ES-Rank: Evolution Strategy Learning to Rank Approach

Osman Ali Sadek Ibrahim
ASAP Research Group,
The University of Nottingham
CS Dept., Minia University, Egypt.
psxoi@nottingham.ac.uk

Dario Landa-Silva
ASAP Research Group,
School of Computer Science
The University of Nottingham
dario.landasilva@nottingham.ac.uk

ABSTRACT

Learning to Rank (LTR) is one of the current problems in Information Retrieval (IR) that attracts the attention from researchers. The LTR problem is mainly about ranking the retrieved documents for users in search engines, question answering and product recommendation systems. There are a number of LTR approaches from the areas of machine learning and computational intelligence. Most approaches have the limitation of being too slow or not being very effective. This paper investigates the application of evolutionary computation, specifically a (1+1) Evolutionary Strategy called ES-Rank, to tackle the LTR problem. Experimental results from comparing the proposed method to fourteen other approaches from the literature, show that ES-Rank achieves the overall best performance. Three datasets (MQ2007, MQ2008 and MSLR-WEB10K) from the LETOR benchmark collection and two performance metrics, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) at top-10 query-document pairs retrieved, were used in the experiments. The contribution of this paper is an effective and efficient method for the LTR problem.

CCS Concepts

•Information systems → Learning to rank;

Keywords

Learning to Rank; Evolution Strategy; Machine Learning; Information Retrieval

1. INTRODUCTION

In information retrieval (IR), ranking the retrieved documents with respect to their relevance in response to the user's query is very important in order to satisfy the user's information needs. This is known as the *Learning to Rank (LTR)* problem in IR. In order to tackle this problem, earlier approaches include term scoring methods such as TF-IDF, Okapi-BM25 and language models among others [19]. However, a limitation of using only one scoring method in IR systems is that such approach is not effective enough in order to retrieve the most relevant documents. Moreover, learning models such as Okapi-BM25 and language models rely considerably on the relevance judgment in order to achieve good retrieval results [11, 25, 26]. This limitation has inspired the use of more than one scoring method for ranking documents. Moreover, using additional features that represent the quality (e.g. reputation of the source in the web) of the retrieved documents according to the user's query helps to accomplish a more effective ranking. Recently, Tao Qin et. al. [23] proposed the LETOR datasets which have been widely adopted by IR researchers interested in document ranking techniques. Each dataset in LETOR is a distilled benchmark originating from search engines and from the well-known TREC conference document collections. These benchmarks contain more than one scoring weighting scheme as part of the benchmark features. They also contain some other features that indicate the importance of the documents on the web. The documents in the LETOR datasets were mapped into fully judged query-document pairs suitable for conducting research on the LTR problem. Hence, given that the LETOR benchmark collection has been used in various previous works proposing LTR methods, three of these datasets are used for the experiments in this paper. A number of machine learning and computational intelligence approaches have been proposed to improve the ranking based features. Section 2 provides more on the background of the LTR problem in IR, while section 3 presents a brief literature review of related work in this subject.

The intended contribution of this paper is to present an Evolutionary Strategy (ES) to tackle the LTR problem. The proposed method is called ES-Rank and consists on evolving a vector of weights where each weight represents a desirable feature. The mutation step-size in ES-Rank has been tuned based on preliminary experimentation. Details of the proposed method are presented in section 4. In order to assess the performance of ES-Rank, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) [23] are used and comparison is carried out against fourteen state-of-the-art LTR approaches from the literature. Experimental results in this paper show that ES-Rank performs very well when compared to those other methods in

SAC 2017,April 03-07, 2017, Marrakech, Morocco

http://dx.doi.org/10.1145/3019612.3019696

terms of MAP and NDCG. Furthermore, most of the other methods consumed very long computation time while ES-Rank was much faster. For example, some of the other methods consumed more than 9 hours on each MSLR-WEB10K dataset fold [23] while ES-Rank consumed only around 30 minutes on each fold. Another feature of ES-Rank is that it has small memory requirements according to the problem size ($2XM$ dimensions where M represents the number of features in the training dataset). Experimental results are presented in sections 5 while conclusions and proposed future work are given in section 6.

2. BACKGROUND

A LTR dataset consists of query-document pairs for a large number of queries [23]. This is illustrated in Table 1 showing the representation of several query-document pairs. Each pair contains a relevance label indicating the relevance degree of the document for each query. In most cases, the relevance labels are binary where 1 means relevant and 0 means irrelevant. There is also a query identifier (id) indicating the corresponding query for each query-document pair. The feature vector refers to M other features such as Term-Weighting Scores (e.g. TF-IDF, Okapi-BM25 and Language Models [23]), PageRank and Host Server Importance, features associated to each query-document pair in the LTR dataset. Each feature in the *Feature Vector* has the form FeatureID:FeatureValue, where FeatureValue contains the contribution value of this feature in the query-document pair. The dataset itself is divided into N folds (usually it contains 5 folds) and each fold contains training, validation and testing set of the query-document pairs. These folds are useful for examining the LTR algorithm behavior and its predictive performance by applying it on test datasets different to the training datasets. More details about LTR datasets and query-document pairs are in [22, 23].

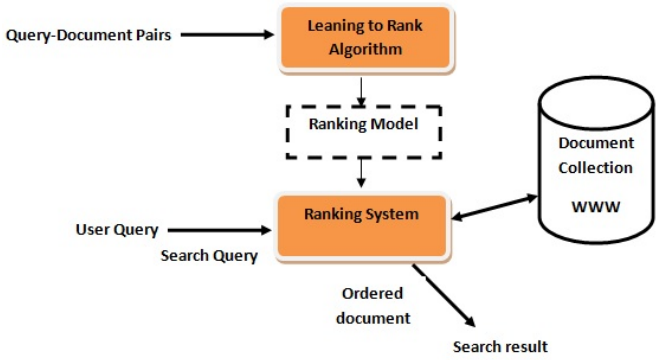

Figure 1: Learning to Rank (LTR) Approach Architecture as Discussed in [18].

In recent years, LTR as a supervised learning-based method has been widely used in IR to produce ranking functions based on the training datasets. The ranking function is used to rank the retrieved documents in response to the user's query. Figure 1 shows the general LTR approach architecture that most learning-based approaches follow to deal with the IR ranking problem. It starts with the training set made of query-document pairs being the input to a computational intelligence or machine learning technique [15]. The ranking model or ranking function is created and then used to rank the search results for the user's queries. The ranking model can also be used in the test phase to measure the predictive performance of the ranking algorithm on the test datasets. Then, the resulting ranking system will produce an ordered list of documents retrieved from the document collection in response to the search. The next section reviews some of the existing LTR approaches in order to set the context for the method proposed in this paper.

Table 1: Learning to Rank (LTR) Query-Document Pairs Representation

Relevance Label	QueryId:id	Feature Vector
1	qid:1	1:0.1 2:0.8 4:0.5 .....N:M
0	qid:1	1:0.9 2:0.6 4:0.2 .....N:M
1	qid:1	1:0.1 2:0.8 4:0.5 .....N:M
1	qid:2	1:0.2 2:0.4 4:0.5 .....N:M
0	qid:2	1:0.3 2:0.7 4:0.3 .....N:M
1	qid:3	1:0.4 2:0.3 4:0.5 .....N:M

3. RELATED WORK

There are three categories of LTR methods [18]: (1) the point-wise method, (2) the pair-wise method and (3) the list-wise method. These categories are based on the loss function or fitness function measurements. The point-wise approach views each single object (query-document pair) as the learning instance. Examples of point-wise approach are Linear Regression (LR) [28], Boosting [8], Gradient Boosted Regression Trees (GBRT or MART) [9, 21] and Random Forests [2]. The pair-wise approach views the pair of objects (two query-document pairs for the same query) as the learning instance. Examples of the pair-wise approach are RankNET (Rank Neural Net) [3], RankBoost and RankSVM (Rank Support Vector Machine) [15]. The list-wise approach takes the entire retrieved list of objects (the list of query-document pairs for each query) as the learning instance. Examples of the list-wise approach are ListNET (Listwise Neural Net) [5], RankGP [17], Coordinate Ascent [20], AdaRank [27] and RankGPES [12]. The proposed ES-Rank method described later in this paper is a list-wise approach because this type has been shown to perform better than point-wise and pair-wise approaches [5].

Due to the need for increasing the performance of LTR approaches, researchers have proposed hybrid techniques by combining methods from the three LTR categories. Sculley [24] proposed an approach (CoRR) combining LR (point-wise approach) with Support Vector Machine (pair-wise approach). This approach is implemented in the Sofia-ml package [24] and it has reasonable computational run-time. However, its performance in terms of NDCG and MAP is limited. In order to achieve better NDCG, Mohan et al. [21] proposed a hybrid machine learning approach for initializing GBRT using RF. However, experiments show that this approach consumes too much computational run-time compared to other approaches from the literature [7, 15]. On the other hand, two proposed hybrid approaches called LambdaRank and LambdaMART combine pair-wise with list-wise approaches [4]. LambdaMART is the boosted tree from

LambdaRank which is based on RankNET. Both LambdaMART and LambdaRank have shown better performance than the method by Mohan et. al. on the Yahoo! Learning to Rank Challenge [6]. Overall, all these approaches have a limitation on the computational run-time or the evaluation accuracy of the predictive results. For example, the computational training time of IGBRT or Coordinate Ascent in Ranklib package among other methods on the MSLR-WEB10K fold is more than 7 hours. The RankGPES method also proposed by Islam [12] combining Genetic Programming with Evolutionary Strategy consumed significant computational time on the LETOR 3 datasets [18]. The consumed training time by the method was between 30 and 40 minutes on the Ohsumed dataset and between 1-2 hours on each .Gov dataset. It is noted that LETOR 3 is smaller than MSLR-WEB10K and other machine learning methods consumed less computational run-time than RankGPES.

4. THE PROPOSED ES-RANK METHOD

Given the limitations in respect of computation time and accuracy of results observed in existing LTR methods as discussed above, the proposal here is to use an Evolutionary Strategy (ES) [10] to tackle the learning to Rank (LTR) problem. The technique developed here is called Evolution Strategy Ranking (ES-Rank). The reason for choosing an ES is their capability for convergence towards a good-quality solution in fast computational run-time. In addition, as mentioned above, list-wise LTR approaches have shown better performance in terms of Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) against pair-wise and point-wise approaches [5].

Algorithm 1 outlines the proposed ES-Rank. This approach is essentially a (1+1)-Evolutionary Strategy that evolves a single vector over a number of generations. The input is the training set of query-document pairs or feature vectors and the output is a linear ranking function. The chromosome *ParentCh* is a vector of M genes, where each gene is a real number representing the importance of the corresponding feature for ranking the document. Steps 1 to 4 initialise the chromosome vector by setting each gene to a value of 0. The Boolean variable *Good*, used to indicate whether repeating the mutation process from the previous generation, is set to FALSE in Step 5. A copy of *ParentCh* is made into *OffspringCh* in step 6. The evolution process for $MaxGenerations$ generations ($MaxGenerations = 1300$ in this paper) starts in Step 7 and ends in Step 24. Steps 8 to 16 show the strategy to control the mutation process by choosing the number of genes to mutate (R), the actual genes to mutate and the mutation step. The mutation step is determined using equation 1 where *Gaussian(0,1):* is a random Gaussian number with 0 mean and 1 standard deviation, and *Cauchy(0,1):* is a cumulative distributed Cauchy random number with value between 0 and 1.

$$Mutated\_Gene\_i = Gene\_i + Gaussian(0,1) * \exp(Cauchy(0,1)) \quad (1)$$

The mutation step defined by equation (1) was chosen based on preliminary experiments in which several ways of combining the Gaussian and Cauchy numbers were tried. The combinations tried involved adding, subtracting and multiplying these numbers. Both random and probabilistic mutation rates were tried in the preliminary experiments. Among the various combinations tried, the one expressed by equation (1) provided the best performance for ES-Rank. A mutation process that is successful (produces a better offspring) in generation $(G-1)$ is replicated in generation G as shown in Step 9. Otherwise the parameters of the mutation process are reset as shown in Steps 11 to 15. Steps 17 to 23 select between the *ParentCh* and the *OffspringCh* according to their fitness function values measured using MAP or NDCG (see subsection 5.2). Finally, ES-Rank returns the ranking function in Step 25, defined by the transpose of the evolved vector of feature weights and the query-document pairs.

Algorithm 1: ES-Rank: (1+1)-Evolutionary Strategy Ranking Approach

Input : A training set $\phi(q,d)$ of query-document pairs of feature vectors.
Output: A linear ranking function $F(q,d)$ that assigns a weight for every query-document pair indicating its relevancy degree.

1 Initialization
2 **for** $(Gen_i \in ParentCh)$ **do**
3 | $Gen_i = 0.0$;
4 **end**
5 Good=FALSE;
6 $OffspringCh = ParentCh$;
7 **for** $G = 1$ *to* $MaxGenerations$ **do**
8 | **if** *(Good==TRUE)* **then**
9 | | Use the same mutation process of generation $(G-1)$ on $OffspringCh$ to mutate $OffspringCh$, that is, mutate the same R genes using the same $MutationStep$;
10 | **else**
11 | | Choose R at random, the number of genes to mutate;
12 | | **for** $j = 1$ *to* R **do**
13 | | | Choose at random, Gen_i in $OffSpringCh$ for mutation;
14 | | | Mutate $Gene_i$ using $MutationStep$ according to equation (1)
15 | | **end**
16 | **end**
17 | **if** *(Fitness(ParentCh,*$\phi(q,d)$*) <Fitness(OffspringCh,*$\phi(q,d)$*))* **then**
18 | | $ParentCh = OffspringCh$;
19 | | Good=TRUE;
20 | **else**
21 | | $OffspringCh = ParentCh$;
22 | | Good=FALSE ;
23 | **end**
24 **end**
25 **return** the linear ranking function $F(q,d) = ParentCh^T \bullet \phi(q,d) = W^T \bullet \phi(q,d)$, that is $ParentCh$ at the end of the G generations contains the evolved vector W of feature weights, T indicates the transpose

Then, in order to apply the proposed LTR approach, the first step is to obtain the datasets which contain the training, validation and test benchmarks. Next, the proposed ES-Rank algorithm is applied to the training set in order to evolve a linear ranking function. Then, the performance of the evolved linear ranking function is assessed using the test set to get the predictive performance of the learning algorithm. The link for ESRank library is: http://www.cs.nott.ac.uk/˜psxoi/ESRank.zip.

5. EXPERIMENTAL RESULTS

This section first describes the three datasets from the LETOR

benchmark collection used to test the proposed ES-Rank. It then describes the metrics MAP and NDCG used for assess the fitness of the evolved vector and also for evaluation in the tests. Finally, the section presents the experimental results in which fourteen other LTR methods from the literature are considered.

5.1 Benchmark Datasets

For the experiments, the three most recent datasets from the LETOR datasets were used: MSLR-WEB10K and LETOR 4 (MQ2007 and MQ2008) [22, 23]. Table 2 summarizes the properties of these datasets. The numbers of features vary in these datasets. In MQ2007 and MQ2008, the number of features is 46, while MSLR-WEB10K contains 136 features. The largest number of queries is in MSLR-WEB10K with 10000 unique queries. On the other hand, MQ2007 and MQ2008 contain 1692 and 784 unique queries respectively.

Table 2: Properties of Datasets Used in Experiments

Dataset	Queries	Query-URL Pairs	Features	Relevance Labels	No. of Folds
MQ2007	1692	69623	46	{0,1,2}	5
MQ2008	784	15211	46	{0,1,2}	5
MSLR-WEB10K	10000	1200192	136	{0,1,2,3,4}	5

5.2 Fitness and Evaluation Metrics

In this study, the *Mean Average Precision (MAP)*, and the *Normalized Discounted Cumulative Gain (NDCG)* were used [1, 14, 3] as two separate fitness functions on the training sets. They also were used as the evaluation metrics for the ranking functions on the test sets.

These metrics are defined as follows. Let $d_1, d_2, ..., d_D$ denote the ranked documents by decreasing order of relevance based on the ranking model, where D is the number of query-document pairs retrieved responding to user's query. The function $r(d_i)$ gives the relevance label of a query-document pair d_i. It returns 1 in binary relevance judgement if d_i is relevant, and 0 otherwise. The precision per query q, denoted $P(q)$, is defined as follows.

$$P(q) \;=\; \frac{1}{D}\;\Sigma_{i=1}^{D}\; r(d_i)\;.\;\Sigma_{j=1}^{D}\;\frac{1}{j} \tag{2}$$

Then, the MAP for a set of queries is the mean of the average precision values over all queries. This is given by the following equation where Q is the number of queries.

$$MAP \;=\; \frac{\Sigma_{q=1}^{Q}\; P(q)}{Q} \tag{3}$$

The NDCG of the top-k documents retrieved (NDCG@k) can be calculated by the following equation where $IDCG@k$ is the ideal (maximum) discounted cumulative gain of the top-k documents retrieved.

$$NDCG@k = \frac{1}{IDCG@k} * \Sigma_{i=1}^{k} \frac{2^{r(d_i)} - 1}{log_2(i+1)} \tag{4}$$

The *Discounted Cumulative Gain of the top-k documents retrieved (DCG@k)* can be calculated by the following equation where $r(d_i)$ returns the relevance label value of the query-document pair in the ranked retrieved list (see table 2 for the relevance label values). In binary relevance judgment labels, $r(d_i)$ returns 1 if the document (d_i) is relevant for the query in the query-document pair, and returns 0 otherwise.

$$DCG@k = \Sigma_{i=1}^{k} \frac{2^{r(d_i)} - 1}{log_2(i+1)} \tag{5}$$

If all top-k documents retrieved are relevant, the $DCG@k$ will be equal to $IDCG@k$.

5.3 Results

The performance of the proposed ES-Rank is compared to that of fourteen machine learning and computational intelligence techniques from the literature. Figures 2 and 3 show the overall results and lists the fourteen methods tested. These are implemented in the packages RankLib [7], rt-Rank [21], Sofia-ml [24], SVMRank [13] and Layered Genetic Programming for Learning to Rank (RankGP) [16]. The parameter values used for the fourteen approaches are the default settings in these packages. Those settings produced the shortest computational run time for each approach. The experimental results presented in Figures 2 and 3 are the average scores of ten runs on 5-folds cross validation. Each dataset fold consists of a training, a validation and a testing data. The number of genes in the chromosome of ES-Rank is equal to the number of features existing in the dataset. For example, the length of the chromosome for the evolved LTR function vector in dataset MSLR-WEB10K is 136 which is the number of features (see table 2).

Table 3: Statistical Summary of the Experimental Results

Algorithms	Mean	SD	SE (Mean)	CV
AdaRank	0.4605	0.0758	0.0138	0.1646
CoRR	0.4318	0.0406	0.0074	0.0941
Coordinate Ascent	0.4776	0.0638	0.0116	0.1330
ES-Rank	0.4779	0.0596	0.0109	0.1247
IGBRT	0.4565	0.0563	0.0145	0.1234
LambdaMART	0.4774	0.0634	0.0116	0.1328
LambdaRank	0.3247	0.0939	0.0171	0.2892
LR	0.4425	0.0527	0.0096	0.1192
ListNET	0.4099	0.1034	0.0189	0.2522
MART	0.4750	0.0624	0.0114	0.1313
RF	0.4771	0.0661	0.0121	0.1384
RankBoost	0.4621	0.0770	0.0141	0.1666
RankGP	0.4197	0.0372	0.0068	0.0887
RankNET	0.4143	0.1060	0.0194	0.2559
RankSVM	0.4043	0.0707	0.0129	0.1748

The results shown in Figures 2 and 3 correspond to the predictive values of the best performance by the tested approaches. For each dataset in the horizontal axis, a different color point represents the performance of one of the

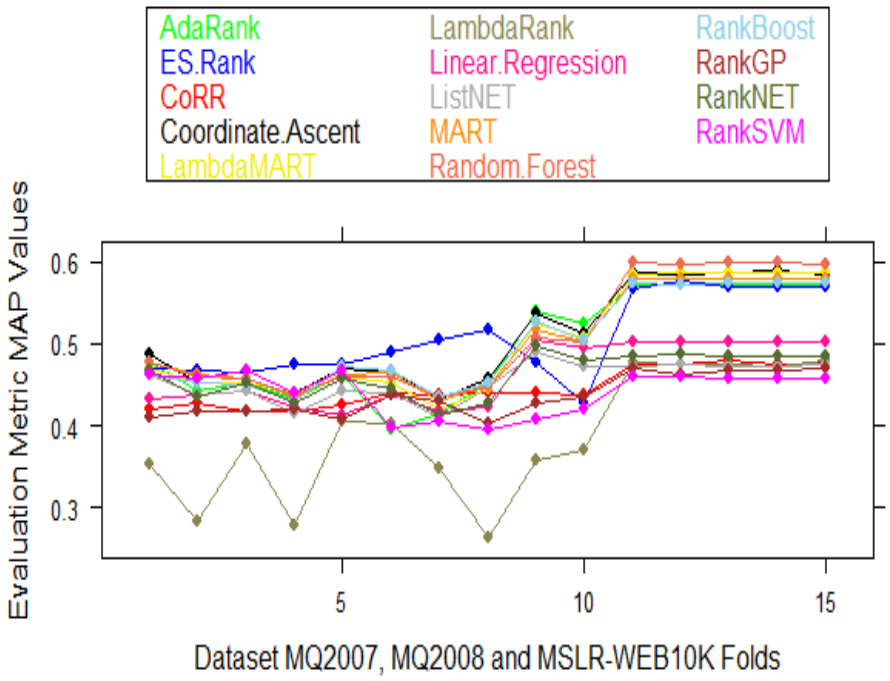

Figure 2: Performance of LTR Algorithms Including the Proposed ES-Rank When Applied to MSLR-WEB10K, MQ2008 and MQ2007 Datasets Using MAP as Fitness and Evaluation Metric.

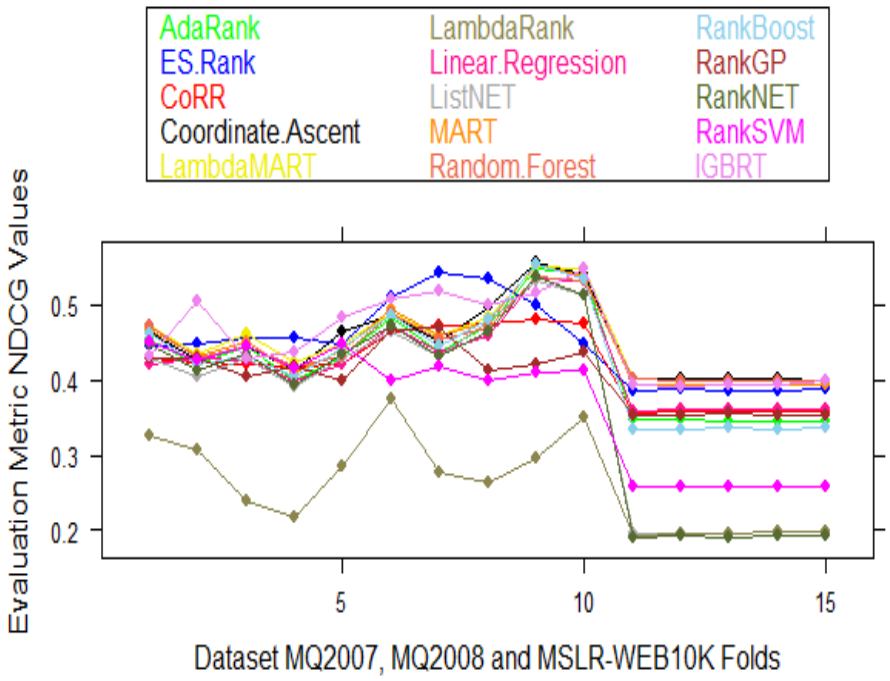

Figure 3: Performance of LTR Algorithms Including the Proposed ES-Rank When Applied to MSLR-WEB10K, MQ2008 and MQ2007 Datasets Using NDCG as Fitness and Evaluation Metric.

Table 4: Average Computational Run-time of the Algorithms in Seconds When Applied to MSLR-WEB10K, MQ2008 and MQ2007 Datasets

Algorithm	MSLR-WEB10K	MQ2008	MQ2007	Average Time
RankBoost	**3720**	**15**	**74**	**1269.667**
RankSVM	**32409**	**19**	**23**	**10817**
ListNET	**18005**	**45**	**95**	**6048.333**
AdaRank	**3600**	**11**	**20**	**1210.333**
MART	**1200**	**8**	**11**	**406.3333**
Coordinate Ascent	**25200**	**37**	**240**	**8492.333**
LambdaMART	**3720**	**9**	**11**	**1246.667**
RankNET	**10800**	**33**	**96**	**3643**
Random Forest	**3660**	**27**	**55**	**1247.333**
Linear Regression	**157**	**2**	**3**	**54**
RankGP	**26020**	**375**	**390**	**8928.333**
CoRR	**10803**	**42**	**51**	**3632**
LambdaRank	**18015**	**46**	**142**	**6067.667**
IGBRT	**36750**	**274**	**253**	**12425.67**
ES-Rank	**1800**	**35**	**51**	**628.6667**

approaches. As mentioned above, the performance is measured with the evaluation metrics MAP and NDCG@10. From these results, it can be seen that ES-Rank is generally the best approach producing the best performance among all methods in 11 out of 30 data fold-metric combinations. From the MAP results in Figure 2, ES-Rank outperforms the other approaches in 7 data folds, while Random Forest outperforms the other approaches in 5 data folds. AdaRank is the third best approach by outperforming the other approaches in 2 data folds, where RankSVM outperforms other approaches in only 1 data folds. From the NDCG@10 results in Figure 3, the best approach is Coordinate Ascent, producing the best performance in 5 out of 15 data folds, while ES-Rank is the second best approach that has the best performance in 4 out of 15 data folds. IGBRT comes in the third position with 3 out of 15, while LambdaMART is the fourth with 2 out of 15. The Random Forest approach is in the fifth position with 1 out of 15 best performance in the data folds. The statistical summary of the results is in table 3. This table shows the mean, standard deviation (SD), standard error mean (SE Mean) and coefficient of variation (CV) of the results on the 30 fold-metric combinations. From this table, we can conclude that the ES-Rank has the highest mean value at 0.4779, while the Coordinate Ascent mean value is 0.4776. Furthermore, the standard error mean, standard deviation and coefficient of variation of ES-Rank are competitive values compared with IGBRT. The values of standard error mean, standard deviation and coefficient of variation of ES-Rank are 0.0109, 0.0596 and 0.1247, while these values in IGBRT are 0.0145, 0.0563 and 0.1234. In terms of computational run-time, ES-Rank used 30 minutes on MSLR-WEB10K fold, 35 seconds on MQ2008 fold and 51 seconds on MQ2007 fold. The detailed computational run-times of the approaches are shown in table 4. These experiments were conducted on a 3.60 GHz Intel (R) core(TM) i7-3820 CPU and the implementation was in Java NetBeans under Windows 7 Enterprise Edition.

6. CONCLUSION AND FUTURE WORK

This paper presented a new LTR approach called ES-Rank which is based on a (1+1) Evolutionary Strategy with a tailored mutation process. The performance of the proposed approach was compared to that of fourteen machine learning and computational intelligence approaches from the literature. The metrics *Mean Average Precision (MAP)* and *Normalized Discounted Cumulative Gain (NDCG)* were used for fitness evaluation within ES-Rank and also for evaluating the performance of the LTR approaches in the comparison. These datasets used here are MSLR-WEB10K (Microsoft ten thousand web queries) dataset, MQ2008 and MQ2007 (TREC Million queries datasets for years 2008 and 2007).

From the experimental results, in general ES-Rank exhibited an overall better performance than the other fourteen methods tested, achieving the best performance in 10 out of 30 data fold-metric. Thus, ES-Rank is a competitive approach to tackle the LTR problem. Given the good results achieved with a (1+1) Evolutionary Strategy in this work. Future research will be focused on improving the approach by investigating its combination with machine learning techniques. Moreover, future work will also consider applying the technique to LETOR 3 datasets (.Gov and Ohsumed datasets) and use three other fitness functions, Reciprocal Rank, Precision and Error Rate.

7. REFERENCES

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern*

Information Retrieval - the concepts and technology behind search. Pearson Education Ltd., Harlow, England, 2nd edition edition, 2011.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] 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*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[4] C. J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010.

[5] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[6] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, pages 1–24, 2011.

[7] V. Dang. RankLib, http://www.cs.umass.edu/~vdang/ranklib.html, 2016.

[8] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, Dec. 2003.

[9] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[10] H. georg Beyer and H. paul Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[11] O. A. S. Ibrahim and D. Landa-Silva. Term frequency with average term occurrences for textual information retrieval. *Soft Computing*, 20(8):3045–3061, 2016.

[12] M. A. Islam. Rankgpes: Learning to rank for information retrieval using a hybrid genetic programming with evolutionary strategies. Master's thesis, Computer Science, University of Windsor, Toronto, Canada, 2013.

[13] T. Joachims. SVM-rank: Support Vector Machine for Ranking, Accessed online (2016).

[14] K. L. Kwok. Comparing representations in Chinese information retrieval. In *SIGIR '97 Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 34–41, New York, NY, USA, 1997. ACM.

[15] H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Morgan & Claypool Publishers, 2014.

[16] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang. Designing a classifier by a layered multi-population genetic programming approach. *Pattern Recognition*, 40(8):2211–2225, 2007.

[17] J. Y. Lin, J. Y. Yeh, and C. C. Liu. Learning to rank for information retrieval using layered multi-population genetic programming. In *Computational Intelligence and Cybernetics (CyberneticsCom), 2012 IEEE International Conference on*, pages 45–49, July 2012.

[18] T.-Y. Liu. Learning to rank for information retrieval. *Foundation Trends of Information Retrieval*, 3(3):225–331, Mar. 2009.

[19] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[20] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[21] A. Mohan, Z. Chen, and K. Weinberger. Web-search ranking with initialized gradient boosted regression trees. In *Journal of Machine Learning Research, Workshop and Conference Proceedings*, volume 14, pages 77–89, 2011.

[22] T. Qin and T. Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013.

[23] T. Qin, T.-Y. Liu, J. Xu, and H. Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

[24] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 979–988, New York, NY, USA, 2010. ACM.

[25] A. Tonon, G. Demartini, and P. Cudre-Mauroux. Pooling-based continuous evaluation of information retrieval systems. *Information Retrieval Journal*, 18(5):445–472, 2015.

[26] J. Urbano. Test collection reliability: a study of bias and robustness to statistical assumptions via stochastic simulation. *Information Retrieval Journal*, 19(3):313–350, 2016.

[27] 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*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.

[28] X. Yan and X. G. Su. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2009.