Feature Selection for Learning-to-Rank using Simulated Annealing

Mustafa Wasif Allvi[1], Mahamudul Hasan[2], Lazim Rayan[3], Mohammad Shahabuddin[4],
Md. Mosaddek Khan[5], Muhammad Ibrahim[6]
Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh[1,2,3,4]
Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh[5,6]

***Abstract*—Machine learning is being applied to almost all corners of our society today. The inherent power of large amount of empirical data coupled with smart statistical techniques makes it a perfect choice for almost all prediction tasks of human life. Information retrieval is a discipline that deals with fetching useful information from a large number of documents. Given that today millions, even billions, of digital documents are available, it is no surprise that machine learning can be tailored to this task. The task of learning-to-rank has thus emerged as a well-studied domain where the system retrieves the relevant documents from a document corpus with respect to a given query. To be successful in this retrieving task, machine learning models need a highly useful set of features. To this end, meta-heuristic optimization algorithms may be utilized. The aim of this work is to investigate the applicability of a notable meta-heuristic algorithm called simulated annealing to select an effective subset of features from the feature pool. To be precise, we apply simulated annealing algorithm on the well-known learning-to-rank datasets to methodically select the best subset of features. Our empirical results show that the proposed framework achieve gain in accuracy while using a smaller subset of features, thereby reducing training time and increasing effectiveness of learning-to-rank algorithms.**

***Keywords*—Information retrieval; learning-to-rank; feature selection; meta-heuristic optimization algorithm; simulated annealing**

I. Introduction

Information retrieval (IR) is a process of retrieving the relevant information from a huge collection of data. Given the sheer amount of digital documents available today, this task is inherently quite difficult. An IR system works as follows. A query is submitted by the user of the system, and the task of the system is to return a ranked list of documents to the user based on the query. The user expects that highly relevant documents are in the top portion of the ranked list. Hence, the job of the system is to decide which document is relevant to the query and to what degree. To accomplish this task, researchers have been using heuristic scoring functions [1].

Machine learning can be thought of a discipline of applied statistics [2]. Given sufficient amount of empirical or historical data, these techniques are able to predict the outcome of unseen events. Various paradigms of machine learning are practised. Amongst these, supervised machine learning is mostly used by common people. In this setting, the training data consists of various information about different events along with the known labels. The job of the training module is to learn the pattern (in the form of a function) of the data that decides the labels. This function or model is then used to predict the labels of the unseen data, which is called the testing or evaluation module.

Learning-to-rank (LtR) is a relatively new area emerged in early 2000 as a successful marriage between information retrieval and machine learning [3]. In this framework, the training examples are query-document pairs, the features are the output scores of various scoring functions (such as tf-idf, bm25 score, etc.), and the labels are relevance scores assigned usually by humans. A model learnt from these data can then be used to generate relevance scores for documents with respect to a user's query. Fig. 1 depicts the scenario.

Today not only are data sets getting bigger and bigger, but also new data types have also been keeping to emerge, such as web-based data streams, genomics and proteomics micro arrays, and social media and system biology networks [4]. Therefore, the choice of features in a supervised machine learning setting is of utmost importance [5], [6]. One the one hand, we want to incorporate as much information as possible in our training set so that the learning algorithm can easily decide which aspects of the training data plays role in producing the labels. On the other hand, if irrelevant and misleading information is as features, the learning module will find it difficult to extract the pattern of the data, thereby reducing predictive accuracy. Moreover, if we can reduce the number of features, the training time in the learning phase will be minimized. For these reasons, a lot of research in supervised machine learning has been devoted to feature selection process [7]. The goals of feature selection include: creating easier and more comprehensible models, and enhancing data mining efficiency and helping to clean and understand data better [8]. It should be noted here that the feature selection is an NP-hard problem [9]. More details about these works will be elaborated in the next section.

The rest of the paper is organized as follows. In Section II, we briefly discuss existing works related to our area. In Section III, we discuss our framework in detail. Section IV presents the experimental settings and discusses the findings. Finally, Section V concludes the paper.

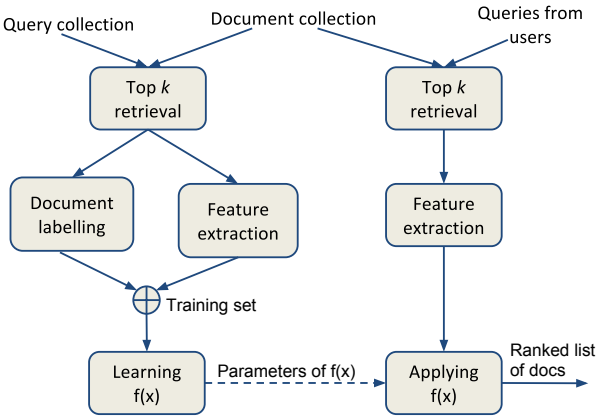

Fig. 1. An LtR-based IR system [10].

II. Background and Literature Review

In this section, we discuss the existing works related to our field which is feature selection in machine learning and learning-to-rank using traditional and meta-heuristic methods. We thus identify the gap in the existing literature.

A. Feature Selection in Supervised Machine Learning

We discuss the relevant papers of this subsection in two categories: (1) using traditional methods, and (2) using meta-heuristic algorithms.

1) Using Traditional Methods: By traditional methods of feature selection we mean filter, wrapper, embedded, forward/backward elimination, etc. methods [11].

Karegowda et al. [12] propose a supervised feature selection approach called wrapper approach. Wrappers take a subset of the function set, evaluates the output of the classifier on this subset, and then evaluates another subset on the classifier. Four different classifiers, namely Decision tree C4.5, Naïve Bayes, Bayes Network and Radial Basis are used. Eleven attributes identified by different wrappers were compared using different classifiers in the validation step. Their experiment discovers that no single standard wrapper approach is the best for different data sets.

Liu et al. [11] explain the importance of feature selection in data mining and briefly describe the methods of feature selection which is filter, wrapper and embedded model. For dimension reduction in data mining, the impact of feature selection is explained. There are brief explanation on feature weighting algorithms or subset selection, single data source algorithms, multi-source feature selection, detecting feature dependency, among other topics. Two research issues with selection features are explored.

Fan et al. [13] discusses the process of selecting features for data sets with millions of features.

2) Using Meta-Heuristic Algorithms: Heuristic optimization algorithms have been designed to solve large-scale optimization problems [14]. Most of these algorithms are nature-inspired. These methods differ from heuristic algorithms in the sense that the heuristic algorithms, working in a purely greedy manner, oftentimes stuck in local optima of the search space whereas the meta-heuristic algorithms use various types of randomization, even sometimes at the cost of apparently bad move, to get out of local optima. The focus of the meta-heuristic algorithms is to find an optimal solution for given problem by exploring maximum number of useful positions of the search space landscape in a given time frame.

Many researchers have investigated meta-heuristic algorithms for feature selection in machine learning. Below we describe some of them.

Emary et al. [15] propose a grey-wolf optimization (GWO) – a meta-heuristic algorithm inspired by natural instinct of wolves – based feature subset selection approach. Three parameters of the algorithm are used to decide the fitness of a candidate feature subset. The GWO algorithm iterates by exploring new regions within the function space and leverages solutions before near-optimal solution is reached. The optimization approach is based on k-nearest neighbor.

Sayed et al. [16] propose an extension of crow search algorithm. Features are selected based on the chaotic crow search algorithm (CCSA). CCSA is an upgraded version of crow search algorithm which is a nature based evolutionary algorithm. The paper use ten different chaotic maps for optimization. 20 data sets with different features and parameters are examined.

Aljarah et al. [17] explores the grasshopper optimization algorithm. A bio-inspired optimization technique is introduced to optimize the performance of Support Vector Machine (SVM) classifier, a powerful supervised machine learning technique. The model's main objective is to maximize SVM's classification accuracy with the minimum number of features. The suggested solution is tested on 18 public datasets and the result is satisfactory.

One of the most effective meta-heuristic algorithms to date is simulated annealing (SA) [18]. Being popular for its capability to find good quality solutions, SA is used by many researchers in multifarious machine learning domains. Gheyas and Smith [19] present a combination of two algorithms. The capability of better exploration in the search space of simulated annealing and the rapid convergence behavior of genetic algorithm are combined to find the feature subset more quickly and precisely. 11 synthetic and 19 real-world high-dimensional data sets to conduct the experiments.

Mafarja and Mirjalili [20] design a hybridization of whale optimization algorithm (WOA) and simulated annealing for feature selection. Whale optimization algorithm has some unique properties such as fewer parameters to control (since it requires only two key internal parameters to be modified), simple implementation and high versatility. The SA algorithm is wrapped with the WOA algorithm in an attempt to find the best solution throughout the neighborhood solutions. 18 data sets are examined for performance evaluation.

Barbu et al. [21] investigates feature selection methods using medical image data set where there is a massive number of features. The authors propose an algorithm that is suitable for big data computing due to its simplicity and ability to reduce the problem size throughout the iterations. The authors show that unlike its competitors such as boosting, the amount

of data which the algorithm requires to use for training is much smaller, making it suitable for large-scale problems.

B. Feature Selection in Learning-to-Rank

Learning-to-rank has a wide area of application. Feature selection plays a vital role in building up a learning-to-rank model. Besides searching, image processing, big data learning and in classification of tweets are also included in the area of learning-to-rank implementation.

Novakovic et al. [22] combine ranking methods and classification algorithms to find the optimal feature subset. Four supervised algorithms, namely IB1, Naïve Bayes, C4.5 Decision Tree and Radia basis and statistical and entropy-based ranking methods are used. Filter-based methods are used for evaluating each subset of features, and irrelevant features are discarded. Duan et al. [23] use advanced greedy feature selection algorithm while exploring the earning-to-rank on tweets. Lai et al. [24] transform the feature selection problem into a joint convex optimization formulation which minimizes ranking errors as well as simultaneously conducting feature selection. Their framework can incorporate various feature similarity and imporance measures. Geng et al. [25] also pose the feature selection problem as an optimization problem by defining a loss function involving feature importance, and then solves it efficiently.

From the above review of existing works related the theme of this paper which is feature selection methods for learning-to-rank, we see that although simulated annealing have been studied to some extent for feature selection in supervised machine learning framework, to the best of our knowledge it has not been investigated in a learning-to-rank paradigm. Our work attempts to fill this gap in the literature.

III. Methodology

In this paper, we focus on finding the optimal subset of features of training data of LtR problem that is likely to yield a higher ranking accuracy during evaluation. For this optimization purpose, we utilize simulated annealing algorithm.

A. Simulated Annealing

Simulated annealing is usually used to solve NP-complete problems such as our feature subset selection problem, The algorithm basically combines two methods, namely hill climbing and random walk [14], [26].

Hill climbing is a greedy algorithm that only search for the local best solution. Hill climbing reaches a solution by recursively choosing the best neighbor based on an evaluation function, until there is no immediate better neighbor than the current one. When there is more than one best successor, a random selection is made from the set of best successors. For this nature of hill climbing algorithm, it often gets stuck in a local optimal point.

To overcome the problem of local optima of hill climbing algorithm, the idea of random walk is introduced [27]. The walk starts at a certain fixed node and moves randomly to a neighbor of the current node at each step. This method, however, has its own limitation because it may arbitrarily jump from one point to another in the search space.

Simulated annealing algorithm combines the merits of both hill climbing and random walk. It applies randomization in a way that allows occasional "bad" movements in an attempt to reduce the likelihood of getting stuck in a mediocre yet locally optimal solution. Specifically, the working procedure of simulated annealing is as follows. A random state is selected first. It then randomly selects a neighbor state (depending on the specific problem at hand, the definition of neighborhood is devised beforehand). If the selected neighbor state is better than the current state in terms of a utility function (again, decided beforehand), then the neighbor becomes the current state and the algorithm iterates over. But if the selected neighbor is worse than the current state, then the algorithm still gives it a chance to be selected as the (next) current state by a probability; the probability depends on the difference between the two states (current and neighbor) and the time during which the algorithm has been in its operation. More specifically, the higher the difference, the less the probability of choosing the (bad) neighbor, and the earlier stage the algorithm is in, the higher the said probability. Mathematically this probability is,

$$probability = e^{\frac{\Delta E}{T}},$$

where,

$$\Delta E = neighbor\ state\ quality - current\ state\ quality,$$

and, T = temperature, which is reduced in every iteration from a very high value to zero. In essence, if the quality of neighbor state is worse than the current state, then the neighbor is selected (or not) with the probability $e^{\frac{\Delta E}{T}}$.

Although simulated annealing offers a way to overcome the striking bottleneck of hill climbing algorithm, but a large amount of time is oftentimes the price to be paid [28]. The idea of considering less value node is that, by doing it, the algorithm gets to explore more area in solution space by not getting stuck in a local optima.

Simulated annealing is known to work better than the bare local search algorithm most of the time. The procedure of simulated annealing is depicted in Fig. 2. To know more details about this exciting algorithm, the interested reader is requested to go though [29].

```
1 function SIMULATED-ANNEALING(problem, schedule) returns a solution state
2   inputs: problem, a problem
            schedule, a mapping from time to "temperature"
3   current ← MAKE-NODE(problem.INITIAL-STATE)
4   for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
5     ΔE ← next.VALUE – current.VALUE
6     if ΔE > 0 then current ← next
      else current ← next only with probability e^(ΔE/T)
```

Fig. 2. Simulated annealing algorithm [30]. "schedule" in line 3 is a monotonically decreasing function of T. "successor" in line 5, in our context, means neighbor. "Value" in line 6 implies quality of a state.

B. Proposed Framework

In this subsection we detail our algorithm that we use for selecting the best subset of features using simulated annealing technique. The procedure consists of broadly three constructs which are described below.

- *Notion of a state.* Here a state in the search space means a subset of features. Ultimately we search for the best subset of features that, when learnt using these features, yields the best predictive accuracy.
- *Definition of neighborhood.* A state's neighbor is defined as altering some features indexes of the current state randomly.[1]
- *Quality of a state expressed as a function.* Here we employ a heavily used IR evaluation metric called Normalized Discounted Cumulative Gain (NDCG) (will be elaborated in Section IV) as the quality of a state. The higher the NDCG of a model learnt from a training data (consisting of only the k features in question), the better.

Armed with these constructs, we are now in a position to detail our framework. Here is how it works. The procedure takes the number of features to select, k, as parameter, from the available pool of features. It then initially chooses random k features which is considered as the initial state in the search space. It then builds the training set using only these selected features, and trains an LtR algorithm on these data. The learnt model is then evaluated on test data (obviously using only the reduced subset of features in question) and stores the NDCG value as the quality of the current state. After that the neighbor state is chosen as per the rule aforementioned rule. The NDCG value in the evaluation stage is computed using the training data triggered by the neighbor state. The difference in these two NDCG values are then used to decide whether to make the neighbor state current state. The procedure stops when, for a particular k value, a predefined maximum number of iterations is reached. This entire procedure is repeated for various k values. While we retain and compare NDCG performance of various k values against corresponding random feature subset selection, ultimately the corresponding feature subset of highest performing k value that gives the best NDCG value across all the iterations is suggested to use instead of all available features. In our implementation we start with $k = 1$ and increase it one by one until we reach the number of available features.

IV. Empirical Results

For the experiments, we use six popular data sets, namely MQ2007, MQ2008, TD2004, HP2004, NP2004 and Ohsumed. These data sets have been made publicly available by Microsoft[2]. The data sets contain a varying number of features which are as follows: MQ2007 and MQ2008: 46, TD2004, HP2004, and NP2004: 64 and Ohsumed: 45. The data sets come with predefined chunks for training, validation and test sets. We maintain these chunks for the sake of better compatibility with the existing research works. To know more details about these datasets, please see [31], [32].

The RankLib LtR implementation[3] is used for evaluating our proposed framework. As for LtR algorithm, we choose LambdaMart because numerous research such as [33], [34] show that tree-ensemble methods in general, and specifically LambdaMart, perform oftentimes better than other LtR algorithms.

As mentioned earlier, as evaluation metric we use NDCG. NDCG is a ranking performance evaluation metric that gives a gradually higher score (out of 1) to a list having the highly relevant documents in the top portion of it. To know more about NDCG and other IR evaluation metric, please see [3].

For each of the six data sets, we generate a plot as follows. For a particular k value starting from 1 and ending in the number of available features, we generate two new training sets – one with the features suggested by the random selection process (i.e., by choosing random k feature indexes) and the other with the features prescribed by simulated annealing algorithm (i.e., by choosing k features of the solution state returned by the SA algorithm after 100 iterations). The LambdaMart LtR algorithm is then learnt for each of the generated training sets, and then the two learnt models are evaluated on the test set (obviously comprising with the same features of each case). Thus we get two NDCG values for a particular k value: one for the random selection process and the other for the SA algorithm. This way all possible k values are examined, and finally the graph is generated (for this data set) by plotting these two curves. Fig. 3, 4, 5, 6, 7, and 8 show these plots for the six data sets. Now, we analyze each of the six plots in details.

1) MQ2007 Data Set: Fig. 3 shows that the curve fluctuate initially at a higher degree which is gradually quelled. This is because when the number of features is small, the benefit of SA algorithm is not evident. Then as the number of features increases the curve get relatively flatter. From the graph we can say that instead of taking all the 46 features, we can get the best accuracy with only 26 features as prescribed by the SA algorithm

2) MQ2008 Data Set: Fig. 4 shows that for MQ2008 data set from the beginning the efficacy of SA approach is evident. With almost only 20 features suggested by the SA algorithm we can reach the accuracy of the training set having all 46 features.

3) Ohsumed Data Set: Fig. 5 shows that the curve of Ohsumed data set tends to vary almost from beginning to end for all the k values. It can be concluded that initially there is a significant difference between the random selection score and SA score. But after coming to about 18 features, the gap diminishes onward. Therefore, from our experiment we can say that using a subset of 18 features instead of all 45 features may be time-efficient.

4) NP2004 Data Set: Fig. 6 demonstrates that from around $k = 8$, i.e., using only 8 or more features with simulated annealing is almost invariably better than the random selection

[1]We note here that there could be other definitions which we intend to investigate as future work.

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

[3]https://sourceforge.net/p/lemur/wiki/RankLib/

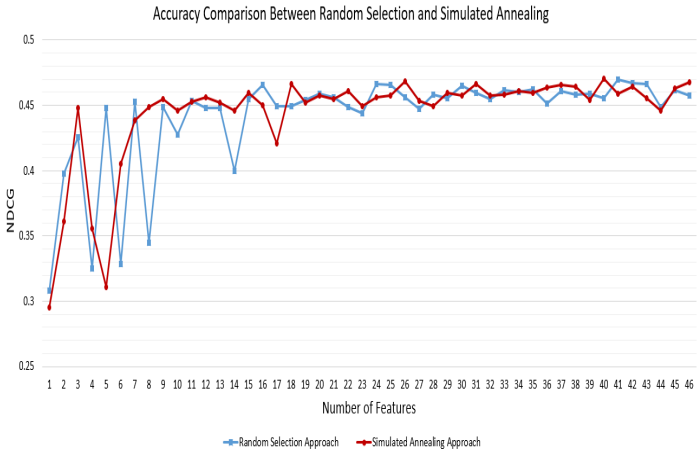

Fig. 3. MQ2007 data: comparison between random selection and simulated annealing in terms of NDCG.

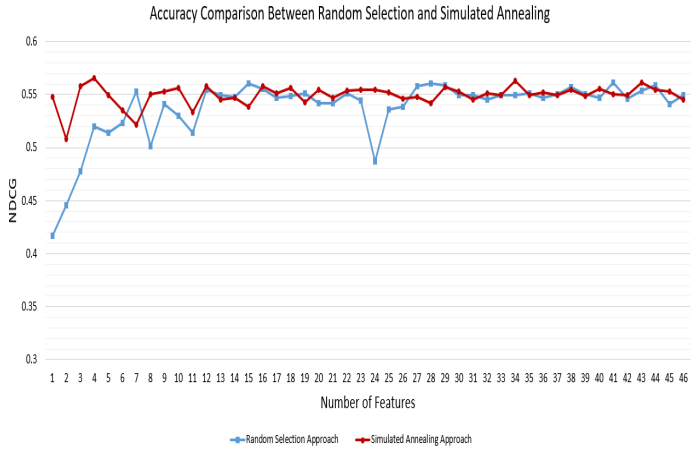

Fig. 4. MQ2008 data: comparison between random selection and simulated annealing in terms of NDCG.

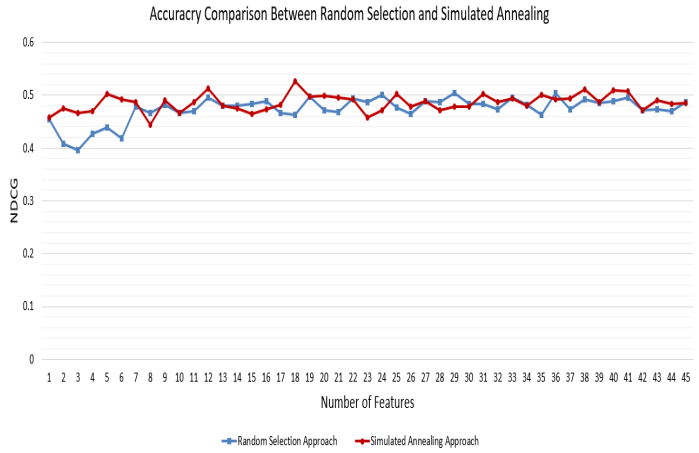

Fig. 5. Ohsumed data: Comparison between random selection and simulated annealing in terms of NDCG.

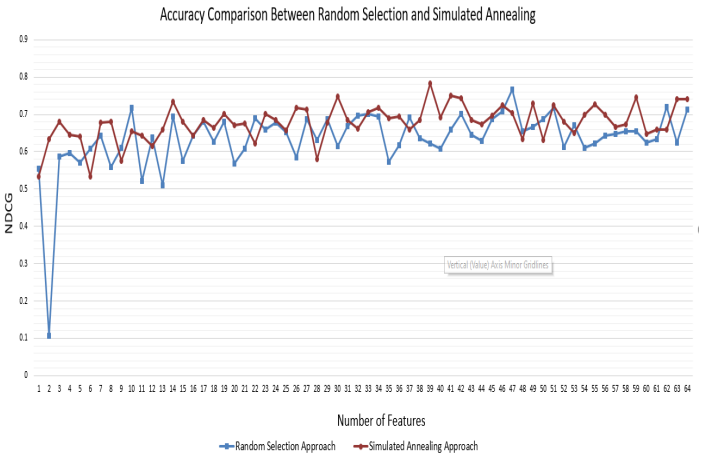

Fig. 6. NP2004 data: comparison between random selection and simulated annealing in terms of NDCG.

approach. Moreover, using 39 features outperforms the setting of using all 64 features.

5) HP2004 Data Set: Fig. 7 demonstrates similar trend that of NP2004 in terms of performance comparison between random selection and simulated annealing approaches. In particular, after $k = 8$ it appears that SA approach almost always outperforms the random selection approach. Taking a subset of only around 22 features is likely to beat the performance of 53 features.

6) TD2004 Data Set: Fig. 8 shows largely similar trend to that of HP2004 and NP2004. From around 8 features, the SA approach seems to outperform the random selection approach. Moreover, it appears that using only around 23 features yield equivalent accuracy to that of using all 64 features.

A. Discussion

The following points can be drawn from the analysis of experimental results.

- For all six data sets, a smaller subset of features work quite well as compared to the setting of using all available features. This indicates that blindly incorporating as many features as possible may not improve the accuracy of LtR systems, rather a careful selection of features is needed.
- Broadly, all six data sets appear to reap benefit of the simulated annealing feature selection method over the random selection method. It should be noted that we have used a basic SA algorithm. Recent variations of SA algorithm and other meta-heuristic algorithm may yield further improvement.
- Fluctuation is present in all the plots which is natural given the simulated annealing is a randomized algorithm. If we average the results of several runs, the fluctuation will be minimized.
- The nature of the features selected by the random selection and simulated annealing based selection has not been investigated.

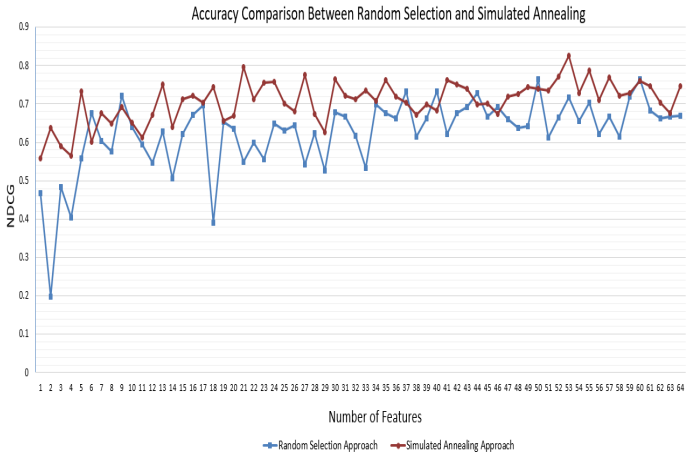

Fig. 7. HP2004 data: Comparison between random selection and simulated annealing in terms of NDCG.

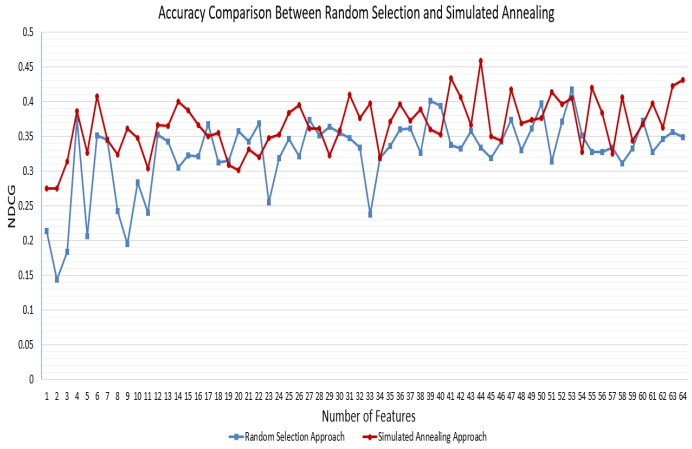

Fig. 8. TD2004 data: Comparison between random selection and simulated annealing in terms of NDCG.

- More iterations in simulated annealing may uncover further insights into our findings.

In essence, we can say that if our proposed framework is deployed, we are able to not only discover better feature subset to learn an LtR model but also to reduce the training time. This investigation thus suggests that commercial IR systems such as search engines that deploy LtR system may apply feature selection methods more seriously and wisely using sophisticated techniques like simulated annealing.

V. Conclusions and Future Work

Recently the area of ranking in information retrieval has earned a lot of attention in the field of machine learning. Learning-to-rank paradigm that is a blend between information retrieval and supervised machine learning has gained much momentum in the research community due to the success in satisfying users of information retrieval systems such as search engines. Performance of these algorithms heavily depend on the features or attributes used in the learning module. Hence a careful selection of features is needed. In this work we have deployed a effective and efficient meta-heuristic algorithm called simulated annealing to select a better subset of features from the available ones. Our experiments on benchmark data sets reveal that using simulated annealing we can extract an effective yet smaller subset of features that performs quite well as compared to the baseline (i.e., the setting where all features are used). This investigation suggests that the features should be chosen carefully so as to improve the predictive accuracy of the LtR algorithms as well as to reduce training time.

This work generated at least three-pronged future research avenues. Firstly, in this work we have examined only one, albeit highly effective and hence popular, LtR algorithm. It is natural to be curious about performance of other LtR algorithms when plugged in into our proposed framework. Secondly, larger LtR data sets need to be investigated. Thirdly, while we have explored the classical simulated annealing algorithm, other contemporary meta-heuristic algorithms may deserve such investigation.

References

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[3] M. Ibrahim and M. Murshed, "From tf-idf to learning-to-rank: An overview," in *Handbook of Research on Innovations in Information Retrieval, Analysis, and Management*. IGI Global, 2016, pp. 62–109.

[4] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, "Advancing feature selection research," *ASU feature selection repository*, pp. 1–28, 2010.

[5] J. Brownlee, "An introduction to feature selection," *Machine Learning Process*, vol. 6, 2014.

[6] V. Kumar and S. Minz, "Feature selection: a literature review," *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014.

[7] T. R. Brick, R. E. Koffer, D. Gerstorf, and N. Ram, "Feature selection methods for optimal design of studies for developmental inquiry," *The Journals of Gerontology: Series B*, vol. 73, no. 1, pp. 113–123, 2018.

[8] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, "Advancing feature selection research," *ASU feature selection repository*, pp. 1–28, 2010.

[9] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in bioinformatics*, vol. 2015, 2015.

[10] M. Ibrahim and M. Carman, "Comparing pointwise and listwise objective functions for random-forest-based learning-to-rank," *ACM Transactions on Information Systems (TOIS)*, vol. 34, no. 4, p. 20, 2016.

[11] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, "Feature selection: An ever evolving frontier in data mining," in *Feature selection in data mining*, 2010, pp. 4–13.

[12] A. G. Karegowda, M. Jayaram, and A. Manjunath, "Feature subset selection problem using wrapper approach in supervised learning," *International journal of Computer applications*, vol. 1, no. 7, pp. 13–17, 2010.

[13] J. Fan, R. Samworth, and Y. Wu, "Ultrahigh dimensional feature selection: beyond the linear model," *Journal of machine learning research*, vol. 10, no. Sep, pp. 2013–2038, 2009.

[14] J. Rajpurohit, T. K. Sharma, A. Abraham, and A. Vaishali, "Glossary of metaheuristic algorithms," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 9, pp. 181–205, 2017.

[15] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, 2016.

[16] G. I. Sayed, A. E. Hassanien, and A. T. Azar, "Feature selection via a novel chaotic crow search algorithm," *Neural computing and applications*, vol. 31, no. 1, pp. 171–188, 2019.

[17] I. Aljarah, A.-Z. Ala'M, H. Faris, M. A. Hassonah, S. Mirjalili, and H. Saadeh, "Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm," *Cognitive Computation*, vol. 10, no. 3, pp. 478–495, 2018.

[18] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[19] I. A. Gheyas and L. S. Smith, "Feature subset selection in large dimensionality domains," *Pattern recognition*, vol. 43, no. 1, pp. 5–13, 2010.

[20] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, 2017.

[21] A. Barbu, Y. She, L. Ding, and G. Gramajo, "Feature selection with annealing for computer vision and big data learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 2, pp. 272–286, 2016.

[22] J. Novaković, "Toward optimal feature selection using ranking methods and classification algorithms," *Yugoslav Journal of Operations Research*, vol. 21, no. 1, 2016.

[23] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum, "An empirical study on learning to rank of tweets," in *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 295–303.

[24] 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.

[25] X. Geng, T.-Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 407–414.

[26] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning," *Operations research*, vol. 37, no. 6, pp. 865–892, 1989.

[27] C. Avin and B. Krishnamachari, "The power of choice in random walks: An empirical study," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, 2006, pp. 219–228.

[28] E.-G. Talbi and T. Muntean, "Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to the mapping problem," in *[1993] Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences*, vol. 2. IEEE, 1993, pp. 565–573.

[29] D. Henderson, S. H. Jacobson, and A. W. Johnson, "The theory and practice of simulated annealing," in *Handbook of metaheuristics*. Springer, 2003, pp. 287–319.

[30] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[31] M. Ibrahim, "Reducing correlation of random forest–based learning-to-rank algorithms using subsample size," *Computational Intelligence*, vol. 35, no. 4, pp. 774–798, 2019.

[32] ——, "Sampling non-relevant documents of training sets for learning-to-rank algorithms," *International Journal of Machine Learning and Computing*, vol. 10, no. 3, pp. 1–10, 2020 (In Press).

[33] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 23-581, p. 81, 2010.

[34] M. Ibrahim, "An empirical comparison of random forest-based and other learning-to-rank algorithms," *Pattern Analysis and Applications*, pp. 1–23, 2019.