [9] the problem of finding relevance score of documents using features is considered as a polynomial regression problem. Quadratic polynomial is used for modeling complex relationships between requests and documents which cannot be approximated using simple linear least square regression. In n dimensional feature space this method first defines a vector space using following basis

$$\bar{V}(x) = \{1, x_1, x_2, x_3, \dots, x_1 x_2\}$$

The retrieval function or scoring function is defined as

$$e(x) = \{a_{1k} + a_{1k} * x_1 + a_{3k} * x_2 + a_{4k} * x_3, \dots, a_{n+3k} x_1 * x_2\}$$

Where a_{ij} are the coefficients of the polynomial terms. Different polynomials represent the complex relationship between the document features and the corresponding relevance scores, but higher powers of the polynomials have a tendency to over-fit, the optimum results were achieved using the quadratic polynomial. Parameters were fitted using the least squares approach for proposed polynomial regression model. For highly non-linear features we have identified that this formulation may not work and the kernel regression methods shall be a better choice. This method has the advantage over the probabilistic models of the ranking, as it gives directly the probabilities of the relevance of the documents with the queries, as well as this model is applicable on multivalued relevance judgment. In [10] the subset ranking problem is considered. The ranking problem of the subsets of the given documents is reduced to the smaller binary classification problems. The problems of finding a consistent ordering of the pairs of the documents with respect to a reference linear ordering using subset ranking are NP-complete as proven by [11], also they proposed a greedy subset ranking algorithm with the error bounds on the pairwise misclassification. In [10] the regret of the model is bounded by the loss expression of the purely random pairwise mistake bound. Regret bounds on the subset ranking is an extensively researched topic with various types of the

proposed loss functions and the formulation of the error bounds under a verity of the assumptions. The major limitation of these models is that their parameter estimation is difficult and these models only learn the approximate ranking. Another class of the pointwise learning to rank models is based on the classification methods to compute the direct scores of the documents using the output of the pairwise or the multi-class classifiers. Support vector machine based discriminative models for turning binary classifiers into the ranking functions are proposed in the past by improving the objective function of the SVM, modifying constraints and posing the problem of ranking as the problem of interleaving pointwise projections of the document scores on the weight vector. These methods do not directly optimize the pairwise concordance between the base ranking and the predicted ranking; instead, they compute the scores from the projection of the classifier outputs on the weight vector. These models work well as long as the ranks are distributed linearly in the features of the documents, but in case of the nonlinear and the non-uniform intervals between the rank boundaries the model performance goes down, if we try to increase the number of the parameters to learn the distribution within each possible rank interval, it may result in a drastic over-fitting. Within a linear rank boundary assumption, in [12] SVM was used to represent the ranking as a modified binary classification problem, It was shown that the empirical performance of the SVM based model is superior to the language modeling approach while the maximum entropy method used for computing the similarity was unable to perform well on the TREC dataset of the home page identification. The language models are probabilistic generative models which learn the joint probabilities of the documents and queries, while SVM being a discriminative model intend to learn the class conditional probabilities and is a more direct method of classification as it doesn't have an extra step of modeling the distribution. For modeling the problem of classification in SVM, they use a discriminant function with the kernel

$$g(R|D, Q) = w * \varphi(F(D, Q)) + b$$

Where,

$$g(R|D, Q) = \{ 1 \text{ if } D \text{ belongs to positive class else } 0.$$

The problem of finding optimum parameters which guarantee the maximum margin between the separating hyperplanes give birth to an optimization problem which is quadratic in the parameters and have a number of constraints equal to the number of the example data points. Above paper used SVM-light with a built-in solver. Non-linear rank boundaries can be easily represented under this formulation because it uses the kernel, however, the author has used the linear kernel and discussed the advantages of the quadratic kernels for the non-linear

cases. Quadratic optimization formulation for the SVM is given by the following objective function

$$\min 1/2 * |w|^2 + \lambda \sum \sum \xi^i$$

Subject to $w^T * x^j(i) \leq -1 + \xi_i$ if label is 0 and $w^T * x^j(i) \geq 1 - \xi_i$ if label is 1 each ξ is a positive quantity

representing the number of mistakes allowed by model. For maximum entropy based classification the following scoring function was used,

$$P(R|D, Q) = 1/Z(Q, D) \exp(\sum \lambda_{ir} * f_i(D, Q))$$

Where λ_{ir} are the weights. The performance of the SVM was better than the maximum entropy method as maximum entropy method failed to discriminate slightly different natural language statements in the documents but the SVM was able to capture even small non-linearity in the occurrence of terms. In [13] logistic regression based classifier is used for learning ranking functions, which define a cross entropy type objective function called log-odds ratio. Logistic regression is very good at modeling non-linear binary and multi-class classification problems by directly computing the probability of a document of being relevant to a query. The objective function of this model is given by

$$P(R|x_j) = 1/1 + e^{-c \sum w^T * x_{jt}}$$

This paper proposed the logistic inference as a mechanism to compute the relevance scores from six specially designed features. The model was trained on Cranfield dataset and tested against tf-idf method. The proposed model on comparison with the counterparts provided a statistically significant improvement in the performance computed on the recall and the precision measures. The paper also proposed to test the logistic model developed after training on one dataset to be tested on the other datasets, but in this setting, VSM (Vector Space Model) was found to be to be more accurate in the predictions. The author points out that if the objective of the logistic regression is modified to learn the mean and the variance of the standard term distribution, then it may be possible to train the model using one dataset and test on the other datasets. This hypothesis involves the assumption that most of the collections are generated by the same distribution, but they differ in the mean and the standard deviations with which terms are distributed. However, this assumption may not be true in the presence of a large number of the features as used in the modern learning to rank problem. The log-odds for a particular term $tk \in D_i, q_i$ is given by

$$\log O(R|q_i, d_j, t_k) = \{c_0 + c_1 v_1 + \dots + c_n v_n\}$$

While log-odds for a complete query containing q terms can be given by the sum of the log odds over all the terms minus the prior log odds of the relevance. The parameter can be learned using maximum likelihood as well as the gradient-based algorithms. Since by nature IR is an unbalanced training problem where a large fraction of the documents is not in the class of the relevant documents, therefore this paper proposed a method of sampling in which more weight is given to the positive samples as compared to the irrelevant samples. The log odd function can be written as

$$\log O(R|q_i, d_j) = \sum \log O(R|q_i, d_j, t_k) - \log O(R)$$

While in case of cross validation with a different dataset, the model could be modified as follows

$$\log O(R|q_i, d_i, t_k) = \{c_0 + c_1(x_i - \mu_{x1})/\sigma_1 + c_2(y_i - \mu_{x2})/\sigma_2\}$$

After computing the parameters, the inverse logistic regression function can be used to derive the probabilities of the relevance of the test sample. Another class of the pointwise methods is based on the multiclass classification and these models tend to directly order the documents on the absolute class probabilities or using a relative ordinal scale. In [14] multiple-classification is used as a technique for finding the expected relevance of the documents with respect to the given queries. Unlike page rank and Hub-authority scores, the author has created dynamic common features among the documents so that all the queries can be used in training the subsets of documents in ranking them. As compared to the regression-based upper bounds on the ranking errors, multiclass classification and the multiclass ordinal regression using logistic regression were found to give better bounds on the ranking error. The author has developed tighter ranking error bounds in terms of the underlying classification error as compared to the previous algorithms. Classification based algorithms are capable to model the preferences in more natural and direct way as compared to the regression-based algorithms for optimizing the DCG. The proposed McRank algorithm is based on the fact that the perfect classification of the documents in the corresponding label classes corresponds to the minimum DCG loss. This research used boosted tree model for finding the expected relevance of the each document to determine its class, while ordinal regression method does not directly find individual class scores, rather it finds relative ordering among documents. McRank algorithm has been shown to give better performance as compared to the pairwise methods like RankNet and LambdaRank, also tight bounds on the regression based DCG error are formulated. The ranker takes permutation of the documents and perform the following mapping

$$\pi : \{1, 2, 3, \dots, n\} \rightarrow \{1, 2, 3, \dots, n\}$$

Initially, the documents are given in random order, ranker has to order them according to the non-increasing relevance scores such that the DCG score is equal to the reference ranking of documents. If $Y = \{0, 1, 2, \dots, m\}$ is the set from which a label is given to each document, then the DCG score can be computed as follows

$$DCG = \sum c_{\pi i} * (2^{y_i} - 1)$$

where

$$c_i = \log 1/1 + i \text{ if } i \leq L \text{ and } c_i = 0 \text{ If } i = 0$$

NDCG is normalized DCG score over all queries. Author formulated a number of classes in the multi-class solution equal to the number of the relevance levels in the DCG

computation. The final bounds on the DCG error computed from the proposed classification model is the square root of the classification error.

$$DCG(g) - DCG(\pi) \leq 15/2\sqrt{(\sum c_i^2 - n \prod c_i^2/n)}\sqrt{(\sum 1_{y_i \neq \bar{y}})}$$

If class labels are mapped to the document relevance scores directly it solves the problem of ordering across labels but if multiple documents have the same label, it creates a problem because similarly classified documents are tied in the ranks. Therefore author proposed to model relevance score as a monotonically increasing function of the features. The author selected linear transform and used training examples to learn this function from the data. This was used to learn the expected relevance scores in case of the test examples. This model was fitted using boosted tree method while ordinal regression was modeled as a problem of learning to rank from cumulative probabilities of the relevance of each document. The proposed method proved to be better than RankNet, RankBoost, and LambdaRank. In [15] ranking is performed by ordinal regression, the author used perceptron update rule in learning the parameters of the proposed model. To represent individual classes, ordered threshold values are used to learn the model parameters in an online fashion, each threshold represents one class and a set of thresholds is defined to represent the given number of the relevance labels. If there are n labels in the relevance column, perceptron computes weighted values of feature so that the sum $\sum w^* x_i$ is the first threshold for which $w^* x_i \leq b_i$ where b_i is the threshold representing the true class of the data point x_i . In principle, if we use multiple thresholds, we may need multiple instances of the perceptron, but in this research, the author has used a vector of indicator binary variables for all possible thresholds to reuse a single instance of the perceptron for all documents of a query. For each feature instance, algorithm update weights in $w^* x_i$ as long as it does not cross the actual level of the data point. For example, if a data point has a level of 3 in the training set, hypothetically thresholds should be multiplied by an indicator vector $\{0, 0, 0, 1, 1, \dots\}$ because first zero is for only class-0 instances, second zero is for only class-1 instances, third zero is for class-2 instances and if $w^* x_i$ crosses all the thresholds b_0, b_1, b_2 , it will be able to see threshold b_3 . To cross this threshold, the parameter of the perceptron should be adjusted as much as that the sum lies in the interval of b_3 and b_4 . To learn this threshold for class 3, we need to update the weights as well as lower the value of the threshold, in these two-sided updates, the algorithm tries to decrease the interval in which features belonging to actual class lies. The update rule of the perceptron learning algorithm is used in the slightly modified form and uses a number of mistaken thresholds in update equation. This research has also provided time averages bounds on mistakes committed by the PRanking algorithm and it is found to be better than the Windrow-Hoff algorithm and MCP algorithm both in online and batch settings. To improve generalization performance, the large margin perceptron algorithm based ranker was proposed in [16] and

an SVM based ordinal rank learning model was proposed by [17]. Large margin principle has played a key role in the improvement of the classification accuracy of the various discriminative classifiers including perceptron, SVM, etc. Large margin principle is derived from the structural risk minimization theory and theory of VC dimension, according to this principle, the generalization error of a classifier can be minimized by maximizing the margin of its decision boundary from the data points. As an improvement to the Pranking algorithm proposed in [15], the large margin classifier is constructed by [16]. The bagging method is used to learn the rank boundary at the mean of each true rank interval, as the mean represents the maximum margin point of an interval. [17] modified the objective function in SVM is used to perform ranking from the given samples using large margin principle. In fixed margin approach author has taken the margin of the nearest pair of the classes and used these pairs to learn the hyperplane equation subject to maximizing the margin within the condition of separability. The thresholds used in this paper for various classes corresponding to the relevance labels are given by $B = \{b_1, b_2, b_3, \dots\}$ while w is the parameter of the SVM, the objective is to maximize $|w^2|$ which represent the margin after the normalization of the distance between the closest pair with the decision boundary. The formulation of the soft margin SVM is given in the form of following objective function

$$\text{Min } w, b_j, e^j, e^{j+1} * 1/2 |w|^2 + C \sum \sum e_i + e^{-j+i}$$

constraints are given by

$$w^* x_{ji} - b_j \leq -1 + e^j \text{ and } w^* x^{j+1}_i - b_j \geq 1 - e^{-j}_{i+1}$$

Two hyper-parameters are used to represent the data points inside the margin and to indicate the relative locations of the wrongly placed data points. Summation in the second part of the objective ensures that every rank boundary is taken care of and every closest pair is used in the objective function to determine the weights of the SVM. This loss function has a form of quadratic linear programming and its solution can be found using two-step optimization procedures like SMO or Lagrangian method of optimization. Author represented this objective with the Lagrange variables for constraints and solved using quadratic and the linear solvers. For a more general case, the same problem was formulated using quadratic kernels, this formulation gives better training and generalization performance. In the sum of margin strategy, the margins from all the pairs of the classes are considered and the formulation of the objective function contains one entry for each pair of the examples. However, this method is designed to solve the problem in one shot, but due to the large and complex objective function the scope of this formulation is limited to the small-scale ordinal regression problems only because in case of large-scale experiments the number of pairs may be very large and to perform comparative evaluation of these problems with the existing pairwise approached may become computationally hard. In the above section, we have discussed the various pointwise learning to rank algorithms. They are simple to use, have less number of parameters and

efficient in learning rank boundaries. All the approaches, including regression, classification and ordinal regression compute the absolute value of the relevance score and therefore suffers from the similar problem of identifying the correct rank boundaries and learning good parameters from the nonlinear rank boundaries of the target ranking. Also, in case of the similar relevance judgments, documents are tied on the ranks, which yields highly unstable rankings. As an alternative to the pointwise approach, researchers have proposed pairwise and listwise approaches for learning ranking functions. These algorithms differ from one another in the basic principle used, the definition of loss function or the objective function used, underlying model, training algorithm used and most importantly the assumptions and the constraints in which their performance was tested.

Pairwise Learning to Rank Methods

Pairwise learning to rank algorithms first computes all $n(n - 1)/2$ pairs of the documents from given labeled training set and use these pairs as input to learn the parameters of a total order generating ranking model. Later, the learned model is used to classify the test pairs and if all the pairs classified by the algorithm are consistent with their target labels, the algorithm is considered to have no error. The parameter learning problem doesn't have a close form solution in most of the cases; therefore the problems are formulated with the help of the loss functions which have a suitable structure for optimization. A loss function is defined over the set of features, their weights (i.e. parameters) and the distance of predicted ranking from the target rankings. Most of the pairwise algorithms rewrite the ranking level loss in terms of the classification loss and tries to minimize the surrogate loss functions of the binary loss function. Hinge loss, logistic loss and many other types of the surrogate loss functions with known approximation error from binary loss functions were used in the past. The design of loss function is specific to a particular problem and how the researchers have recognized the constraints, but the general principle in most of the pairwise learning to rank algorithm is to optimize proposed ranking under given preferences. We try to perform a survey of existing pairwise algorithms with emphasis on the formulation of the problem, objective function, and optimization, in some cases we also discuss the learnability and error bounds proposed by the authors. [11] identified that it is easy for a human to provide comparative judgments about the relevance of the two documents as compared to the absolute degree of the relevance of each document. Therefore, it is more intuitive to learn pairwise preferences directly into the model rather than the computing the degree of the relevance of each document or complete ranks from the training examples. The proposed method involves learning the preferences, which are already given in the form of pairs. They proposed a parametric model to compute a total order of the documents from the pairwise preferences. In fact, the problem of computing the total order from the binary preference aggregation is an NP-Hard, therefore, the author has proposed a greedy algorithm which guarantees at-least half good labels in the predicted ranks as compared to the target ranking. Let two documents d_1 and d_2 with features x_1

and x_2 such that $x_1 \leq x_2$ is the given preference relation between the two documents for a query q . The preference relation is defined on the set of all the documents. The loss of a learned function f with respect to the label y_{12} is given as

$$L(l, x_1, x_2, y_{12}) = |y_{12} - f(x_1, x_2)|/2$$

Where, $f(x_1, x_2) = 1$ if $x_1 \leq x_2$ otherwise 0. Similarly, y_{ij} is defined as a target preference label for given preference representing the order of the document pair x_i and x_j . The objective of this research was to minimize the disagreement between the target rank and the predicted rank using this loss function. They proposed weighted majority and winnow algorithm based rules for the optimization of the disagreements. The disagreement is defined on all the possible pairs which have a difference in the position of a document in the target pair and predicted pair $x_1 \leq x_2$. Disagreement is given by

$$Disagreement = \max_{\pi} \sum f\{\pi^{-1}(x_1) - \pi^{-1}(x_2)\}$$

This disagreement is minimized with the help of a greedy preference aggregation algorithm. We hope that if the dictionary based dynamic programming algorithm is used to construct the total order from the binary preferences, it will give an opportunity to reuse some of the partial orders in a more efficient manner and should yield better bounds on the disagreement between predicted ranking and the optimal ranking. In [1] author used pairwise preference relation to model the relevance of documents in response to a query. The author proposed a two output neural network whose one output is on at a time, each output refers to the one preference relation out of two possible between two documents given by i.e $x_1 < x_2$ and $x_2 < x_1$. Since this neural network tries to learn preferences, therefore the objective function of the neural network is defined as the loss between actual preferences to the predicted preferences for both of the neurons given as follows

$$L(f, x_i, x_j, y_{ij}) = (y_{ij} - f(x_1 > x_j))^2 + (y_{ij} - f(x_1 < x_j))^2$$

Learning is performed by the back-propagation algorithm and after training the three layers neural network, it was tested on the LETOR data set, the output of the network was sorted to give a final ranking. They named this algorithm as SortNet. The performance measures they have used to evaluate the SortNet algorithm are P@K, AP, MAP, and NDCG. This method does not consider the relative frequency of the occurrence of two types of the preferences in the learning. We believe that the two possibilities must be taken in proportion of their occurrence and the loss function should be modified to

$$L(f, x_i, x_j, y_{ij}) = \lambda(y_{ij} - f(x_1 > x_j))^2 + (1 - \lambda)(y_{ij} - f(x_1 < x_j))^2$$

In [18] the preferences are learned using arbitrary pairs of the documents computed from the flat training data. Also, this method learns a scoring function $f(\cdot, w)$ in a neural network, which scores each document for its relevance, this score is used to compute the preference of one document on other

document using an exponential function of the softmax type. The probability of predicting a preference is optimized using the gradient descent learning algorithm on cross-entropy loss function. For any two documents x_1 and x_2 the modelled relevance score is given by

$$P(x_1, x_2) = \exp((f(x_1) - f(x_2)) / (1 + \exp((f(x_1) - f(x_2))))$$

While the loss function is given as

$$L(f, x_1, x_2, y_{12}) = -\bar{P}_{12} * \log P_{12}(f) - (1 - \bar{P}_{12}) * \log(-P_{12}(f))$$

This loss function is considered to be more robust for the outliers as compared the square loss as used in the SortNet. Also cross entropy loss is convex, smooth and differentiable function making it easy to optimize using the gradient descent method unlike zero-one classification loss which is very hard to optimize. An algorithm with the improved performance, but non-convex loss function is proposed in [19], theoretically, this model guarantees better loss bounds but this loss was hard to optimize. In consequence, the loss was optimized using tree ensemble method. In another ensemble approach for ranking RankBoost was proposed by [20] which is based on the famous ensemble method called Adaboost. GBrank, the gradient boosting based ranker is proposed in [21]. Application of SVM in designing the pairwise learning to rank function using clickthrough data is proposed by [22]. The main problem in the pairwise setting is the preference information is lost in case of tied labels because the documents of the same labels will be classified into the same class, but there is no information that how they will be ranked together. Also, pairwise rankers are sensitive towards noise which led to the development of the robust rankers later by improving the design of the loss function. Also, some documents have a higher density of some query terms as compared to the others, causing imbalance. For some queries, more number of the documents are returned while for other queries very less number of the relevant documents are available in the collection. This cause imbalance in the distribution of query terms across the document collection to be nonuniform. This imbalance may influence the ranker in favor of some queries for which document density is higher. To counter this problem query level normalization was proposed so that the queries which have more number of documents should not cause the problem of biasing in the model parameters. In most of the ranking scenarios, a mistake on the higher positions of search results is more costly as compared to the mistake on the lower positions. In pairwise approach, some algorithms are proposed which takes care of discounted loss using DCG or OWA operator with existing loss function. In the next section, we have given loss functions for each of these methods and compared them on the basis of common attributes.

Listwise Learning to Rank Methods

In a ranking problem, ranks are discretely mapped to the relevance scores of the documents. If the ranks were continuous, it was easy to define a function of the ranks which could be optimized using standard gradient based

methods. In fact, most of the evaluation measures used to judge the goodness of a ranking as compared to the ground truth are also discrete in nature, including MAP and NDCG. If we consider a ranked list in its entirety and try to minimize the difference between predicted list and the target list it will become a combinatorial problem because of discrete nature of the scores. Since the number of permutations of a list grows exponentially as the size of the list grows, therefore it is computationally not feasible to optimize these lists combinatorially. Therefore, the researchers have designed a class of methods called listwise learning to rank methods, which tries to create smooth, differentiable and convex surrogate of the loss function for the whole list. Once the surrogate loss function is defined for a particular evaluation measure, it is used to learn optimal ranking. On the basis of measure used, smoothing method, classifier used and the error bounds there are various types of listwise algorithms proposed in past like SoftRank, AproximateRank, and smoothRank. Various types of specialized SVM models like SVM-MAP, SVM-ROC, SVM-NDCG etc. are developed in the past which directly tries to optimize the listwise losses of various IR evaluation measures by providing a surrogate loss function in the form of traditional SVM loss function. The performance of the listwise methods are proven to be better than the pointwise and the pairwise methods as they directly optimize the evaluation measure. In pairwise learning to rank approach the number of training pairs grows quadratically with the size of the collection. Therefore, training these models on huge training sets is a computationally expensive task. In Ranking SVM, the number of constraints grows quadratically as the number of the feature instances in the training set is increased. Also, All the pairwise methods give an indirect method of optimizing ranking level loss, by writing it in the form of the pairwise loss function. This may cause certain errors while using surrogate loss functions of zero-one binary classification loss. The listwise methods are intended to directly optimize ranking quality measure such as DCG and NDCG loss. Some of the representative works in list-wise learning to rank is described here. In [24] author proposed an algorithm called SoftRank which tries to directly optimize the NDCG. Since NDCG is rank position discounted method of evaluation of IR results, therefore it is a discontinuous, nonsmooth and non-differentiable function of the relevance scores. Above research provides a smoothed version of NDCG by modeling a normal distribution across the ranks for computing NDCG scores. The approximate normal distribution makes it possible to convert the NDCG function as a continuous function of ranks, smoothing is controlled by mean and variance of normal distribution used to design surrogate NDCG. Since infinite possible realizations of NDCG are possible in this case, therefore, expected value is taken as representative. Author, assumes x_i is the document for a query q , the score of a document is approximated with a normal distribution given by

$$p(s_i | q) = N(s_i | f(x_i, q), \sigma^2),$$

Where σ is a parameter fitted for modeling the variance. Since scores are probabilistic, therefore ranks are also non-

deterministic and there is a probability associated with each pair of the documents to be ranked in both ways. We can find the probability of each pair to be ranked in a consistent way to target ranking as well as in opposite ordering. After expectation of a pair of the documents to be ranked against each other is computed, next documents can be ranked iteratively based on the probability of beating any one of previously ranked documents. The expected value of the probability of order $x_1 < x_2$ under the standard deviation of σ is given by

$$P(x_1, x_2) = \int N(s | f(x_1) - f(x_2), 2\sigma^2) * ds$$

After rank distribution is computed, the researcher tries to maximize the expectation of NDCG or minimize $(1 - \text{softNDCG})$. The smooth form of the loss function is given by following equation

$$L(f, x, y) = 1 - 1/Z \sum (2^{y_i} - 1) * \sum \eta(r) Pi(r)$$

Where f is the estimated ranking model. This model is fitted using the gradient descent algorithm. In [25] NDCG can be written in terms of multiplication of a label wise gain function $G(y_i)$ and a position wise discount functions $\pi(x_i)$ where the position wise discounted function is a discrete function of the ranks. Author tries to smooth position discount by using an approximation of the discrete discount with a softmax function given by

$$\pi(x_i) = 1 + \sum \exp_\alpha(f(x_i) - f(x_u)) / 1 + \exp_\alpha(f(x_i) - f(x_u))$$

In this case, approximate NDCG is maximized using the following surrogate loss function using the gradient descent algorithm

$$L(f, x, y) = 1 - 1/Z \sum G(y_i) / \log 1 + (\bar{\pi}_i)(x_i)$$

Since $\pi(x_i)$ is a continuous function, therefore, approximate NDCG loss becomes smooth and differentiable. It has been found that the empirical performance of the listwise loss function is better than the predecessors like pointwise and the pairwise learning to rank models. In [26] a boosting algorithm known as Adarank is proposed which is based on the AdaBoost algorithm. Adarank uses a non-smooth version of the optimization of the NDCG measure using several weak models in an iterative fashion. These methods try to generate most fit ranking using the combinatorial approach with local gradient information in the non-smooth version of the NDCG. The algorithm defines a probability distribution on the examples and trains a weak learner on the samples taken from the initial distribution, after computing the loss on the current measure, it tries to compute the weight for the current model in the ensemble. After the current weak model is incorporated into the ensemble, the algorithm tries to increase the weights of wrongly ranked pairs and decrease the weights of rightly ranked pairs. The new distribution is used to learn another weak model which is guaranteed to improve the measure. Finally, a weighted sum of all the classification rules is output as a learned model. In addition to the design and optimization of the new loss functions, various other researchers have proposed feature selection for learning to rank problem,

dimensionality reduction, relational ranking, query-dependent ranking, semi-supervised ranking, transfer ranking etc. The survey of these algorithms and applications of learning to rank algorithms are out of the scope of our current work. In our paper, we have considered the problem of finding a consistent ranking function for all possible compositions of the features using an ensemble of pairwise rankers like SVM-rank and RankNet as our base learners as they are more intuitive in modeling human preferences on the pairs of the documents. Our method can be used to optimize the list-wise and any other surrogate loss function.

Mathematical Formulation of the Problem

We have a set of queries $Q = \{q_1, q_2, \dots, q_n\}$ and for each query q_i the retrieved set of the documents is given by $D = \{d_{i1}, d_{i2}, \dots, d_{in}\}$. There is a label assigned to each document from a set of labels, where each label y_{ij} takes a value of. Let there are n number of features in the description of each document, represented by a set $X = \{x_1, x_2, \dots, x_n\}$. Each subset of features in composition $c_i \in C$ represents a class of users supported by the search engine. The objective of this research is to learn a ranking model $f_i: (q, D, C, \alpha, Y) \rightarrow R$ that maps the labelled query-document pairs in i^{th} composition of features with their given ranks with minimal loss on concordant pairs of documents in case of pairwise classification model weighted by user feedback α . In case of listwise models, the objective is to minimize the listwise loss for each feature composition. The ensemble model f is learned by aggregating $f_i \in F$ where F is the collection of learners for compositions of features. The objective of f is to minimize loss of aggregate ranks with the target ranks learned by considering all compositions. The ranks learned from individual subsets of features in a te composition should be aggregated in the proportion of their subset sizes and feedbacks. Finally, aggregate ranks should be compared with ranks learned from a full set of features as well as the rank learning functions designed from ensemble methods on subsets of features.

Definitions

Definition 1: Shared Feature Model A model $M(X, \theta, Y)$ where θ is the set of the parameters, X is the feature vector and Y is the list of labels given to documents then model M is called shared feature learning to rank model if each $x \in X$ is a member of objective function and is used to learn the optimum value of θ in learning Y .

Definition 2: Mean Average Precision (MAP) Mean average precision is an evaluation measure generally used by search engines to define the proportion of the relevant documents in the search results. Precision at k is defined as

$$P@k(\pi, l) = \sum I_{\pi-1=1/k}$$

Average precision is defined on the all documents of query q , let there are m documents for q then,

$$AP(\pi, l) = \sum P@k(\pi, l) * I_{\pi-1=1/m}$$

MAP is the mean of average precision over all queries.

Definition 3: Normalized Discounted Cumulative Gain (NDCG) Discounted cumulative gain (DCG) is a family of position discounted evaluation measures originally proposed by [27] in the year 2002. For a given query q , if π is the predicted list and l the relevance grades, Z is a normalization constant, then G is the gain function which defines the gain on different relevance labels and η is a discount function. Generally, a discount function tries to give higher penalties for mistakes done at higher ranks of the predicted list as compared to the mistakes made on the lower ranks.

$$DCG(\pi, l) = 1/Z \sum G(l_{\pi-1})\eta(j)$$

NDCG@ k is a ratio of DCG@ k of predicted rank and DCG@ k of the ground truth. By careful choice of gain and discount functions, a number of NDCG@ k measures can be defined.

Definition 4: User Feedback Let α_i be the user feedback computed over implicitly observed feedback parameters on search results given by a vector $\alpha = \varphi(\mu_{1i}; \mu_{2i}; \mu_{3i}; \mu_{4i}; \mu_{5i}; \mu_{6i}; \mu_{7i}; \mu_{8i})$ Where φ is a heuristically defined function which maps positive and negative user feedback parameters on the presented search result at the list level to a real value.

The positive feedback parameters are given by

- μ_{1i} = Average number of clicks on a document in the rank list
- μ_{2i} = Avg. number documents made bookmarks in the list
- μ_{3i} = Avg. time to stay on a document in the list
- μ_{4i} = Avg. number of downloads from the list
- μ_{5i} = Avg. number of sharing of documents from search result

Negative feedback parameters which can be implicitly observed are given by

- μ_{6i} = Avg. number of documents ignored from the list
- μ_{7i} = Avg. number of times a back button is used on presenting a list.
- μ_{8i} = Avg. number of times a document of the list is closed within 5 seconds of opening.

After normalization on simulated feedback data, for a composition i , we get $\alpha_i \in [0,1]$, the feedback on a composition is used to compute the feedback on each subset

contained in the composition. In this research, we have divided feedback among each subset in proportion to their cardinality in a composition.

Definition 5: Composition Let $X = \{x_1, x_2, x_3, \dots, x_n\}$ be the set of the features, then we define a set of all possible compositions C of the feature set X where a composition of the feature set X is a factorization of the feature set in such a way that the sum of cardinalities of all the subsets generated by the compositional factorization is equal to the original cardinality of the feature set. Since cardinality of the feature set is an integer, therefore by the principle of integer composition we can say that there are $2^{(n-1)}$ unique feature compositions possible for a feature set of the size n .

Compositional Feature Subset based Ranking System(CFBRs)

The proposed model named as CFBRs is based on the **enumeration of integer compositions** which is a well-researched topic in the enumerative combinatorics. The composition of an integer n is defined as an ordered set of subsets which add to n . We used the recursive composition algorithm with given feature set and find all the subsets of the query-document features forming valid compositions. For each subset of the features in the composition, there exists a one to one mapping with respect to a group of the users. The rank lists for each subset of the composition of the features are prepared by taking composition columns from corresponding shared feature sets of the training, validation and the test data. There are two possible labelling of the documents, first is the global labeling which is also used in the shared feature model as target ranking. But the way one group of user map features to labels may be different from another user group as well as from the global relevance scores given in the shared feature models. Therefore, we decided to include this aspect by defining a non-uniform distribution of the ranks over the subsets of the features. The ranks generated by this model are compared with the traditional shared feature model. Similarly, we used a user group specific ranking of the documents and test the performance of the learned ensemble model with respect to each user. We also performed this experiment on the shared feature model and shown that the feature composition based model provides more flexible architecture as we can apply various types of the filters on the user groups whose expected error on the training rank lists is more than some threshold. The Proposed model is based on an ensemble of pairwise rankers trained on the unique composition $ci \in C$ of the feature set in the presence of the implicit user feedback ai . A learner takes input ranked lists and the ground truth to compute and evaluate the ranked lists on all the subsets of a composition. Each subset of a composition is used to learn a ranking and have its NDCG@k computed. User feedback given to each document is divided among the constituent subsets of a composition in the fraction of cardinalities. Each rank list is weighted by this fraction of user feedback and its NDCG@k is multiplied with this feedback to obtain a maximum possible gain in NDCG under feedback

assumption. Finally, the weighted models are aggregated using a suitable method (here we used average NDCG@k method) like Kendall's tau and Spearman's footrule ranking. We defined the population risk as the summation of the individual compositional risks on the whole population (each possible composition of the feature subset).

$$Risk = \int_{c^*y} Pr(dc, dy) * f(loss(Y, M(ci)), \alpha_i)$$

Where $M(ci)$ is the model learned from the composition ci with the feedback ai , Y the target labels, f redefines the risk of each composition and the loss of underlying pairwise model in the light of user feedback as

$$f : loss * feedback \rightarrow risk$$

The modified loss function of a composition learner f_i of the cardinality N can be written as

$$1/N \sum_{S^c * Y} L(f, c, r^*) = n1 / \alpha_1 * L_1(f_1, s_1, r^*) + n2 / \alpha_2 L_2(f_2, s_2, r^*) \dots n_k / \alpha_k L_{|k|}(f_k, s_k, r^*)$$

where, $ci \in C$ is the composition such that after factorization, it forms the subsets s_1, s_2, s_k of the cardinalities n_1, n_2, n_k so that $n_1 + n_2 + \dots + n_k = N$ and, L_1, L_2, \dots, L_k are the loss functions of each subset based ranker with respect to the target ranks r^* , L is the loss function of the composition ranker. $\alpha_1, \alpha_2, \dots, \alpha_k$ are the feedbacks on the subsets of the composition ci . Since we only have a single feedback on a composition ci so we computed each subset based loss function which gets this feedback ai in the proportion of the cardinality. Also, good feedback indicates less loss, therefore we divided individual subset loss by the fraction of the composition feedback weighted by the cardinality of that subset, resulting into the following equations:

$$\alpha = n_1 * (\alpha / N) + n_2 * (\alpha / N) \dots + n_k * (\alpha / N)$$

$$1/N \sum_{S^c * Y} L(f, ci, r^*) = N / \alpha n_1 * L_1(f_1, s_1, r^*) + N / \alpha n_2 * L_2(f_2, s_2, r^*) \dots N / \alpha n_k L_{|k|}(f_k, s_k, r^*)$$

We can minimize this compositional loss by minimizing the individual loss and before aggregation of the individual subset loss, we should multiply the scores by a fraction of feedback. In fact, the loss is divided by the fraction of the feedback as loss and the feedback are inversely related. But in case, if we first compute the NDCG@k on each subset model with the minimum loss, we can multiply the gain with the fractional feedback factor on each component of a composition and this weighted average represents an upper bound on the NDCG@10 possible under this composition. The ensemble of all the compositions can be found in the same way. In the equation, one more summation represents that each

compositional model is treated as a weak learner for the final ensemble learner.

$$1/N \sum_{s \in Y} \sum_{i=1}^{|c|} L(f, c_i, r^*) = N / \alpha_{i1} L_1(f_1, s_1, r^*) +$$

$$N / \alpha_{i2} L_2(f_2, s_2, r^*)$$

$$\dots N / \alpha_{ik} L_{|k|}(f_k, s_k, r^*)$$

Above equation represents an ensemble of the compositional models, while each compositional model is learned by minimizing the weighted sum of the risks of the individual ranking models on the subsets. The underlying rank list learner can be based on the pointwise, pairwise and the listwise approach and its parameters can be learned to minimize the corresponding loss function. In this paper, cross-entropy based pairwise classification model (RankNet) modified for weighted compositions of the feature set is used to learn the patterns into a set of the parameters of ANN. Learned model can be used for prediction of the rank lists for the new documents, the evaluator is used to judge the performance of the learned model against ground

truth of the new documents. The architecture of the proposed ranking system is given in fig 1.

Architecture of CFBRs

We used bagging as an ensemble technique and sampled the compositions of each cardinality subsets from the population of the feature compositions. An exhaustive list of the compositions is given for five feature set as an example. It is possible to learn all possible weighted compositions for smaller feature sets, but very difficult to learn all possible compositions of the large feature sets because a number of compositions is exponential in the size of the feature set. In that case, our model relies on a sampling of the composition space. Our hypothesis is that, if the sample is the representative of the population of the compositions, we can guarantee some kind of upper bound on the generalization error of our ensemble model. The figure 1 represents the architecture of our bagging based CFBRs ensemble learner.

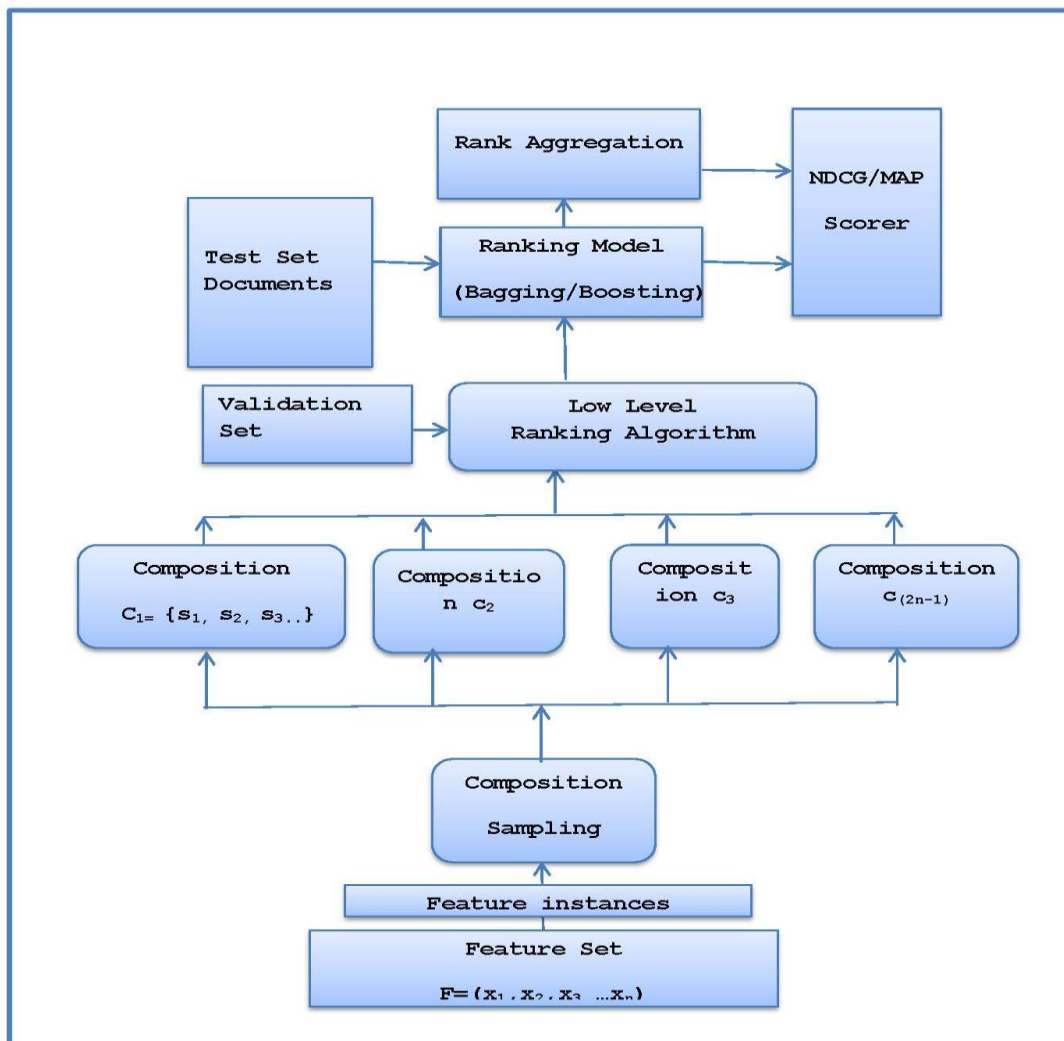


Fig 1: Architecture of CFBRs

Experiments on feature composition Models

The composition of a feature set of size n is given by C where $|C| = 2^{n-1}$. Each composition of the feature set covers all the features by making the boundaries of the subsets at the different elements of the feature set. The obtained compositions can be used to represent various users who have different choices on the documents. Feature composition of the size k is the maximum number of ways in which a set can be factored into k components so that the sum of the cardinalities of the components is equal to the size of the feature set. In our research, we used this property of the compositions to represent users who judge the documents on a different set of features. For $n=5$ there are totally 15 compositions.

Each composition is an ordered set of the numbers which sum to 5. We used feature compositions and learned 15 different models for 5 feature dataset. The number of features used in learning each component model is defined by the composition of the feature set. We selected feature composition method because it is a simple and a powerful method for enumerative learning to rank models as it gives samples of all possible ways in which a feature set can be used to train the ensemble models. For feature composition model with only five features, the size of each composition is very small such that we had to train base learners on a single feature much time, which is not a standard way of training the rankers. Therefore, for actual experiments on the feature composition based ranking models, we have trained the compositional models on the 46 features of LETOR 4.0, with a minimum factor of size 5 to 46. For 46 features, there are 2^{45} possible compositions. It is a very large number of the models as each composition is a unique set of elements which add to 46. We used samples of the compositions and learned individual models on the subsets of the features belonging to a composition and then combined the results using the averaging methods. We have used the rank aggregation methods to combine ranking of the various models. For example, a particular composition $\{1, 2, 1, 1\}$ represents that this user can use different ordered subsets of the features of sizes 1, 2, 1, 1 in order to judge the documents. The first feature is used to judge some documents, then next two features, then the next one and again next one feature to judge the same document list. Feedback on all the documents is averaged to find the feedback on a composition, which is later divided into the components of the composition. Next subsection provides the compositions of 46 features with factors 5 to 46. We used these compositions exhaustively in learning the bagging model called CFBRs.

Compositions of LETOR

LETOR 4.0 MQ2008 dataset has 46 features, the total number of compositions of this data set must be $2^{(45)}$ which is equal to 3,51,84,37,20,88,832. In the exhaustive compositional model, we must take the minimum subset size of 1. But as we know that due to the computational power needed for such a huge number of models to be learned, we decided to limit the size of a factor in sampled compositions. With minimum factor size of 5, there are 51320 total compositions out of 46

features, with factor size 10, there are 265 compositions and only 8 compositions of minimum factor size of 20. Here, we give the results of training the ensemble model with the first composition of each factor size starting from 5. In fact, this will work as a baseline for better sampling plans and also a comparative starting point for the shared feature model which is trained on 46 features together. In our experiments, the compositions of the minimum factor size 5 are taken and composition of each cardinality is taken at-least once while performing the bagging. Table 1 provides the details of each experiment in the compositional setting as well as in the shared feature model.

RESULTS AND DISCUSSIONS

In table 1, it can be observed that the Training time NDCG on various compositions lies between the values of .40 to .49 which on bagging gives the NDCG@10 value of .45. Similarly, bagging of individual compositions gives average NDCG@10 of .49 and .44 for validation and test data. While the original shared feature model gives NDCG value of .48 for all the compositions. We can further enhance the compositional model by appropriately weighting the importance of frequently used compositions. The Shared feature model gives more NDCG@10 for those compositions which it should not and it gives less NDCG@10 for Those compositions for which it should give higher NDCG. This is because in SFM we tried to satisfy all the compositions with single predicted rank. But in our model we can see, it better fits individual compositions, as well as it gives the opportunity to combine NDCG scores of individual rankings generated using bagging of individual composition models.

Table 1: Results on the compositions

Min Composition size	Training NDCG@10	Validation NDCG@10	Test NDCG@10	SFM-NDCF@10
5	0.40	0.44	0.40	0.48
6	0.40	0.43	0.38	0.48
7	0.40	0.47	0.42	0.48
8	0.43	0.48	0.43	0.48
9	0.43	0.46	0.43	0.48
10	0.44	0.48	0.43	0.48
11	0.43	0.48	0.43	0.48
12	0.45	0.50	0.45	0.48
13	0.45	0.50	0.45	0.48
14	0.45	0.49	0.45	0.48
15	0.46	0.49	0.45	0.48
16	0.44	0.49	0.45	0.48
17	0.43	0.49	0.43	0.48
18	0.44	0.48	0.43	0.48
19	0.45	0.49	0.45	0.48
20	0.45	0.48	0.44	0.48
21	0.46	0.50	0.46	0.48
22	0.48	0.52	0.45	0.48
23	0.48	0.52	0.47	0.48
24	0.48	0.52	0.47	0.48
25	0.48	0.52	0.47	0.48
26	0.47	0.51	0.47	0.48
27	0.47	0.52	0.46	0.48
28	0.48	0.52	0.47	0.48
29	0.48	0.52	0.47	0.48
30	0.48	0.52	0.47	0.48
31	0.48	0.52	0.47	0.48
32	0.48	0.52	0.47	0.48
33	0.48	0.52	0.47	0.48
34	0.48	0.52	0.47	0.48
35	0.48	0.52	0.47	0.48
36	0.48	0.52	0.47	0.48
37	0.48	0.52	0.47	0.48
38	0.48	0.52	0.47	0.48
39	0.48	0.52	0.47	0.48
40	0.48	0.52	0.47	0.48
41	0.48	0.52	0.47	0.48
42	0.48	0.52	0.47	0.48
43	0.48	0.52	0.47	0.48
44	0.48	0.52	0.47	0.48
45	0.48	0.52	0.47	0.48
46	0.48	0.52	0.47	0.48

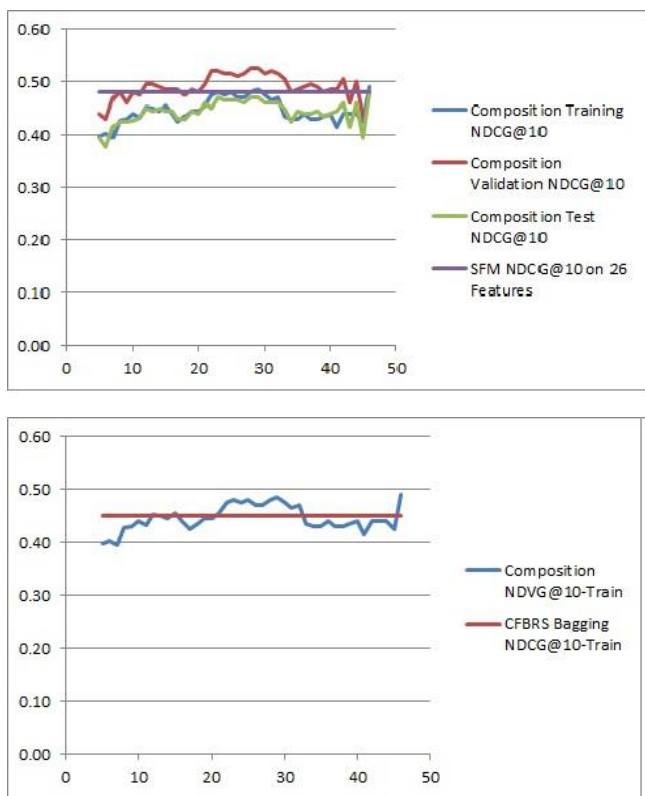


Figure 2: Bagging with training and validation

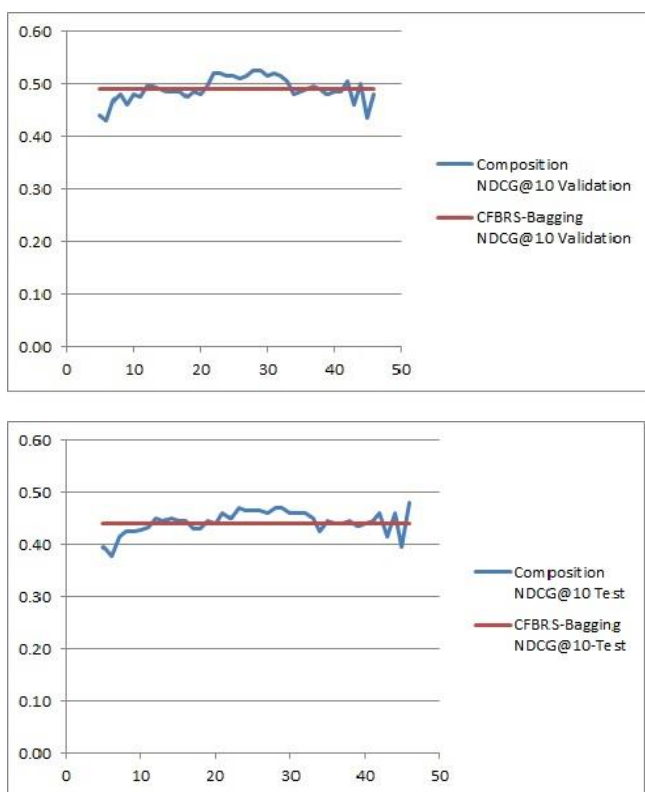


Figure 3: Bagging vs. Boosting against Avg.NDCG@10 and Shared feature model

CONCLUSIONS

In this paper, we proposed a new model for learning to rank documents in accordance with the feedback given by the population of users, called compositional feature based ranking system. The above model learns the parameters in accordance with the ranking generated by each weighted composition with the user feedback. A new loss function is designed to perform ensemble learning of the composition rankings. Bagging is used as a method of ensemble and the average value of NDCG@10 is computed on each composition. For the sake of baseline, we have taken a uniform feedback distribution across various compositions of the feature set and generated the results. The results show that the proposed compositional model better fit on the subsets of the features in terms of the NDCG@10 of the produced ranks, Also it shows that the average value of the NDCG score of the aggregate ranking can be taken as the final score for all the subsets of the users. The aggregate ranking can be computed using more sophisticated methods like Kendall's tau and Spearman's rho in order to better fit with the user interest. The feedback from the users can be used to give a more discriminating NDCG@10 score for the rankings generated by the compositions of a feature set in order to generate better-fitted ranking models.

REFERENCES

- [1] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, "Sortnet: Learning to rank by a neural preference function," *IEEE Transactions on Neural Networks*, vol. 22, no. 9, pp. 1368-1380, Sept 2011.
- [2] F. Radlinski and T. Joachims, "Query chains: learning to rank from implicit feedback," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM,2005, pp. 239-248.
- [3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452-461.
- [4] T. Joachims and F. Radlinski, "Search engines that learn from implicit feedback," *Computer*, vol. 40,no. 8, 2007.
- [5] H.-J. Lai, Y. Pan, Y. Tang, and R. Yu, "Fsmrank: Feature selection algorithm for learning to rank,"*IEEE transactions on neural networks and learning systems*, vol. 24, no. 6, pp. 940-952, 2013.
- [6] B. Bai, J. Weston, D. Grangier, R. Collobert, K.Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger, "Learning to rank with (a lot of) word features," *Information retrieval*, vol. 13, no. 3, pp. 291-314,2010.
- [7] M. S. Beg and N. Ahmad, "Web search enhancement by mining user actions," *Information Sciences*,vol. 177, no. 23, pp. 5203-5218, 2007.

- [8] D. Cossock and T. Zhang, "Subset ranking using regression," in *Proceedings of the 19th Annual Conference on Learning Theory*, ser. COLT'06. Berlin, Heidelberg:Springer-Verlag, 2006, pp.605-619. [Online]. Available: <http://dx.doi.org/10.1007/11776420.44>
- [9] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Trans. Inf. Syst.*, vol. 7, no. 3, pp. 183-204, Jul. 1989. [Online]. Available: <http://doi.acm.org/10.1145/65943.65944>
- [10] N. Ailon and M. Mohri, "An efficient reduction of ranking to classification," *arXiv preprint arXiv:0710.2889*, 2007.
- [11] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," *CoRR*, vol. abs/1105.5464,1999,2011. [Online]. Available: <http://arxiv.org/abs/1105.5464>
- [12] R. Nallapati, "Discriminative models for information retrieval," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 64-71. [Online]. Available: <http://doi.acm.org/10.1145/1008992.1009006>
- [13] F. C. Gey, "Inferring probability of relevance using the method of logistic regression," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc., 1994, pp. 222-231.
- [14] P. Li, Q. Wu, and C. J. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in neural information processing systems*, 2008, pp. 897-904.
- [15] K. Crammer and Y. Singer, "Pranking with ranking," in *Advances in neural information processing systems*, 2002, pp. 641-647.
- [16] E. F. Harrington, "Online ranking:collaborative filtering using the perceptron algorithm," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 250-257.
- [17] A. Shashua and A. Levin, "Ranking with large margin principle: Two approaches," in *Advances in neural information processing systems*, 2003, pp. 961-968.
- [18] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89-96.
- [19] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma, "Frank: a ranking method with fidelity loss," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 383-390.
- [20] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol. 4, no. Nov, pp. 933-969,2003.
- [21] Z. Zheng, K. Chen, G. Sun, and H. Zha, "A regression framework for learning ranking functions using relative relevance judgments," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 287-294.
- [22] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 133-142.
- [23] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*,pp.1189-1232, 2001.
- [24] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "Softrank: optimizing non-smooth rank metrics," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 2008, pp.77-86.
- [25] T. Qin, T.-Y. Liu, and H. Li, "A general approximation framework for direct optimization of information retrieval measures," *Information retrieval*, vol. 13, no. 4, pp. 375-397, 2010.
- [26] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*.ACM, 2007, pp. 391-398.
- [27] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422-446, 2002.