Sampling Non-relevant Documents of Training Sets for Learning-to-Rank Algorithms

Muhammad Ibrahim

***Abstract*—Learning-to-rank (LtR) is a framework that spans from training set generation from unlabelled data to learning a model to runtime efficiency. While a good number of learning-to-rank algorithms have been developed over the past decade, "out-of-the-box" topics such as investigation into the properties of training sets have not been given much attention to date. In this article we investigate the imbalanced nature of LtR training sets, which generally contain very few relevant documents as compared to the non-relevant ones. Since the relevant documents are rare in a document collection, the need to include in the training set as many relevant documents as possible is well-understood. However, the lower bound of the number of non-relevant documents needed to learn a good ranking function is still largely uninvestigated. This article aims at addressing this question with a special focus on random forest based LtR algorithms. We employ both random and deterministic undersampling techniques to reduce the number of non-relevant documents. Minimization of training set size reduces the computational complexity (i.e., learning time) which is an important factor, especially for large scale LtR. Extensive experiments on Letor benchmark datasets reveal that in many cases the performance of a LtR algorithm trained on a much smaller training set (in terms of presence of non-relevant documents) remains largely similar to that of a model learnt from the original training set. This investigation thus suggests that for large scale LtR tasks, we can leverage undersampling techniques to reduce training time with oftentimes negligible effect on predictive accuracy. We further examine the potential benefit of random forest based LtR algorithms in the context of undersampling which demonstrates that the inherent structure of a random forest is conducive to using undersampling, and thereby allows for even more scalability.**

***Index Terms*—Learning-to-rank, undersampling, random forests, scalability, imbalanced data.**

I. Introduction

The task of an information retrieval (IR) system is to take a query from the user and eventually to return a list of documents ordered by their predicted relevance to that query. To accomplish this task, traditionally various heuristic (eg. tf-idf score) and probabilistic models (eg. BM25, Language Models), are used[1]. Such a model is essentially a scoring function that takes a document and a query representation as input and produces a relevance score for that document with respect to that query. The documents are then sorted in descending order of these scores before presenting to the user.

In the past decade, due to their effectiveness, the use of machine learning techniques for computing the abovementioned relevance scores of documents has received much attention both in academia and industry [2]-[4] which is called the learning-to-rank (LtR) problem [5]. This framework adopts a statistical learning approach which uses labeled data (i.e., relevance predictions given by a set of simple ranking functions along with the true relevance labels of the documents) to predict the degree of relevance of unseen documents with respect to a query. Today most of the commercial search engines are known to be using LtR methods. In academia, the use of LtR algorithms in a variety of tasks is ever-increasing [2]. Learning-to-rank algorithms are thus at the core of a modern IR system.

The rest of the article is organized as follows. Below we explain as to why we need to investigate the skewed distribution of relevance labels in an LtR training set, which is followed by a list of major contributions of this investigation. Section II illustrates the methodology of this research. Section III discusses the existing works. Section IV explains the learning model. Section V describes the datasets and analyses the experimental results. Section VI shows the efficacy of a random forest framework in yielding further scalability. Section VII summarizes the findings of this investigation.

A. Motivation

The task of developing an LtR-based IR system can be viewed as a two stage process [6], [7]. The first stage involves the following steps:

1. *Top-k retrieval*. An initial retrieval approach (involving one or more base rankers such as BM25 score) is used to retrieve top *k* documents for each query from the entire collection of (unlabelled) documents.

2. *Human labeling and feature extraction*. Relevance judgments for the retrieved *k* documents are collected usually from human judges [8], [9]. Also, features (i.e., relevance scores predicted by various base rankers) are extracted for each of these query-document pairs. The features are then normalized on a per query basis. These features along with the relevance labels constitute a training data set.

The following step is then performed in the second stage:

3. *Learning*. An LtR algorithm is employed to learn a ranking function $f(x)$ from the training set.

Once the system has been trained, the following steps are performed during real time evaluation:

1. *Top-k retrieval*. For a query submitted by the user, the

Manuscript received on April 26, 2019; revised on February 7, 2020.

Muhammad Ibrahim is with the Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh (e-mail: ibrahim@cse.du.ac.bd).

[1] Manning *et al*. [1] nicely explain these models in details.

same top-k retrieval approach (i.e., step 1 of the learning phase) is applied.

2. *Feature extraction*. Features are extracted for the retrieved documents using the same base rankers mentioned in Step 2 of the first stage.

3. *Application of the learnt model*. A relevance score for each document is generated using the learned model. The documents are then ranked using these scores and returned to the user. Fig. 1 depicts the complete scenario.

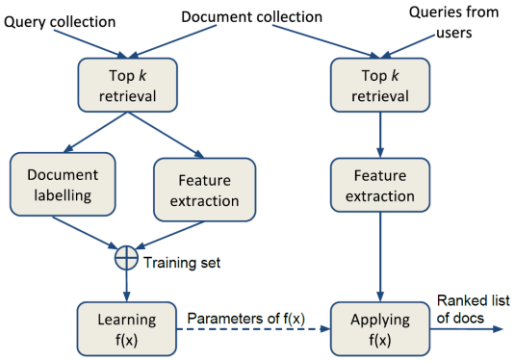

Fig. 1. An LtR-based IR system.

Most of the existing works on LtR is focused on developing better algorithms for learning a ranking function given a training set, whereas relatively little research has been devoted to the out-of-the-box engineering topics such as how to improve the quality of the training data, or how to tackle the scalability issue for large scale LtR etc. Some studies have been conducted on the first stage (i.e., how to improve the initial retrieval approach) of LtR which will be discussed in the Section III. We are, however, interested in an aspect that lies between the first and second stages. After applying the initial approach to retrieve the top *k* documents from a document collection, the resultant training set usually contains very few relevant documents associated with a query as compared to the non-relevant ones. To demonstrate this characteristic, in Fig. 2 we show the relevance label distribution of the prominent LtR datasets (details of these datasets will be discussed in Section V.A). We see that for all eight datasets the discrepancy in label distribution is stark (with varying degree).[2] Now the question becomes, why might imbalanced training data be a problem for LtR? The answer is, commercial IR systems have billions of documents in their collections. As such, a representative training set (found after applying the initial top-*k* retrieval) is typically very large. It is, therefore, computationally challenging for a learning algorithm to learn a ranking function from such a large training set.[3] Hence keeping the training set size smaller is lucrative from the perspective of scalability. On the other hand, the larger the training set, the more information is given to a learning algorithm, and hence sufficient amount of training data are needed to learn a good ranking function. Since the number of relevant documents is limited (while the non-relevant documents for a query is unlimited), all the relevant documents of a training set are useful. Thus the following research question can be raised: are all of the non-relevant documents found from the top-*k* retrieval (which are currently included in the LtR training sets) necessary to learn a good ranking function? That is to say, could a smaller subset of non-relevant documents be sufficient for learning a ranking function capable of well-distinguishing between relevant and non-relevant documents? If we discover that we can use such a smaller subset without compromising the accuracy of the learning algorithm significantly, the time complexity for LtR algorithms (which is an important issue for large scale LtR [2], [3], [11]) would be significantly reduced.

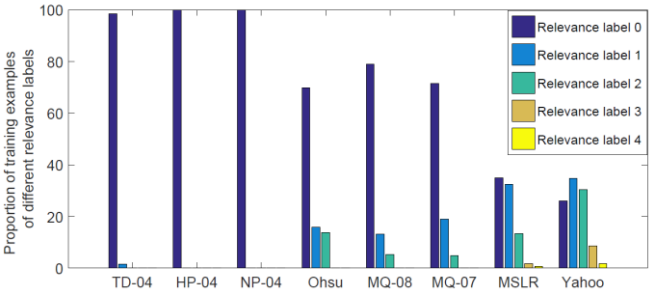

Fig. 2. Relevance label distribution of different datasets. Dataset names are given in the x-axis.

B. Contributions

The following contributions are made in this article:

We employ both random and deterministic undersampling techniques to examine if a ranking function can successfully distinguish between the relevant and non-relevant documents with a smaller subset of non-relevant documents in the training set. Experimental results suggest that the random undersampling of non-relevant documents on a per-query basis works quite well. In many cases, much smaller training data (even as little as 19-32% of the training data) can be used without significant degradation of accuracy.

While our investigation suggests that all LtR algorithms are likely to reap benefit from the use of undersampling techniques, the inherent structure of a random forest offers a more effective way to achieve this as compared to other LtR algorithms. By properly utilizing the tree-forest structure of an RF-based LtR algorithm we may achieve even more scalability without significant degradation of accuracy.

II. Approach

Our goal is to reduce the number of non-relevant documents which comprise the vast majority of the training set, and then to examine the effect of learning from the (reduced) training set on a separate (non-reduced) test set. Pictorially, Fig. 3 describes the procedure. In order to achieve

[2] Note that for graded relevance labels, the usual practice (e.g. [10]) is to treat only some of the higher relevance labels as relevant while the rest as non-relevant. Specifically, among the labels in the range 0-4, labels 3 and 4 can be considered as relevant, and among labels in the range 0-2, labels 1 and 2 can be considered as relevant.

[3] Despite possessing huge computational resources, the commercial search engines are still in need of better scalability for their learning algorithms [2], [3], [11], because they need to frequently test newly developed systems (before releasing them to relevant departments) to check if the new setup works well in practice. As additional computation comes at a monetary cost, training time of the algorithms is crucial.

this goal we exploit the concept of undersampling techniques for imbalanced data from the machine learning literature [12]. Undersampling techniques aim at making the training set more balanced in terms of the number of instances from each class. In this study we investigate two approaches: (1) a random undersampling, and (2) a deterministic undersampling.

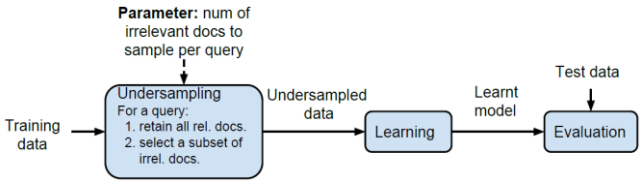

Fig. 3. Undersampling process of our approach.

The existing literature on improving the classification accuracy in the presence of imbalanced data can broadly be divided into two categories [12]: (1) a data-level approach, also known as sampling techniques, and (2) an algorithmic-level approach, also sometimes called as cost-sensitive learning. In this work we investigate the undersampling techniques from the first category – we do not consider oversampling techniques (and related techniques such as SMOTE [13]) as they increase the training set size.

Since the LtR data are divided by queries, we perform undersampling at the query level. That is, for each query, we retain all the relevant documents in the new training set, and then include a subset of non-relevant documents according to a criterion (either randomly or using a deterministic technique). We then learn a model on this (reduced) training set, and evaluate the model on a (non-reduced) test set to compare performance at different amounts of training data.

The procedure of random undersampling is, as the name implies, to randomly select a subset of non-relevant documents. As for the deterministic undersmapling, our goal is to retain the most "informative" non-relevant instances (i.e., query-document pairs) so that the learning algorithm can effectively learn to distinguish between relevant and non-relevant documents with fewer non-relevant documents in the training set. We identify the informative non-relevant documents using an effective feature which is BM25 score. The rationale is that the lower the BM25 score of an non-relevant document, the "more non-relevant" the document, and hence the more informative this document is (as an non-relevant document).

The configuration which uses the original training set is considered as the baseline.

III. Related Works

Aslam *et al.* [8] investigate different methods for top-*k* retrieval using a large corpus. That is, they study techniques for generating a better training set from a large unlabelled document collection. The training sets produced by their methods are, however, still highly imbalanced. Hence our intended work is complementary to theirs.

McDonald *et al*. [7] focus on the properties of a good training set through extensive empirical study on several large document collections. They examine the number of documents per query (i.e., different values of parameter *k*) to be used in top-*k* retrieval stage, and empirically search for the optimal values of *k* for different tasks and datasets. Their conclusions include: retrieval performance in general increases with increasing size of training sample (i.e., the values of *k*) up to a certain point (depending on the datasets), and afterwards the performance plateaus.

Dang *et al.* [6] develop an improved initial retrieval method by retrieving more relevant documents than the existing methods such as BM25. Their method utilizes some complex features like proximity based retrieval functions [14].

Long *et al*. [15] propose a technique based on Active Learning framework that includes an (unlabelled) example in the training set if this inclusion minimizes a ranking loss over the training set. The main motivation of using active learning is to reduce the large cost associated with manual labeling of documents. Their proposed method is compared with mainly the depth-*k* pooling method. Some related works such as Donmez *et al.* [16] and Yu [17] also try to find the (unlabelled) examples which, if added to a training set, increase the quality of the learned ranking function. This category of works does not differentiate between relevant and non-relevant documents, whereas our intended investigation is specifically concerned with the importance of non-relevant documents. That is, the training set generated by these methods is still likely to be highly imbalanced, and thus eligible for our investigation.

We emphasize the point that although one of the goals of most of the abovementioned works is to retrieve as many relevant documents as possible in a training set, in practice the disparity between the numbers of relevant and non-relevant documents is still high. As an example, in an attempt to increase the relevant documents, Chapelle and Cheng [18] employ the depth-*k* pooling method which uses more than one base ranker in the initial retrieval stage, but still get an imbalanced training set.

We thus observe that none of the studies that investigate into improving the quality of the training data are primarily concerned with the non-relevant documents, i.e., none of them are concerned with the imbalanced nature of the training data. As our techniques are applicable after applying those methods (and before the learning phase), this work is distinctive and complementary to the existing works.

IV. Model

The main focus of this article is the random forest based LtR algorithms. A random forest [19] is a conceptually simple but highly effective and efficient learning framework. It aggregates the predictions of a large number of (independent and variant) decision trees.

It has been shown that the ranking error is bounded by both the classification error [20] and regression error [21]. Hence practitioners of LtR oftentimes approach to the LtR problem with a classification or a regression model. In this setting, a classification (or regression) algorithm learns to

predict the relevance label of an individual query-document pair. During evaluation, the documents associated with a query are ranked in decreasing order of their relevance scores predicted by the learnt model. Since it treats the instances (i.e. feature vectors corresponding to the query-document pairs) independently from one another, this approach is called pointwise in the literature [3]. Although lacks objective functions tailored to LtR problem, this approach is still used by researchers because of its lower computational resource requirement and conceptual simplicity.

In this article we employ an LtR algorithm which utilizes concepts of both the classification and regression settings [22], [23] at the same time. To partition the data of a node of a tree of a random forest, a classification setting (entropy-based gain) is used. However, to assign a label to a data partition during evaluation phase, a regression setting is used in the following way: when the test instance falls into a data partition (i.e., a leaf node), the algorithm assigns (as the score to the test instance) the average of the relevance labels of all the documents of that data partition. Finally, as mentioned earlier, the documents are sorted in decreasing order of the predicted scores and presented to the user. Thus the training and the testing phase adopts a classification and a regression setting respectively. With regard to the data to begin with, each tree is learnt using a bootstrapped (without replacement) sample of the training set, and the bootstrapping is performed on a per-query basis. We call this algorithm *RF-point*.

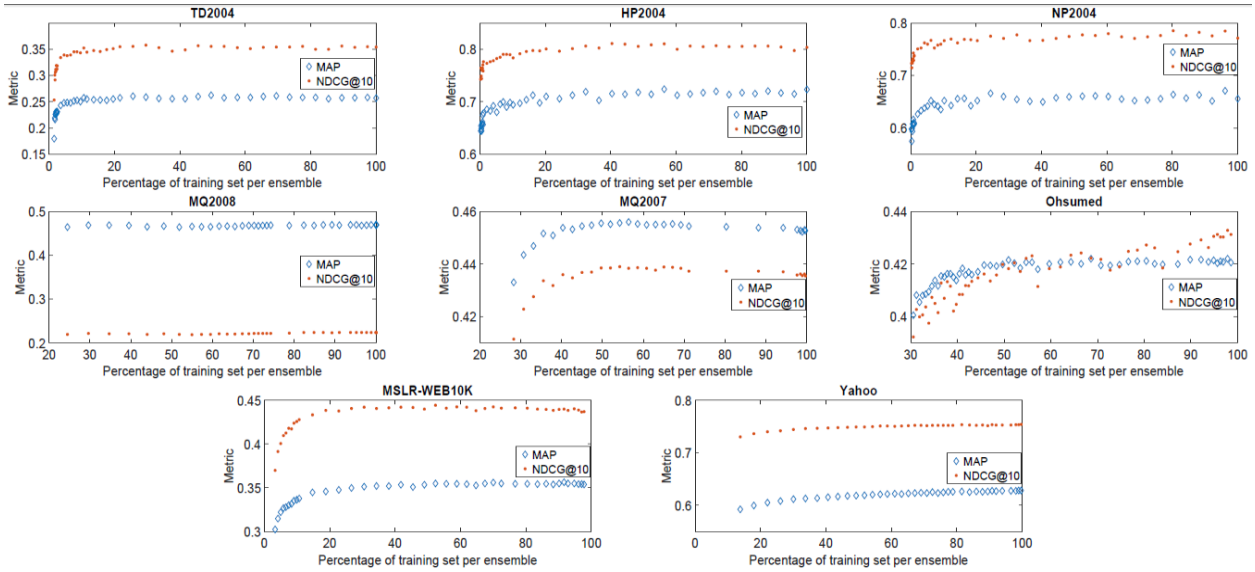

Fig. 4. Results of random undersampling approach with RF-point algorithm.

V. Experiments

This section describes the experimental settings and analyses the results.

A. Datasets

In total, eight prominent LtR datasets are used for our experiments that span a variety of IR tasks, namely topic distillation (TD2004), homepage finding (HP2004), named page finding (NP2004), domain-specific search (Ohsumed), and general web search (MQ2007, MQ2008, MSLR-WEB10K and Yahoo). Table 1 shows their statistics. For further details about these datasets, please see Qin *et al*. [5], in Letor website[4], Microsoft Research website[5], and Chapelle *et al*. [18].

B. Setup

As rank learners, in addition to the RF-point algorithm discussed in the previous section, we employ another rank-learner called RankSVM[6] [24] which is a popular pairwise LtR algorithm that formalizes LtR as a problem of binary classification on instance pairs, and solves the problem using support vector machines (SVM).

C. Evaluation Metrics

As evaluation metrics, we use two widely known measures, namely Normalized Discounted Cumulative Gain at 10th position (NDCG@10) and Mean Average Precision (MAP). Since the contemporary IR literature largely prefers NDCG over MAP, we give more emphasis on NDCG in our result analysis, although we make use of both of them. Readers may go through Ibrahim and Murshed [4] and Ibrahim [23] to know details about these metrics.

D. Using RF-point Algorithm

This section discusses results of RF-point algorithm with random and deterministic undersampling approaches.

Random Undersampling. Fig. 4 shows plots of random undersampling approach for the eight datasets.

We first discuss results of the six smaller datasets. Along the *x*-axis of a plot is the percentage of training set used for learning, and along the *y*-axis is performance. To suppress random fluctuations, each point in a plot corresponds to the average metric of five independent runs. We observe that initially, i.e., with very few non-relevant documents performance is comparatively poor, as expected; however, the level of degradation depends on the dataset. As the

[4] http://research.microsoft.com/en-us/um/beijing/projects/letor/
[5] http://research.microsoft.com/en-us/projects/mslr/
[6] http://research.microsoft.com/en-us/um/beijing/projects/letor/Baselines/RankSVM-Primal.html.

number of non-relevant document increases, so does the performance. In general, after the initial increase, relatively minor improvement in performance is gained as we increase the number of non-relevant documents (note the restricted interval on the y-axis); and this phenomenon is pronounced in the datasets where the distribution of the relevance labels is highly skewed, namely in TD2004, HP2004 and NP2004.

Table II shows the percentage of original training set required to reach a given level of performance. We observe that the percentage of training set required to achieve close performance to the baseline varies from task to task. The TD2004, HP2004 and NP2004 datasets benefit most from undersampling – only 10.6%, 7.2% and 4.2% training data are required to reach within 98% of baseline performance respectively. Ohsumed, MQ2007 and MQ2008 datasets are moderately benefitted due to the comparatively less imbalance nature of their data – approximately half, one-third and one-fourth of the original data is needed for these three datasets to reach within 98% of the baseline performance respectively. As for the training time, the TD2004, HP2004 and NP2004 datasets achieve 10-21 times gain over the baseline.

For the two larger datasets, namely MSLR-WEB10K and Yahoo, we report results of one run (instead of five) as these datasets are sufficiently large to suppress the possible fluctuation in the plots (indeed, their plots are found to be comparatively smoother). Also, for better comparison with the baseline, here we focus on the amount of training data required to reach within 98% and 99% of the baseline performance. From the plots we observe that the undersampling approach works better for MSLR-WEB10K dataset than for Yahoo dataset because the relevance label distribution is more skewed in the former case than the latter (cf. Table I). On the MSLR-WEB10K dataset, approximately one-fifth of the data is required to reach within 99% of baseline performance (with 8 times smaller training time), whereas on the Yahoo dataset an increased amount of non-relevant documents is found to be always useful.

TABLE I: STATISTICS OF THE DATASETS (SORTED BY # QUERIES). IN THE LAST ROW, 973/15 FOR TD2004 MEANS THAT THERE ARE 973 AND 15 DOCUMENTS OF LABEL 0 AND 1 RESPECTIVELY

Characteristic	**TD2004**	**HP2004**	**NP2004**	**Ohsumed**	**MQ2008**	**MQ2007**	**MSLR-WEB10K**	**Yahoo**
Task	Topic Distillation	Homepage Finding	Named Page Finding	Medical Corpus	Web Search	Web Search	Web Search	Web Search
# Queries (overall)	75	75	75	106	784	1692	10000	29921
# Queries (train)	50	50	50	63	470	1015	6000	19944
# Features	64	64	64	45	46	46	136	519
# Rel. labels	2	2	2	3	3	3	5	5
# Query-doc pairs (overall)	75000	75000	75000	16000	15211	69623	1200192	709877
# Query-doc pairs (train)	50000	50000	50000	9684	9630	42158	723412	473134
# Docs per query	988	992	984	152	19	41	120	23
# Docs of diff. labels (0/1/2/3/4) per query	973/15	991/1	983/1	106/24/21	15/2.5/1	30/8/2	42/39/16/2/0.9	6/8/7/2/0.4

TABLE II: RANDOM UNDERSAMPLING WITH RF-POINT: % OF TRAINING DATA REQUIRED (PER ENSEMBLE) TO ACHIEVE A GIVEN PERCENTAGE OF PERFORMANCE OF BASELINE NDCG@10 (I.E., WITH 100% TRAINING DATA) ALONG WITH TRAINING TIME IMPROVEMENT

Dataset	**%Rel. Docs**	**%of training set required (per ensemble) for 95%-98%-99% of baseline performance**	
		Random undersampling	**Training time improvement**
Ohsumed	30	37 - 56 -98	2.8 - 1.8 - 1 times
MQ2007	26	30.7 - 33.1 -40.3	3.5 - 3.2 - 2.6 times
MQ2008	19	24.5 - 24.5 -95.2	3 - 3 -1 times
TD2004	1.5	4.5 - 10.6 - 21.7	13 - 10 -6 times
HP2004	0.3	0.31 - 7.2 -32.4	50 -18 -4 times
NP2004	0.1	0.82 - 4.2 - 24.4	40 - 21 - 6 times
Dataset	**%Rel. Docs**	**% of training set required (per ensemble) for 98%-99% of baseline performance**	
		Random undersampling	**Training time improvement**
MSLR-WEB10K	3	15-19	11 - 8 times
Yahoo	12	22-100	7 - 1 times

TABLE III: COMPARISON BETWEEN STANDARD RANDOM UNDERSAMPLING (I.E., ENSEMBLE LEVEL) AND TREE LEVEL RANDOM UNDERSAMPLING WITH RF-POINT: % OF TRAINING DATA REQUIRED (PER TREE) TO ACHIEVE A GIVEN PERCENT OF BASELINE (I.E., 63% TRAINING SET) PERFORMANCE (NDCG@10) ALONG WITH TRAINING TIME IMPROVEMENT. THE BEST VALUE IS IN ***BOLD AND ITALIC*** FONT

Dataset	**% of training set required (per tree) for 95%-98%-99% of baseline (i.e., 63% sample) performance**	
	Ensemble level	**Tree level**
Ohsumed	23 - 35 -62	***13 - 14 - 19***
MQ2007	19.3 - 20.9 - ***25.4***	***17.8 - 19.3 25.4***
MQ2008	***15.4 - 15.4*** - 60.0	***15.4 - 15.4 - 18.6***
TD2004	2.9 - 6.7 - 13.6	***2.2 - 4.0 - 7.2***
HP2004	0.2 - ***4.5*** - 20.4	***0.19*** - 5.7 - ***17.6***
NP2004	0.51 - ***2.6 - 15.4***	***0.32*** - 3.8 - 30.2
Dataset	**% of training set required (per tree) for 98%-99% of baseline (i.e., 63% sample) performance**	
	Ensemble level	**Tree level**
MSLR-WEB10K	9 - 12	***6-9***
Yahoo	14 -63	***11-45***

This sub-section thus reveals that when RF-point employs the random undersampling approach, both the smaller and larger datasets are candidates for getting improved training time given that their relevance label distributions are sufficiently skewed.

Fig. 5 shows that for all the datasets the training time increases linearly with increasing training set size.

Deterministic Undersampling. Our second approach performs a deterministic selection of non-relevant documents. The procedure is as follows. For a query, we, as before, include all the relevant documents in the training set. We then sort the non-relevant documents in ascending order of the values of an effective individual ranker, namely the BM25 scorer. In the first configuration, we thus include only the first non-relevant document (i.e., with the minimum feature value) for that query to the training set. For subsequent configurations, additional non-relevant documents are included in the previously sorted order. We call this approach the ascending order approach.

Fig. 6 shows the plots for the ascending order approach. All the datasets, in contrast to the random undersampling approach, exhibit an increasing trend up to the last point of a curve. A possible explanation for rising performance up to the last configuration is as follows. When we continue to include non-relevant documents having comparatively larger BM25 scores, the performance still continues to slightly increase because even though BM25 is a good ranker, there do exist some non-relevant documents which have high BM25 scores, and including these documents in the training set helps the learning algorithm to learn these patterns in addition to the normal patterns (i.e., low BM25 scores) of the non-relevant documents. Another observation is that for most of the datasets for the very initial points of a curve there seems to be no systematic trend of performance. This is possibly due to the fact that the missing values of the features have been replaced by 0.0 by the dataset providers; so when we include the initial documents in ascending order, the documents with missing values are included which act as "noise" to our intended goal (i.e., to include documents with lower BM25 scores). Once the documents having genuine BM25 scores start to be included, the systematic trends appear.

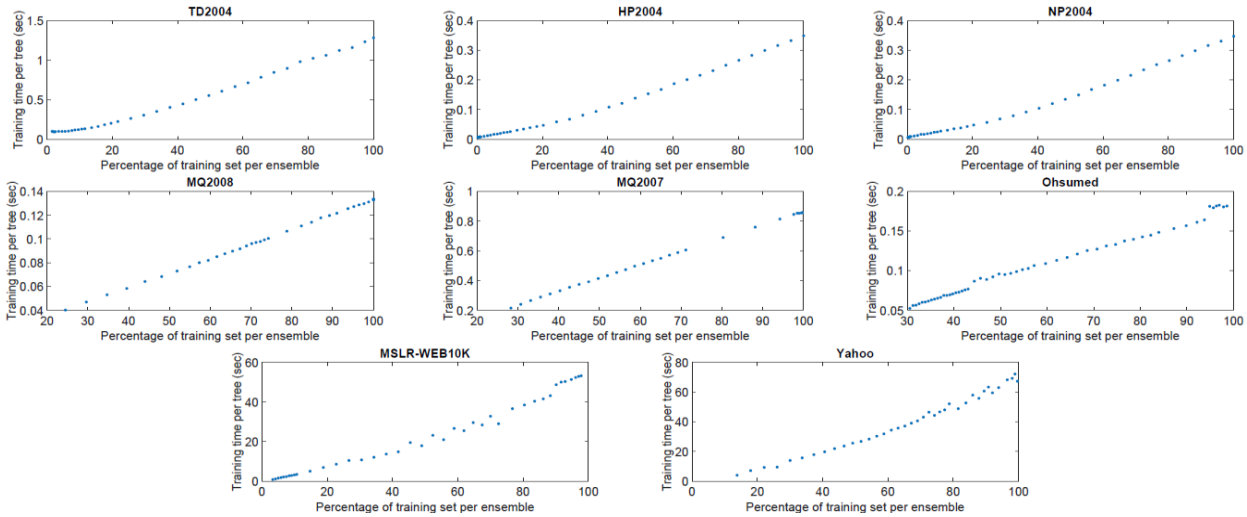

Fig. 5. Training time of RF-point against the percentage of training set used to learn the ensemble.

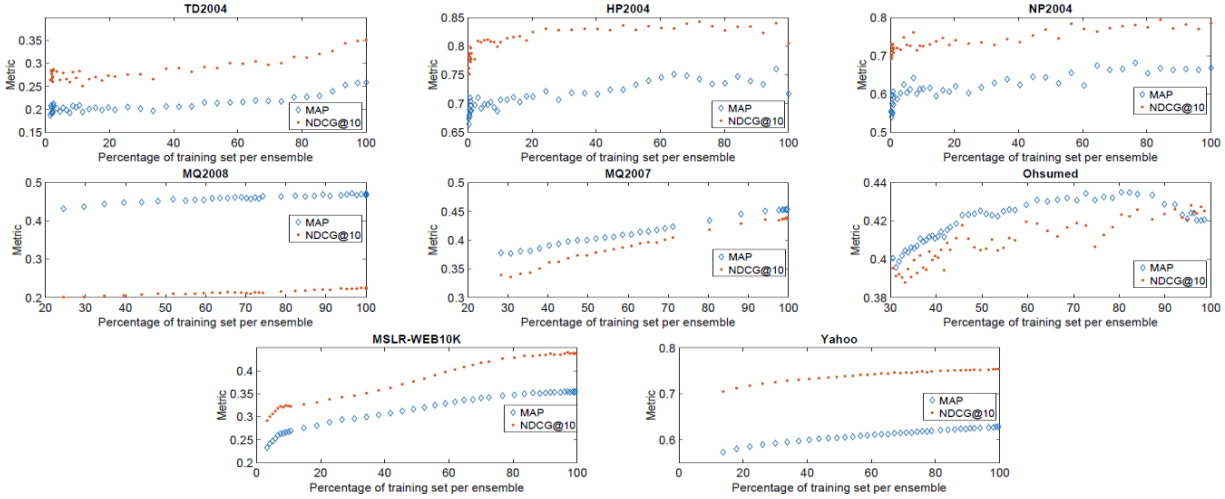

Fig. 6. Results of deterministic undersampling approach with RF-point algorithm.

Comparison between Two Approaches. In Fig. 7 we compare the performance of the two undersampling approaches investigated. We notice that in terms of requirement of percentage of training set to achieve close performance to the baseline, the random undersampling wins almost consistently over the deterministic one (with an

exception of HP2004). This behavior is likely to be due to the fact that the random sampling approach preserves the original distribution of non-relevant documents, and as explained during the analysis of ascending order approach that although the importance of non-relevant documents having lower feature values are perceived, the higher feature values are also required to enhance the hypothesis space from which the ranking model is picked.

We reiterate that our results are not directly comparable with that of existing methods (explained in Section III) since we investigate a previously unexplored area of undersampling only the non-relevant documents, in the post-labelling phase, whereas the existing works focus on sampling both relevant and non-relevant documents, before labelling.

E. Using RankSVM Algorithm

In order to claim that the findings from RF-point are generalizable to any rank-learner, we conduct an experiment with a state-of-the-art LtR algorithm, namely RankSVM. Fig. 8 shows the results for random undersampling approach. We see that the trends found in RF-point algorithm are mostly observed in the plots of RankSVM as well. This indicates that the findings emerged from our technique of undersampling are most likely to be invariant to different types of rank-learning algorithms.

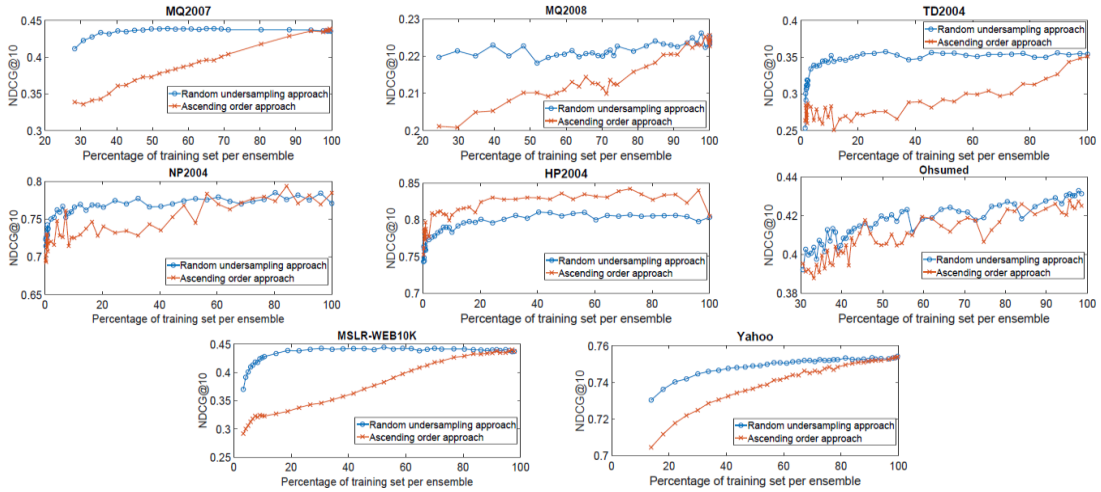

Fig. 7. Performance comparison of RF-point between the random undersampling and deterministic undersampling on different datasets in terms of NDCG@10.

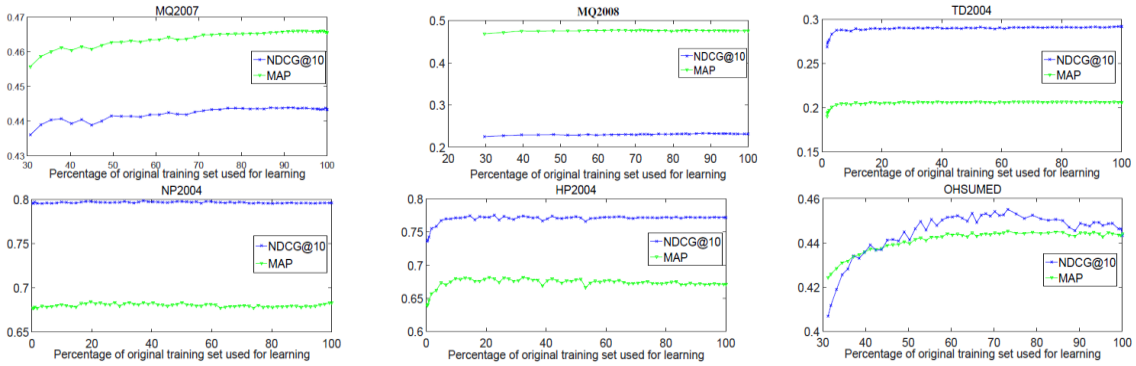

Fig. 8. With RankSVM algorithm, results for random undersampling approach across six datasets.

TABLE IV: PERFORMANCE COMPARISON AMONG RF-POINT (BASELINE), RF-POINT WITH OVERSAMPLING (OVERALL), AND RF-POINT WITH OVERSAMPLING (PER QUERY). AN AVERAGE OVER FIVE INDEPENDENT RUNS IS REPORTED (AND EACH RUN IS THE RESULT OF FIVE-FOLD CROSS-VALIDATION), AND THE WINNING VALUE IS GIVEN IN *ITALIC* FONT

Dataset	Metric	Baseline	Oversampling (overall)	Oversampling (per query)
MQ2007	**NDCG@10**	0.4360	0.4332	*0.4377*
	MAP	0.4524	0.4518	*0.4547*
MQ2008	**NDCG@10**	0.2234	0.2238	*0.2271*
	MAP	0.4693	0.4728	*0.4736*
Ohsumed	**NDCG@10**	*0.4306*	0.4096	0.4109
	MAP	*0.4213*	0.4057	0.4057
TD2004	**NDCG@10**	*0.3513*	0.3204	0.3263
	MAP	*0.2574*	0.2349	0.2358
NP2004	**NDCG@10**	*0.7860*	0.7071	0.7339
	MAP	*0.6647*	0.5773	0.6038
HP2004	**NDCG@10**	*0.8082*	0.7904	0.8001
	MAP	*0.7174*	0.6894	0.7016

VI. A SCALABLE UNDERSAMPLING WITH RANDOM FOREST

In the previous section we have shown that for imbalanced datasets we can select a random subset of non-relevant documents (per query) without significantly degrading the performance. This section shows that the inherent

architecture of a random forest offers a better way to achieve this effect.

A. Approach

Since a random forest is an aggregation of many independent base learners, our idea is to exploit undersampling at the level of base learners instead of considering the learner (i.e., the ensemble) as a black-box - this way the effect of information loss due to bootstrapping may be minimized. Recall that our standard approach to undersampling (cf. Fig. 3) was to undersample a subset of non-relevant documents per query, and thus to generate a new (smaller) training set for an RF (or any other rank-learner). We henceforth call this approach the ensemble-level-undersampling (ELU). In this section, we investigate another approach that performs undersampling for each individual tree of an RF. Elaborately, to learn a tree, the first step is as usual: to generate a bootstrapped sample. After that we impose a second step which retains only a random subset (of pre-defined cardinality) of non-relevant documents (per query) from the bootstrapped sample along with all the relevant documents. Pictorially, Fig. 9 describes the new approach which we term as the tree-level-undersampling (TLU). Thus unlike the ELU approach, the TLU approach does not need to exclude any non-relevant documents altogether from learning, rather each of them gets a chance to be used in some trees (given the ensemble is sufficiently large). We conjecture that the TLU approach will converge to the baseline (i.e., without using any undersampling) performance more quickly than the ELU approach.

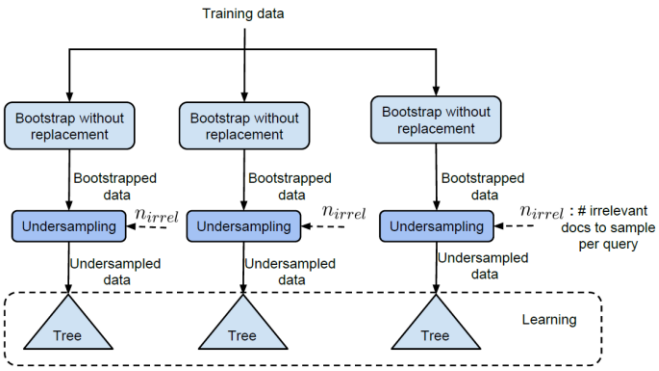

Fig. 9. Tree-level undersampling with RF-point.

Chen *et al.* [25] study some modifications of a random forest to tackle the class imbalance problem for classification. Since we are concerned with undersampling, below we summarize one of their techniques which is relevant to ours. For a tree, their method firstly uses a bootstrap sample of only the minority class examples, and after that it randomly samples a subset of majority class examples whose cardinality is equal to the number of minority class instances of the bootstrapped sample. However, this idea is not likely to be effective in the context of LtR because we need to perform sampling on a query-level, and in some datasets there is only one relevant document (i.e., minority class example) per query. Hence we do not consider their approach further.

B. Result Analysis

Table III shows the comparison between the ELU and TLU approaches. We see that as conjectured, in most of the datasets TLU approach achieves a given level of performance with smaller training data per tree (which in turn further reduces training time) as compared to the ELU approach. Using the data from Table III, Fig. 10 highlights this comparison for the 98% performance level.

We note that for most of the datasets we found a good number of cases where the TLU approach (with smaller data per tree) performs even better than the baseline (i.e., RF with no undersampling at all). The reason for this phenomenon is, we conjecture, due to the fact that the setting of the TLU approach reduces the correlation among the trees which is favorable to getting better performance.

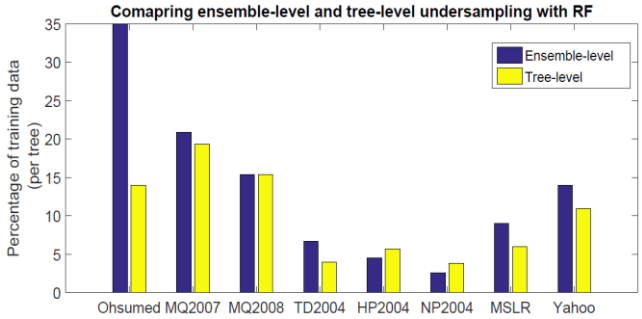

Fig. 10. Comparison between ensemble-level and tree-level undersampling with a random forest. The bars indicate the amount of training data (per tree) used to achieve 98% of NDCG@10 of the baseline (i.e., RF learnt with all training data); cf. Table IV.

VII. Discussion

Training time of the LtR algorithms is considered to be an important issue [2], [3], [11], [27]. Undersampling techniques reduce the training set size thereby decreasing the training time. Also, smaller training set requires smaller feature computation time, and some features are computationally costly to calculate [2], [7], [11], [26].

Our investigation is distinct from and complementary to the existing works on using a better initial retrieval approach (i.e., the top *k* retrieval as explained in Section I) in that our techniques are applicable after using these methods. The existing works focus on the quality of the training set by treating both relevant and non-relevant documents alike and before labelling the documents, whereas we focus on the training set size (and thus on learning time) in the context of the necessity of including large number of non-relevant documents. Moreover, our methods can also be extended to focus on the quality of the training set by incorporating more advanced undersampling methods in our framework.

The observations of our investigation are summarized below:

In many cases much less training data (as little as 19-32% of the training set for TD2004, NP2004, HP2004 and MSLR-WEB10K datasets) can be used without significant degradation of performance, namely retaining at least 99% of

the baseline performance. The findings also suggest that our method is more applicable to the tasks where the training data are highly imbalanced.

The random undersampling approach which preserves the true distribution of non-relevant documents in the undersampled training set appears to be better than a deterministic undersampling which selectively includes the non-relevant document.

The tree-forest hierarchy of a random forest is more conducive to undersampling. When properly used, this structure further increases scalability (cf. Fig. 10).

Applying sampling techniques take only linear time in terms of the size of the training set.

The datasets for which we have found considerable gain in training time (e.g. TD2004, HP2004 and NP2004) are not in the same scale of size of that used in commercial IR systems. The bigger the datasets (having highly skewed relevance label distribution), the more positive effect is likely to emerge from our technique. Result of one of the two big datasets, namely MSLR-WEB10K corroborates this conjecture – in spite of containing relatively more relevant documents than the abovementioned three datasets, here the undersampling approach worked quite well (cf. Table II).

Our approach is applicable after the relevance judgements of the documents are labelled. Hence a natural concern is, our approach does not minimize that large cost associated with human labelling. As such, this investigation is more useful where click-through data are available so that methods for automatic labeling [28] can be used.

In the literature of classification, oversampling of minority class examples is a closely related topic to undersampling. However, this increases the learning time, and the learning time is the main focus of this article. Yet for the sake of comprehensiveness of our investigation, we performed oversampling on six datasets by randomly duplicating the minority class instances until their presence becomes equal to that of the majority class instances. We also implemented a query-level version of it. After applying RF-point with these two settings, we mostly found slightly poorer performance than the baseline. Table IV shows the results of RF-point after applying an oversampling technique on the training set.

VIII. Conclusion

The core motivation of this research was to investigate a particular characteristic of an LtR environment - namely the relevance label distribution of a training set - given a limited budget of computational resources, so that an IR system developer can take a more informed decision on the preparation of the training set. We have employed undersampling techniques to reduce training set size in order to achieve better scalability. This investigation reveals that for highly imbalanced datasets our method will be useful. We have also successfully utilized the hierarchical tree-forest structure of a random forest to perform undersampling that resulted in better even scalability. Our work can be further extended by using more advanced undersampling methods.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Muhammad Ibrahim carried out the research.

References

[1] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*, Cambridge Uni. Press, vol. 1, 2008.

[2] T. Y. Liu, *Learning to Rank for Information Retrieval*, Springerverlag Berlin Heidelberg, 2011.

[3] H. Li, "Learning to rank for information retrieval and natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 4, no. 1, pp. 1–113, 2011.

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

[5] 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*, vol. 13, no. 4, pp. 346–374, 2010.

[6] V. Dang, M. Bendersky, and W. B. Croft, "Two-stage learning to rank for information retrieval," *Advances in Information Retrieval*, pp. 423–434, Springer, 2013.

[7] C. Macdonald, R. L. Santos, and I. Ounis, "The whens and hows of learning to rank for web search," *Inf. Retrieval*, pp. 1–45, 2012.

[8] J. A. Aslam, E. Kanoulas, V. Pavlu, S. Savev, and E. Yilmaz, "Document selection methodologies for efficient and effective learning-to-rank," in *Proc. 32nd intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2009, pp. 468–475.

[9] V. Pavlu, "Large scale IR evaluation," *ProQuest LLC*, 2008.

[10] A. Mohan, Z. Chen, and K. Weinberger, "Web-search ranking with initialized gradient boosted regression trees," *Proceedings of the Learning to Rank Challenge*, pp. 77–89, 2011.

[11] O. Chapelle, Y. Chang, and T. Y. Liu, "Future directions in learning to rank," in *Proc. JMLR Workshop*, 2011, vol. 14, pp. 91–100.

[12] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowledge and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority oversampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[14] M. Bendersky, D. Metzler, and W. B. Croft, "Learning concept importance using a weighted dependence model," in *Proc. 3rd ACM intl. conf. on Web Search and Data Mining*, ACM, 2010, pp. 31–40.

[15] B. Long, O. Chapelle, Y. Zhang, Y. Chang, Z. Zheng and B. Tseng, "Active learning for ranking through expected loss optimization," in *Proc. the 33rd International ACM SIGIR Conf. on Research and Development in Information Retrieval*, ACM, 2010, pp. 267–274.

[16] P. Donmez and J.G. Carbonell, "Optimizing estimated loss reduction for active sampling in rank learning," in *Proc. 25th International Conf. on Machine Learning*, ACM, 2008, pp. 248–255.

[17] H. Yu, "Svm selective sampling for ranking with application to data retrieval," in *Proc. the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ACM, 2005, pp. 354–363.

[18] O. Chapelle and Y. Chang, "Yahoo! learning to rank challenge overview," *Journal of Machine Learn. Research-Proc*, pp. 1–24, 2011.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] P. Li, C. Burges, and Q. Wu, "Learning to rank using classification and gradient boosting," *Advances in Neural Information Processing Systems*, vol. 19, 2007.

[21] D. Cossock and T. Zhang, "Subset ranking using regression," *Learning Theory*, pp. 605–619, 2006.

[22] P. Geurts and G. Louppe, "Learning to rank with extremely randomized trees," in *Proc. JMLR: Workshop*, 2011, vol. 14.

[23] 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, 2016.

[24] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. 8th ACM SIGKDD INTL. CONF. on Knowledge Discovery and Data Mining*, ACM, 2002, pp. 133–142.

[25] C. Chen, A. Liaw, and L. Breiman, *Using Random Forest to Learn Imbalanced Data*, University of California, Berkeley, 2004.

[26] F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato, "Feature selection for ranking using boosted trees," in *Proc. the 18th ACM Conf. on Inf. and Knowledge Management*, ACM, 2009, pp. 2025–2028.
[27] M. Ibrahim, "Reducing correlation of random forest-based learning-to-rank algorithms using sub-sample size," *Computational Intelligence*, vol. 35, no. 2, Wiley, 2019.
[28] J. Xu, C. Chen, G. Xu, H. Li, and E. R. T. Abib, "Improving quality of training data for learning to rank using click-through data," in *Proc. the third ACM International Conference on Web Search and Data Mining*, ACM, 2010, pp. 171–180.

Muhammad Ibrahim obtained his Ph.D. degree from Monash University, Australia. Prior to that, he obtained his M.Sc. and B.Sc. Hons. degrees from the Dept. of Computer Science and Engineering, University of Dhaka, Bangladesh. He has so far published 15 research papers. At present, he is working as a lecturer at the Dept. of Computer Science and Engineering, University of Dhaka, Bangladesh. His current research interest includes machine learning, information retrieval, and artificial intelligence.