Evolutionary Algorithms and Machine Learning Techniques for Information Retrieval

Osman Ali Sadek Ibrahim, MSc

A thesis submitted for the degree of
Doctor of Philosophy

ASAP Research Group
School of Computer Science
The University of Nottingham
United Kingdom

September, 2017

Dedicated to

My Parents, my wife and my daughters

Evolutionary Algorithm and Machine Learning Techniques for Information Retrieval

Osman Ali Sadek Ibrahim

Submitted for the degree of Doctor of Philosophy
September, 2017

Abstract

In the context of Artificial Intelligence research, Evolutionary Algorithms and Machine Learning (EML) techniques play a fundamental role for optimising Information Retrieval (IR). However, numerous research studies did not consider the limitation of using EML at the beginning of establishing the IR systems, while other research studies compared EML techniques by only presenting overall final results without analysing important experimental settings such as the training or evolving run-times against IR effectiveness obtained. Furthermore, most papers describing research on EML techniques in IR domain did not consider the memory size requirements for applying such techniques. This thesis seeks to address some research gaps of applying EML techniques to IR systems. It also proposes to apply (1+1)-Evolutionary Strategy ((1+1)-ES) with and without gradient step-size to achieve improvements in IR systems. The thesis starts by identifying the limitation of applying EML techniques at the beginning of the IR system. This limitation is that all IR test collections are only partially judged to only some user queries. This means that the majority of documents in the IR test collections have no relevance labels for any of the user queries. These relevance labels are used to check the quality of the evolved solution in each evolving iteration of the EML techniques. Thus, this thesis introduces a mathematical approach instead of the EML technique in the early stage of establishing the IR system. It also shows the impact of the pre-processing procedure in this mathematical approach. The heuristic limitations in the IR processes such as in pre-processing procedure inspires the demands of EML technique to optimise IR systems after gathering the relevance labels. This thesis proposes a (1+1)-Evolutionary Gradient

Strategy ((1+1)-EGS) to evolve Global Term Weights (GTW) in IR documents. The GTW is a value assigned to each index term to indicate the topic of the documents. It has the discrimination value of the term to discriminate between documents in the same collection. The (1+1)-EGS technique is used by two methods for fully and partially evolved procedures. In the two methods, partially evolved method outperformed the mathematical model (Term Frequency-Average Term Occurrence (TF-ATO)), the probabilistic model (Okapi-BM25) and the fully evolved method. The evaluation metrics for these experiments were the Mean Average Precision (MAP), the Average Precision (AP) and the Normalized Discounted Cumulative Gain (NDCG).

Another important process in IR is the supervised Learning to Rank (LTR) of the fully judged datasets after gathering the relevance labels from user interaction. The relevance labels indicate that every document is either relevant or irrelevant in a certain degree to a user query. LTR is one of the current problems in IR that attracts the attention from researchers. The LTR problem is mainly about ranking the retrieved documents in search engines, question answering and product recommendation systems. There are a number of LTR approaches from the areas of EML. Most approaches have the limitation of being too slow or not being very effective or presenting too large a problem size. This thesis investigates a new application of a (1+1)-Evolutionary Strategy with three initialisation techniques hence resulting in three algorithm variations (ES-Rank, IESR-Rank and IESVM-Rank), to tackle the LTR problem. Experimental results from comparing the proposed method to fourteen EML techniques from the literature, show that IESR-Rank achieves the overall best performance. Ten datasets; which are MSLR-WEB10K dataset, LETOR 4 datasets, LETOR 3 datasets; and five performance metrics, Mean Average Precision (MAP), Root Mean Square Error (RMSE), Precision (P@10), Reciprocal Rank (RR@10), Normalised Discounted Cumulative Gain (NDCG@10) at top-10 query-document pairs retrieved, were used in the experiments. Finally, this thesis presents the benefits of using ES-Rank to optimise online click model that simulate user click interactions. Generally, the contribution of this thesis is an effective and efficient EML method for tackling various processes within IR. The thesis advances the understanding of how EML techniques can be applied to improve IR systems.

October 30, 2017

Declaration

The work in this thesis is based on research carried out at the ASAP Research Group, the School of Computer Science, The University of Nottingham, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

I appreciate for ALLAH who give me the ability to do this research. HE helped me in every moment in my life and giving me the blessing, the motivation and the power for doing this research. Secondly, I am much appreciate to my family for their help in my whole life. I think that without their support, I can not finish this work.

I am deeply grateful to my main supervisor Dr. Dario Landa-Silva for his guidance, encouragement and patience throughout the doctoral programme. In particular, for his help in securing all the necessary funding. He is always being willing to review my writing and giving me a feedback before the deadlines. His support has been much more than his academic duties.

I would like to thank as well my second supervisor Dr. Steven Bagley for his insightful comments and suggestions before each annual review. My gratitude goes also to my viva voce examiners Professor Bob John and Dr. Craig Macdonald. Their valuable comments were included in the final version of this thesis. I also more grateful for my ASAP group colleagues because they help me more than expected during my PhD journey. Furthermore, I am appreciate for Professor Bob John as the Head of ASAP group who support all ASAP researchers with every support available not only as Professor and supervisor, but also as a father for every PhD student in ASAP.

Finally, this research programme would not have been possible without the financial support provided by: Minia University, Egypt via the scholarship (M11/11). Thus, I appreciate their support to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In many contexts, such as textual documents on the web, data changes rapidly over time. Hence, there is a growing need to develop effective methods of automated Information Retrieval (IR) systems that provide reasonable results and meet users needs. An IR system is an information system used to store items of information that need to be processed, indexed, searched, retrieved and disseminated according to various users' needs. Most IR systems store collections of text documents acting as a data repository.

Previous studies of IR research employed *Mathematical Models (MM), Probabilistic learning Models (PM), Language learning Models (LM), Evolutionary and Machine Learning models (EML)* to store and then rank the retrieved documents. Various Term-Weighting Scheme (TWS) methods have been used in these models. The TWS is a mathematical or probabilistic or Boolean equation used to assign real numbers to each significant word in IR system documents. This real value denotes the importance of the word within the document to allow efficient document retrieval.

The ability to rank the retrieved documents with respect to their relevance according to the user query is an important requirement in the domain of IR research. The historical user interaction data with the IR system can be used as a method to measure the relevance degree for each clicked and browsed document. However, the historical user interaction data suffers from many problematic issues, such as random user decisions, unfamiliarity with the system, education level and user intelligence (Lorigo et al., 2006; Al-Maskari

and Sanderson, 2011). The relevance judgements are values that indicate the relevance degrees of the user queries to specific documents in the IR test collection. These values are extracted from the historical user interaction data with the IR system. The relevance judgements of the documents are limited and expensive to have fully judged test collections at the start of the IR system. Thus, standardised test collections have been produced to simulate actual IR test collections. In these collections, the relevance judgement value represents the degree of relevance of each document relative to user query.

Traditionally, the process of ranking retrieved documents was based on learning document or query representations using probabilistic and EML techniques. These techniques are based on *Term Vector Model (TVM)* representation. The TVM approach represents each document and query as a vector of term-weights. Recent evidence suggests that the use of only one term-weighting model is not sufficient for effective IR systems (Qin et al., 2010; Qin and Liu, 2016; Liu, 2011; Liu et al., 2007; Hofmann, 2013; Schuth, 2016). Thus, recent IR research has investigated data features for learning, to develop rank models based on combinations of MM, PM and LM term-weighting models. As a consequence, recent test collections contain more than one of the MM, PM and LM term-weighting models as feature vector representations. This feature representation is called *Feature Vector Model (FVM)*.

Finally, recent research provides fully judged collections of documents that simulate those available from existing search engines. This type of fully judged test collection is usually obtained some time (maybe several years) after establishing the IR system (or search engine), also following the gathering of historical user data interactions and reducing the noise from user behaviours. These collections can then be used in supervised *Learning to Rank (LTR)* models, applying various EML techniques.

Several attempts have been made to evolve document representations but without fully considering the limitations of EML techniques. These limitations are as follows:

1. IR test collections are only partially judged at the beginning of IR system. This

because the lack of existing user interaction data with the IR system;

2. The memory usage of some EML techniques in IR is typically large when considering all the training documents; some techniques instead do not consider all the training documents in each learning iteration which can affect effectiveness; and

3. The computational runtime for most EML applications is typically large and increases substantially according to the size of the test collection.

1.1 Research Motivation

As mentioned above, ranking retrieved documents with respect to their relevance to the user query is an important issue in an IR system. A huge amount of time and effort is spent searching for relevant documents, if the IR system cannot efficiently rank the retrieved documents. This inspires the need for developing EML approaches to improve the performance of IR systems.

A considerable amount of research has been published regarding the use of EML techniques to improve the performance of IR systems. However, the problem size for *Evolutionary Computation (EC)* techniques in the IR literature research has yet to be addressed (Cummins and O'Riordan, 2006; Oren, 2002; Escalante et al., 2015). Thus, this particular limitation can hinder the application of these techniques to large test collections. This is because using the whole test collection will usually slow down significantly the evolutionary process of EC techniques in TVM approaches. EC techniques have been used to evolve all the document or the query representations of the test collections.

On the other hand, the increase in the problem size of the technique increases the computational evolving runtime (Nopiah et al., 2010). According to (Escalante et al., 2015), the computational runtime problem for evolving TWS, using a subset of 20-Newsgroup collection (Rennie, 2015), with population-based *Genetic Programming (GP)* was 18 hours. The other GP approaches (Cummins and O'Riordan, 2006; Oren,

2002) were applied to very small collections using either a cluster of computers or taking long computational runtimes. However, evolving the term weights of the document representations and evolving TWS using EC have resulted in better IR effectiveness in terms of *Mean Average Precision (MAP)* (Cummins, 2008; Cordon et al., 2003).

Another limitation of EML approaches for IR relates to the relevance judgement values collected from user interactions with IR systems. These values cannot be gathered in the early stages of system development when building the IR systems for large test collections. The relevance judgement requires sufficient historical user interactions with the test collection to maximise the IR system's effectiveness. Thus, there is a need for a new perspective when applying the mathematical models and EML in the early stage of IR systems which have partially relevance judged values or do not have any. In this thesis, Chapters 5 and 6 cover this new perspective on applying EML techniques for TVM with well-known partially judged test collections. The mathematical model is proposed as a starting point for document representations to construct an IR system without the need to have the user feedback at the beginning. Then, Chapter 6 proposes a new methodology for evolving document representation in partially judged test collection stage considering the problem size and the limitation of existing the relevance judgement values.

As more time passes using the IR system, the test collection gradually becomes a fully or nearly fully judged collection, with comprehensive user relevance feedback. Thus, research trends can make use of the new challenge of test collection representation after the fully judged relevance feedback became available from users. In this representation, each document is represented as a vector of features that include numerous of TWS, document reputation and business importance features on the web. In this thesis this representation is referred to as Feature Vector Model (FVM) or Bag-of-Features (BoF). The research field using this representation is referred to as supervised LTR. This field typically uses EML techniques. This is because the use of an IR system by one TWS without including the business importance and web page reputation of the document is not sufficient for effective IR system on the web (such as in Search engines).

Another motivation for research work in this thesis is the need for comparing computational efficiency of the various EML techniques proposed in the literature for LTR. Most comparisons to date, present only the accuracy for EML techniques without mentioning the corresponding computational training runtimes. Moreover, some studies comparing EML techniques did not reveal the experimental settings used to produce the reported experimental results. In other words, numerous researchers have considered the accuracy values when comparing EML techniques but they have not reported on the computational runtime for the proposed approach. Additionally, this thesis is the first to investigate the LTR problem using an adaptive *Evolutionary Strategy (ES)*. The adaptive EC techniques have outperformed non-adaptive EC techniques (such as GP and Differential Evolution) in many Combinatorial Optimisation Problems, which was an additional motivation for this research. The ES-Rank algorithm proposed in this thesis is much faster than those for most of the population-based EML techniques in the LTR problem, as it requires the lowest memory size. In the FVM approach, some of the EML techniques also experienced the same issues and limitations as those in the TVM approach. These issues are related to the computational runtime and the effect of the problem size on the performance of the EML technique. This causes some alternative EML techniques do not consider the entire training instances (documents) in each learning iteration to overcome these issues. Unfortunately, the accuracy values of the ranking model by these techniques are not effective compared to methods that consider all training instances (documents) in each learning iteration. The computational runtime required to have an effective approach increases exponentially with increases in the size of the input data, when the technique considers all training documents in each learning iteration. A further limitation when handling large data-set is the memory size. These issues have not been investigated in previous research published in the existing literature.

1.2 Thesis Statement and Scope

This thesis hypothesis is that evolutionary and machine learning (EML) techniques can improve the accuracy of IR systems using the historical user interaction data. In

particular, it suggests that the proposed (1+1)-Evolutionary Strategy with/without gradient mutation step-size will have the lowest memory size and the lowest computational runtime requirements when improving IR systems, if compared with the other EML techniques reported in the literature. Furthermore, we hypothesise that the use of static collection characteristics when performing term-weighting functions has a negative effect on the accuracy of a TVM IR system. Moreover, that relevance judgement values or historical user interactions are vital for determining the use of evolutionary and machine learning or the use of mathematical term-weighting functions. We also illustrate the importance of the initialisation procedure in ES technique in LTR problem. Finally, we illustrate that the ES technique can improve the accuracy of the user interaction simulation (Dependent Click Ranking Model). This click model is a well-known click model related to an online LTR package (Loret) and this thesis compared its performance with ES-Rank performance.

The current study of this thesis examines the use of evolutionary and machine learning techniques in an offline LTR evaluation. It means that the explicit relevance judgement labels existing in the standardised test collections are used to evaluate the quality of the solutions produced by EML techniques. However, the current online LTR evaluation research uses the same explicit relevance judgement values to produce user click models first and then evaluate the quality of LTR technique based on click models which simulate user clicks (Schuth, 2016; Hofmann, 2013). This means that the online LTR research in the literature uses click models to simulate online user interaction and then evaluate the EML by the explicit relevance judgement labels. This is because the cost and time demands of real online user interaction evaluation are daunting. Moreover, real user relevance judgement assessment is biased by the user familiarity with the IR system and user education levels, as mentioned in the literature (Lorigo et al., 2006; Al-Maskari and Sanderson, 2011; Kharitonov, 2016). In addition to the time consuming for having interactive user interaction during the training time is significant. Thus, the test collections used in this thesis are standardised collections created by TREC, Microsoft IR research teams and well-known IR researchers (Liu et al., 2007; Qin et al., 2010; TREC, 2016*a*; Voorhees, 2000; Soboroff, 2007; Tonon et al., 2015; Urbano, 2016; Cleverdon, 1960;

Sanderson, 2010; Hersh et al., 1994). The relevance judgement values of these collections are collected by trained and expert human annotators in the test collection topic domains and validated by multiple IR systems. These test collections are also used in online LTR research in the literature due to the limitations of having online non-biased relevance judgement labels (Schuth, 2016; Hofmann, 2013). These test collections include many types of user relevance judgements, provided by expert users from Microsoft and the National Institute of Standards and Technology (Liu et al., 2007; Qin et al., 2010; TREC, 2016*a*; Voorhees, 2000; Soboroff, 2007; Tonon et al., 2015; Urbano, 2016). In addition, the test collections used in this thesis have been created, validated and tested over several years before dissemination for IR researchers (Liu et al., 2007; Qin et al., 2010; TREC, 2016*a*; Voorhees, 2000; Soboroff, 2007; Tonon et al., 2015; Urbano, 2016; Cleverdon, 1960; Sanderson, 2010; Hersh et al., 1994). Chapter 3 discuss the details of creating these test collections. Moreover, the cognitive level variation issues for creating the datasets (test collections) relevance judgement values is out of the thesis scope.

1.3 Research Questions

The research questions identified in this research are as follows:

1. What are the limitations of applying EML techniques on IR systems for TVM representation (Bag-of-Words)? This question is addressed in Chapters 5 and 6.

2. Why there is a need for mathematical (non-learning) term-weighting schemes in IR systems? This question is addressed in Chapters 3 and 5.

3. What is the importance of relevance judgement and relevance labels in evolutionary and machine learning approaches? This question is addressed in Chapters 3, 4, 5 and 6.

4. What is the limitation in respect of static collection characteristics for different IR weighting functions on TVMs? How will this parameter affect dynamic variation in test collection? These questions are addressed in Chapter 5.

5. What is the impact of the pre-processing procedure (stop-word removal) in term-weighting functions? This question is addressed in Chapter 5.

6. What is the importance of EML approaches in IR to overcome the impact of pre-processing (stop-word removal and stemming) to enhance effectiveness? This question is addressed in Chapters 4 and 6.

7. What are the limitations of applying EML techniques on IR systems for FVM representation (Bag-of-Features)? This question is addressed in Chapters 4 and 7.

8. How can the adaptive (1+1)-evolutionary technique be used to improve the IR system with the lowest problem size and the lowest computational time? This question is addressed in Chapters 6 and 7.

9. What is the importance of the initialisation procedure in the (1+1)-ES technique? This question is addressed in Chapter 7.

10. Can the (1+1)-ES technique improve the user click ranking model? This question is addressed in Chapter 8.

1.4 Thesis Contributions

The research presented in this thesis contributes to understanding the application of evolutionary and machine learning techniques in the IR research field by:

1. An analysis of commonly used test collections, provides an argument in favour of using heuristic (non-learning) TWS instead of TWS and term weights evolved via EC techniques at the beginning of establishing IR system (Section 3.1.3). We believe that this analysis also supports the argument that more appropriate test test collections, instead of general IR test test collections, should be considered when using EC techniques to evolve TWS or term weights. This analysis is related to research questions 1, 2 and 3.

2. A new non-learning (mathematical) TWS is proposed termed the *Term Frequency With Average Term Occurrence (TF-ATO)*, with a *Discriminative Approach (DA)*

for removing less significant weights from the documents. A study is conducted to compare the performance of TF-ATO to the widely used TF-IDF approach, using various types of test collections such as sampled, pooled (Soboroff, 2007) and those from real IR systems (Hersh et al., 1994). Our experimental results show that the proposed TF-ATO gives higher effectiveness in both cases of static and dynamic test collections. This TWS can be used at the early IR stage to gather the user interaction feedback data. This mathematical TWS is related to the research questions 1, 2 and 4.

3. Using various test collections, we study the impact of our DA and the stop-words removal process for IR system effectiveness and performance when using the proposed TF-ATO and when using the TF-IDF. We find that these two processes have a positive effect on both TF-ATO and TF-IDF for improving the IR performance and effectiveness. This study is related to the research question 5.

4. A new method has been proposed to evolve better representations of documents in the collection for the trained queries with less computer memory usage. This is accomplished by evolving the Global Term Weights (GTWs) of the collection rather than evolving document representations for the whole collection as is typically done in other EC approaches in the literature. Hence, the main contribution of Chapter 6 is the development of a *(1+1)-Evolutionary Gradient Strategy ((1+1)-EGS)* with *Adaptive Ziggurat Random Gaussian Mutation* (Kuo and Zulvia, 2015; Doornik, 2005; Loshchilov, 2014) to evolve GTWs. The proposed methodology reduces the problem size, from evolving $(N \times M)$ document representation vectors to evolving $(1 \times M)$ vector, where N is the number of documents in the collection and M is the number of index terms in the collection. This method also considers the limitation of the relevance judgement of the partially judged test collections in EC techniques. This method is concluded from the research questions 1, 2, 3, 6 and 8.

5. An Evolutionary Strategy (ES) technique has been proposed 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 7.2. In order to assess the performance of ES-Rank, *Mean Average Precision (MAP)*, *Root Mean Square Error (RMSE)*, *Normalised Discounted Cumulative Gain (NDCG@10)*, *Reciprocal Ranking (RR@10)* and *Precision (P@10)* at top-10 query-document pairs retrieved (Liu, 2011; Li, 2014) are used and comparison is carried out against fourteen state-of-the-art LTR approaches from the literature. Experimental results in Chapter 7 show that ES-Rank performs very well when compared to those other methods in terms of MAP, RMSE, RR@10, NDCG@10 and P@10. 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 (Qin et al., 2010) 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). This technique is produced after addressing the limitation identified from the research question 7 and 8.

6. The importance of initialisation procedure in ES-Rank to tackle the LTR problem is investigated. The initialisation procedure in ES-Rank has been tuned based on Linear Regression and Support Vector Machine ranking model to produce IESR-Rank and IESVM-Rank methods. In order to assess the performance of IESR-Rank and IESVM-Rank, MAP, RMSE, NDCG@10, RR@10 and P@10 at top-10 query-document pairs retrieved (Liu, 2011; Li, 2014) are used and comparison is carried out against fourteen state-of-the-art LTR approaches from the literature. Experimental results in Chapter 7 show that the use of machine learning ranking model as an initialisation procedure has a positive impact on ES-Rank technique. Furthermore, the initialisation procedure using *Linear Regression (LR)* machine learning ranking model has a better positive impact for improving ES-Rank in most cases than *Support Vector Machine (SVM)* and zero values initialisation methods. This study is related to the research question 9.

7. The ES-Rank technique has been used to optimise Click Dependent Model (DCM) to tackle click ranking models problem. This click model is an extension of Cascade

Click Model (CCM). These click models are used in online LTR techniques as in Lerot package (Schuth et al., 2013). At the beginning, online LTR models are produced by Lerot package using online probabilistic interleave and DCM technique (Hofmann, 2013; Schuth, 2016). Then, ES-Rank optimises these ranking models. This study is related to the research question 10.

1.5 Publications

The following submitted/published articles have resulted from the research work described in this thesis

1. Osman A. S. Ibrahim and Dario Landa-Silva, A New Weighting Scheme and Discriminative Approach for Information Retrieval in Static and Dynamic test Collections, Proceedings of the 14th UK Workshop on Computational Intelligence (UKCI2014), Bradford UK, September 2014, doi: 10.1109/UKCI.2014.6930160.

2. Osman A. S. Ibrahim and Dario Landa-Silva, Term Frequency with Average Term Occurrences for Textual Information Retrieval, Soft Computing Journal, Volume 20, Issue 8, August 2016, doi: 10.1007/s00500-015-1935-7

   These first two publications cover the findings and research analysis existing in Chapter 5 and Section 3.1.3.

3. Osman A. S. Ibrahim and Dario Landa-Silva, (1+1)-Evolutionary Gradient Strategy to Evolve Global Term Weights in Information Retrieval, Advances in Computational Intelligence Systems Contributions presented at the 16th UK Workshop on Computational Intelligence, September 7-9, 2016, Lancaster, UK, doi: 10.1007/978-3-319-46562-3_25

   This publication presents the research outcomes in Chapter 6 which includes evolving Global Term-Weights (GTW) using (1+1)-Evolutionary Gradient Strategy.

4. Osman A. S. Ibrahim and Dario Landa-Silva, ES-Rank: Evolution Strategy Learning to Rank Approach, ACM Symposium on Applied Computing (SAC 2017), Marrakech, Morocco, April 03-07, 2017, ISBN: 978-1-4503-4486-9/17/04.

5. Osman A. S. Ibrahim and Dario Landa-Silva, An Evolutionary Strategy with Machine Learning for Learning to Rank in Information Retrieval, Accepted in Soft Computing Journal

   These two publications demonstrate the research findings in Chapter 7 which illustrates the using of ES to improve LTR problem.

1.6 Thesis Outline

This section summarises the contents of the remaining chapters of this thesis.

1.6.1 Chapter 2

This chapter presents the background material for IR and Evolutionary Strategies (ES). It begins by detailing the IR architecture and then explains various IR models, such as Boolean, vector space and language models. Then, the chapter introduces the introductory material for EML techniques in IR. Finally, the historical development of ES from method to another is presented in the last section (section 2.3).

1.6.2 Chapter 3

This chapter describes the available methods to create test benchmarks. It introduces the two available methods for creating the relevance judgement in the partially judged test collections. Then, the fully judged LETOR Benchmarks are presented. The partially judged test collections are used for both TVM in learning and mathematical techniques. They are used in Chapter 6 ((1+1)-Evolutionary Gradient Strategy for evolving global term weights) as a learning approach, while they are used in Chapter 5 for mathematical term-weighting schemes and to identify the limitations of EC applications in TVM.

The fully judged LETOR benchmarks are then used with supervised LTR techniques in Chapters 7 and 8.

1.6.3 Chapter 4

This chapter presents the relevant literature about EML in TVM and FVM. The chapter commences with the TVM and introduces the limitations that arise when applying EML in TVM when evolving term weights. The limitations are summarised in relation to the accuracy, the problem size and the computational runtime when evolving the whole document representations. This is in addition to the relevance judgement limitations that arise when applying EC or *Machine Learning (ML)* techniques to a newly established IR system. These limitation inspires the need for proposing a new methodology of evolving the document representations using EML techniques. This chapter also introduces the various LTR techniques on FVM. In this research domain, the EML techniques are used to learn the most suitable ranking model weights for the training data and for testing the performance of the techniques on the test data. The limitations of these techniques relate to their comparability with each other. This is because the majority of the literature research did not identify the experimental settings nor the training computational runtimes when making their comparisons. Furthermore, most of the EML techniques deliver high accuracy with a higher training runtime.

1.6.4 Chapter 5

This chapter presents the benefits of using non-learning term-weight schemes and it proposes a new TWS. TF-ATO is a Term Frequency-Average Term Occurrence weighting scheme. This TWS is proposed to avoid the limitations of the mathematical models with dynamic document variation that causes the collection size parameter to vary by adding/removing a document and consequently causing the need to re-compute the term weights for large variations in the IR test collection. However, TF-ATO is not sufficiently effective when compared with EC, Probabilistic and EML weighting functions. This generates a need for Discriminative Approach (DA) to overcome some of these limitations. Furthermore, the removal of stopwords as a pre-processing component in IR has an im-

portant role in TF-ATO performance variation, which should be considered when using the TF-ATO weighting function. The improved accuracy inspires the need for an EC approach to improve IR while using TF-ATO as local TWS, as shown in Chapter 6.

1.6.5 Chapter 6

Due to the need to improve TF-ATO, this chapter proposes (1+1)-EGS to improve global term weight and TF-ATO as a local term weight. The term weight can be divided into two parts, the Global Term Weight (GTW) and the Local Term Weight (LTW). The GTW is a value assigned to each index term to indicate the topics of the documents. It has the discrimination value of the term to discriminate between documents in the same collection. The LTW is a value used to measure the contribution of the index term in the document. This approach considers the limitation of previous studies that used EC approaches for TVM. However, this approach consumed more computational time than probabilistic learning models and language models but it is less time consuming and less memory demanding compared to EC approaches in the previous studies. Furthermore, this chapter considers the limitations of relevance judgement when using partially judged test collections.

1.6.6 Chapter 7

This chapter presents an efficient EC application to produce a ranking model in LTR research field based on data feature weights. This ranking model represents the importance value of each feature in search engine query-document pairs. This approach is faster than other machine learning approaches. The test benchmarks are LTR benchmarks containing more than one TWS and other features that indicate the business importance and reputation of the documents on the web. ES-Rank ((1+1)-ES for Learning to Rank) application is a supervised learning approach in the LTR problem.

1.6.7 Chapter 8

This chapter shows the capability of ES in LTR domain as a tool to improve user Click Models to achieve improved accuracy values. The ES-Rank can improve the performance

of the *Dependent Click ranking Model (DCM)* in five evaluation fitness metrics. This approach is called ES-Click. It begins with a learning DCM ranking model and then the ES-Rank uses the DCM ranking models to evolve better ranking models. In most cases of IR system evaluation performance, click models replace the interactive user clicks without bias caused by neither user education level, intelligence nor familiarity with the IR system.

1.6.8 Chapter 9

The final chapter presents the conclusion from the research findings and the outcomes of this thesis. It started with answering the research questions presented in this chapter and then it reviews the research contributions. Finally, it presents a summary of conclusions of the thesis and presents the future work avenues.

Chapter 2

Background

This chapter presents a discussion of background knowledge about Information Retrieval (IR) research field. This discussion begins with presenting the general IR architecture in sections 2.1 and 2.2. Then, the evolution Strategies background is presented in section 2.3.

2.1 General Information Retrieval Overview

An Information Retrieval (IR) system is an information system that stores, organises and indexes the data for retrieval of relevant information responding to a user's query (*user information need*) (Salton and McGill, 1986). Generally, an IR system consists of the following three main components (Baeza-Yates and Ribeiro-Neto, 2011):

- **Document Set**. It stores the documents and their *Information Content Representations*. It is related to the indexer module, which generates a representation for each document by extracting the document features (terms). A *term* is a keyword or a set of keywords in the document.

- **Similarity Matching and Ranking Mechanism**. It evaluates the degree of similarity to which each document in the test collection satisfies the user information need. It also ranks the retrieved documents, according to their relevance to the user query.

- **User Query Log**. It is a user's query or set of queries where users can state their

information needs that stored in online IR systems. This component also transforms the users query into its *information content* by extracting the query's features (terms) that correspond to document features.

2.1.1 Information Retrieval Architecture

The implementation of an IR system may be divided into a set of main processes as shown in Figure 2.1. Some of these processes (dotted lines rectangles) can be implemented using *Machine Learning (ML) or Computational Intelligence (CI)* approaches. An outline of the core processes in IR (solid lines rectangles) is also presented in this figure.

The *User Interface* module manages the interaction between the user and the IR system. In this module, the user can request information after *Pre-processing* and *Query Transformation* from the index file. The result of this query is in the form of links or document numbers referring to documents in the test collection or World Wide Web (WWW). The *Pre-processing* module represents the lexical analysis, stop-words removal and stemming procedures that are applied to the user query and the test collection. The *Indexing* module processes the documents in the collection using a term-weighting scheme (TWS) in order to create the index file. Such index file contains information in the form of inverted indexes where each term has references to each document in the collection where that term appears and a weight representing the importance of that term in the document. Similarly to indexing, the user query undergoes a process of *Query Transformation* after pre-processing for building queries of terms and their corresponding weights for those terms. The *Searching* module conducts a similarity matching between the query of terms with their weights and the index file in order to produce a list of links or document numbers referring to documents in the test collection. In the past, the *Ranking* of the matching list depended only on the degree of similarity between the documents and the user query. Nowadays, this ranking may depend on some additional criteria such as the host server criteria among others (Liu, 2009).

After outlining the core processes in the implementation of an IR system, we now focus on the aspect where machine learning and meta-heuristic techniques applied to

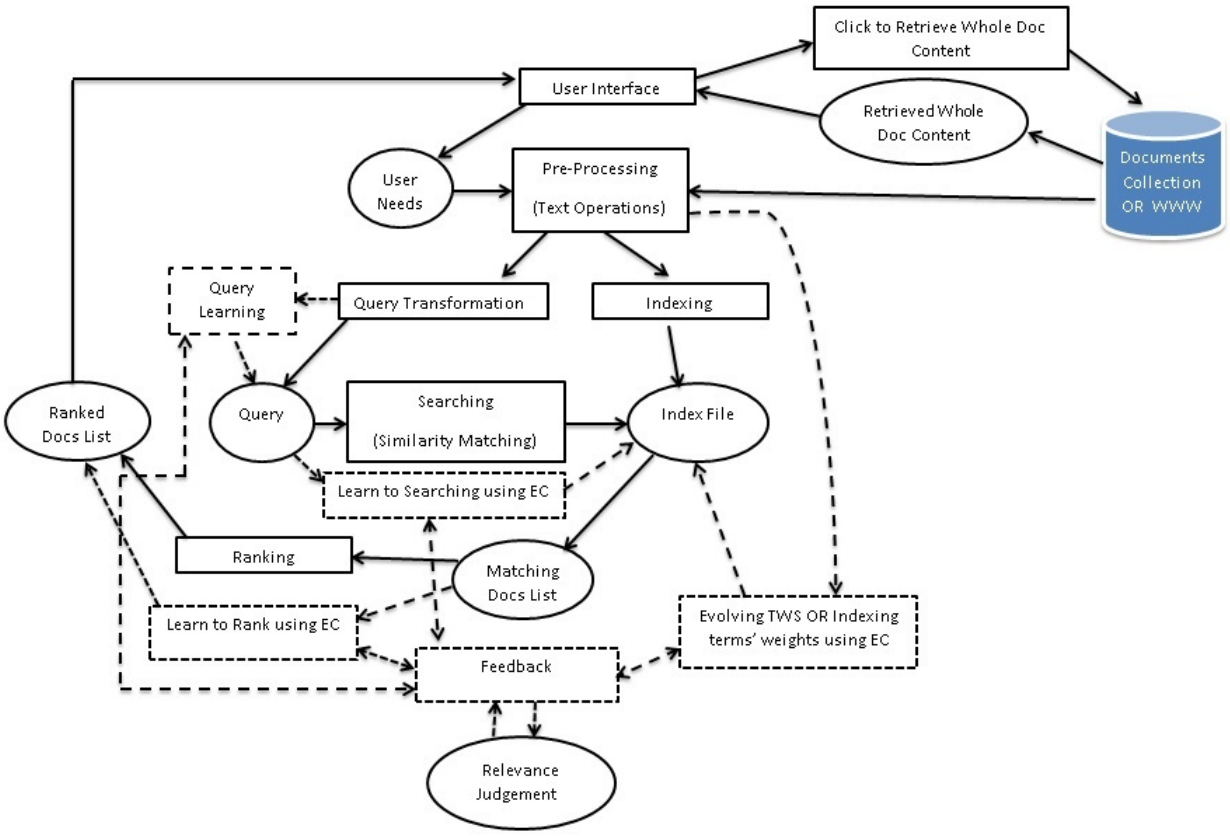

Figure 2.1: Main Processes Involved in the Implementation of an IR System Architecture.

some IR processes exhibit some weakness in our opinion. The *Relevance Judgement* file is the file that contains the set of queries for the test collection and their corresponding relevant documents from the collection. Also, this file sometimes contains the degree of relevancy of documents for the queries (i.e. some number indicating that the document is non-relevant, partially or totally relevant). However, all IR test test collections are partially judged as it is not feasible to have fully judged test collections as mentioned in (Qin et al., 2010). Since machine learning, meta-heuristic, probabilistic and language model techniques applied to IR depend on the relevance judgement file, the efficiency of such techniques for IR is limited. The following sections explain the details of the core components of the IR system.

2.1.2 Natural Language Pre-Processing in Information Retrieval

Natural language pre-processing (Baeza-Yates and Ribeiro-Neto, 2011) for classical IR models is the procedure of text transformation that may utilise the following steps:

1. Each document is tokenised into data elements such as words or group of words or n-letters (n-grams). Moreover, the digits, punctuation marks and the case of letters of these data elements are treated to produce lower-case letters of tokens.

2. Stop-words (negative words) are removed. These stop-words have poor discrimination values that would make poor index terms for similarity matching and retrieval purposes (Fox, 1989). Examples for the stop-words (common words) are the and a. These keywords are common in every English document and they do not have discrimination capability between the documents each other.

3. If the data elements are words it can be stemmed by removing prefixes and suffixes by using any of stemming algorithm such as Porter Stemmer (Porter, 1997), Lovins Stemmer (Lovins, 1968) among others.

4. Then the unique data elements (keywords or a group of keywords) are chosen to be index terms for the IR system.

5. Some IR systems use thesaurus terms and term categorisation for alternative terms and meanings of keywords or data elements in index terms that may be helpful in similarity matching between users' queries and index terms on IR systems (Baeza-Yates and Ribeiro-Neto, 2011).

This process is one of the important steps that affects the IR system efficiency. Research work has concluded that inappropriate pre-precessing can affect the text classification accuracy (Uysal and Gunal, 2014; Makrehchi and Kamel, 2008; Toman et al., 2006; Toraman et al., 2011). It has also been demonstrated that pre-processing is topic domain dependent in text classification problem (Uysal and Gunal, 2014).

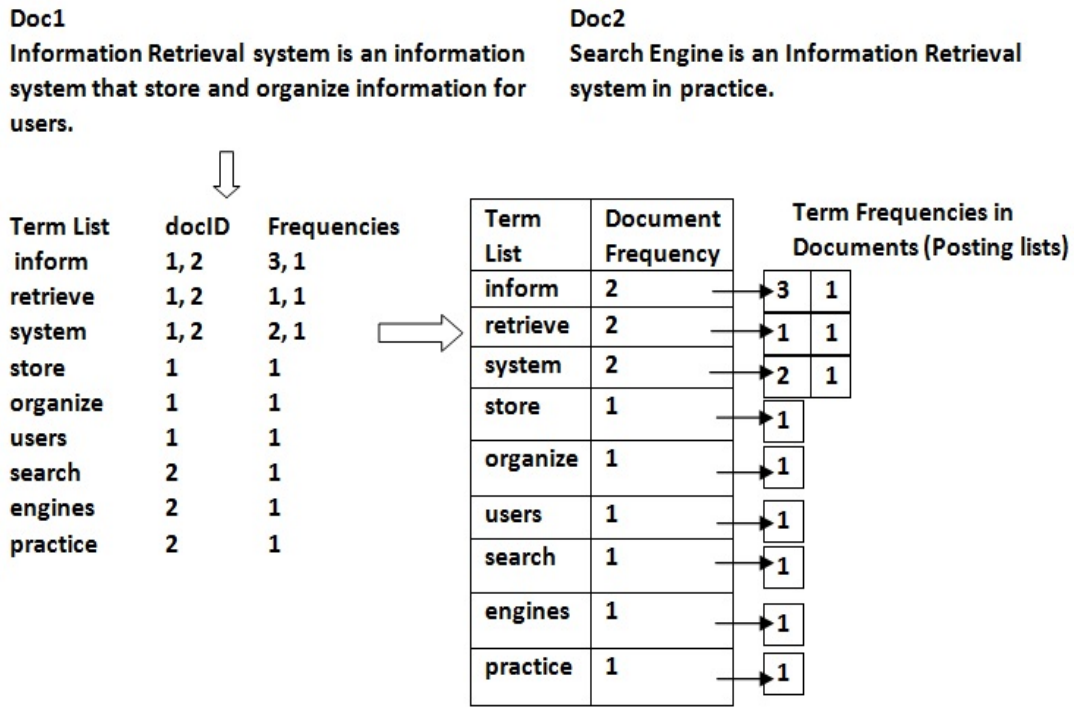

Figure 2.2: Indexing Procedure from Assigning Weights (Term Frequency) to Producing Inverted Index Containing Term List and Postings Lists.

2.1.3 Indexing and Query Transformation

Generally, the index (i.e. an inverted index) for the test collection is a structure that stores, each term (keyword) that occurred somewhere in the collection combined with its weight (Yan et al., 2009). The weight value represents the information content or the importance of the term in the document. As shown in Figure 2.2, the two main components of an inverted index are the term list and the postings lists. For each term in the test collection, there is a postings list that contains information about the term's occurrences or the term's weights in the collection (Buettcher et al., 2010). The information found in these postings lists is used by the system to process search queries. The term list serves as a look-up data structure on top of the postings lists. For every query term in an incoming search query, the information retrieval system first needs to locate the term's postings list before it can start processing the query. The term list job is to map between terms and their corresponding posting list locations in the index. In Figure 2.2, the Document Frequency (DF) is the simplest kind of *Global Term Weight (GTW)* that represents the number of documents in which the term occurred. The GTW contribute to identify the

topic of the documents. DF is the number of documents in which a term is existed, while the Term Frequency (TF) is the number of repetitions of a term in a document. From Figure 2.2, after the test collection undergoes the pre-processing procedure, the TF and DF for each term are calculated. Then, the inverted index is created from term list and posting list. The term list consists of the index terms and their document frequencies, while the posting list consists of the term frequencies in each document for each term. A more detailed discussion about the local and the global term weights will be presented in chapters 4 and 6.

2.1.4 Matching and Ranking for Information Retrieval

Matching and ranking in IR is the process of searching the IR data resource (IR test collection) for the matched documents responding to the user query and then ranking them according to their degree of similarity match and/or their importance. The main categories of matching and ranking methods are as follows:

1. Exact matching which is the matching between the query and the documents according to the terms (keywords) existing in both or not. If the terms exist in both the query and the document, then the document is more similar to the query hence matching the query requirement. This category of matching can be done in the standard Boolean Model. Thus, the Boolean exact matching uses AND and OR Algebra between the binary term occurrence value (1 if the term exists, 0 if the term does not exist). The problem with Boolean exact matching is that there is not ranking or distinguishing between the matched documents that are returned as relevant to the query, i.e. all are considered 'equally' relevant which would not be always precise enough (Greengrass, 2000).

2. Similarity matching, this produces a real numerical value that represents the degree of similarity between the document and the query. This category can be used in *Vector Space Model (VSM)*. Examples of the similarity matching functions used in this category are: Cosine Similarity, Jaccard Similarity Coefficient, Inner Product

and Dice's Coefficient (Baeza-Yates and Ribeiro-Neto, 2011).

3. PageRank matching (Page et al., 1998), this measure calculates the importance of the web-document on the web. This approach is based on assigning a value for the document based on its relation with other web-pages on the web.

4. Probabilistic relevance matching, this method is based on the probability distributions of the data. This category considers the probability distribution of the terms existing in the test collection with respect to the collection relevance judgement. Example of this category are: *Best Match version 25 (BM25)*, a probabilistic model (see Section 2.2.3) and *Dirichlet Similarity Matching (DSM)* on Dirichlet language model (Zhai and Lafferty, 2004).

5. Learning to rank based on Term Vector Model (TVM) approach is one of the similarity matching methods. An example of this category is the Genetic Programming approach that is used for evolving similarity matching functions (Fan et al., 2004). This method is discussed in more details in Chapter 4.

6. Learning to Rank based on Feature Vector Model (FVM) approach is one of the state-of-the-art research directions in IR (Bollegala et al., 2011). The ranking function in this methodology is called ranking model. This model will be discussed in details in the following chapters.

Section 2.2 demonstrate the well-known retrieval models existing in the literature which based on the matching and ranking procedure.

2.1.5 Relevance Judgement in Information Retrieval

The relevance judgement component of the IR test collection contains the list of relevant and irrelevant documents. The relevance judgement is identified by monitoring the user behaviours (interaction) and their corresponding user information needs (queries) on the IR systems. There are three paradigms for producing the relevance judgements for test collections.

The first category is the Cranfield paradigm (Cleverdon, 1960; Sanderson, 2010). In this category, the test collections were produced by collecting the abstracts of scientific articles as the document set. The query set was produced by the authors of these scientific articles. Those authors are asked to assign all relevance values to their queries with their articles. These values indicate which document is relevant or irrelevant for each query in the query set. Then, experts in the articles' topic assess and validate the authors relevance judgement values. This paradigm is the most accurate method for producing the relevance judgements as claimed by (Cleverdon, 1960; Sanderson, 2010). However, this method is very expensive and time consuming for producing relevance judgements for large test collections.

The second paradigm is produced by TREC (Text REtrieval Conference) and the National Institute of Standards and Technology (NIST) for large test collections. This paradigm is called Pooling paradigm (Voorhees, 2000; Soboroff, 2007). The pooling paradigm starts with crawling the web or specific web domains by queries for collecting the document set for these queries. Then, multiple term-weighting schemes were used to retrieve the matched documents for human assessors. Then, the human assessors examine the relevance degree of the top k (for example top 100 or 1000) documents retrieved from the document pool retrieved. After-that, TREC conference used the test collection for TREC conference tracks. Finally, TREC validates the conference outcomes with multiple IR systems for the test collections using multiple TWS. The limitation of this method is that the pool of the documents retrieved and the pool depth are biased to the retrieval strategy used (Sanderson, 2010; Buckley et al., 2007). Recent research identified that TREC Disk 4 and 5 with Robust 2004 and Crowdsource relevance judgement is the most reliable pooled collection to compare between the performances of the IR systems (Tonon et al., 2015; Urbano, 2016).

On the other hand, the learning IR models such as language models, computational intelligence models and probabilistic models which do not consider the limitation of the relevance judgement on their supervised and semi-supervised learning approaches as it will be discussed and investigated later in this thesis (chapters 3 and 5). The reason is that

the traditional test collections are partially judged for their query set. The pool size and the number of queries in the query set cause this limitation. In addition, the number of index terms in the relevance judgement (relevant/irrelevant documents and queries) do not cover the whole term space of the collection. This inspired the need to the third paradigm which is fully judged distilled collection benchmarks for supervised and semi-supervised machine learning approaches (Qin et al., 2010). These benchmarks contain features for various IR techniques and relevance labels. These labels show the relevance degree of the document with the query-document pair as a replacement for relevance judgement values.

2.1.6 Information Retrieval System Evaluation

The final step after implementing an IR system is the system evaluation process. There are several measurements to evaluate IR systems and all of these measurements depend on the objectives of building the IR system. One of these objectives is the system performance and the measurements of this objective are related to the computational runtime and the problem size. The accuracy is the second evaluation metric that has been widely used. The evaluation metrics related to accuracy depend on the relevance degree of the retrieved documents responding to the user queries (user information needs). This type of evaluation is referred to retrieval accuracy evaluation or in other words, system effectiveness (Baeza-Yates and Ribeiro-Neto, 2011).

The most common metrics for measuring IR system effectiveness are *Recall* and *Precision*. Figure 2.3 illustrates the use of recall and precision as a measurement for IR effectiveness. There are two types of classical Recall-Precision approaches to measure the IR system effectiveness: Singular Value Recall-Precision and Interpolated Recall-Precision (Baeza-Yates and Ribeiro-Neto, 1999; Kwok, 1997; Chang and Hsu., 1999). Singular Value Recall-Precision is the way of computing both the recall and corresponding precision for a given user query from the IR system. Interpolated Recall-Precision is the way of retrieving the documents responding to the user query until reaching a specific recall value assigned to the IR system, and then measuring the corresponding precision ratio. In the web-scale IR, there are another metrics such as the precision of the retrieved list at the first k documents retrieved (P@k) (Li, 2014). In the experimental study of

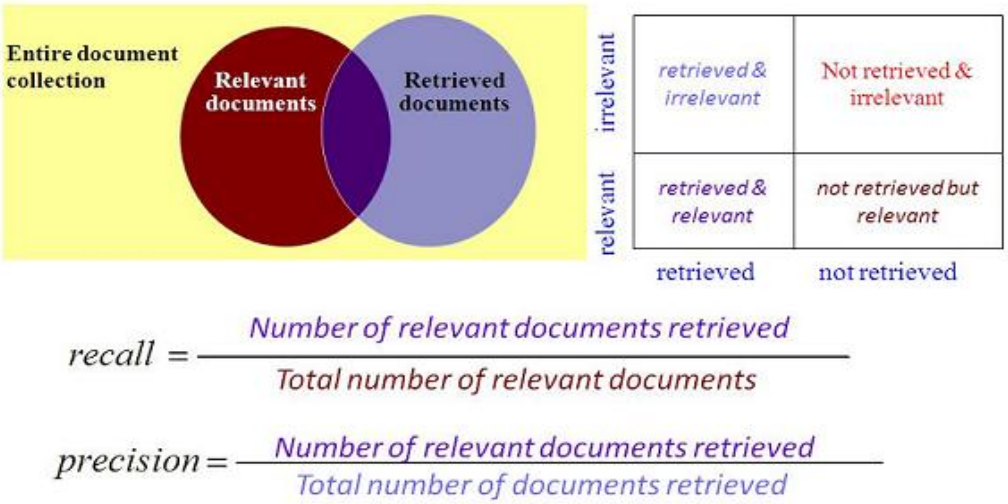

Figure 2.3: Recall and Precision Ratios in IR Systems.

the thesis, *Mean Average Precision (MAP)*, *Average Precision (AP)*, *Precision at top-10 document retrieved (P@10)*, *Normalised Discounted Cumulative Gain (NDCG@10)*, *Reciprocal Rank (RR@10)*, *Error Rate (Err@10)* and *Root Mean Square Error (RMSE)* were used (Li, 2014; Baeza-Yates and Ribeiro-Neto, 2011; Chai and Draxler, 2014). We now turn to the explanation of these evaluation measures.

Let $d_1, d_2, ..., d_k$ denote the sorted documents by decreasing order of their similarity measure function value, where k represents the number of retrieved documents. The function $r(d_i)$ gives the relevance value of a document d_i. It returns 1 if d_i is relevant, and 0 otherwise. The Precision of top-k relevant query-document retrieved per query q $(P@k)$ is defined as follows:

$$P@k \;=\; \frac{\sum_{i=1}^{k} r(d_i)}{k} \tag{2.1.1}$$

On the other hand, the Interpolated Average Precision at specific recall point $r = \bar{r}$ can be calculated as follows:

$$AvgP \;=\; max_{r=\bar{r}} P(\bar{r}) \tag{2.1.2}$$

where $P(\bar{r})$ is the precision at recall point $r = \bar{r}$ over all queries Q. The $AvgP$ value is calculated for a point recall value. In this thesis, we calculated the $AvgP$ for nine-point

recall values as threshold for top k document retrieved. The interpolated mean of the average precision values for M-point recall values (MAP) can be given by the following equation:

$$MAP = \frac{\sum_{L=1}^{M} AvgP}{M} \tag{2.1.3}$$

For considering the graded relevance levels in the datasets for LTR techniques evaluation $r(d_j)$ returns graded relevance value (not binary relevance value as in MAP and $P_q@k$ equations) in equations 2.1.4, 2.1.5 and 2.1.6 for other fitness evaluation metrics. The Normalized Discounted Cumulative Gain of top-k documents retrieved (NDCG@k) in equation 2.1.4 can be calculated by:

$$NDCG@k = \frac{1}{IDCG@k} \cdot \sum_{i=1}^{k} \frac{2^{r(d_i)} - 1}{log_2(i+1)} \tag{2.1.4}$$

where $IDCG@k$ is the ideal (maximum) discounted cumulative gain of top-k documents retrieved. The *Discounted Cumulative Gain of top-k documents retrieved (DCG@k)* can be calculated by the following equation:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{r(d_i)} - 1}{log_2(i+1)} \tag{2.1.5}$$

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

The Reciprocal Rank metric at top-K retrieved query-document pairs (RR@K) is as follows:

$$RR@K = \sum_{i=1}^{k} \frac{1}{i} \prod_{j=1}^{i} (1 - r(d_j)) * r(d_j) \tag{2.1.6}$$

The Error Rate (Err) is usually used to measure the error of the learning model if it is used on another benchmark different from the training set. It is the subtraction between the training evaluation value to the predictive evaluation value, while the Mean Absolute Error and Root Mean Square Error are calculated by equations 2.1.7 and 2.1.8.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|ERR_i| \tag{2.1.7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(ERR_i)^2} \tag{2.1.8}$$

where n is the number of benchmark instances (documents) used for evaluating the IR system effectiveness. Each evaluation metric has a purpose for measuring the quality of the proposed ranked model and the retrieved search results by this model. P@K is used to measure how many relevant documents in the top-K documents. However, this metric does not consider the graded relevance levels of each retrieved document, but it considers if the query-document retrieved if the relevant or not. The MAP evaluation metric considers the average precision on the whole search results rather than top-K query-document pair retrieved. On the other hand, NDCG@K and RR@K metric take in their calculations the graded relevance level of each query-document pair into consideration for the top-K query-document retrieved. The difference between NDCG@K and RR@K is that RR@K considers the impact of the position for each retrieved query-document pair in the search list more than NDCG@K metric. Finally, MAE and RMSE calculate the difference between the relevance labels produced by the ranking model with the query-document pair features against the ground truth relevance labels. The MAE and RMSE consider the ranking problem as ranking and regression problem. In this thesis, we used MAP, NDCG@10, P@10, RR@10 and RMSE as fitness evaluation metrics for extensive evaluation and optimisation to produce by the proposed techniques.

2.2 Information Retrieval Models

An IR model refers to the way in which the IR system organises, indexes and retrieves information when responding to user queries. The IR model also specifies the method used for user query representation. There are five prominent IR model categories: the first is referred to as Boolean model, the second is known as Vector Space Model (VSM), the third is called Probabilistic model, the fourth are Language Models (LM) and the fifth are EML Models (including Evolutionary Computation and Machine Learning Models)

(Baeza-Yates and Ribeiro-Neto, 2011; Li, 2014; Ibrahim and Landa-Silva, 2017). All these model categories use the same procedure for document/query pre-processing and they differ in one or more of the other processes mentioned in section 2.1. They also differ in the ways of assigning term weights in the indexing process, similarity matching and retrieving the similar document.

2.2.1 Boolean Model

This was the first Term Vector Model (TVM) representation for IR systems suggested by researchers (Harter, 1986; Greengrass, 2000). This model is called Boolean model. Each document/query in this model is represented by a vector of term weights. The model is based on Boolean Algebra for user query representation and for finding an exact matching between documents and user queries. The user query may contain logical (AND, OR, NOT, etc.) operators to combine terms. These terms can be represented by 0 or 1 indicating whether the term appears on each query or not. This model depends on searching by exact match between documents in IR system data resource and user query.

(Salton, 1988) mentioned some of the conventional disadvantages in this Boolean model which are summarised as follows:

1. There is no ranking of the retrieved documents according to their relevance to the user query.
2. There is no obvious way of controlling the size of the retrieved output according to some threshold of relevance.
3. There is no weighting of search terms that indicates the information content for every term according to its importance.

Finally, the retrieved search results for the user query is made by an exact Boolean matching between the user query and documents in the IR test collection. This exact matching procedure produces some problems for retrieving the accurate search results.

These problems can be illustrated in the following two types of cases:

1. In response to an OR-query such as (A or B or...or Z), a document containing all the query terms is not treated any better than a document containing only one term.

2. In response to an AND-query such as (A and B and ... and Z), a document containing all but one query term is treated just as badly as a document containing no query term at all.

Salton (Salton, 1988) tried to overcome the disadvantages of the standard Boolean model by assigning weights to index terms in documents and by trying to reformulate the user query. Salton et al. proposed to extend the Boolean retrieval model to overcome some of these disadvantages (Salton et al., 1983). They proposed adding weights to each index term and query term instead of the binary representation. These weights represent the term's information content or the term importance in a given document or a query. As a result, the retrieved documents can be ranked in similarity order by the similarity matching measure. Salton et al. used a normalised Euclidean distance between the documents and the queries. They suggested using the p-norm model to evaluate the degree of documents matching (satisfying) a query. This process is more in accordance with the way a human judges the matching compared to the traditional Boolean model. The extended Boolean functions for similarity matching of the p-norm similarity measures are given by (Greengrass, 2000):

$$SIM_{AND}(d,(t_1,w_{q1})AND...AND(t_n,w_{qn})) = 1-(\frac{\sum_{i=1}^{n}((1-w_{di})^p \cdot w_{qi}^p)}{\sum_{i=1}^{n} w_{qi}^p})^{\frac{1}{p}}, (1 <= p <= \infty) \quad (2.2.9)$$

and

$$SIM_{OR}(d,(t_1,w_{q1})OR...OR(t_n,w_{qn})) = 1 - (\frac{\sum_{i=1}^{n}(w_{di}^{p}\cdot w_{qi}^{p})}{\sum_{i=1}^{n} w_{qi}^{p}})^{\frac{1}{p}},\ (1 <= p <= \infty) \tag{2.2.10}$$

where $t_1, t_2, , t_n$ are the query terms and their corresponding weights $w_{q1}, w_{q2}, ..., w_{qn}$ for a given user query and d is a document in document space with its corresponding weights $w_{d1}, w_{d2}, ..., w_{dn}$ for the same n number of terms. The p-norm model also defines similarity functions for the extended Boolean AND and extended Boolean OR of the n terms and each similarity is computed as a number in the closed interval [0, 1]. When $p = \infty$ the model becomes as a standard Boolean model and if $p = 1$ this will reduce the p-norm to the vector space model with its cosine similarity function (discussed in Section 2.2.2).

There are some other approaches to extend the Boolean model, like (Waller and Kraft, 1979) and (Paice, 1984) but with no improvement compared to p-norm. There is also some research on extending the standard Boolean model using Fuzzy theory such as (Bordogna et al., 1995; Kraft and Buell, 1983; Bookstein, 1980; Bordogna and Pasi, 1993). The state-of-the-art research work related to the Boolean model is reported in (Pohl et al., 2012, 2010; Smith, 1990; Lee, 1994).

2.2.2 Vector Space Model (VSM)

Vector Space Model (VSM) is a type of TVM representation. In VSM, the document and the query are represented as vectors in the document space (Baeza-Yates and Ribeiro-Neto, 2011; Salton et al., 1975). Each dimension in the document space represents the weights given to the term in the test collection. In other words, the documents in the IR collection contain text words. After the pre-processing procedure, we obtain index terms or keywords representing each document. Then, we assign weights for each index term in the test collection. This weight represents the importance or the information content of

that index term in a given document. Each document is stored in the following form:

$$d = (w_1, w_2, ..., w_n)$$

where d is a document in the test collection, w_i is the weight value of term i in the index terms collection and n is the number of index terms in the index terms collection that represent the information content of the test collection. The term weight can be assigned statistically or manually by trained indexer with expertise in the content of the test collections. When users type their queries as textual data, the IR system automatically assigns weights for each search keyword to build the query vector.

Term-Weighting Schemes in VSM

A good index term is a term that has a high discrimination value or weight that decreases the similarity between documents when assigned to the collection (Salton et al., 1975). The simple term weighting scheme uses the number of term occurrences in a given document which is called term frequency (tf). However, there is a drawback in this scheme. It may be that the term gets high weight value in every document at the same time because the term is repeated in every document and this makes it not a good discriminator term for documents. (Jones, 2004) proposed another weighting scheme called Inverse Document Frequency (idf) represented by $\log(N/n)$ where N is the total number of documents in the collection and n is a number of documents to which a term is assigned.

(Salton and Buckley, 1988) proposed several weighting schemes for automatic text retrieval; these are shown in Figure 2.4. Salton and Buckley classified a term weighting scheme according to three main components: term frequency, collection frequency and normalisation components. One of these combinations is Term Frequency-Inverse Document Frequency (TF-IDF). The TF-IDF weighting scheme is now the most well-known term-weighting scheme in VSM that has been widely used in the literature such as in (Liu, 2011; Reed et al., 2006*a*; Greengrass, 2000).

Term Frequency Component		
b	1.0	Binary weight equal to 1 for terms present in a vector (term frequency is ignored).
t	tf	Raw term frequency (number of times a term occurs in a document or query text).
n	$0.5 + 0.5 \frac{tf}{maxtf}$	Augmented normalized term frequency (tf factor normalized by maximum tf in the vector, and further normalized to be between 0.5 and 1.0).
Collection Frequency Component		
x	1.0	No change in weight; use original term frequency component (b, t, or n).
f	$\log \frac{N}{n}$	Multiply original tf factor by an inverse collection frequency factor (N is total number of documents in collection, and n is number of documents to which a term is assigned).
p	$\log \frac{N-n}{n}$	Multiply tf factor by a probabilistic inverse collection frequency factor
Normalization Component		
x	1.0	No change; use factors derived from term frequency and collection frequency only (no normalization).
c	$1 \Big/ \sqrt{\sum_{vector} w_j^2}$	Use cosine normalization where each term weight w is divided by a factor representing Euclidian vector length

Figure 2.4: Term Weighting Components By (Salton and Buckley, 1988).

(Lee, 1995) discussed each term weighting component proposed by Salton and Buckley and experimentally investigated whether cosine normalisation played an important role in retrieving different sets of documents or not. Lee concluded that cosine normalisation is a more important factor than maximum normalisation in retrieving different set of documents. Additionally, Lee studied the properties of different weighting schemes and showed that the significant improvements are obtained by combining the results retrieved from different properties of weighting schemes.Prior studies presented by (Fox and Shaw, 1994) used multiple document representations and multiple query representations for improving the retrieval effectiveness. (Belkin et al., 1993) achieved improvement in the effectiveness by using multiple query representations using different Boolean query formulations. (Harman, 1993) suggested that using multiple retrieval runs and combining them can be used for improving the retrieval effectiveness.

Lee also classified weighting schemes into three classes according to the term-weighting component used. These classes are as follows:

1. Class C: weighting schemes of that class perform cosine normalisation. The advantage of that class is in retrieving single topic documents being relevant in the collections with varying document length. However, its disadvantage is the difficulty in retrieving relevant multiple topic documents and longer documents.

2. Class M: weighting schemes of that class perform maximum normalisation but do not perform cosine normalisation. The advantage of this class is that it may alleviate the problem of cosine normalisation in retrieving relevant documents that have multiple topics but it cannot normalise documents length. Moreover, it will retrieve longer documents regardless of their relevance.

3. Class N: this class of weighting schemes does not perform either cosine normalisation or maximum normalisation. This class favours longer documents to be retrieved instead of short documents and this may have an effect on retrieving relevant documents corresponding to users' queries.

(Reed et al., 2006*a*) studied the relationship between the number of terms in the test collection and the document frequency distribution, where the document frequency is the number of documents in which the term occurs in the test collection. They concluded that there is a major effect on document frequencies by adding new documents to small test collections. Whereas, there is a minor effect on document frequencies when adding new documents to large test collections. Their studies led them to name the idf in large test collection as inverse corpus frequency ICF and they conducted experiments by assigning term weight scheme as follows:

$$w_{ij} \;=\; \log(1+tf_{ij}) \;\cdot\; \log(\frac{N+1}{n_j+1}) \tag{2.2.11}$$

where tf_{ij} is the term frequency of term j in document i and $ICF = \log(\frac{(N+1)}{(nj+1)})$ since N is the number of documents in the corpus and n_j is the number of documents in which term j appears in the corpus. Reed et al. (Reed et al., 2006*a*) conducted their studies on three test collections: Reuters-21578 (Lewis, 1997), SMART (Salton, 2013) and 20

Newsgroup (Rennie, 2015). In addition, in their work they used two similarity functions, Euclidean distance and Cosine similarity.

Similarity Matching Between Document and Query in VSM

Once the document vectors and query vector of term-weight have been computed using a TWS, the following step is to calculate the similarity matching value between the query vector and the document vectors in the test collection. Then, the documents are retrieved in descending order of their similarity values. The highest ranking document will be the most similar document to the query. The similarity matching procedure simulate the automatic system measurement for the relevance levels of the documents to the query. The more accurate similarity matching function is the more of effective IR accuracy obtained. There are five well-known similarity matching functions that are widely used in VSM. These functions are inner product, cosine similarity, Dice, Jaccard and Euclidean distance between document/query vectors (Greengrass, 2000; Baeza-Yates and Ribeiro-Neto, 2011). Several research have been reported about these functions in (Greengrass, 2000). Chapters 5 and 6 uses the most widely used and the most efficient similarity function in VSM which is the cosine similarity. The cosine similarity function between document d and query q ($Cosine\_Similarity(d, q)$) is defined by:

$$Cosine\_Similarity(d, q) = \frac{\Sigma_{i=1}^{n} W_{id} \cdot W_{iq}}{\sqrt{\Sigma_{i=1}^{n} W_{id}^2 \cdot \Sigma_{i=1}^{n} W_{iq}^2}} \tag{2.2.12}$$

In the above equation, n is the number of index terms that exist in the document d and query q, W_{id} is the weight of term i in document d and W_{iq} is the weight of the same term i in query q.

2.2.3 Probabilistic Models (PM)

Probabilistic models in information retrieval are learning approaches based on the TVM representation in which each query/document is a vector of term weights. Probabilistic models are based on the theory of probability with some statistical basis. Cooper et. al. (Cooper et al., 1992) argued that the probabilistic approach is better than other

mathematical models such as vector space model approach in getting more relevant results responding to the same query. However, there are some advantages and disadvantages of probabilistic models comparing to other statistical approaches. Greengrass (Greengrass, 2000) discussed these advantages and disadvantages in a survey. According to Greengrass and Cooper et. al (Greengrass, 2000; Cooper et al., 1992) the advantages are as follows:

1. The result of a probabilistic IR approach is near to optimal result regarding its retrieval effectiveness.

2. It should be less reliant on traditional trial-and-error retrieval experiments as it depends on a theoretical approach for building each model.

3. It has more powerful statistical indicators for predictive and goodness of fit than traditional effectiveness measures such as recall and precision.

From another point of view, Cooper (Cooper, 1994) identifies various disadvantages of using probabilistic models:

1. Some researchers prefer the mathematical models because they do not need statistical assumptions eliminating the theoretical burden. They save the time and effort spent on the statistical analysis on ad hoc experimentation instead of on probability theory.

2. Probabilistic IR models usually involve some statistical analysis to simplify assumptions in their estimation procedures.

3. Some underlying IR models widely used such as *Binary Independence* of IR model can lead to logical inconsistency (Fernandez-Reyes et al., 2015; Cooper, 1991, 1995) since the actual assumptions used in practice are different from the assumptions of the theoretical model. *Binary or Linked Independence* means that each index term is independent of its value in appearing within document than other terms on the same document.

Furthermore, Greengrass (Greengrass, 2000) illustrated some of the well-known probabilistic IR models such as Okapi. The next section presents the Okapi probabilistic model.

Two-Poisson Probabilistic Model (Okapi-Best Match Model)

Robertson et al. (Robertson and Walker, 1994) proposed a term weighting function based on Two-Poisson distribution Models. This term weighting function was used in the Okapi IR system in City London University. It is the most successful term weighting function in TREC (Text Retrieval Conference) competitions. It has been widely used by the IR research community (Cummins, 2008; Robertson, Walker, Beaulieu, Gatford and Payne, 1995; Robertson et al., 1994). In TREC, the Okapi function has been used as a weighting scheme for document representation in TREC collection for query optimisation problems (Robertson, Walker, Beaulieu, Gatford and Payne, 1995; Robertson et al., 1994). Furthermore, it has been widely used in other TREC tracks such as in Robertson et al. (Robertson et al., 1998). Robertson et al. extended the Linked Independence assumption in their probabilistic model for *Eliteness*, where the *Elite* document for index terms is the document that is mainly about these index terms. In other words, if an index term or a group of index terms represents the topic of the document, this document is called *Elite* document for this or these index terms. More details of the Okapi weighting function can be described using the the following notations. Let N be the number of documents in the collection and n_i is the number of documents that contain term i. Moreover, Let Rq be the total number of relevant documents to query q and r_i the number of relevant documents that contain term i. Further, dtf_i is the term frequency of term i in the document d which has length dl. The length of the document means the total number of index terms frequencies in the document. If the document has the average length of the documents existing in the document collection, it will be called $Avgdl$. Using the previous notations, the Okapi weight for term i in document d can be represented by the following equation:

$$w = \frac{dtf_i}{(K_1 \cdot dl)/(Avgdl) + dtf_i} \cdot w^{(1)}, and \tag{2.2.13}$$

$$w^{(1)} = \log \frac{(r_i + 0.5)/(Rq - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - Rq + r_i + 0.5)} \tag{2.2.14}$$

where K_1 represents a constant and usually has a value of either 1 or 2 for TREC test collections. In addition, Okapi weight for term i in the query q has a different equation represented as follows:

$$w = \frac{qtf_i}{K_3 + qtf_i} \tag{2.2.15}$$

where qtf is the term frequency of term i in query q and K_3 is unknown constant that can be determined by a trial and error procedure. Furthermore, the phenomena of the strong variation of the documents in their length than $Avgdl$ with more words is called *Verbosity Hypothesis*. This Phenomena affects the performance of Okapi weighting function which require a correction factor for the term weights. This correction factor is given by the following equation:

$$correctionfactor = K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.16}$$

where K_2 is unknown constant and it should be determined by trial and error (its value in TREC collections is between 0 and 2). Further, ql, dl and $Avgdl$ are the query length, document length and average document length. Robertson et. al used a new similarity matching function in their research (Robertson et al., 1998). This similarity matching function is called *Best Match (BM)*. In their experimental study, three formula for *Best Match* have been used. They are called BM1, BM11 and BM15. These matching functions represent the *Dot Product* similarity function (Greengrass, 2000; Baeza-Yates and Ribeiro-Neto, 2011) between the document d and query q for a vector of weights (BM15 which is the *Dot product* plus correction factor). In general, the *Dot Product* function used by various similarity matching and ranking functions forms such as *Cosine Similarity*, *Inner Product*, *Euclidean Distance*, *Language Model (LM)* matching functions (Greengrass, 2000; Baeza-Yates and Ribeiro-Neto, 2011) and *Linear Learning to Rank* (Bollegala et al., 2011) functions. The equations of the BM similarity matching functions are given by:

$$BM1(Q,D) = \sum_{term\ i\ \in q} \log \frac{(r_i + 0.5)/(Rq - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - Rq + r_i + 0.5)} \cdot \frac{qtf_i}{K_3 + qtf_i} \tag{2.2.17}$$

$$BM11(Q,D) = \sum_{term\ i\ \in q} \frac{dtf_i}{K_1 + dtf_i} \cdot \log \frac{(r_i + 0.5)/(Rq - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - Rq + r_i + 0.5)} \cdot \frac{qtf_i}{K_3 + qtf_i} + K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.18}$$

$$BM15(Q,D) = \sum_{term\ i\ \in q} \frac{dtf_i}{(K_1 \cdot dl)/(Avgdl) + dtf_i} \cdot \log \frac{(r_i + 0.5)/(Rq - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - Rq + r_i + 0.5)} \cdot \frac{qtf_i}{K_3 + qtf_i} + K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.19}$$

In the case of the absence of relevance judgements in the test collections, these equations become simpler by assigning zero values for r_i and Rq in the above equations. The above equations after assigning zero values for r_i and Rq will be given by:

$$BM1(Q,D) = \sum_{term\ i\ \in q} \log \frac{N - n_i - +0.5}{n_i + 0.5} \cdot \frac{qtf_i}{K_3 + qtf_i} \tag{2.2.20}$$

$$BM11(Q,D) = \sum_{term\ i\ \in q} \frac{dtf_i}{K_1 + dtf_i} \cdot \log \frac{N - n_i - +0.5}{n_i + 0.5} \cdot \frac{qtf_i}{K_3 + qtf_i} + K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.21}$$

$$BM15(Q,D) = \sum_{term\ i\ \in q} \frac{dtf_i}{(K_1 \cdot dl)/(Avgdl) + dtf_i} \cdot \log \frac{N - n_i - +0.5}{n_i + 0.5} \cdot \frac{qtf_i}{K_3 + qtf_i} + K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.22}$$

From BM11 and BM15, Robertson et. al. (Robertson and Zaragoza, 2009; Robertson et al., 2004) proposed a new similarity matching function called BM25. This function has been widely used in the literature (Cummins, 2008; Reed et al., 2006*a*). The BM25 means *Best Match* version 25 and its equation is given by:

$$BM25(Q,D) = \sum_{term\ i\ \in q} \frac{(K_1 + 1) \cdot dtf_i}{K1 \cdot ((1-b) + b \cdot \frac{dl}{Avgdl}) + dtf_i} \cdot \log \frac{(r_i + 0.5)/(Rq - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - Rq + r_i + 0.5)} \cdot \frac{qtf_i}{K_3 + qtf_i} + K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.23}$$

where K_1, K_2, K_3 and b are constants that are usually chosen by a trial and error procedure. The simplest form of BM25 by assigning zero values for Rq and r_i is:

$$BM25(Q,D) = \sum_{term\ i\ \in q} \frac{(K_1 + 1) \cdot dtf_i}{K1 \cdot ((1-b) + b \cdot \frac{dl}{Avgdl}) + dtf_i} \cdot \log \frac{N - n_i - +0.5}{n_i + 0.5} \cdot \frac{qtf_i}{K_3 + qtf_i} + K_2 \cdot ql \cdot \frac{(Avgdl - dl)}{(Avgdl + dl)} \tag{2.2.24}$$

TREC tracks such as TREC-9 (Greengrass, 2000; Robertson and Walker, 2000), have

determined the default values for K_1 and b as 1.2 and 0.7 respectively, while K_3 is often set to either 0 or 1000 and K_2 has often set to 0. Okapi-BM25 was the best term-weighting function by tuning its constants to the suitable values based on the test collection. However, the need for adjusting the suitable constants will require the prior knowledge for the relevance judgements of the queries with the test collections regardless of using the relevance judgement in the BM25 equations. If the constants $K1$, $K2$ and $K3$ have zero values in Okapi-BM25, the function will be an *Inner Product* similarity function in VSM with *Inverse Document Frequency (IDF)* term-weighting scheme.

2.2.4 Language Models (LM)

The probabilistic model described in the previous section (Section 2.2.3) is based on the term weight assignment in document indexing. Ponte and Croft (Ponte and Croft, 1998) discussed the limitations of the probabilistic models. The probabilistic model considers the *elite document* for the term in indexing, but it does not consider the distribution of the terms in user queries. Thus, Ponte and Croft (Ponte and Croft, 1998) proposed the probabilistic *Language Model (LM)*. The LM is a probabilistic retrieval and indexing model that considers the distribution of the index terms in the documents and the user queries. It assumes that a document is a language sample as in speech language models. Then, LM estimates the probabilities of the term weights in the document. After-that, the query used to estimate the rank of the retrieved documents based on the query likelihood. Ponte and Croft model is represented by the following equation:

$$\hat{p}(Q|M_d) = \prod_{t \in Q} \hat{p}(t|M_d) \cdot \prod_{t \in Q} (1.0 - \hat{p}(t|M_d)) \tag{2.2.25}$$

where $\hat{p}(Q|M_d)$ is the probability matching function and t represents index terms existing in the query Q, while $\hat{p}(t|M_d)$ is the language weighting function for index term t in document d. The $\hat{p}(t|M_d)$ function has two parts. The first part is for the probability weight for term t if the term exists in the query and the document d with term frequency $tf_{(t,d)} > 0$, while the second probability is for otherwise. The equation for the probability weight function is as follows:

$$\hat{p}(t|M_d) = \begin{cases} p_{ml}(t,d)^{(1.0-\hat{R}_{t,d})} \cdot \hat{p}_{avg}(t)^{\hat{R}_{t,d}} & if\ tf_{(t,d)} > 0 \\ \frac{cf_t}{cs} & \text{otherwise} \end{cases} \tag{2.2.26}$$

In this equation, $p_{ml}(t,d)$ represents the probability of the maximum likelihood of the term t in the document d, while $\hat{p}_{avg}(t)$ is the average probability of the maximum likelihood of term t in the test collection. Whereas, $\hat{R}_(t,d)$ is the geometric distribution of term t in the document. The details of these parameters are given in the following equations. Firstly, the maximum likelihood probability of term t in document d is defined as:

$$\hat{p}_{ml}(t|M_d) = \frac{tf_{(t,d)}}{dl_d} \tag{2.2.27}$$

where $tf_{(t,d)}$ represents the term frequency of term t in the document d which has the total number of tokens dl_d. The average of the maximum likelihood values for term t in the collection is given by:

$$\hat{p}_{avg}(t) = \frac{\sum_{d_{t\in d}} \hat{p}_{ml}(t|M_d)}{df_t} \tag{2.2.28}$$

where df_t is the document frequency of term t in the collection (i.e. df_t is the number of documents from the test collection in which term t is exist). The equations 2.2.27 and 2.2.28 did not have the discrimination of term t between various documents. This problem causes the weakness and the risk of distinguishing between retrieved documents. To eliminate this risk, the probability of matching between the documents and the query should include the occurrence of the term in the document. Ponte and Croft (Ponte and Croft, 1998) proposed a geometric distribution to eliminate this risk. Their geometry distribution is represented by the following equation:

$$\hat{R}_{t,d} = \left(\frac{1.0}{(1.0+\bar{f}_t)}\right) \cdot \left(\frac{\bar{f}_t}{(1.0+\bar{f}_t)}\right)^{tf_{t,d}} \tag{2.2.29}$$

The symbol $\bar{f}_t$ in equation 2.2.29 is the mean term frequency of term t in the documents where term t occurs. The second part of the equation 2.2.26 represents the weight value of term t in case of the absence of that term in the document. This weight is the fraction value of total number of term t occurrences in the collection cf_t divided by the

collection size cs. Ponte and Croft compared their language model with TF-IDF term-weighting function using TREC Disk 2 and 3. From the experimental results, their language model outperformed slightly the TF-IDF. The largest improvement was obtained when using TREC Disk 2 and 3 was 21.5% and the lowest improvement was -11.9%. During the last 30 years, the language models have been improved for long and short documents and queries. This is because the problem of estimating an accurate matching retrieval value based on the existence and non-existence of the index terms in the short/long documents/queries. These issues are called verbosity and scope. One of the state-of-the-art language model called SPUD (Smoothed Polya Urn Document) model discussed these problems (Cummins et al., 2015; Cummins, 2016). Cummins compared his SPUD language model to Dirichlet-prior smoothing, BM25+, BM25, Jelinek-Mercer smoothing and multi-aspect tf language models (Zhai and Lafferty, 2004; Cummins, 2016). SPUD outperformed these techniques in terms of MAP and NDCG on TREC Robust-2004, WT10g and Gov2 collections. The limitation of the language models is the choice of the appropriate parameters (constants) values in each model to produce better performance. Moreover, the verbosity and data sparseness caused by missing existing index terms in the queries from the documents are the additional limitations to produce accurate performance using the language models.

2.2.5 Evolutionary and Machine Learning Models (EML)

EML techniques have been widely used in IR (Li, 2014; Cordon et al., 2003; Cummins, 2008). These techniques are based on two different query and document representation categories in the IR systems. The first representation category is the *Term Vector Model (TVM)*. The second category is the *Feature Vector Model (FVM)*. In TVM, each document or query is a vector of term weights. Examples of TVM are VSM and PM models. In TVM, the EML techniques were used to optimise the document or the query weight representation (Cordon et al., 2003; Robertson, Walker, Beaulieu, Gatford and Payne, 1995; Robertson et al., 1994). On the other hand, the FVM category uses query-document pair vector of features. These features represent the relation between a query with a document (Li, 2014; Liu, 2009; Qin et al., 2010). Examples of these features are the TF-IDF, LM, Okapi, BM25 features. The other features that represent the reputation and the business

importance of the document on the web. Chapters 4 and 6 discuss the EML on TVM and FVM models.

2.3 Evolution Strategies Overview

Evolution Strategy (ES) is an adaptive Evolutionary Algorithm (EA) which imitates the natural evolution principles. The ES technique can be considered as a scalable alternative to Reinforcement Learning technique with which the optimisation problem converges to near optimal solutions in less runtime than other evolutionary computation techniques (Salimans et al., 2017; Beyer and Schwefel, 2002). This technique was developed in Germany in 1960 and it is widely used in various optimisation applications (Back et al., 2013; Schwefel, 1965). One of these library packages in data mining is the SHARK library package (Igel et al., 2008). In the SHARK package, the (1+1)-ES is used. To the best of our knowledge, the ES-Rank and the (1+1)-Evolutionary Gradient Strategy for optimising the ranking model and evolving global term weight presented later in this thesis are the first approaches for optimising IR Systems in FVM and TVM. The ES technique will be discussed in more details in Section 2.3.2. The following section presents the EA in general.

2.3.1 Evolutionary Algorithms (EAs)

Algorithm 1: Evolutionary Algorithm Pseudo-Code.

```
1  Initialisation individual_1, ..., individual_μ
2  while Stop criterion do
3      for j = 1 to λ do
4          Choose at random ρ parent individuals;
5          recombination → individual_j^o;
6          mutate individual_j^o;
7          evaluate individual_j^o → f(individual_j^o);
8      end
9      select μ parent individuals from {individual_j^o}_{j=1}^{ρ} → {individual_j}_{j=1}^{μ};
10 end
```

EAs belong to the class of heuristic optimisation methods. They are based on biological assumptions of evolution selection. EAs are based on three main processes

which are: recombination, mutation and selection. Algorithm 1 shows the general overview of EAs. The EA has a set of $\{individual_1, ..., individual_\mu\}$ of parent and a set $\{individual^o_1, ..., individual^o_\mu\}$ of offspring individuals or candidate solutions. These solutions undergo recombination and mutation changes and then selection process for the best offspring proposed solutions. Algorithm 1 starts with the initialisation procedure for the parent candidate solutions as in step 1. Then, steps 2 to 10 show the generation or the evolving loop to optimise the candidate solutions. Each iteration in the generation loop is called an evolving iteration and it starts with selecting ρ parent individuals (candidate solutions) to undergo the recombination and mutation to produce a new offspring candidate solution as in steps 4 to 6 in the algorithm. Then, a fitness function $f(.)$ is used to evaluate the performance of the offspring as solvers for the problem as in step 7. Following to the evaluation process, the best offspring individuals are chosen as parent individuals for the next evolving iteration. These steps are repeated until stop criterion is reached. The stop criterion can be the number of iteration or a threshold based on the fitness values. Section 2.3.2 presents the details of the recombination, mutation and selection in ES technique.

2.3.2 Evolution Strategies Techniques

The early application of ES technique was (1+1)-ES which consists of two individuals. One of them is the parent individual, which produces one offspring individual each of multiple generations (iterations). Each individual is represented by a pair of float-valued vectors $ind_i = (X, \sigma)$, where $i = parent$ for parent or current individual and $i = offspring$ for offspring individual. The first float-valued vector X represents the proposed solution in the search problem space. It is also called the proposed solution chromosome. Thus, the parent proposed solutions chromosome X_{parent} is for the parent individual $ind_{parent} = (X_{parent}, \sigma)$, while the offspring chromosome $X_{offspring}$ is for the offspring individual $ind_{offspring} = (X_{offspring}, \sigma)$. The second float-valued vector σ represents the standard deviations of random Gaussian numbers that are used in the mutations of the current parent chromosome to produce the new offspring chromosome. This mutation procedure can be represented by the following equation:

$$X_{offspring} = X_{parent} + N(0, \sigma), \tag{2.3.30}$$

where $N(0, \sigma)$ represents a vector of random Gaussian numbers with zero means and standard deviation equal to σ. Examining the quality of the parent and offspring chromosomes (the evolved solutions) is the next step in each evolving iteration in (1+1)-ES technique. The function used to check the quality of the proposed solution chromosomes is called the fitness or the objective function. In IR problem domain, the accuracy and the similarity matching functions such as Mean Average Precision, Precision, Error Rate, and Cosine Similarity among others (Baeza-Yates and Ribeiro-Neto, 2011) are used as the fitness functions for evolutionary and machine learning techniques (Li, 2014; Cordon et al., 2003; Cummins, 2008). For example, the $Cosine\_Similarity(d, q)$ fitness function is defined by:

$$Cosine\_Similarity(d, q) = \frac{\Sigma_{i=1}^{n} W_{id} \cdot W_{iq}}{\sqrt{\Sigma_{i=1}^{n} W_{id}^2 \cdot \Sigma_{i=1}^{n} W_{iq}^2}} \tag{2.3.31}$$

In the above equation, $Cosine\_Similarity(d, q)$ is the similarity function between the query q and document d vectors, n is the number of index terms that exist in the document d and query q, W_{id} is the weight of term i in document d and W_{iq} is the weight of the same term i in query q. The optimisation target for (1+1)-ES is to find an evolved document representation for the relevant document d corresponding to its query q. Assuming that $n = 2$ in Equation 2.3.31, the document d and the query q have only index terms t_1 and t_2. The query q vector has weight vector representation as $q = (0.25, 0.35)$. For the current evolved iteration j in (1+1)-ES, the proposed current evolved representation (parent chromosome) for d is $w_{1d}^{parent_j} = 0$ and $w_{2d}^{parent_j} = 0.45$. If the σ vector in equation 2.3.30 is $\sigma = (1, 1)$. Then, the Offspring chromosome of weight representations after mutation in the current evolving iteration is given by:

$$\begin{aligned} w_{1d}^{offspring_j} &= w_{1d}^{parent_j} + N(0,1) = 0 + 0.4 = 0.4, \\ w_{2d}^{offspring_j} &= w_{2d}^{parent_j} + N(0,1) = 0.45 - 0.1 = 0.35 \end{aligned} \tag{2.3.32}$$

where N(0,1) is a random Gaussian number with zero mean and 1 as standard deviation. The cosine similarity functions for the parent and offspring chromosomes are given by:

$$\begin{aligned} Cosine\_Similarity(d^{parent_j}, q) &= 0.814, \\ Cosine\_Similarity(d^{Offspring_j}, q) &= 0.9733 \end{aligned} \tag{2.3.33}$$

From Equation 2.3.33, the fitness function value for the offspring chromosome $d^{Offspring_j}$ is higher than the fitness function value for the parent chromosome d^{parent_j}. In other words, if the offspring chromosome which represents the relevant document D is more similar to the query q than the parent chromosome, the parent chromosome d^{parent_j} is replaced by the offspring chromosome $d^{Offspring_j}$ for the next evolving iteration $j+1$ in the ES technique. In (1+1)-ES, the standard deviation vector $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$ of the mutation is usually updated in each iteration according to the performance of the offspring chromosome. Moreover, the success rule differs in each fitness function in the hope of achieving an increased performance of the search for converging to a better evolved solution. The first success rule for the corridor model and sphere model was proposed by Rechenberg (Beyer and Schwefel, 2002). This success rule is called *1/5 success rule*. This rule is used to reduce or increase the standard deviation vector components $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$ based on the real value 1/5 for the sphere or corridor fitness functions using (1+1)-ES.

The multi-membered ES differs from the previous (1+1)-ES in the population size (the number of individuals) in each iteration (generation) (Beyer and Schwefel, 2002; Schwefel, 1981). Furthermore, each individual has a random probability to be selected for mating. Such multi-membered ES also use a recombination procedure which is similar to crossover in Genetic Algorithms. The most well known multi-membered ES techniques are: $(\mu+\lambda)-ES$ and $(\mu,\lambda)-ES$. In the $(\mu+\lambda)-ES$, the parent individuals μ are used to create λ offspring individuals in each evolving generation, where $\lambda \geq 1$. The worst λ individuals are discarded out of all $(\mu+\lambda)$ individuals. Then, the best μ individuals are used as parent individuals for the next generation. On the other hand, the $(\mu,\lambda)-ES$ has a different selection procedure. The parent individuals μ are used to create λ offspring individuals in each evolving generation. Then, parent individuals

are discarded and the selection of the best μ individuals from the λ offspring are used as the parent individuals for the next generation. Similar to (1+1)-ES, the fitness functions are represented as $(\mu + \lambda) - ES$ and $(\mu, \lambda) - ES$ to check the quality of the proposed evolved solutions. The standard deviations of the mutation parameters are no longer constant, nor changed by a deterministic rule such as the "1/5 success rule". They are incorporated in the individual evolution process. For creating the offspring individuals from parent individuals, the ES technique works as follows:

1) The algorithm selects two or more individuals for recombination. Assuming that the following two individual are selected:

$$\begin{aligned}(x^1, \sigma^1) = ((x_1^1, ..., x_n^1), (\sigma_1^1, ..., \sigma_n^1))\ and \\ (x^2, \sigma^2) = ((x_1^2, ..., x_n^2), (\sigma_1^2, ..., \sigma_n^2))\end{aligned} \tag{2.3.34}$$

where x^1 and x^2 are the chromosomes for individuals one and two, while σ^1 and σ^2 are the standard deviation step-sizes vectors of the mutations in individuals one and two respectively. There are two well-known ways of applying the recombination (crossover) operator:

A) Discrete recombination which produces the new offspring

$$(x', \sigma') = ((x_1^{pop_1}, ..., x_n^{pop_n}), (\sigma_1^{pop_1}, ..., \sigma_n^{pop_n})), \tag{2.3.35}$$

where $pop_i = 1\ or\ pop_i = 2$, while $i = 1....n$. Thus, each component was selected from individual one or individual two.

B) Intermediate recombination is similar to the uniform crossover in Genetic Algorithm (GA) (Le, 2011). When using the intermediate recombination, the new offspring becomes:

$$(x', \sigma') = (((x_1^1 + x_1^2)/2, ..., (x_n^1 + x_n^2)/2), ((\sigma_1^1 + \sigma_1^2)/2, ..., (\sigma_n^1 + \sigma_n^2)/2)). \tag{2.3.36}$$

These recombination types can be used to converge the proposed evolved solutions to the global optimal solutions. They can be applied on each of the two individual pairs or multiple individuals to produce a new population of offspring individuals.

2) The following procedure is used for mutation of the offspring (x', σ') obtained from the recombination. The mutation is done as follows:

$$\begin{aligned} \sigma^{Offspring} &= \sigma' \cdot e^{N(0,\Delta\sigma')}, \ and \\ x^{Offspring} &= x' + N(0, \sigma^{Offspring}) \end{aligned} \tag{2.3.37}$$

where $\Delta\sigma'$ is the variation parameter value in the mutation standard deviation. For controlling the convergence rate, Schwefel proposed an additional controlling parameter (Beyer and Schwefel, 2002). Assuming that the additional control parameter is θ and each individual is (x, σ, θ). The mutation Equation 2.3.37 becomes as follows:

$$\begin{aligned} \sigma^{Offspring} &= \sigma' \cdot e^{N(0,\Delta\sigma')}, \\ \theta^{Offspring} &= \theta' + N(0, \Delta\theta'), \ and \\ x^{Offspring} &= x' + C(0, \sigma^{Offspring}, \theta^{Offspring}) \end{aligned} \tag{2.3.38}$$

where $C(0, \sigma^{Offspring}, \theta^{Offspring})$ is a vector of Random Gaussian numbers with zero mean and appropriate probability density. Recently, Hansen proposed the use of the covariance matrix adaptation to adapt and to control the convergence rate (Hansen, 2016; Back et al., 2013). This is called Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). The historical developments of (1+1)-ES to the the state-of-the-art ES (p-sep-lmm-CMA-ES) is presented in (Back et al., 2013). However, the problem memory size and the run-time are issues in CMA-ES. This is because the size of Covariance Matrix and its calculations in each evolving iteration require more memory and more computational runtime. Thus, recent research (Back et al., 2013) proposed (1+1)-CMA-ES with various adaptations to reduce the problem memory size and the computational runtime. Christian

Igel et al. proved that (1+1)-Cholesky-CMA-ES is more efficient than multi-membered CMA-ES in some unimodal fitness functions (Ackley, Rastrigin and Griewangk) (Igel et al., 2006). Recently, Ilya Loshchilov proposed a computationally efficient limited memory CMA-ES for large scale optimisation technique (LM-CMA-ES) (Loshchilov, 2014). In this technique, vectors of random weights are used for the adaptation of the mutation and convergence rate. One of these vectors is a vector of random Ziggurat numbers. The Ziggurat random numbers are positive random Gaussian numbers within the range between 0 and 1. This type of random number is also used in (1+1)-Evolutionary Gradient Strategy for evolving global weight technique 6. The use of Gradient step-size helps to give more control in the converging rate of the new fitness functions such as Cumulative Cosine Similarity. The proposed ES techniques in this thesis are presented in details in Chapters 6 and 7.

2.4 Chapter Summary

This chapter presented the background material for IR and ES. It starts with the IR architecture followed by the various IR models such as Boolean, vector space and language models. Then, the chapter introduced the introductory material for the EML techniques in IR. Finally, the development of ES is presented in the last section (Section 2.3). The following chapter introduces the test benchmarks with the various type of creation them.

Chapter 3

Test Collections

This chapter describes the dataset types used for IR research. It also presents the various paradigms used for creating these IR benchmarks. There are two types of IR dataset benchmarks. The first type is textual test collections, the second type is distilled fully judged datasets from the textual test collections. The rest of this chapter provides knowledge about the paradigms used for creating these benchmarks.

3.1 Textual Test Collections

The IR research community started to create test collections to evaluate the performance of IR systems around 60 years ago (Cleverdon, 1960; Sanderson, 2010). The creation of test collections was a very expensive and time-consuming process. The reason is that each test collection requires relevance judgement values to measure the quality of the IR systems. The relevance judgement values require hiring experts in the test collection domain (collection topic) and training them for how they can assess the test collections. After the assessing process, the relevance judgement values and the proposed test collection should be validated by multiple IR systems. The validation process confirms that the relevance judgement values are accurate and represent the actual expert user interactions. Thus, the assessing and the validation processes require more money, time and effort for having the accurate relevance judgement values. Furthermore, the relevance judgement values were required to build intelligent IR systems using EML techniques and also for evaluating the IR research. The developments of test collections were static test collections in which

the document set and the query set were static (not changeable by time). It means that the number of documents and queries in the collection are constant numbers. These constant values are two of the static characteristics of the test collections. The IR research was using these static characteristics of the test collections for producing various TWS. Thus, the performance of the TWS is limited to the static characteristics of the collection at the beginning of IR system. This causes that there is a vital demand for re-weighting the document-term representation in the case of dynamic variation by adding/removing documents from the collection for keeping the same level of IR system effectiveness. However, the reason for applying IR research on static standardised test collections is that the standardised test collections are lower regarding cost, effort and more accurate regarding relevance judgement values to compare IR research contributions than applying IR research on online dynamic IR test collections. The noise and bias in historical user interaction data in online IR systems cause the limitation of having accurate relevance judgement values (Kharitonov, 2016). In other words, the historical user interaction data such as number of user clicks represent the online relevance judgement values (implicit relevance judgement values). The noise and bias in these implicit relevance judgement values in online IR test collections cause the limitation of having an accurate and reliable test collection evaluation in online IR systems (Li, 2014; Kharitonov, 2016; Hofmann, 2013). The user familiarity with IR system and education level are two of the parameters that cause the bias and noise in the implicit relevance judgement (Al-Maskari and Sanderson, 2011; Lorigo et al., 2006). Thus, the IR researchers (Hofmann, 2013; Schuth, 2016) tended to do their offline and online IR research on standardised test collections that judged by experts and validated by multiple IR systems. To the best of our knowledge, there are two well-known paradigms for creating standardised textual test collections. The first paradigm is the Cranfield paradigm and the second paradigm is the Pooling paradigm. The following sections describe these two paradigms in more details. Then, the distilled fully judged datasets are described.

3.1.1 Cranfield Paradigm

Cleverdon and his Cranfield University research team started in early 1960 to evaluate their IR system using their test collection (Cleverdon, 1960; Sanderson, 2010). They

created a test collection called Cranfield I and then they extended it to the second version called Cranfield II. At present, Cranfield II is called Cranfield collection. The Cranfield was composed of 1,400 documents (titles, author, names and abstracts) derived from the reference lists of around 200 recent published research papers at that time. The authors of each of those published papers were asked to write a question for each reference cited in their papers. These questions became the query set of the test collection. The authors were also asked to provide the relevance judgement values from their papers and their papers reference lists. The relevance judgement values used a scale from 1 to 5 to reflect the relevance degree of the papers to the query set (authors questions). Cleverdon's team checked all documents against all the questions. Furthermore, they asked the authors if they can provide more relevant documents corresponding to the query set. Intensive research for more than 10 years on Cranfield collection was done by Cleverdon research team before publishing the Cranfield collection as an open access data resource for other researchers after validating and evaluating their sponsored research (Cleverdon, 1960).

Other test Collections Using the Cranfield Paradigm

Many test collections have been created using the Cranfield Paradigm. Those collections are now widely used in IR research using EML techniques (Cordon et al., 2003; Cummins, 2008). Table 3.1 shows a list of the well-known test collections created with the Cranfield Paradigm (Hersh et al., 1994; Glassgow, 2014). Each test collection has three main components: a set of documents, a set of queries and the relevance judgement file. The creation of these collections and their relevance judgements has been done using different approaches including sampling (Cranfield, NPL, CACM, CISI and Medline) and extracting from real IR system (Ohsumed) (Soboroff, 2007; Hersh et al., 1994). The sampling procedure means the documents were chosen from published paper abstracts and their authors give samples or example of user queries, while Ohusmed were created by experts in the field with Boolean searching in MEDLINE system and validated by physicians. The choice of Boolean search for Ohsumed creation reduces the barrier of bias to any term-weighting scheme used in the relevance judgement procedure by the expert physicians.

Table 3.1 shows the topic domain of the test collection, the number of documents and

Table 3.1: Cranfield Paradigm Test Collections General Characteristics.

ID	Description	No. of Docs.	No. of Queries
Cranfield	Aeronautical engineering abstracts	1,400	225
Ohsumed	Clinically-Oriented MEDLINE subset	348,566	105
NPL	Electrical Engineering abstracts	11,429	93
CACM	Computer Science ACM abstracts	3,200	52
CISI	Information Science abstracts	1,460	76
Medline	Biomedicine abstracts	1,033	30

the number of queries in the test collection. Figures A.1 and A.2 in Appendix A show an example of a document and a query in the Cranfield collection. Each document contains 5 fields and each field is identified by a symbol. The first field (.I) is the document number. The second field (.T) is the title of the paper. The paper's author names are preceded by (.A) symbol, while the paper publisher and the publishing date are proceeded by (.B) symbol. Finally, the paper abstract content is preceded by the symbol (.W). Each query in the Cranfield collection consists of two fields. The symbols (.I) and (.W) are the query number and the query textual content respectively. For each test collection, there are specific symbols to identify each document and query fields. The details of each document and query symbols are provided in (Hersh et al., 1994; Glassgow, 2014). The largest test collection from table 3.1 is the Ohsumed collection. This collection is a subset of MEDLINE which is a database of medical publications. The Ohsumed collection consists of 348,566 documents (out of over 7 million records) from 270 medical journals during the period of 1987-1991. It includes a document identifier, title, abstract, MeSH index terms, author, content, and publication details. In addition, Ohusmed contains 106 queries and each query has a query identifier, patient information and information request (query's content).

The test collections produced by Cranfield paradigm are reliable and realistic test collections for evaluating IR system accurately but these collections cost more effort, time and money than have been created using Pooling paradigm (Sanderson, 2010). Moreover, it is difficult and it is not feasible for having very large test collections produced by Cranfield paradigm. However, the reliable of relevance judgement values to repre-

sent expert users in the Cranfield collections was the reason of the extensive IR research based on this type of test collection. Thus, the test collections that illustrated in table 3.1 were used in most of mathematical, probabilistic and evolutionary computation research to improve IR system as in (Cordon et al., 2003; Cummins, 2008), but the limitation of Cranfield Paradigm test collections as supervised EML test collections is discussed in Section 3.1.3.

3.1.2 Pooling Paradigm

The Pooling technique has been widely used in TREC tracks to evaluate the test collections before submitting it as test benchmarks for IR researchers (TREC, 2016*a*). The Pooling Paradigm starts with crawling the web or a specific web domain for creating the document set of the test collection. Then, TREC organiser committee uses their IR system with the crawled document set and expert human annotators to produce the query set and its relevance judgement. It begins with using a retrieval method of their IR system for retrieving the documents responding to the queries (topics) created by the expert annotators. Then, the top-k documents retrieved (k pool depth) are judged by the TREC human expert annotators to determine the relevant and irrelevant documents to each query. Finally, the test collection becomes ready for TREC track competitions by using multiple IR systems to validate the test collections and the research outcomes. (Buckley et al., 2007) argued that the test collection size affects the degree of bias related to the relevance judgement values. If the test collection is large the pool size should be large respectively to provide an accurate relevance judgement for each query. This issue inspired the need for various statistical analysis of the results using various retrieval methods on pooling benchmarks such as in TREC Disks 4 and 5 (Soboroff, 2007; Sanderson, 2010). The TREC Disks 4 and 5, .GOV, ClueWeb09 and ClueWeb12 test collections are well-known document collections used in various TREC and SIGIR tracks (TREC, 2016*b*; Soboroff et al., 2003; Habernal et al., 2016). The TREC and SIGIR are the most well-known international conferences that produce standardised test collections for IR research. The most widely used standardised pooling collection from TREC and SIGIR is TREC Disk 4 and 5, while ClueWeb12 is the newest textual pooling test collection. The range size of TREC Disks 4 and 5 is about half a million documents, while the set of queries and

their relevance judgements varied between various TREC tracks (TREC, 2016*b*,*c*). The TREC Disks 4 and 5 document set was crawled and created from news and Broadcast websites such as Financial Times website for FT document set in TREC Disks 4 and 5. They were combined by multiple relevance judgement set and query set for multiple purpose of IR research in TREC tracks such as TREC 1 to 8, Robust-2003 and Robust-2004 (TREC, 2016*b*,*c*). ClueWeb12 has been used in TREC 2014 and contains 733,019,372 pages acting as documents with 27.3 Terabytes of storage on hard disks (TREC, 2016*d*; Lemur, 2016*a*). The comparison between TREC Disks 4 and 5 with ClueWeb12 shows that the TREC Disks 4 and 5 test collections are more accurate for evaluating IR research than ClueWeb12 (Urbano, 2016). The common research issues in the ClueWeb12 to act as real IR test collection are as follows:

- ClueWeb12 is a pooling judged collection which has only 50 queries in the query set. However, the small pool size compared to the collection size causes the bias to the retrieval method used to assess the relevance judgement of the test collection as discussed above. The total unique number of relevant/irrelevant documents existing in relevance judgement for ClueWeb12 of Web Track 2014 is only 5666 documents, while the document set contains 733,019,372 documents. This confirm the limitation and bias for having an accurate evaluation with small pool size containing only 5666 relevant/irrelevant documents of 733,019,372 documents existing in the document set. This means that there may be a lot of unjudged relevant documents in the test collection which are corresponding to some queries in the query set but they are not appear in the relevance judgement file. The unjudged relevant documents will be considered as irrelevant documents in the evaluation procedure of a new IR model which causes inaccurate evaluation for IR research. On the other side, TREC Disks 4 and 5 were judged and evaluated for various TREC tracks by multiple IR systems. The pool size for these tracks are reasonable comparing to the document set size. According to the statistical analysis based on results produced by multiple IR systems in (Urbano, 2016), the Disks 4 and 5 with the relevance judgement produced in Robust 2004 track are the most stable and accurate pooling test collection for assessing and comparing between IR research.

Table 3.2: Characteristics of the Pooling test Collections Used in this Thesis.

ID	Description	No. of Docs	No. of Queries
TREC Disks 4&5 (Robust 2004)	News and Broadcast Web-Pages	472525	230
TREC Disks 4&5 (Crowdsource 2012)	News and Broadcast Web-Pages	18260	10

- ClueWeb12 document set has been collected using five instances of the Internet Archive Heritrix web crawler that were running on five Dell PowerEdge R410 machines with 64GB RAM (Lemur, 2016*b*). Furthermore, a huge computational cost will be required for adapting ClueWeb12 to be a fully judged collection for a large number of queries.

In this thesis, TREC Robust 2004 and Crowdsource 2012 relevance judgements for Disks 4 and 5 and Cranfield paradigm test collections were used in Chapter 5 and 6 (TREC, 2004; Smucker et al., 2012; Hersh et al., 1994; Glassgow, 2014). The reason is that they have stable and accurate relevance judgements between various IR systems which are proved and validated by (Urbano, 2016), TREC tracks (TREC, 2016*b*,*c*) and the previous IR research (Cleverdon, 1960; Sanderson, 2010). In addition, the Cranfield paradigm collections are the most suitable test collections to simulate the real expert user feedback without bias. Moreover, they can be used on normal PC for IR systems. The detailed pooled test collection characteristics used in this thesis are shown in table 3.2. On the other hand, we identified the limitation of TREC Disks 4 and 5, and Cranfield Paradigm test collections to act as supervised EML datasets. This limitation is discussed in Section 3.1.3. These collections can only simulate the early stage of the IR test collections before having extensive relevance feedback by user interactions for unsupervised learning techniques. The later stage of IR systems can use supervised EML technique when having fully judged test collection from extensive historical user interactions. The fully judged collections can be used to extract fully judged query-document pairs for applying supervised Learning to Rank and creating ranking models for query auto-completion searching (Kharitonov, 2016; Liu et al., 2007; Qin et al., 2010).

3.1.3 Analysis of Textual Collections for Evolving TWS

This section presents the motivation for having non-learning (statistical) IR approaches instead of Evolutionary and Machine Learning (EML) techniques for evolving TWS at the early stage of the establishing the IR system. Evolutionary computation approaches have been applied for evolving term weights or evolving a TWS such as in section 4.1. The relevance judgement is the set of queries for the test collection and their corresponding relevant documents from the collection. The objective functions of the learning IR approaches use relevance judgements to check the quality of the evolved solutions. However, as mentioned earlier, real and test IR test collections are partially judged as it is not feasible to have fully judged test collections at the beginning of any IR system (Qin et al., 2010). Consequently, EML techniques for evolving TWS are limited because the trained queries and their corresponding relevant documents do not cover the whole term space of the collection.

In the evolving TWS and term weights research, the EML techniques should be trained using queries and the corresponding relevant documents containing the whole term space (index terms) that exists in the collection (see section 4.1.1). Then, the IR system should be tested with a test dataset different to those used in the learning process (training dataset). To the best of our knowledge, it appears that works applying evolutionary computation to IR systems use the same training set from the learning stage to then test the candidate solution that represents the documents. The index terms that do not exist in relevant documents are also given random weights. Hence, these index terms cannot be judged by the fitness function because they do not exist in relevant documents nor the query set. Thus, the number of random weights created in the evolutionary learning process are not really applicable to measure the relevancy for any query. Moreover, evolving term-weighting function using EML is based on the collection or relevance judgement values that were used in the evolving process. Thus, the evolved term-weighting function can not be generalised as the optimal solution for all textual collection. The ideal solution for this issue is the EML technique should be applied for every textual collection. However, there is a need for using a term-weighting function (TWS) before evolving it to gather the user relevance judgement values by establishing IR system. Thus, optimising

document representation regarding term-weights rather evolving TWS itself after establishing IR system and gathering the relevance judgement values is the ideal solution for this issue. Although, most of the documents in the test collection have no relevance values to optimise them using EML techniques at the beginning of IR system. This issue is not only for evolving TWS and term-weights but also for using probabilistic and language models to establish IR systems (see Section 2.2.4). This is because probabilistic and language models are probabilistic models that rely on the relevance judgement distribution between the relevant documents and their corresponding queries to propose its TWS. The problem with evolving TWS and term weights is likely to arise in any test collection created at the beginning of establishing IR system.

Table 3.3: Test Collections General Basic Characteristics.

ID	Description	No. of Docs.	No. of Queries
Cranfield	Aeronautical engineering abstracts	1,400	225
Ohsumed	Clinically-Oriented MEDLINE subset	348,566	105
NPL	Electrical Engineering abstracts	11,429	93
CACM	Computer Science ACM abstracts	3,200	52
CISI	Information Science abstracts	1,460	76
Medline	Biomedicine abstracts	1,033	30
FBIS	Foreign Broadcast Information Service	130,471	172
LATIMES	Los Angeles Times	131,896	230
FT	Financial Times Limited	210,158	230

On the other hand, Table 3.3 lists the eight test collections (see Section 3.1) used in our analysis and that have also been used in most literature for evolving TWS and term weights (Cordon et al., 2003; Cummins, 2008). Each test collection has three main components: a set of documents, a set of queries and the relevance judgement file. The creation of these collections and their relevance judgements has been done using different approaches including sampling (Cranfield paradigm), extracting from real IR system and pooling paradigm (Soboroff, 2007; Hersh et al., 1994). FBIS, FT and LATIMES were the most recent test collections used in evolving local and global term weights in IR (Cummins, 2008). These test collections are the test collections existing in TREC Disk 4 and 5 with Robust relevance judgement values as discussed in Section 3.1.2. A number of additional characteristics about the test collection was not taken into consideration for evolving TWS or evolving term weights. Table 3.4 gives the values for such additional characteristics which are defined as follows.

NoUR is the number of unique occurrences of relevant documents that exist in the test

collection.

NoDR is the number of duplicates occurrences of relevant documents between queries in the query set.

NoInDC is the total number of index terms that exist in the whole test collection.

NoInDr is the number of unique index terms that exist in the relevant documents set.

NoInNR is the number of index terms that were not covered by relevance judgement and is given by the difference NoInD − NoInDr. This is the number of index terms that get a random weights in documents representations without testing them with the objective function.

Table 3.4: Characteristics of Test Collections to Consider When Evolving TWS.

ID	NoUR	NoDR	NoInDC	NoInDr	NoInNR
Cranfield	924	914	5,222	4,236	986
Ohsumed	4,660	177	227,616	22,760	204,856
NPL	1,735	348	7,697	3,536	4161
CACM	555	241	7,154	3,189	3,965
CISI	1,162	1,952	6,643	5,709	934
Medline	696	0	8,702	6,907	1,795
FBIS	4,506	42,873	177,065	41,272	135,793
LATIMES	4,683	497	211,909	56,255	155,654
FT	5,658	55,819	287,876	45,564	242,312

Table 3.4 shows characteristics for the collections that were created with a pooling technique, such as FBIS, FT and LATIMES collections and with Cranfield (Sampling) technique such as Cranfield and Ohsumed. From this table, the majority of index terms that exist in the test collections were not covered by the relevance judgements. Thus, the majority of index terms of the test collections did not exist in the queries nor their corresponding relevant/irrelevant documents. As discussed above, this is an issue for evolved TWS because the trained queries and their corresponding relevant documents do not cover the whole term space of the collection. Hence, it is an argument that having non-learning IR approaches instead of learning ones at the start of building IR system is vital. Then, the relevance feedback can be gathered and used for improving the system by partially learning model as described in Chapter 6.

3.2 Distilled (LETOR) Benchmark Datasets

From the previous paradigms, (Liu et al., 2007; Qin et al., 2010) proposed producing fully judged distilled benchmark datasets called LETOR datasets. They used sampled and pooled test collections to create LETOR datasets. These datasets contain a feature vector for each query-document pair rather than textual document/query and they were used for the Learning to Rank (LTR) problem. Each LETOR dataset usually consists of query-document pairs for a large number of queries (Qin et al., 2010). Table 3.5 shows the representation of several query-document pairs in LETOR datasets. Each row (query-document pair vector) contains a relevance label indicating the relevance degree of the document for a query. In most cases, the relevance labels have one of three values which are: 2 means the query is high relevant to the document, 1 means the query is relevant to the document and 0 means the query is irrelevant to the document. There is also a query identifier (id) indicating the corresponding query number for each query-document pair. The feature vector refers to M features such as Term-Weighting Scores (e.g. TF-IDF, Okapi-BM25 and Language Models (Qin et al., 2010)), PageRank, Host Server Importance and other features associated to the similarity matching between the query and the document such as BM25. The query-document pair also contains features represent the recent IR research added in SIGIR conference papers such as Language Model with Absolute Discounted Smoothing (LMIR.ABS), Language Model with Jelinek-Mercer smoothing (LMIR.JM), Language Model with Bayesian smoothing using Dirichlet priors (LMIR.DIR) and User Click features (Liu et al., 2007; Qin et al., 2010; Qin and Liu, 2013; Chen and Goodman, 1996; Chuklin et al., 2015). 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 whole features contribute to identify the relevance level of the document to the query.

The dataset itself is divided into N folds (usually it contains five folds) and each fold consists of training, validation and testing set of the query-document pairs. These folds are useful for examining the LTR algorithm behaviour and its predictive performance by applying it on test sets different to the training sets. The creation of LETOR dataset starts with choosing a textual test collection usually from TREC test collections such

as Ohsumed, .Gov and .Gov2 collections. Then, the relevant and irrelevant documents with their corresponding queries that exist in the relevance judgement were selected as a fully judged test collection. This process is called document sampling or document selection. There was another sampling method by extracting the documents and their corresponding queries from a real search engine such as Microsoft Bing datasets. Following to this selection process, the selected fully judged test collections were used to extract low-level features such term frequency in each query with its relevant/irrelevant document and high-level feature such BM25 similarity matching between each query with its relevant/irrelevant document. They also were used to create hybrid features such as mentioned in recent SIGIR and TREC papers. The purpose of having this list of features in each query-document pair vector is to simulate the actual query-document pair representation in search engines with the recent research work published in the well-known IR conferences such as SIGIR and TREC conference. On the other hand, the creation of feature list in Microsoft Bing datasets was different because the features were extracted from real Microsoft Bing search engine dataset. The search engines usually use multiple retrieval and representation models (low-level, high-level and hybrid features) in their system for effective performance. This is because depending on only one IR model in their systems is not sufficient for effective and efficient performance. More details about LTR datasets creation as query-document pairs are in (Qin and Liu, 2013; Qin et al., 2010). The LETOR datasets represent the later stage of the IR system after the test collections become fully judged and they becomes ready for applying supervised EML techniques for ranking retrieved documents problem.

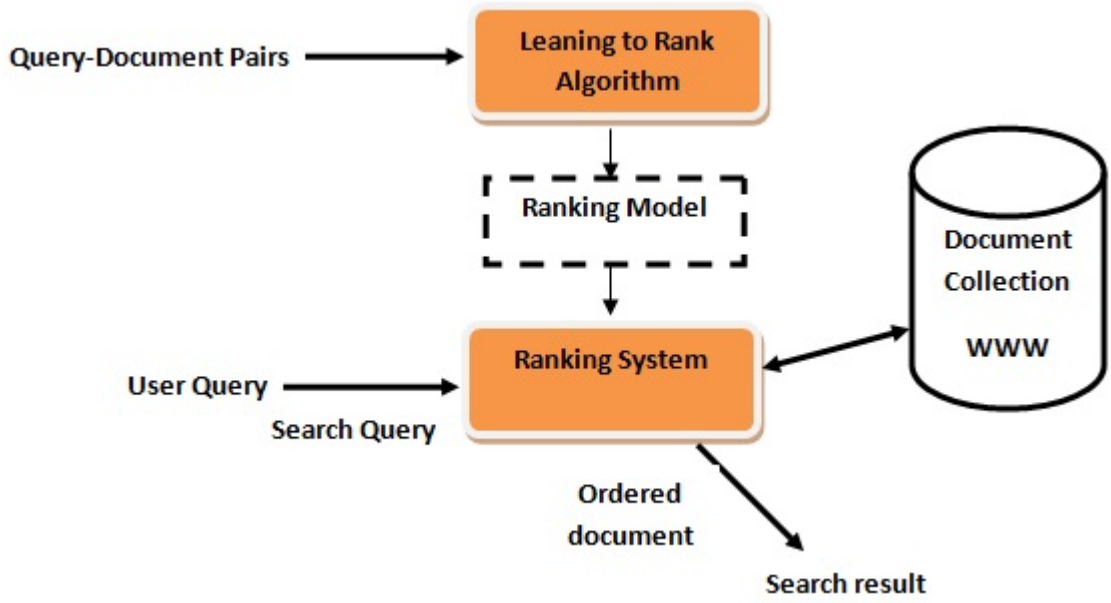

Figure 3.1: Learning to Rank (LTR) approach architecture as discussed in Liu (2009).

Table 3.5: 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

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 query. Figure 3.1 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 a machine learning technique (Li, 2014). The ranking model or ranking function is created and then used to rank the search results for the user 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 test collection in response to the user search query. The LTR datasets represent the later stage of IR system test collection when they become fully judged datasets and IR systems

uses LTR techniques for retrieving document responding to user queries. The next section reviews the LTR benchmark datasets used in this thesis in order to set the context for the method proposed.

3.2.1 LETOR 3 Datasets

LETOR 3 datasets were created from different two test collections: Ohsumed and .Gov corpus. The .Gov collection were used in TREC 2003 and TREC 2004 Web tracks. The query sets of these tracks were selected for LETOR 3. The .Gov collection itself was collected from crawling the .Gov US domain on January, 2002. This collection uses two query sets to create six LTR datasets. In other words, each TREC Web track query set is used for creating three different LTR datasets. These datasets are related to the Topic Distillation (TD), Homepage Finding (HP) and Named Page finding (NP). These three categories were used in TREC 2003 and TREC 2004 Web track competitions. The TD aims to create a dataset from the list of documents on the same topic domains of the queries. The HP category is related to creating test collection using query set home pages, while the NP is related to creating test collection from the web pages that have names identical to the queries contents. The details of LETOR 3 TREC datasets are shown in Table 3.6. These datasets are called TD2003, HP2003, NP2003, TD2004, HP2004 and NP2004. The limitation of these datasets is the number of queries in each dataset is smaller than number of queries in LETOR 4 datasets. However, the number of features in each query-document pair in .Gov datasets is 64 features. These features includes more feature than LETOR 4 datasets. The feature list includes the similar low-level, high-level and hybrid features that exist in LETOR 4 and Microsoft Bing datasets. The feature low-level and high-level features represent the characteristics of the various parts existing in the document (webpage). The feature list of the query-document pair is shown in Table A.3 in Appendix A.

Finally, the last LETOR 3 dataset is the dataset distilled from Ohsumed test collection. The Ohsumed corpus is a subset of MEDLINE medical publication database (Hersh et al., 1994). MEDLINE database consists of about 0.3 million records (out of over 7 million research articles) from 270 medical journals during the period between 1987 and 1991. Each record includes title, abstract, MeSH index terms, author, the publisher details and

Table 3.6: The properties of LETOR 3 datasets used in the experimental study.

Dataset	Queries	Query-Document Pairs	Features	Relevance Labels	No. of Folds
TD2003	50	49058	64	{0, 1, 2}	5
HP2003	150	147606	64	{0, 1, 2}	5
NP2003	150	148657	64	{0, 1, 2}	5
TD2004	75	74146	64	{0, 1, 2}	5
HP2004	75	74409	64	{0, 1, 2}	5
NP2004	75	73834	64	{0, 1, 2}	5
Ohsumed	106	16140	45	{0, 1, 2}	5

publication date. The Ohsumed query set consists of 106 queries and each query consists of the search request and it is associated with patient disease information. The Ohsumed relevance set is judged by human annotators on three level: 0 for irrelevant, 1 for partially relevant and 2 for definitely relevant. The number of query-document pairs in Ohsumed collection is 16,140. It has five cross-validation folds as they will describe in Table 3.7 in Section 3.2.3. The number of features in each Ohsumed query-document pair is 45 features. The query-document feature list details are shown in Table A.4.

3.2.2 LETOR 4 Datasets

LETOR 4 is the most recent LTR datasets that have been extracted and distilled from .Gov2 webpage collection (25 Million pages) (Qin and Liu, 2013). LETOR 4 datasets use two query sets from Million Query track of TREC 2007 and TREC 2008. The two datasets produced by these query sets are called MQ2007 and MQ2008. There are about 1692 and 784 unique queries in MQ2007 and MQ2008 with labelled documents respectively. The advantage of choosing these test collections to produce MQ2008 and MQ2007 are the large number of queries existing in Million Query TREC tracks. Similarly to the Microsoft Bing Search datasets, MQ2007 and MQ2008 contains five cross validation folds. The number of query-document pairs in MQ2007 and MQ2008 are 69,623 and 15,211 respectively, while the number of features in each query-document pair is 46 features in MQ2007 and MQ2008. The feature list contains low-level, high-level and hybrid features which are similar to the feature list in Microsoft Bing datasets but they have

lower number of commercial feature list than Microsoft Bing datasets. They also have not user click and dwell time features that exist in Microsoft Bing datasets. However, they include most of recent language models, probabilistic models and mathematical models of term-weighting and similarity matching features. Table A.2 in Appendix A shows the feature list of the query-document pair in MQ2007 and MQ2008. The relevance label set consists of value 0 for irrelevant, value 1 for partially relevant and value 2 for totally relevant. The datasets were constructed by Tao Qin et al. This team cooperated with TREC2007/2008: Northeastern University team, University of Massachusetts team, I3S_Group_of_ICT team, ARSC team, IBM Haifa team, MPId5 team, Sabir Buckley team, HIT team, University of Amsterdam team, University Melbourne team to produce LETOR 4 datasets (Qin and Liu, 2013).

3.2.3 Microsoft Bing Search Engine Dataset

Tie-Yan Liu and Tao Qin (Qin and Liu, 2016) created the most recent LETOR datasets using the Microsoft Bing search engine. Their datasets are called MSLR-WEB30K which consists of 30,000 queries. A random sample that contains 10,000 queries has been created from MSLR-WEB30K and it is called MSLR-WEB10K. The two datasets are large scale benchmarks for evaluating the performance of LTR ranking models. The relevance label set for each query-document pair is created by the Microsoft Bing search engine. The relevance label takes one of five values from 0 (irrelevant) to 4 (perfectly relevant). The feature vector for each query-document pair is selected by Tie-Yan Liu and Tao Qin. The feature list in each query-document pair contains low-level features such as query term frequency in the webpage (document) parts (body, anchor, title, url and whole document). It also contains high-level features such as PageRank, BM25 and Language Model matching. Furthermore, it also contains hybrid feature such dwell time for gathering user relevance labels, number of user clicks and SiteRank. They include most of recent language models, probabilistic models and mathematical models for term-weighting schemes and similarity matching features with commercial importance features. Table A.1 in Appendix A shows the feature list and a column called Comments in this table provides more explanation about these features that exist in Microsoft Bing search engine dataset. This feature list simulate the real dataset existing in the search

engines (real IR system). The number of query-document pairs in the MSLR-WEB10K and MSLR-WEB30K datasets are 1,200,192 and 3,771,125 query-document pairs respectively. Each query-document pair consists of 136 features and one relevance label. The MSLR-WEB10K or MSLR-WEB30K dataset is partitioned into five parts S1, S2, S3, S4 and S5 using five-fold cross validation. In each fold, three parts are: one part used for training set, one part is used as validation set and the remaining part is used as a test set. Table 3.7 shows the details of MSLR-WEB10K or MSLR-WEB30K cross-validation folds. These datasets was created by Tao Qin, Tie-Yan Liu, Wenkui Ding, Jun Xu, Hang Li with Bing search engine team including Nick Craswell as leader for that team (Qin and Liu, 2016). The training set is used to train the ranking model using LTR algorithm, while the validation set is used to tune the hyper-parameter of the learning algorithm. Finally, the test set is used to evaluate the predictive performance of the learning model.

Table 3.7: LTR Dataset Folds From Dataset Partitions S1, S2, S3, S4 and S5.

Folds	**Training Set**	**Validation Set**	**Test Set**
Fold1	S1,S2,S3	S4	S5
Fold2	S2,S3,S4	S5	S1
Fold3	S3,S4,S5	S1	S2
Fold4	S4,S5,S1	S2	S3
Fold5	S5,S1,S2	S3	S4

3.3 Chapter Summary

This chapter demonstrated the test benchmarks with various type of creation them. It started with Cranfield and Pooling paradigm for creating textual test collections. It also showed various LTR datasets details. It also introduced an analysis for textual collections that have been used for evolving TWS using EML techniques. From this analysis, we argue that there is a preference for using mathematical TWS rather than evolving it when the relevance judgement queries and documents do not cover the whole term-space.

Chapter 4

Related Work

Evolutionary and Machine Learning (EML) approaches have been widely used in IR. The *Term Vector Model (TVM) and Feature Vector Model (FVM)* are methods of data representation employed when applying these approaches. For the TVM, the document and the query are represented as vectors of term-weights. In TVM, EML techniques are usually used with the traditional Vector Space Model (VSM). Whereas, in the FVM, the term-weights are replaced by feature-weights to produce feature vectors of query-document pairs. These features represent the query-document similarity matching values (such as cosine similarity and BM25), the query-document term-weights (such as TF-IDF, Okapi and Language Models) and the reputation of the document on the web. The rest of this chapter presents some of the literature related to using EML techniques on the TVM and FVM.

4.1 Learning Approaches Based on the Term Vector Model (TVM)

The application of EML techniques in IR research, using TVM has been on numerous problem domains, the most important ones are as follows:

1. Document Indexing Problem. This problem domain is mainly about evolving or learning the most appropriate document vector representations in IR system.

2. Query Optimisation Problem. This problem area is primarily about evolving or

learning the most appropriate query vector representations in IR system. These queries are saved for the next search to be used in query auto-completion suggestion by the IR system.

3. Similarity Matching Functions Problem. This problem domain concerns about optimising or evolving a similarity matching function based on the query-document vector representation of term-weights.

The most of the literature related to using EML techniques on the TVM and FVM are population-based techniques which require large memory sizes or requiring large computational evolving/training time periods. The next sections describe some of the work related to these problem domains in details.

4.1.1 Document Indexing Problem

EML techniques are commonly used to improve the effectiveness or the accuracy of various IR problem areas such as in document indexing problem (Cordon et al., 2003; Liu, 2009). The objective functions used in *Evolutionary Computation (EC) and Machine Learning (ML)* techniques usually rely on the relevance judgements to determine the quality of the evolved candidate solutions. The following sections outline previous research carried out regarding the document indexing problem (Zobel and Moffat, July, 2006) using EC techniques. The document indexing problem refers to the process of assigning weights to each term that exists in every document in the collection. This type of problem can be divided into: 1) evolving term-weighting schemes (TWS), and 2) evolving term weights.

Evolving Term-Weighting Schemes

In this category, researchers have tried to evolve the best TWS for improving IR effectiveness using Genetic Programming (GP). However, these TWS can be considered as collection-based functions, because each test collection has different characteristics. Furthermore, all the test collections are partially judged to simulate real test collections at the beginning of IR systems. As a result, most of the index terms in a test collection do not exist in the training queries and their relevant documents (see section 3.1.3).

Furthermore, there is a need for using a TWS to collect the relevance judgement values before using EC techniques and it is impractical to evolve TWS without relevance judgement values. Moreover, evolving TWS technique does not guarantee having better IR effectiveness than mathematical TWS in test collections different from the ones used in evolving procedure. Hence, research work in (Fan et al., 2000; Oren, 2002; Cummins and O'Riordan, 2006) has been carried out for evolving TWS in IR research field and they did not consider these issues.

The first approach for evolving a weighting function using GP was developed by (Fan et al., 2000) using two test collections. One was the Cranfield collection containing 1,400 documents and 225 queries. The other was the Federal Register (FR) text collection from TREC 4 containing a huge number of documents (55,554 documents) compared to its queries (50 queries). Fan et al. argued that few documents were relevant for these queries so they chose a larger number of documents (2,200 documents) than the number of relevant documents as a training set. They used the precision based on collections relevance judgement with a threshold as a fitness function in their application. The evolved TWS created with their GP approach was used to test the same trained queries on the whole test collections. Their results outperformed TF-IDF. However, no results for the Cranfield collection have been shown with this approach (Fan et al., 2000). This technique is population-based EML method which requires large memory size and consequently large evolving time. These limitation are not existing in (1+1)-Evolutionary Algorithms.

(Oren, 2002) proposed employing GP to evolve the term-weighting function using a terminal set similar to the one used by Fan et al. discussed above, but with an additional function operator (square root). Oren used the Cystic Fibrosis database (Shaw et al., 1991) which consists of 1239 documents and 100 queries, comparing his approach to the TF-IDF term-weighting scheme. His method outperformed TF-IDF with regards to recall-precision values. In that experiment, a cluster of computers was used due to the problem size. Thus, the computational cost of Oren's approach even for the small collection used, was very high.

(Cummins and O'Riordan, 2006) proposed a methodology for evolving local and

global term-weighting schemes from small test collections. They showed that their global weighting function evolved on small collections also increased the average precision on larger test collections. However, their local weighting function evolved on small collections did not perform well on large collections. They conducted experiments on five test collections: Medline, Cranfield, CISI, NPL and Ohsumed. The computational runtime required by their approach on the smallest training set from the Medline collection was significant: 6 hours on a standard PC. Thus, the main limitations of their approach are: 1) long computational time and large problem size on medium and large test collections, 2) the issue of test collections being partially judged, 3) evolving local and may be global TWS can not be generalised from test collection to another, and hence poor performance on collections other than the training set. Cummins and O'Riordan identified that full term weighting scheme evolved on small test collections did not outperform Okapi-BM25 on large test collections (Ohusmed88, Ohsumed89, Ohsumed90-91 and NPL collections).

Evolving Term Weights

Genetic Algorithms (GA) have been used for evolving term weights to produce better document representations for whole test collections. These approaches are also based on the relevance judgement. The same drawbacks noted previously arise when using these approaches: the reliance on partial relevance judgement for the collection and the need to run the GA again after changes occur within the collection (adding more documents to the test collection). This because the added documents to the collection requires document-weight representation that should be assigned by GA rather than traditional TWS.

(Gordon, 1988) proposed the first approach of applying a GA to IR for adapting the term weights for every document in the corpus. He demonstrated the value of using a GA for adapting term weights instead of using probabilistic models. He also highlighted some issues of using probabilistic models, such as dependencies among index terms, dependency on the estimation of probabilities, relevance judgement based on a small set of queries and high computational cost of automated probabilistic models. In this research, the GA used a probability of crossover equal to 1 with no mutation

and relevance feedback adaptation as the fitness function. Findings showed that the GA improved document representation to distinguish between relevant and non-relevant queries. The problem size was very large, more than the document space as it consisted of multiple representations for each document.

(Vrajitoru, 1998) also applied a GA to adapt term weights. The approach used a new dissociated crossover and tested different ways to generate the initial document descriptions. Vrajitoru conducted experiments using two test collections (CISI and CACM collections), which were both larger than Gordon's chosen collection (Gordon, 1988). However, this approach also had the same limitation related to the relevance judgement due to the nature of the test collections.

The research limitation identified above of evolving TWS and evolving document-term weights was the motivation for proposing a new perspective of evolving global term weights rather than evolving TWS and evolving document-weight representations. Chapter 6 shows a new method for optimising document-weight representations by evolving global term weights using (1+1)-Evolutionary Gradient Strategy. This technique has the lowest problem size than the above techniques and it considers the limitation of relevance judgement values at the beginning of IR systems.

4.1.2 Query Representation Problem

Optimising query representation has been a dominant problem in the TVM of IR research. The EML applications have been used to modify the query representation by adding and removing terms or adapting better weights for the existing query terms. This is done while considering the relevance judgements (such as users clicks on retrieved relevant documents). This problem has been devised from the Ide dec-hi and the other mathematical methods ideas (Salton and Buckley, 1997) for improving query representation for future user searches. This modified query term/weight vector representation has been used to retrieve more relevant documents than the original user query. These improved query representations were saved in the IR system as modified representations, in order to improve future searches using the same set of queries. However, this approach only

improved IR system effectiveness for the trained set of queries. This boosted the need to re-run EML application for every new added query to the IR system relevance to update the representation for the new query. This type of problem is divided into three groups (Cordon et al., 2003) which are as follows:

- Query Weight Learning (Weight Selection). This problem area is primarily about adapting the query weight representation to the most appropriate representation. Thus, the query can retrieve the relevant documents in the top of the search list and the irrelevant documents in the bottom of the search list.
- Query Term Selection. This problem area is primarily about adapting the query terms representation to the most appropriate representation by adding/removing terms with their weights to the query from its relevant/irrelevant documents. Thus, the query can retrieve the relevant documents in the top of the search list and the irrelevant documents in the bottom of the search list.
- Mixed Term and Weight Selection. This problem domain is mainly about mixing between query weight learning and term selection for better-saved query representation in the IR systems.

The following sections will discuss some of the approaches used to tackle these problems.

Hill-Climbing Using Okapi Approach

Hill-Climbing is a local search algorithm used for identifying better solutions in the solution search space by improving query vector representations (Talbi, 2009; Robertson, Walker, Hancock-Beaulieu, Jones and Gatford, 1995; Robertson et al., 1996). Hill-Climbing technique starts with initial solution which is the original query. Then, hill-Climbing technique searches the neighbourhood solutions by adapting the original query for better IR effectiveness in several evolving iterations using mutation procedure. In TREC-3 (Robertson, Walker, Hancock-Beaulieu, Jones and Gatford, 1995), the hill-climbing approach was used to optimise query representation via a term selection method. The experimental settings of this approach were as follows:

1. The weight values of the terms in term-set were static which is Okapi term-weights in the corresponding relevant documents to the query. This means that each selected term from relevant document to be added or removed had a fixed weight which is the weight existing in the relevant document.

2. The terms in the term-set were ordered as a list in a descending order of their Okapi term-weights. It was updated in every evolving iteration in hill-climbing using the index terms of the test collection.

3. The three top weighted terms from each relevant document in the collection were used to build the query's (topic's) term-set.

4. The terms were considered just once during the iterations of the evolution of the new query vector.

5. After the first top three terms, the successive term in the term-set was added to the query. Then the fitness value was calculated, which was the average precision of the top 1000 documents retrieved.

6. The above steps were repeated until a stopping criterion was satisfied.

The stopping criterion could be one of the following thresholds:

- The maximum number of terms on the term-set was the max-terms =30 terms.
- The maximum number of successive iterations that have worse offspring query representation was max-bads = 8 successive iterations.
- The maximum total number of terms in each query was MaxTerms = 150 terms in the chromosome representation.
- The maximum runtime which has been set to the max-time= 1 or 2 hours per each evolved query (topic).

Robertson et al. (Robertson et al., 1996) tried to minimise the computational run time of the above approach by using additional constraints. The constraints used were as follows:

1. Terms were sorted in descending order of their Retrieval Selection Value (RSV) (Robertson, 1991).

2. The maximum number of terms in the selection process was 200.

3. Terms were added or removed one-by-one in a specified computational runtime limit.

This technique has been applied on TREC Disk 1, 2 and 3 with 50 queries. Robertson et al. compare their technique against Okapi-BM25 approach using average precision, precision at top-5, top-30 and top-100 document retrieved and recall evaluation metrics. From their results, their proposed technique outperformed Okapi-BM25 for all evaluation metrics. The computational runtime of this technique was 34 minutes per 48 documents used. However, there is no show for the computational runtime comparison between Okapi-BM25 and the proposed approach.

Simulated Annealing Using Okapi Approach

The hill-climbing technique is usually stuck in local optima solutions and consuming too much runtime for jumping from local optimum solution to global one. Several remedies have been proposed for this issue and Simulated Annealing (SA) is one of these proposed techniques. SA is a local search technique that is similar to hill-climbing but with accepting a worst solution in the evolving iterations under specific circumstances. The SA technique inspired from metal annealing process in Physics (Talbi, 2009). This technique has been used for optimising query representations using term and weighting selection (Walker et al., 1997). The term-weighting function used in document and query representations is Okapi. This approach employs two methods: a simple SA and a mild SA. In simple SA, the query representations are optimised by searching for the most appropriate query representations. In this approach, the query representations with lower average precision values were accepted under a certain temperature (T). This procedure is generally used for adapting the term selection procedure from local optima to global optima. The acceptance criteria for the worst score (average precision) is the probability of $exp(-(best\_score - new\_score)/T)$. Unfortunately, when tested this approach produced disappointing results compared to hill-climbing approach in the previous

studies (Robertson, Walker, Hancock-Beaulieu, Jones and Gatford, 1995; Robertson et al., 1996). This is because the computational runtime was higher. It appeared that the annealing process was over-fitting the terms and there was no deterministic re-weighting process, which consumed too much runtime. A second method that has been used is a mild simulated annealing approach, in which a deterministic re-weighting process is combined with the SA mechanism. This approach gave a noticeable improvement in IR effectiveness. Walker et al. compare both simple and mild SA using average precision, precision at top-5, top-10, top-15, top-20 and top-30 document retrieved and recall evaluation metrics. They applied their experiments on TREC-5 routing test collection. The mild SA outperformed simple SA in all cases. There is no show for computational runtime in their paper.

Genetic Algorithm for Term and Weight Selection

In this approach, Genetic Algorithm (GA) is used to adapt an optimised query representation. Several research efforts have been introduced to solve this problem using GA (Cordon et al., 2003). (Radwan et al., 2006) proposed a new fitness function for evolving better query optimisation, which involved minimising the difference between the query vectors and their corresponding relevant documents. It is also maximising the difference between the query vectors and their top-30 irrelevant documents retrieved from VSM model based on TF-IDF weighting scheme. The results were compared with non-evolving IR approaches and the GA that used cosine similarity as a fitness function. Findings showed that the new fitness function outperformed the GA approach using cosine similarity as a fitness function. The main features of this investigation were as follows:

- Three test collections (CISI, CACM and NPL) were used to demonstrate the results.
- The selection mechanism was a roulette wheel selection.
- The probabilities of crossover and mutation were $P_c = 0.8$ and $P_m = 0.7$
- The approach was a mixture of learning the most appropriate weight and term selection methods.

In this approach, there is no show comparison between the runtime of the proposed approach.

4.1.3 Similarity Matching Functions Problem

In this research field, EC has been used to evolve better functions for measuring the similarity between documents and queries. These evolved functions are used to retrieve the relevant documents at the top of the retrieved documents. In other words, they have one purpose, to retrieve similar documents in terms of relevance at the top of retrieved document list. Whereas, the less relevant documents will be retrieved at the bottom of the retrieved list.

Evolving Similarity Matching Function Using Genetic Programming

(Fan et al., 1999) proposed the first approach for evolving similarity matching functions using GP. They started by using the automatic generation of GP applications to evolve a general similarity function in (Fan et al., 1999). Then, they customised their approach to be suitable for each personal user profile and the context of the search, with search keywords, context and so on (Fan et al., 2004). The main characteristics of their GP approach were as follows:

- The fitness function of their GP application was the effectiveness measure (Van Rijsbergen, 1977) given by the following equation:

$$E = 1 - \frac{1}{[\frac{\alpha}{P} + \frac{(1-\alpha)}{R}]} \tag{4.1.1}$$

  where α is a constant that expresses the degree of user preference for precision(P) or recall (R) component.

- Terminal set contained term frequency (TF), document frequency (DF), inverse document frequency (IDF), maximum term frequency in the document (TF_{max}), average term frequency (TF_{avg}) and random constants.

- Function set contained +, -, *, /, log and sqrt.

Unfortunately, the evolved similarity matching functions were not presented in their paper. Furthermore, these evolved functions cannot be generalised for other document collections. This is because the evolved function is a collection-based similarity matching function. Thus, it depends on the relevance judgement of a specific dataset and on the dataset characteristics. Evolving similarity matching functions using GP has a similar limitation to the other EC applications for automatic indexing through relevance judgement. It has a weakness of random IR effectiveness with queries that have different index terms than the trained ones. The reason for this is that the fitness functions do not cover the whole term space because all test collections are partially judged.

4.2 Learning to Rank Based on Feature Vector Model (FVM)

The most common issue in IR research is ranking the retrieved documents responding to the user query with regard to their relevance. In early IR research, the unsupervised TVM techniques such as VSM based on TF-IDF or Okapi-BM25 and language models were used (Manning et al., 2008). These models were used to rank the retrieved documents based on their matching similarity to user queries. However, using only one scoring method (TWS) in IR systems was not effective enough for effective IR systems. The reason is that the scoring methods such as Okapi-BM25 and various language models are limited to the relevance judgement in terms of retrieving accurate search results (Tonon et al., 2015; Urbano, 2016). This highlights the need for using more than one TWS method for ranking the documents with respect to the user queries. In addition, the importance of the documents on the web and the host server, among other desirable features, should be considered to rank the documents. Recently, (Qin et al., 2010) proposed a new trend of research into ranking documents by producing LETOR datasets. These datasets are distilled benchmarks from search engines and from the well-known TREC conference collections. These benchmarks contain more than one term-weighting scheme (scoring methods) 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 these datasets were mapped into fully judged query-document pairs for Learning to

Rank (LTR) research problems. The scoring method features and the other desirable features in these datasets are tuned and optimised by EML techniques in LTR research to produce efficient ranking models for effective IR systems.

There are three categories of LTR approaches (Liu, 2009): (1) the pointwise method, (2) the pairwise method and (3) the listwise method. These categories are based on the loss function or fitness function measurements. The pointwise approach views each single object (query-document pair) as the learning instance. Examples of pointwise approaches are Linear Regression (LR) (Yan and Su, 2009), Boosting (Freund et al., 2003), Gradient Boosted Regression Trees (GBRT or MART) (Friedman, 2001; Mohan et al., 2011) and Random Forest (RF) (Breiman, 2001). The pairwise approach views the pair of objects (two query-document pairs for the same query) as the learning instance. Examples of the pairwise approaches are RankNET (Rank Neural Net) (Burges et al., 2005), RankBoost and SVMRank (Rank Support Vector Machine) (Li, 2014). The listwise approach takes the entire retrieved list of objects (the list of query-document pairs for each query) as the learning instance. Examples of the listwise approaches are ListNET (Listwise Neural Net) (Cao et al., 2007), RankGP (Lin et al., July, 2012; Mick, 2016), Coordinate Ascent (Metzler and Croft, 2007), AdaRank (Xu and Li, 2007) and RankGPES (Islam, 2013). The proposed ES-Rank method described later in Chapter 7 is a listwise approach because this type has been shown to perform better than pointwise and pairwise approaches (Cao et al., 2007).

Although listwise methods have been shown to perform better regarding accuracy than point-wise and pair-wise approaches (Cao et al., 2007), the need to improve the performance of LTR approaches has motivated researchers to propose hybrid methods as well. For example, Sculley proposed an approach (CoRR) combining linear regression (point-wise) with support vector machine (pair-wise) (Sculley, 2010). That approach is implemented in the Sofia-ml package and while it executes in reasonable computational time, its performance in terms of NDCG and MAP is limited. In order to achieve better NDCG, Mohan et al. proposed a hybrid machine learning approach for initialising GBRT using Random Forest (Mohan et al., 2011). However, experiments showed that their

approach consumes too much run-time compared to other approaches from the literature (Dang, 2016; Li, 2014). Two other hybrid approaches are LambdaRank and LambdaMART which combine pair-wise with list-wise methods (Burges, 2010). LambdaRank is based on RankNET while LambdaMART is the boosted tree from LambdaRank. Both LambdaMART and LambdaRank have shown better performance regarding IR accuracy than the method by Mohan et. al. on the Yahoo! LTR Challenge (Chapelle and Chang, 2011). Thus, the combination of listwise and pointwise techniques has shown to be promising. Muahmmed and Carman conducted experiments combining listwise with pointwise Random Forest (Hybrid RF) showing that the their hybrid outperformed other both pointwise and listwise RF in computational run-time and accuracy (Ibrahim and Carman, 2016). Most of the LTR approaches still have some limitation on the computational run-time or the achieved accuracy of the predictive results. In Chapter 7, we proposed two hybrid methods by initialising ES-Rank with LR (pointwise) and SVMRank (pairwise) for achieving better predictive accuracy.

The previous approaches relate to offline LTR based on FVM. (Schuth et al., 2013) proposed a new research trend in LTR based on the simulation of user click models. This research trend is called online LTR and it simulates online LTR in search engines. However, this research is based on offline relevance labels existing in the LTR datasets to check the quality of the proposed LTR models. Thus, online LTR techniques can be implemented for offline LTR techniques by adding user click simulation to these techniques. However, offline LTR techniques learn explicitly from the relevance labels and online LTR techniques learn implicitly from the relevance labels. Chapter 8 provides details about one of the click models in online LTR Lerot package and how to optimise its LTR models using ES-Rank technique.

Most of the offline and online LTR techniques are based on sampling methods to check the quality of the proposed LTR model in each learning/evolving iteration. The sampling methods are used to pick up samples of the training instances (query-document pairs) from the training set rather than taking the whole training set instances in each learning iteration. However, sampling methods such as bootstrap Bagging or Boosting cause over-

fitting and under-fitting problems. The proposed ES-Rank evolves better ranking models with smooth fitting and better performance regarding run-time and accuracy. In addition, ES-Rank has the lowest problem size compared to other evolutionary techniques because it is (1+1)-Evolutionary Strategy approach. The following sections provide details about LTR methods based on FVM.

4.2.1 SVMRank: Support Vector Machine for Ranking

(Joachims, 2016) proposed a pairwise approach for LTR based on a Support Vector Machine, called SVMRank. The approach compares every two query-document pairs in order to rank them in a retrieved query-document pair list. This approach uses the error rate between the actual ranking and the ranking from its model as a loss function. The objective of the SVMRank technique is to minimise the loss function value between the actual relevance labels and the ranking model labels on the training dataset. This approach produces a linear ranking model of weights. Assuming the vector of weights that are adjusted by the SVMRank technique is $\overrightarrow{w}$. The ranking model is represented by $f_{\overrightarrow{w}}(q)$, where q is the query set of the training data. The ranking of two documents d_i and d_j that have query-document pairs $\Phi(q, d_i)$ and $\Phi(q, d_j)$ can be represented by:

$$(d_i, d_j) \, \epsilon \, f_{\overrightarrow{w}}(q) \Leftrightarrow \overrightarrow{w}\Phi(q, d_i) > \overrightarrow{w}\Phi(q, d_j) \tag{4.2.2}$$

If the training set contains n queries, the target of the SVMRank is to find the weight vector $\overrightarrow{w}$ that maximises the number of the fulfilled inequalities in:

$$\begin{aligned} (d_i, d_j) \, \epsilon \, r_1^* : \overrightarrow{w}\Phi(q_1, d_i) > \overrightarrow{w}\Phi(q_1, d_j) \\ ..... \\ ..... \\ (d_i, d_j) \, \epsilon \, r_n^* : \overrightarrow{w}\Phi(q_n, d_i) > \overrightarrow{w}\Phi(q_n, d_j) \end{aligned} \tag{4.2.3}$$

This direct generalisation in equation 4.2.3 for equation 4.2.2 shows that this problem is complex (NP-hard) problem to solve. However, it can be simplified based on the

classification problem using SVM. Thus, the optimisation problem of SVMRank can be represented as follows:

$$minimise: \quad V(\overrightarrow{w}, \overrightarrow{\xi}) = \frac{1}{2}\overrightarrow{w} \cdot \overrightarrow{w} + C \sum \xi_{i,j,k} \tag{4.2.4}$$

subject to:

$$\begin{aligned}
(d_i, d_j)\, \epsilon\, r_1^* : \overrightarrow{w}\Phi(q_1, d_i) &\geq \overrightarrow{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\
&\ldots\ldots \\
&\ldots\ldots \\
(d_i, d_j)\, \epsilon\, r_n^* : \overrightarrow{w}\Phi(q_n, d_i) &\geq \overrightarrow{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \\
\forall i, \forall j\ and\ \forall k : \xi_{i,j,k} &\geq 0
\end{aligned} \tag{4.2.5}$$

where C is a constant that adjusts the margin size against the training error and $\xi_{i,j,k}$ is the slack variable. Thus, the problem is to minimise the upper bound of $\sum \xi_{i,j,k}$. This problem is a convex problem that has no local optima. For clarifying, constraints in equation 4.2.5 can be re-arranged as:

$$\overrightarrow{w}(\Phi(q_k, d_i) - \Phi(q_k, d_j)) \geq 1 - \xi_{i,j,k}, \tag{4.2.6}$$

In the beginning, Joachims proposed a support vector machine called svmlight library package (Joachims, 2015). However, this package was slower than other LTR techniques. Thus, he proposed a new library package for ranking called SVMRank (Joachims, 2016). SVMRank package is faster because it does not include all query-document pairs of the training set in each learning iteration. Another issue of SVMRank is that the range of the relevance label for each query-document pair is 0 or 1 or 2. Therefore, a large dataset such as LETOR MSLR-WEB10K can not be tested with this method. In the literature, SVM-Rank and SVM-light did not compared to other LTR techniques in terms of computational runtime. In Chapter 7, SVMRank and SVM-light (in MSLR-WEB10K as SVMRank) was in comparison with ES-Rank and other LTR in terms of accuracy and computational runtime.

4.2.2 RankBoost: An Efficient Boosting Algorithm for Combining Preferences

The RankBoost approach (Freund et al., 2003) is the extension of the Adaptive Boosting (AdaBoost) (Le and Smola, 2007) approach for the classification of LTR in IR. The RankBoost is a pairwise approach in which the loss function is an exponential loss between every two query-document pairs of the learning sample. This learning technique combines many weak rankers of the given training set to produce a strong learning ranking model. Each weak ranker $h_t(x)$ is a linear ranking model of vector weights, while the final strong ranking model is $H(x) = \sum_t \alpha_t h_t(x)$. The symbol x represents the training query-document pairs set, while t is the number of weak rankers $h_t(x)$ used to produce $H(x)$. On the other hand, the parameter α_t represents the importance weight value of the weak $h_t(x)$ in $H(x)$. In the RankBoost approach, the exponential pairwise loss function is used to measure the quality of the proposed $h_t(x)$. The calculation of the parameter α_t is based on a distribution D_t and the initial distribution value of D_1 has a value of one. There are three methods for computing α_t in each learning iteration. These methods are based on minimising the exponential pairwise loss function. The first method used a simple linear search for assigning α_t values, the second method assumes the ranking model produces one of two values either are zero or one. The third method assumes that the loss function has a real value of between zero and one. As with most LTR techniques, RankBoost is based on sampling technique to check the quality of weak ranker rather using the whole training set in each learning iteration. Thus, there is a drawback regarding accuracy for LTR models produced by this approach. Furthermore, this technique was not compared before with other LTR approaches regarding computational runtime versus accuracy obtained by it. More details about RankBoost can be found in (Freund et al., 2003) and RankBoost was implemented in the RankLib package (Dang, 2016).

4.2.3 RankNET: Learning to Rank Using Neural Nets

RankNET was proposed by Burges et al. (Burges et al., 2005; Burges, 2010) and it is a pairwise approach. The technique uses the neural network combined with gradient descent steps. These gradient descent steps are used to control the learning rate in each

iteration. RankNET is an extension of a back-propagation neural network with probabilistic loss function that can handle pairs of instances (query-document pairs). This means that the loss function relies on two query-document pairs for measuring the quality of the proposed ranking model in each iteration. RankNET has two hidden layers and it uses backpropagation to minimise the pairwise loss function. Given two query-document pairs d_i and d_j associated with the training query q, the target probability of the ranking sequence (d_i, d_j) is $\overline{P}_{d_i,d_j}$. The value of $\overline{P}_{d_i,d_j}$ is calculated based on the ground truth labels of d_i and d_j. For example, $\overline{P}_{d_i,d_j} = 1$, if the difference between the ground truth labels $y_{d_i,d_j} = 1$, while $\overline{P}_{d_i,d_j} = 0$, otherwise. The modelled probability P_{d_i,d_j} is calculated based on the difference between the scores of these query-document pairs as follows:

$$P_{d_i,d_j} = \frac{exp(\overrightarrow{w}\Phi(d_i) - \overrightarrow{w}\Phi(d_j))}{1 + exp(\overrightarrow{w}\Phi(d_i) - \overrightarrow{w}\Phi(d_j))} \tag{4.2.7}$$

Then, the loss function is represented by the cross entropy between the target probability and the modelled probability can be calculated by:

$$Loss_{d_i,d_j} = -\overline{P}_{d_i,d_j} log(P_{d_i,d_j}) - (1 - \overline{P}_{d_i,d_j}) log(1 - P_{d_i,d_j}) \tag{4.2.8}$$

The Backpropagation neural network uses the loss function to learn the ranking model. Burges et al. did not provide the dataset that was used in their experiments to be available for other researchers. In addition, their comparison was based only on one evaluation metric which is NDCG with its computational runtime. The comparison was made against other gradient descent methods which are PRank and RankProp techniques. In this comparison, one layer and two layer versions of RankNET outperformed PRank and RankProp techniques regarding NDCG values, but they were slower than linear Prank and RankProp. There is no extensive comparison for this technique with recent LTR techniques in the literature. RankNET is similar to other pairwise techniques that checks the quality of the ranking model for each two query-document pairs separately in each learning iteration rather than the whole retrieved list. Thus, this technique has a drawback in terms of evaluation metric values on datasets that have multiple relevance labels than binary relevance labels. Further details about this approach can be found in (Burges et al., 2005; Burges, 2010). The method has been implemented in the RankLib package (Dang,

2016).

4.2.4 ListNET: Listwise Learning to Rank Based on Neural Nets

ListNET is a listwise and probabilistic technique for LTR proposed by Cao et al. (Cao et al., 2007). This technique is different than RankNET in its way for calculating the loss function. The loss function in ListNET is a listwise loss function. This technique is based on the probability distribution in the ranking list of the query-document pairs. Suppose that each query i has the instance (X_i, y_i), where X_i is the feature vector of the query-document pair and y_i is the ground truth label. Then, the training data that contains N queries is given as $S = \{(X_i, y_i)\}_{i=1}^{N}$. The ListNET technique is used to create a ranking model which has a vector of weight $W = (w_1, ..., w_M)$, while M is the number of features in the training data. The ranking model function can be represented by $F(X, W)$. The ListNET calculates the KL Divergence probability of all training query-document pairs as the total loss function value. Then, it attempts to minimise the total loss by updating the learning ranking model weights. The total loss function is given by:

$$L(w) = \sum_{i=1}^{m} L(y_i, F(x_i, W)) \tag{4.2.9}$$

Here $L(y_i, F(X_i, W))$ is the cross-entropy loss function for each query. This loss function for top-K query-document pairs for query i is given by:

$$L(y_i, F(X_i, W)) = - \sum_{y_{i,j} \epsilon GroundTruth_j^K} \prod_{j=1}^{K} \frac{exp(y_{i,j})}{\sum_{L=j}^{n_i} exp(y_{i,L})} \cdot log \prod_{j=1}^{K} \frac{exp(F(X_{i,j}, W))}{\sum_{L=j}^{n_i} exp(F(X_{i,L}, W))} \tag{4.2.10}$$

where n_i is the number of query-document pairs for each query i. The ListNET updates the ranking model weight vector in each learning iteration for better accuracy by $W = W - \eta \bigtriangledown L(W)$, where η is a learning rate parameter that can be chosen in the training time and $\bigtriangledown L(W)$ is the Gradient of the total loss. Cao et al. (Cao et al., 2007) compared their technique with RankBoost, RankNET and RankSVM using NDCG and

MAP evaluation metrics. They argued that ListNET outperformed RankBoost, RankNET and RankSVM on LETOR2 (Ohsumed, TD2003 and TD2004) benchmarks (Qin et al., 2010). However, they did not mention the parameter settings of each technique nor the training computational runtime of each technique. This method has been implemented in the RankLib package (Dang, 2016).

4.2.5 AdaRank: A Boosting Algorithm for Information Retrieval

The AdaRank technique is a listwise approach based on Adaptive Boosting (Ada-Boost) in text classification (Xu and Li, 2007). The main difference between RankBoost and AdaRank is their loss functions. In the RankBoost technique the loss function is an exponential pairwise loss function, while the loss function in AdaRank is an exponential listwise loss function. Similar to RankBoost, AdaRank combines the linear weak rankers $h_t(x)$ to produce an effective ranker model $H(x) = \sum_t \alpha_t h_t(x)$. The symbol x represents the training query-document pairs set, while t is the number of weak rankers $h_t(x)$ used to produce $H(x)$.

On the other hand, the parameter α_t represents the importance weight value of the weak rankers $h_t(x)$ in $H(x)$. In the learning procedure, the AdaRank repeats the process of re-weighting the training samples to create each weak ranker. Then, it calculates the weight (the importance) for the weak ranker in the learning ranking model. Furthermore, the AdaRak technique is used to optimise an exponential loss function based on the IR evaluation metrics such as MAP, NDCG, Error Rate, Reciprocal Rank, Precision. The exponential loss function is the upper bound of the normal loss function based on the evaluation metrics. In each learning iteration, AdaRank maintains a weight distribution over the training set. This distribution is used to identify the importance of each weak ranker in the ranking model. Xu and Li (Xu and Li, 2007) compared their approach with Okapi-BM25, SVMRank and RankBoost approaches on four benchmarks: Ohsumed, WSJ, AP and .GOV datasets. The AdaRank outperformed these approaches on these datasets using MAP and NDCG evaluation fitness metrics. However, AdaRank has not been tested using more fitness evaluation metrics nor state-of-the-art large LETOR datasets. They also did not mention the parameter settings of each technique nor the training computational

runtime of each technique. Furthermore, the implementation of AdaRank and other LTR approaches does not consider the whole training instances (query-document pairs) in each learning iteration to check the quality of the proposed solution, which causes a drawback in the evaluation values (accuracy) of AdaRank. More details about this approach can be found in (Xu and Li, 2007) and the technique has been implemented in the RankLib Package (Dang, 2016).

4.2.6 RankGP: Learning to Rank Using Genetic Programming

Yeh et al. proposed a new research trend for evolving ranking function using Genetic Programming called RankGP (Yeh et al., 2007; Mick, 2016). This approach is a listwise LTR approach. They used LETOR2 (TD2003 and TD2004) benchmarks (Qin et al., 2010). Their approach outperformed the traditional, probabilistic and machine learning ranking functions (BM25, RankBoost and SVMRank) in terms of the IR system effectiveness. The system effectiveness was measured by three IR evaluation measures which are Precision of each top-10 query-document pair retrieved, Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 2011) and Normalised Discounted Cumulative Gain (NDGG) (Jarvelin and Kekalainen, October 2002).

However, the computational cost of their approach was high in comparison with other approaches. It cost approximately 35 hours to learn a better evolved ranking function for the TD2003 benchmark. The equipment used for these experiments was a 1.8 GHz Intel Core 2 CPU and 2GB memory PC. The main characteristics of this approach were as follows:

1. Before applying the GP approach, all the features existing on the trained and validation subsets were normalised into values between 0 and 1.

2. This approach used Layered Genetic Programming (Lin et al., 2007*a*, July, 2012; Mick, 2016) with ramped half-and-half for creating the initial population for the proposed function with a maximum depth of 8 terminals and operators.

3. The function set contained {+, -, *} and the division was neglected to evolving linear solutions with less computational cost. The terminal set contained all bench-

mark features (44 features) and 44 constant values between 0 to 1. In addition, the fitness function was *Mean Average Precision (MAP)* for all queries.

4. The crossover rate, mutation rate, and the number of generations and reproductions were set according to (Lin et al., 2007*a*). Furthermore, the mutation rate was the adaptive mutation rate tuning AMRT (Lin et al., 2007*a*, July, 2012).

The limitations of this approach and all learning to rank approaches using EC referenced in the literature are as follows:

- The computational runtime is higher than for other machine learning applications as mentioned by Yeh et al. (Yeh et al., 2007). In addition, this technique requires a large problem size to represent a population of the proposed solutions in each evolving iteration compared to (1+1)-Evolutionary Algorithms (ES-Rank technique).
- The state-of-the-art machine learning techniques outperformed this approach in terms of NDCG and MAP metrics theoretically from the results recorded in the literature papers and documented in (Tax et al., 2015). However, there is no practical comparison between the state-of-the-art LTR techniques and RankGP on the same datasets that considers the computational runtimes and the accuracy values.

This technique has been implemented in the LAGEP Package (Mick, 2016).

4.2.7 LambdaRank

Burges et al. proposed the LambdaRank technique, which is based on the RankNET technique (Burges et al., 2006). The LambdaRank is a pairwise technique that utilises the minimisation of the surrogate loss function which is equal to $L(W) = -\lambda$, where λ is based on the Normalised Discounted Cumulative Gain (NDCG) of the training query-document pairs on each learning iteration. The λ parameter is equal to $\sum_{j=1}^{K}(\frac{1}{1+exp(s_i-s_j)} * (NDCG_i - NDCG_j))$, where K is the number of query-document pairs in the retrieved truncated ranking list and the parameters S_i and S_j are the score rankers for documents i and j. Suppose the gradient of the loss function is $\bigtriangledown L(W)$, then, the LambdaRank updates the ranking model weight vector in each learning iteration for better accuracy, through $W = W - \eta \bigtriangledown L(W)$, where η is a learning rate parameter that can be chosen in

the training time. The experiment settings for Burges et al. technique were 1 layer and 2 layers (with 10 hidden nodes) nets experiments and they run on a 2.2GHz 32 bit Opteron machine. Burges et al. only compared this technique with the RankNET technique and LambdaRank outperformed it in terms of NDCG and runtime across speedup procedures. However, the detailed characteristics and its source (feature type and the name of the search engine) of the dataset did not state in his paper. In addition, there is no show for the total runtime of the technique on the dataset.

4.2.8 Random Forest

Random Forests (RF) (Breiman, 2001; Ganjisaffar et al., 2011) is a pointwise LTR approach that combines decision tree rankers and determines their average, in order to produce a strong ranking model. The RF technique is an ensemble method that utilises rankers based on bagging and sampling features. Bagging refers to the procedure of combining multiple decision trees and calculating their average. The technique takes a sample of the training set and then uses randomly chosen features to build a decision regression tree as a ranker. This procedure is repeated M times, which is the number of bagging. Then, the ranking model is the average value of the rankers produced by the decision trees. The random sampling of the features adds an additional control to the variance of the bagging. Recently, (Ibrahim and Carman, 2016) extended RF from pointwise to listwise. The computational complexity of listwise RF was higher than the pointwise RF. The lowest training time of listwise RF per tree for 30% of MSLR-WEB10K queries was 137 minutes. Thus, Ibrahim and Carman proposed a hybrid pointwise-listwise RF to overcome the computational complexity on large datasets. They compared the listwise, the pointwise and the hybrid RF. The results show that hybrid RF outperformed both pointwise and listwise RF in terms of NDCG@10, MAP and Error rate. They also compared hybrid RF with MART, Coordinate Ascent, RankSVM, AdaRank, RankBoost, LambdaMART techniques on MSLR-WEB10K (Microsoft Bing search engine) and Yahoo datasets. The comparison shows that LambdaMART outperformed other techniques, while hybrid pointwise-listwise RF was the second best performance technique. However, the comparison did not include the computational runtime of each technique. The RF was developed in rt-rank and ranklib packages (Mohan et al., 2011; Dang, 2016).

4.2.9 Gradient Boosted Regression Tree (MART or GBRT)

Gradient Boosted Regression Tree (MART or GBRT) technique is another pointwise LTR technique. GBRT combines the boosted technique with random regression trees from the sampled features. This technique was first proposed in data mining (Friedman, 2001), then it was developed in rt-rank and ranklib packages (Mohan et al., 2011; Dang, 2016). The loss function of this technique is the RMSE values. Similar to the RankBoost, this technique combines the weak ranker to produce a strong ranker. It starts with initial ranker. Then, it uses the gradient of RMSE to produce the following rankers. The difference between RankBoost and GBRT is that each ranker in GBRT is produced from random regression tree. The algorithm and the details of this technique are shown in (Friedman, 2001; Mohan, 2010). The computational runtime has not been investigated in the literature on LTR datasets for that technique. Furthermore, there is no extensive comparison for this technique with other LTR techniques for various evaluation metrics.

4.2.10 LambdaMART

LambdaMART is the extended listwise version for the GBRT technique. The difference between GBRT or MART and LambdaMART is the loss function. The loss function of GBRT is the RMSE or the Error Square value, while the loss function for LambdaMART is the negative NDCG value. In LambdaMART and GBRT techniques, the boosted method based on the random regression tree of the samples is employed. The details of this technique are presented in (Li, 2014; Burges, 2010). LambdaMART is a powerful LTR technique that won the Yahoo Challenge for LTR problems (Chapelle and Chang, 2011). It won Yahoo challenge for LTR techniques in terms of IR effectiveness (Reciprocal Rank) (Chapelle and Chang, 2011). The other LTR techniques in this challenge were IGBRT, AdaBoost and YetiRank. However, the performance of LambdaMART in terms of computational runtime has not been investigated in the literature. LambdaMART is also one of the most effective LTR applications after Coordinate Ascent in terms of effectiveness in Ranklib package (Dang, 2016).

4.2.11 Coordinate Ascent

Coordinate Ascent (CA) is a local search technique in which the learning model may reach to the global maxima if the fitness evaluation function is concave (Metzler and Croft, 2007). Metzler and Croft proposed this technique to produce a linear ranking model. Their approach uses multinomial manifolds to propose the weights of the ranking model. The multinomial manifold is a parameter space from a multinomial distribution. The summation of the proposed weights of the ranking model on the training set features is one. The benefit of multinomial manifold is the ability to converge the solutions without repeating similar ranking models that have the same evaluation values. In other words, if $W_{11}, ..., W_{1m}$ is the current ranking model and its fitness evaluation value is E_1, the next proposed learning ranking model must have weights $W_{21}, ..., W_{2m}$ with different fitness evaluation value than E_1. This is due to the properties of the multinomial manifold parameter values. However, this approach consumes too much time in large datasets. In addition, the CA cannot guarantee the optimal solution for every fitness function. Metzler and Croft only compared this technique with SVMRank and Language Model without considering the computational time of each technique in the comparison. In that comparison, Coordinate Ascent outperformed SVMRank and Language Model regarding MAP evaluation metric on various TREC document collections (TREC Disk 1 to 5, .Gov2, WSJ, WT10g and AP). (Dang, 2016) implemented this technique in Ranklib library package. Further details of this approach can be found in (Metzler and Croft, 2007).

4.2.12 CoRR: Combined Regression with Ranking

Schulley proposed a new technique for improving the performance of Support Vector Machine (SVM) using regression (Sculley, 2010). This approach uses the stochastic subgradient descent SVM solver (SGD SVM) with linear regression optimisation function. The reason for using SGD SVM is its performance compared to the other three SVM approaches (Sculley and Inc, 2009). The regression CoRR optimisation function is given by Equation 4.2.11. In this equation, the parameter $\alpha \epsilon [0, 1]$ is the trade-off between the optimising by regression loss and optimising pairwise loss. In the case of using $\alpha = 1$, the equation recovers only the standard regression, while the equation recovers only pairwise

ranking when $\alpha = 0$. The D parameter represents the training set data, while P represents the two query-document pairs, λ is the learning parameter and L(.,.) is the loss function. Equation 4.2.11 is used in the Stochastic Gradient Step to modify the weight of the LTR ranking model in each iteration.

$$\min_{w \,\epsilon\, \mathbb{R}} \alpha L(w, D) + (1 - \alpha) L(w, P) + \frac{\lambda}{2} ||w||_2^2 \tag{4.2.11}$$

Schulley used two loss functions in his technique; the squared loss function and the logistic loss function. He assessed the accuracy of his approach by using three evaluation metrics (Mean Square Error (RMSE), Mean Average Precision (MAP) and Normalised Discounted Cumulative Gain (NDCG)). The LETOR MQ2007 and MQ2008 have been used to compare the accuracy of this approach with SVMRank. The results indicated that Ranking by assigning $\alpha = 0$ in Equation 4.2.11 only outperformed the Regression (for $\alpha = 1$), SVMRank and CoRR techniques in the two datasets in MAP and NDCG, while his Regression approach outperforms SVMRank, Ranking and CoRR in RMSE. That approach is implemented in the Sofia-ml package and while it executes in reasonable computational time, its performance in terms of NDCG and MAP is limited. This approach has been compared with RankSVM, while there are numerous available LTR packages that implement various EML techniques in LTR. Moreover, the pairwise techniques do not consider the whole list of query-document pairs when ranking the documents. Thus, pairwise approaches have limitations for multiple relevance label levels, but are more appropriate for binary relevance labels. The details of CoRR approaches are demonstrated in (Sculley, 2010).

4.2.13 RankDE: Learning to Rank Using Differential Evolution (DE)

In this approach, Differential Evolution is used to learn the most appropriate weights for each feature existing in the LETOR dataset (Bollegala et al., 2011). This approach outperformed Okapi-BM25, SwarmRank, RankBoost, SVMRank and RankGP techniques in terms of precision, NDCG and MAP using LETOR TD2003 and TD2004 datasets (Qin et al., 2010). The evaluation measures used were three effectiveness measures which were also used in the RankGP approach. Some of other characteristics of that approach are as

follows:

- Feature weight vectors are evolved as real number values.
- The chromosome dimension space has 44 genes. The initial population of these genes in each chromosome has a value of between [-1,1].
- Similar to the RankGP approach (Yeh et al., 2007), the fitness function is the *Mean Average Precision (MAP)*.

However, the computational runtime comparison between this technique and other techniques has not been presented. It may consume less runtime than the RankGP approach, but it may consume more runtime than (1+1)-Evolutionary Algorithms. The parameter settings such as probability of mutation and crossover have not been presented in their paper.

4.2.14 Linear Regression

The *Linear Regression (LR)* technique was introduced in the Ranklib library package (Dang, 2016), but there is no paper discussing its usefulness compared to other LTR techniques. The method used in Ranklib is the least square LR technique (Miller, 2006). In this method, the ranking model weight vector is chosen based on minimising the total distance between the ground truth labels of the training query-document pairs and the labels produced by ranking the ranking model. The ranking model produced by LR technique has the objective to minimise $loss = \frac{1}{N}\sum_{j=1}^{N}|y_j - \sum_{i=1}^{n}(w_i x_{ij})|$, where N is the number of query-document pairs in the training set, n is the number of features in each query-document pairs, w_i is the weight for feature i in the ranking model proposed by LR and x_{ij} is the feature value for feature i in query-document pair j. Finally, y_j is the ground truth label for query-document pair j. From our findings, the LR technique in Ranklib is the fastest approach, but it is not the most effective one in Ranklib package.

4.2.15 Initialised GBRT using Random Forests (IGBRT)

This application is a hybrid pointwise LTR technique (Mohan et al., 2011). It uses the ranking model produced by the pointwise RF technique in the initialisation procedure of

the GBRT technique. Mohan (Mohan et al., 2011) developed this technique with pointwise GBRT and pointwise RF in rt-rank package (Mohan et al., 2011). This technique has been introduced in Yahoo challenge for LTR problems (Chapelle and Chang, 2011). It was in comparison with LambdaMART, AdaBoost and YetiRankand approaches. The LambdaMART outperformed IGBRT technique regrading reciprocal rank evaluation values in this challenge. This comparison did not consider the computational time of each technique. However, IGBRT outperformed GBRT and RF in terms of receiprocal rank. From our findings, IGBRT technique also consumes considerable computational runtime for large LETOR datasets.

4.3 Chapter Summary

In this chapter, the related works of *Evolutionary Computation (EC)* and *Machine Learning (ML)* in TVM and FVM are presented. The chapter started with the TVM and introduced the limitation of applying EC and ML in TVM for evolving term weights. The limitation is summarised in the problem size and the computational run-time for evolving the whole document representations. Besides to the relevance judgement limitation for applying EC or ML techniques at the beginning of establishing a new IR system. These cause the need for proposing a new methodology for evolving document representations using EML techniques. This Chapter also introduces the various LTR techniques on FVM. In this research domain, the EML techniques are used to learn the most suitable ranking model weight for the training data and testing the performance of the techniques on the test data. The literature research was not stated the experimental settings nor the training computational run-time in their comparison. Furthermore, most of the EML techniques give high accuracy more than other heuristic techniques. The following chapter presents a new TWS called TF-ATO and it shows the heuristic issues caused by the pre-processing procedure in IR.

Chapter 5

Term Frequency With Average Term Occurrences (TF-ATO)

5.1 Introduction

In the context of Information Retrieval (IR) from textual documents, the term-weighting scheme (TWS) is a key component of the matching mechanism when using the TVM (Term Vector Model) representation. At the beginning of establishing an IR system, there is a need for using a non-learning (mathematical) term-weighting scheme (TWS). This is because there are no relevance judgement values provided by the users for the IR test collection. The preferable non-learning TWS is the term-weighting function that considers most of non-noisy document words as index terms and assigns a discriminate weight value for it. As discussed in Section 2.2.2, an effective TWS is crucial to make an IR system more efficient. There are various TWS proposed in the literature and some have been implemented in search engines. To the best of our knowledge, the most widely used approach is the term frequency-inverse document frequency (TF-IDF) as a non-learning TWS. However, TF-IDF and its variations may remove some significant keywords before user relevance feedback is gathered by the IR systems. This may cause the bias of relevance judgement values based on the used TWS in the IR system (Buckley et al., 2007; Urbano, 2016).

An analysis of commonly used test collections for evolving TWS and term weights is

demonstrated in subsection 3.1.3. This analysis shows that test collections are not fully judged as achieving that is expensive and may be unfeasible for large collections at the early stage of establishing the IR system. A test collection being fully judged means that every document in the collection acts as a relevant document to a specific query or a group of queries. Some Evolutionary Computation (EC) techniques have been used for evolving TWS or evolving term weights using those test collections (Cummins, 2008; Cordon et al., 2003). However, such approaches have an important drawback. These EC approaches usually use the relevance judgements for the test collection on their fitness functions for checking the quality of the proposed solutions. The relevance judgement of a collection gives the list of relevant/irrelevant documents for every query. Furthermore, the real IR test collection have not relevance judgement values at the beginning of IR systems. This means that TWS can not be evolved at the beginning of establishing IR system. This provokes that when using EC techniques most documents have random term weight representations. In addition, TWS evolved with Genetic Programming (GP) as in (Cummins and O'Riordan, 2006; Cordon et al., 2003) are based on the characteristics of the test collections and hence, not easily generalisable to be effective on collections with different characteristics.

This is what motivates the work presented in this chapter on the development of such the proposed TWS. In this work, the *Term Frequency With Average Term Occurrence (TF-ATO)* is proposed which computes the average term occurrences of terms in documents and uses a *Discriminative Approach (DA)* based on the document centroid vector to remove less significant weights from the documents. This TWS does not require any prior knowledge about relevance judgement values.

This chapter evaluates the performance of TF-ATO and investigates the effect of stop-words (or negative words) removal (Fox, 1992) and the DA as procedures for removing non-significant terms and term weights in heuristic TWSs. The performance of the proposed TF-ATO and the well-known TF-IDF approach are compared in this chapter. It is shown that using TF-ATO results in better effectiveness in both static and dynamic test collections. In addition, this chapter investigates the impact that stop-words removal and our DA have on TF-IDF and TF-ATO. The results show that both,

stop-words removal and the DA, have a positive effect on both term-weighting schemes. More importantly, it is shown that using the proposed DA is beneficial for improving IR effectiveness and performance with no information in the relevance judgement for the collection. The intended contributions of this chapter are contributions 2 and 3 in section 1.4.

5.2 The Proposed Term-Weighting Scheme

How to assign appropriate weights to terms is one of the critical issues in automatic term-weighting schemes. In this section, a new TWS called *Term Frequency Average Term Occurrences (TF-ATO)* is proposed and is expressed by:

$$W_{ij} = \frac{tf_{ij}}{\#\ ATO\ in\ document\ j} \tag{5.2.1}$$

and

$$\#\ ATO\ in\ document\ j = \frac{\Sigma_{i=1}^{m_j} tf_{ij}}{m_j} \tag{5.2.2}$$

where tf_{ij} is the term frequency of term i in document j, ATO is the average term occurrences of terms in the document and is computed for each document, m_j represents the number of unique terms in the document j or in other words it is the number of index terms that exist in document j. This TWS does not require any relevance judgement values for establishing IR system. TF-ATO considers long/short document/query variation length by normalising term frequency by the average term occurrence in the term-weighting scheme. The global part of TF-IDF scheme and its variations depends on the test collection characteristics, while the proposed TF-ATO scheme considers that global weights are the same in any term weight that has a value of 1 for any existing term in the collection. The Discrimination Approach (DA) incorporated into TF-ATO uses the documents centroid as a threshold to remove less-significant weights from the documents. The DA can discriminate between the term weight representations in the documents for better document representations, while IDF discriminates between the terms themselves by removing most repeated terms between documents. Thus, if the term is significant in

a specific document or a group of documents but repeated in all documents, it will have 0 TF-IDF value in all documents regardless its importance in some documents. However, the documents, in which this term is a discriminative term for them, should have a term-weight value that indicates its importance as a discriminative term. Thus, we proposed DA to discriminate terms in term-weight level in the documents rather than IDF that discriminate the terms in term level in the collection.

5.3 Discriminative Approach (DA)

The proposed DA is a non-learning heuristic approach for improving documents representation. To the best of our knowledge, this is the first non-learning discriminative approach for improving documents representation. It is similar to the heuristic method Ide dec-hi (Salton and Buckley, 1997) for improving queries representation. However, our DA is for documents representation instead of queries. It does not require any relevance judgements information and it depends only on test collection representations. Thus, it depends on the topic domain of the test collection to remove less significant words from it. This DA can be represented by:

$$W_{ij} = \begin{cases} W_{ij} & \text{if } c_i < W_{ij} \\ 0 & \text{if } c_i \geq W_{ij} \end{cases}$$

where c_i is the weight of term i in the documents centroid vector and W_{ij} is the term weight of term i in document j as calculated with equation 5.2.1. This DA is applied to every term weight W_{ij} in every document in the collection. Assuming that M is the number of index terms in the collection and N is the number of documents existing the collection, the document centroid vector of the collection is given by:

$$C = (c_1, c_2, ..., c_i, ..., c_M) \tag{5.3.3}$$

and

$$c_i = \frac{1}{N} \Sigma_{j=1}^{N} W_{ij} \tag{5.3.4}$$

This proposed DA is somehow based on Luhn's approach (cuts-off) (Luhn, 1957) for removing non-significant words from text (see subsection 5.5.1). However, it takes into account that some *non-significant* words can become *significant* in different context according to some documents domains (Saif et al., 2014). Thus, our DA is used to remove non-significant term weights when they are non-significant compared to the centroid of the term weights, instead of removing the terms totally from the document representations.

5.4 Implementation and Experimental Study

5.4.1 Building the IR System

Information Retrieval systems manage their data resources (test collections) by processing words to extract and assign a descriptive content that is represented as index terms to documents or queries. In text documents, words are formulated with many morphological variants, even if they refer to the same concept. Therefore, the documents often undergo a pre-processing procedure before building the IR system model. The model here is based on the vector space model (VSM) as explained in Chapter 2. The following procedures are applied to each document in our IR system:

1. Lexical analysis and tokenization of text with the objective of treating punctuation, digits and the case of letters.

2. Elimination of stop-words with the objective of filtering out words that have very low discrimination value for matching and retrieval purposes.

3. Stemming of the remaining words using Porter stemmer (Jones and Willett, 1997) with the objective of removing affixes (prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms.

4. Index terms selection by determining which words or stems will be used as index terms.

5. Assign weights to each index term in each document using one given weighting scheme which gives the importance of that index term to a given document.

6. Create document vectors of term weights in the test collection space (create inverted and directed files using term weights for documents from the test collection).

7. Apply the previous steps (1-6) to queries in order to build query vectors.

8. For the proposed weighting scheme only (TF-ATO), there are two additional steps:

   - Compute the document centroid vector from document vectors by using equations (9) and (10).
   - Use the documents centroid for normalising document vectors. This can be done by removing small non-discriminative weights using the documents centroid as a threshold.

9. Matching between document vectors and each query using cosine similarity and rank them according to their cosine similarity values in descending order. Then, the precision was calculated under fixed 9-points recall values. In other words, the ranked document retrieved until one recall point of the nine recall points is reached (cut-off according nine recall points). Then, precision is calculated for the retrieved list. The precision values are for the retrieved documents for each corresponding recall value (recall point) for each query.

10. Compute the average precision values for the whole query set in 9-points recall values for the retrieved documents. Then compute the Mean Average Precision (MAP) value. The average precision under fixed 9-points recall values as evaluation metric is an accurate evaluation metric for the textual test collections compared to measuring the the precision for the whole document retrieved. This is because there are variations in the number of relevant documents per each query.

11. Repeat steps 5 to 12 for each weighting scheme tested and compare results.

The above procedure has been used for experiments with static data stream. For the case of dynamic data stream, there are two approaches. The first one is to re-compute terms weights for each document in the collection by conducting the above procedure for each update to the collection using a non-learning approach. This of course, adds extra computation cost for every data update in a dynamic data stream. The second approach

involves using IDF or the documents centroid in the next approach that is measured from the initial test collection. Then assign term weights to the new documents using the term frequency in the document multiplied by the corresponding IDF for the term that computes by the initial test collection or alternatively, use the DA. In addition to the term-weighting approach proposed here, the old documents centroid vector is used for eliminating non-discriminative term weights from the added documents. The second approach costs less in computation time but it may give lower effectiveness in terms of MAP in both the proposed TF-ATO and TF-IDF. The reason for this drawback is the variation between the actual values of IDF or documents centroid in dynamic test collection compared with the old values that are computed for the initial collection. We think that all proposed term-weighting schemes have drawbacks in their effectiveness if they do not re-compute their weighting scheme after every large update to the collection. However, this issue has not been investigated in the previous work when considering dynamic data streams as well as static ones. Thus, this work investigates the issue behind of using dynamic test collection in IR system to check its impact on IR effectiveness when using TF-IDF and TF-IDF with DA.

5.4.2 Experimental Results and Analysis

In this Section, two experiments are conducted using the overall procedure described in subsection 5.4.1. The purpose of the first experiment was to compare the average recall-precision values achieved by the proposed TF-ATO with and without the DA to the ones achieved by TF-IDF. This experiment considered the test collection as static. For this first experiment the largest test collection created by Cranfield paradigm was used which is Ohsumed and the pooling paradigm LATIMES test collection is also used in this experiment with their query sets (outlined in Table 3.3). The experiments were conducted on a PC with 3.60 GHz Intel (R) core(TM) i7-3820 CPU with 8GB RAM and the implementation was in Java NetBeans under Windows 7 Enterprise Edition. The computational run-time for indexing using TF-ATO, TF-ATO with DA and TF-IDF on Ohsumed collection are 25, 31 and 40 minutes respectively, while the indexing run-time using TF-ATO, TF-ATO with DA and TF-IDF on LATIMES collection are 24, 29 and 43

minutes.

Tables 5.1 and 5.2 present the results from the first experiment applied on Ohsumed and LATIMES. The tables show the Average Precision (AvgP) value obtained by each TWS method for nine Recall values as well as the corresponding *Mean Average Precision (MAP)* value. It is observed that the proposed weighting scheme TF-ATO gives high effectiveness compared to TF-IDF. The tables show that TF-ATO without the DA does not achieve better precision values than TF-IDF for some recall values, but when the DA is used then TF-ATO always outperforms TF-IDF for all recall values. Considering all the recall values, the average improvement in precision using Ohsumed collection (given by the MAP value) achieved by TF-ATO without DA is 6.94% while the improvement achieved by TF-ATO using the DA is 41%. Moreover, the average improvement in precision using LATIMES collection achieved by TF-ATO without DA is 2.2% while the improvement achieved by TF-ATO using the DA is 29.4%.

Table 5.1: Average Recall-Precision and MAP for Static Experiment Applied on Ohsumed collection.

Recall	**AvgP and MAP for static document experiment**		
	TF-IDF	**TF-ATO without DA**	**TF-ATO with DA**
0.1	0.648	0.713	0.816
0.2	0.445	0.47	0.61
0.3	0.343	0.361	0.472
0.4	0.253	0.259	0.362
0.5	0.216	0.196	0.288
0.6	0.176	0.153	0.24
0.7	0.156	0.13	0.199
0.8	0.136	0.114	0.154
0.9	0.123	0.108	0.13
MAP	0.277	0.278	0.364

From the results of this first experiment, it is clear that the proposed TF-ATO weighting scheme gives better effectiveness (higher average precision values) when

Table 5.2: Average Recall-Precision and MAP for Static Experiment Applied on LA-TIMES collection.

Recall	AvgP and MAP for static document experiment		
	TF-IDF	TF-ATO without DA	TF-ATO with DA
0.1	0.528	0.563	0.764
0.2	0.431	0.441	0.658
0.3	0.392	0.393	0.51
0.4	0.345	0.348	0.41
0.5	0.305	0.32	0.329
0.6	0.261	0.29	0.268
0.7	0.172	0.172	0.222
0.8	0.158	0.158	0.201
0.9	0.126	0.126	0.196
MAP	0.305	0.312	0.395

compared to TF-IDF in static test collections. Furthermore, there is an improvement by using the document centroid as a DA with the proposed weighting scheme. Moreover, the proposed DA reduces the size of the documents in the test collections by removing non-discriminative terms and less significant weights for each document. These reduction ratios are illustrated in Section 5.5.2.

The purpose of the second experiment was to investigate the average recall-precision values achieved by the proposed TF-ATO with the DA to the ones achieved by TF-IDF but now considering the test collection as dynamic. In order to conduct this experiment considering the test collection as dynamic, the given document sets in the test collections are split into parts. Then, the first part of the test collection is taken as the initial test collection to apply steps 1-8 of the procedure described in section 5.4.1. This allows to compute the index terms IDF values and document centroid vector of term-weights for the collections. The test collections are then updated by adding the other parts but without updating the index terms IDF values or document centroid vector weights computed for the initial collections. So, no recalculation is done even after adding a large number (remaining parts) of documents to the initial collections. The reason for this is

that re-computing IDF values and assigning new weights (for updating documents in the collection) would have a computational cost of $O(N^2 * M * Log(M))$, where N is the number of documents in the collection and M is the number of index terms in the term space (Reed et al., 2006*b*). So, there would be a cost for updating the system in both IDF and document centroid values but there is no extra cost for using them for assigning term weights without updating.

In order to determine the ratio for splitting the test collections into parts, some preliminary experiments were conducted. The document set in the test collections were split into 2, 5, 10 and 30 parts and observed that if the ratio was small (few parts), the variation in MAP values was small and less significant. That is, the simulated effect of having a dynamic data stream was better achieved by splitting the collection into a larger number of parts. Thus, for the second experiment, the document sets in the test collections were split into 30 parts, i.e. the ratio between the initial document set in the test collection and the final updated document set in the collection was 1:29.

Table 5.3: Average Recall-Precision Using TF-IDF and TF-ATO with DA in Dynamic Experiment for Ohsumed.

Recall	**AvgP and MAP for Dynamic Experiment**	
	TF-IDF	**TF-ATO with DA**
0.1	0.516	0.776
0.2	0.329	0.561
0.3	0.26	0.402
0.4	0.202	0.283
0.5	0.159	0.213
0.6	0.138	0.17
0.7	0.126	0.146
0.8	0.117	0.125
0.9	0.111	0.11
MAP	0.217	0.309

Tables 5.3 and 5.4 present the results from the second experiment applied on Ohsumed and LATIMES collections. The tables show the results using TF-IDF or

Table 5.4: Average Recall-Precision Using TF-IDF and TF-ATO with DA in Dynamic Experiment for LATIMES.

Recall	AvgP and MAP for Dynamic Experiment	
	TF-IDF	TF-ATO
0.1	0.403	0.663
0.2	0.217	0.449
0.3	0.15	0.292
0.4	0.101	0.182
0.5	0.1	0.132
0.6	0.059	0.109
0.7	0.05	0.061
0.8	0.041	0.055
0.9	0.035	0.03
MAP	0.128	0.219

TF-ATO with DA for dynamic simulation experiment by adding more documents in the document set without re-weighting neither IDF nor DA. The tables show the average precision values obtained by the given TWS method for nine Recall values as well as the corresponding MAP value.

From these tables, it is observed that there is a drawback in the effectiveness compared to the case with static data streams. The MAP drawback ratios from static to dynamic using TF-IDF and TF-ATO with DA on Ohsumed collection are 21.8% and 15% respectively, while, the MAP drawback ratios from using TF-IDF and TF-ATO with DA on LATIMES collection are 57.9% and 44.5%. This means only large variation on the document size by adding a large number of documents may cause an impact on the IR effectiveness. However, the proposed weighting scheme TF-ATO with DA still gives better effectiveness values than those produced with the TF-IDF weighting scheme. It can be also seen from these tables that the average improvement in precision of TF-ATO with DA compared to TF-IDF is 42.38% for Ohsuemd collection, while the average improvement in precision of TF-ATO with DA compared to TF-IDF is 70.7% for LATIMES collection. Furthermore, the dynamic experiment shows the effect of the strong dynamic variation and drawback in IR effectiveness. Adding a large number

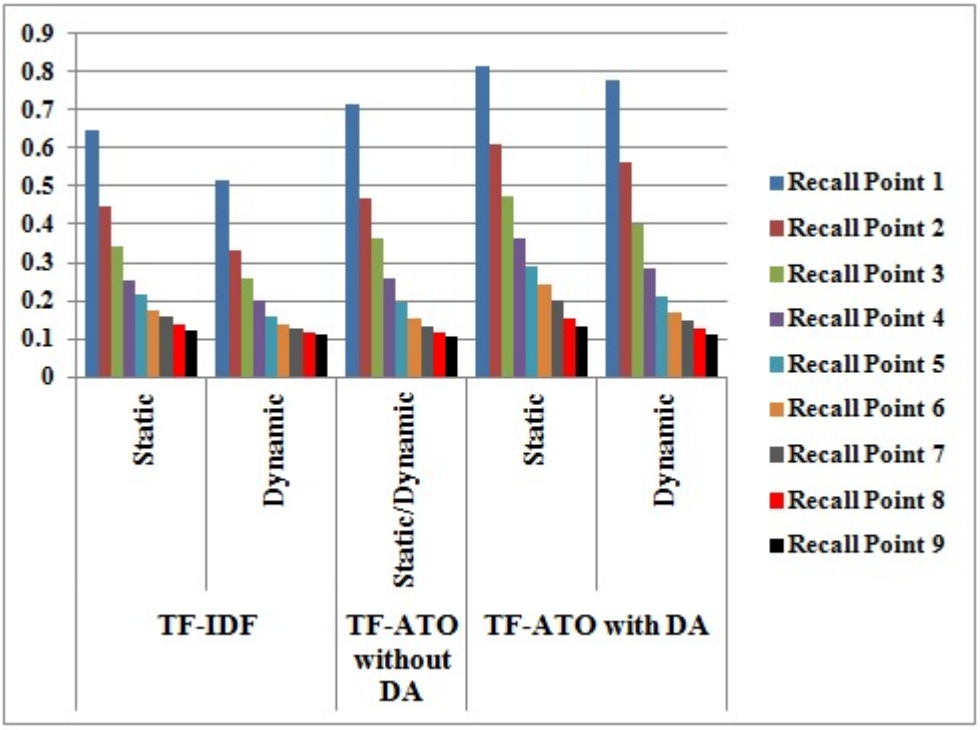

Figure 5.1: Illustrating the Average Precision performance for Static/Dynamic Experiments on Ohsumed Collection.

of documents to the document set (the index file) can cause a drawback in IR system effectiveness.

Figures 5.1 and 5.2 illustrate the bar chart for static/dynamic experiments on Ohsumed and LATIMES results reported in the tables mentioned above. In these figures, higher values correspond to better performance. From these figures it can be observed that the TF-ATO with DA TWS exhibits the overall best performance. On the other hand, the p-values of paired t-test for experiments is shown in Table 5.5. From the table, we can observe that the improvements using TF-ATO with DA are significant comparing with TF-IDF.

5.5 Stop-words Removal and DA Case Studies

The performance of the proposed term-weighting scheme TF-ATO was further investigated in terms of its DA and the impact of stop-word removal. The related work is

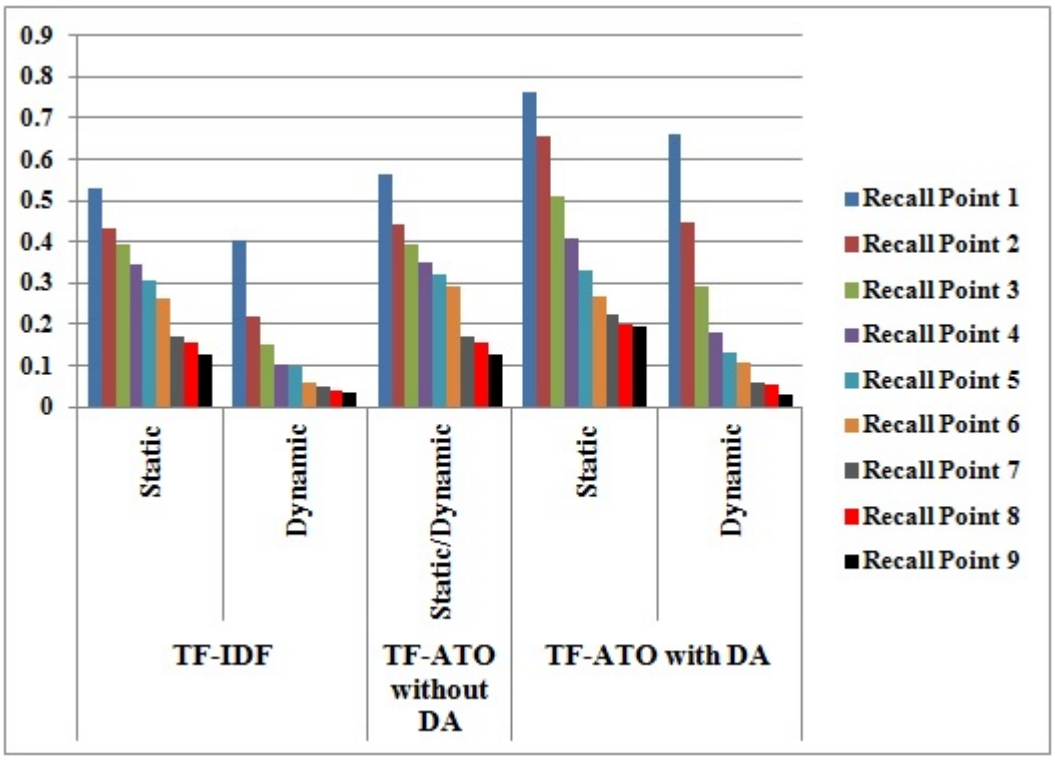

Figure 5.2: Illustrating the Average Precision performance for Static/Dynamic Experiments on LATIMES Collection.

Table 5.5: Paired T-test for Static and Dynamic Experiments

Paired	Ohsumed		LATIMES	
	static	dynamic	static	dynamic
TF-IDF and TF-ATO without DA	0.932	NA	0.049	NA
TF-IDF and TF-ATO with DA	0.0026	0.0223	0.011	0.02465
TF-ATO without /with DA	0.0001	NA	0.155	NA

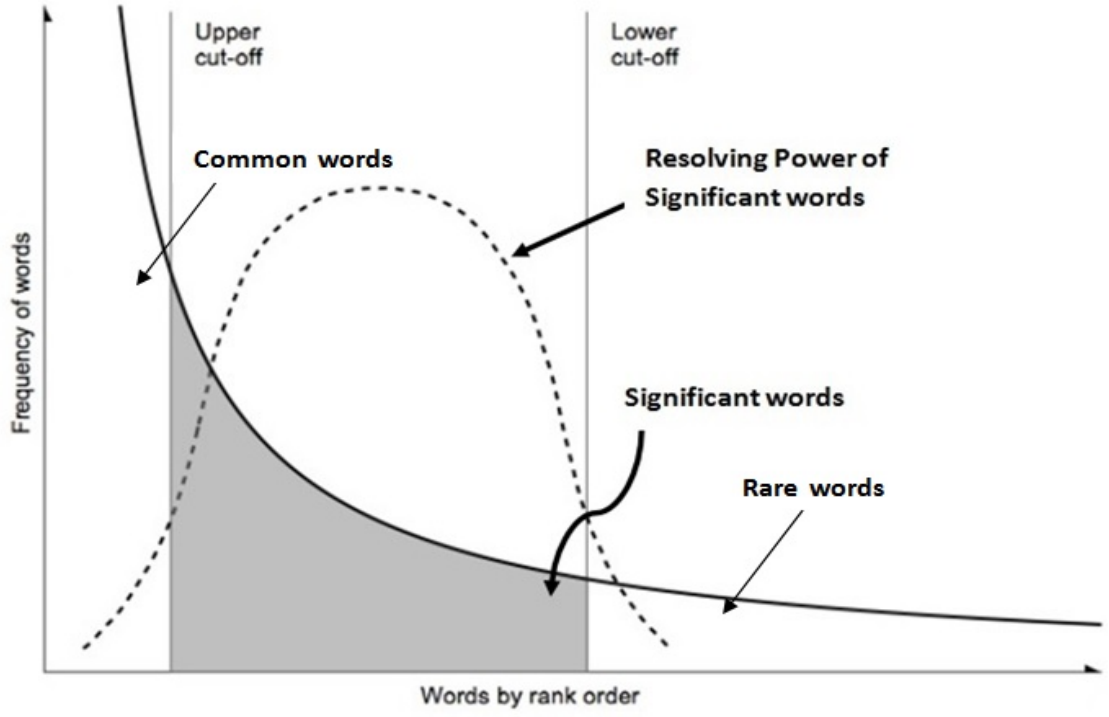

Figure 5.3: Zipf's Relationship Frequency vs. Rank Order for Words and Luhn's Cut-off Points for Significant and Non-significant Words on Text as in (Rijsbergen, 1979).

reviewed first and then the conducted experiments are introduced to compare the effectiveness of TF-ATO and TF-IDF in respect to the issues mentioned.

5.5.1 Related Work on Stop-word Lists

Zipf's Law and Luhn's Hypothesis

Zipf states that the relation between the frequency of the use of words and their corresponding rank order is approximately constant (Zipf, 1949). Zipf based his study on American English Newspapers. Based on Zipf's law, Luhn suggested that words used in texts can be divided into significant and non-significant keywords. He specified upper and lower cut-off points on Zipf's curve as shown in Figure 5.3. The words below the lower cut-off point are rare words that do not contribute significantly to the content of articles. The words above the upper cut-off point occur most frequently and cannot be good discriminators between articles because they are too common in texts. From Zipf's and Luhn's works, researchers have proposed lists of stop-words that should be removed

from texts for better effectiveness (accuracy) in natural language processing (NLP). From the literature, stop-words lists (stoplists) can be divided into three categories as follows.

1. ***General Stoplists***: These general purpose stoplists are generated from large corpus of text using term ranking scheme and high Document Frequency (high-DF) filtering among other methods inspired by Zipf's law. Examples are the Rijsbergen (Van Rijsbergen, 1975), SMART's (SMART, 2014) and Brown's (Fox, 1992) stoplists. Later, (Sinka and Corne, 2003*b*) generated two ranked list of words in ascending order of their entropy and constructed modern stoplists based on Zipf's and Luhn's work. They showed that their stoplists outperform Rijsbergen's and Brown's stoplists in text clustering problem with respect to accuracy. However, Rijsbergen's and Brown's stoplists perform better on other case studies. Sinka and Corne did not make their stoplists available. It should be noted that the computational cost to build new stoplists from large corpus by this method is high compared to the slight improvement in accuracy.

2. ***Collection-Based Stoplists***: These stoplists are generated from the test collection and can be applied on the test and real IR test collections. The challenge here is in choosing the cut-off points to classify the words in the collection into stop-words, rare (non-significant) words and significant. Four approaches based on Zipf's law and Luhn's principle for choosing corpus-based stop-words list were proposed by (Lo et al., 2005). Further, they used Kullback-Leibler (KL) divergence measure (Cover and Thomas, 1991) to determine the cut-off on these approaches. Their study concluded that the approach using normalised inverse document frequency (IDF) gave better results. It should be noted that the computational time and efforts including mathematical and probabilistic calculations to build collection-based stoplists for each test collection is high compared to using general stoplists. Furthermore, Lo et al. illustrated that there is no significant differences between using general stoplists and collection-based stoplists in terms of average precision when using TREC Disk 4&5 and .Gov collections.

3. ***Evolving Stoplists***: In this category, meta-heuristic techniques are used for evolving a group of general stoplists with the aim of producing better stoplists. To the best

of our knowledge, only (Sinka and Corne, 2003*a*) have used this approach. Their method starts by combining the top 500 stop-words in the stoplists of (Sinka and Corne, 2003*b*) with the stoplists of Rijsbergen's and Brown's into one group to be evolved. Then, they applied Hill Climbing (HC) and Evolutionary Algorithm (EA) with 2000 documents in 2-mean clustering problem. This approach has been applied on text classification problem in which each document is belong to a specific class. The similar approaches for evolving stoplists in IR require the relevance judgement values in each test collection to check the quality of the proposed stoplist comparing to the IR system effectiveness.

Hence, the general stoplists are the most appropriate stoplist in IR at the beginning of establishing IR system. The general stoplists can be used with less computational cost and without the need for having relevance judgement values to use it. They are available and easy to apply stopword removal with them at the start of establishing IR system.

5.5.2 Experimental Results and Analysis

In these experiments, the impact of our DA was investigated as a heuristic method for improving documents representations. The system effectiveness and performance in terms of the Mean Average Precision (MAP) and the size of the index file were measured. In order to apply the DA no information about relevance judgement is needed. In these experiments, the impact of stop-words removal is also examined. As discussed above, this is an important process for improving the performance and effectiveness of IR systems. Then the impact of the DA and the removal of stop-words were examined on two TWS, our proposed TF-ATO and also TF-IDF. The experiments were conducted using the following five test collections: Ohsumed, Cranfield, CISI, FBIS and LATIMES (see Table 3.3). These test collections are used by researchers on mathematical (non-learning) and on Computational Intelligence domain (Cummins, 2008; Reed et al., 2006*b*; Smucker et al., 2012; Voorhees, 2004). The following four case studies are used in the experiments where TWS is either our TF-ATO or TF-IDF:

- Case 1: apply TWS without using stop-words removal nor DA.
- Case 2: apply TWS using stop-words removal but without DA.

- Case 3: apply TWS without using stop-words removal but using DA.
- Case 4: apply TWS using both stop-words removal and DA.

Table 5.6: Mean Average Precision (MAP) Results Obtained From Each Case in the Experiments. Using and Not-using Stop-words Removal is Indicated With sw(y) and sw(n) Respectively, Similarly for the DA.

Case No.	TWS	Ohsumed	Cranfield	CISI	FBIS	LATIMES
Case 1: sw(n)/da(n)	TF-IDF	0.215	0.275	0.282	0.287	0.269
	TF-ATO	0.188	0.233	0.241	0.249	0.22
Case 2: sw(y)/da(n)	TF-IDF	0.268	0.3	0.307	0.348	0.34
	TF-ATO	0.279	0.355	0.34	0.392	0.35
Case 3: sw(n)/da(y)	TF-IDF	0.277	0.282	0.295	0.293	0.306
	TF-ATO	0.278	0.301	0.315	0.295	0.312
Case 4: sw(y)/da(y)	TF-IDF	0.349	0.356	0.358	0.394	0.386
	TF-ATO	0.364	0.4	0.362	0.427	0.395

Detailed results from our experiments are shown in Tables 5.7, 5.8, 5.9, 5.10 and 5.11. Each table reports for one test collection, the average recall-precision values obtained with the four case studies as described above. The last row in each of these tables shows the MAP values for TWS on each case study across 9-points recall values. Then, the MAP values are collated and presented in Table 5.6. The p-values of the paired t-test for Case1, Case2, Case3 and Case4 are 0.0004, 0.025, 0.7 and 0.048 respectively. These values indicate the significant variation in each case study. The lower value is the highest significant.

Several observations can be made from the results in Table 5.6. First, it is clear that for both TWS in all five collections, using both stop-words removal and the DA (case 4) gives the better results. When comparing cases 2 and 3 (using only one of stop-word removal or DA), better results in general are obtained when using stop-words removal (case 2) than when using the DA (case 3). It is noted that when comparing TF-ATO and TF-IDF on cases 2, 3 and 4, our proposed TWS produces better results. Specifically, in case 2 (using stop-words removal only) TF-ATO outperforms TF-IDF by 2-18%, in case 3 (using DA only) TF-ATO outperforms TF-IDF by 0.3-7% and in case

Table 5.7: Average Recall-Precision Results Obtained on the Ohsumed Collection.

Recall	Average Precision In Ohsumed Collection For Cases Studies							
	Case1		Case2		Case3		Case4	
	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO
0.1	0.547	0.536	0.663	0.742	0.648	0.713	0.797	0.816
0.2	0.331	0.267	0.456	0.47	0.445	0.47	0.584	0.61
0.3	0.246	0.183	0.348	0.346	0.343	0.361	0.442	0.472
0.4	0.176	0.142	0.233	0.238	0.253	0.259	0.343	0.362
0.5	0.151	0.127	0.192	0.191	0.216	0.196	0.26	0.288
0.6	0.133	0.117	0.16	0.154	0.176	0.153	0.241	0.24
0.7	0.124	0.112	0.133	0.14	0.156	0.13	0.194	0.199
0.8	0.117	0.107	0.118	0.121	0.136	0.114	0.145	0.154
0.9	0.111	0.104	0.11	0.111	0.123	0.108	0.134	0.13
MAP	0.215	0.188	0.268	0.279	0.277	0.278	0.349	0.364

Table 5.8: Average Recall-Precision Results Obtained on the LATIMES Collection.

Recall	Average Precision In LATIMES Collection For Cases Studies							
	Case1		Case2		Case3		Case4	
	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO
0.1	0.522	0.406	0.57	0.58	0.528	0.563	0.723	0.764
0.2	0.474	0.337	0.54	0.579	0.431	0.441	0.685	0.658
0.3	0.352	0.316	0.496	0.503	0.392	0.393	0.432	0.51
0.4	0.294	0.27	0.387	0.391	0.345	0.348	0.367	0.41
0.5	0.243	0.182	0.317	0.319	0.305	0.32	0.328	0.329
0.6	0.183	0.159	0.259	0.264	0.261	0.29	0.289	0.268
0.7	0.142	0.143	0.203	0.208	0.172	0.172	0.254	0.222
0.8	0.111	0.106	0.162	0.161	0.158	0.158	0.217	0.201
0.9	0.095	0.064	0.125	0.144	0.126	0.126	0.181	0.196
MAP	0.269	0.22	0.34	0.35	0.305	0.312	0.386	0.395

Table 5.9: Average Recall-Precision Results Obtained on the FBIS Collection.

Recall	Average Precision In FBIS Collection For Cases Studies							
	Case1		Case2		Case3		Case4	
	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO
0.1	0.487	0.458	0.582	0.601	0.513	0.507	0.623	0.669
0.2	0.422	0.403	0.501	0.578	0.457	0.491	0.559	0.633
0.3	0.381	0.31	0.493	0.55	0.418	0.428	0.532	0.599
0.4	0.343	0.291	0.41	0.428	0.377	0.348	0.407	0.517
0.5	0.295	0.247	0.372	0.421	0.21	0.22	0.383	0.429
0.6	0.206	0.195	0.302	0.397	0.195	0.19	0.389	0.39
0.7	0.207	0.166	0.21	0.271	0.161	0.172	0.254	0.292
0.8	0.138	0.117	0.15	0.159	0.158	0.158	0.217	0.193
0.9	0.106	0.05	0.11	0.121	0.144	0.144	0.181	0.117
MAP	0.287	0.249	0.348	0.392	0.293	0.295	0.394	0.427

4 (using both) TF-ATO outperforms TF-IDF by 2-12%. It is believed that the DA and stop-words removal are capable of removing more non-significant keywords compared to the traditional IDF method. However, it was recognised that the TF-IDF outperforms TF-ATO by 14-22% in case 1 (not using stop-words removal nor DA). This is due to the ability of IDF to remove some non-significant words from the documents by assigning values of 0 to words that are repeated in all documents in the collection.

Table 5.10: Average Recall-Precision Results Obtained on the Cranfield Collection.

Recall	Average Precision In Cranfield Collection For Cases Studies							
	Case1		Case2		Case3		Case4	
	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO
0.1	0.643	0.454	0.659	0.698	0.653	0.664	0.729	0.765
0.2	0.425	0.426	0.485	0.53	0.456	0.464	0.548	0.655
0.3	0.373	0.343	0.403	0.457	0.362	0.402	0.461	0.526
0.4	0.292	0.277	0.332	0.402	0.296	0.331	0.362	0.41
0.5	0.205	0.237	0.268	0.359	0.257	0.29	0.317	0.361
0.6	0.183	0.128	0.195	0.267	0.153	0.162	0.27	0.293
0.7	0.154	0.109	0.14	0.205	0.117	0.147	0.229	0.226
0.8	0.107	0.072	0.112	0.156	0.135	0.133	0.156	0.192
0.9	0.095	0.048	0.108	0.119	0.107	0.122	0.129	0.173
MAP	0.275	0.233	0.3	0.355	0.282	0.301	0.356	0.4

The stop-words removal and DA have a large impact on the efficiency of the IR

Table 5.11: Average Recall-Precision Results Obtained on the CISI Collection.

Recall	Average Precision In CISI Collection For Cases Studies							
	Case1		Case2		Case3		Case4	
	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO	TF-IDF	TF-ATO
0.1	0.56	0.468	0.643	0.624	0.51	0.603	0.727	0.74
0.2	0.421	0.355	0.521	0.56	0.46	0.544	0.626	0.621
0.3	0.395	0.308	0.427	0.457	0.423	0.431	0.526	0.537
0.4	0.342	0.266	0.319	0.403	0.39	0.36	0.442	0.465
0.5	0.289	0.235	0.264	0.331	0.203	0.298	0.339	0.263
0.6	0.183	0.186	0.208	0.267	0.189	0.169	0.245	0.246
0.7	0.14	0.175	0.14	0.186	0.171	0.147	0.149	0.147
0.8	0.111	0.108	0.127	0.122	0.163	0.154	0.108	0.123
0.9	0.095	0.067	0.11	0.11	0.148	0.127	0.058	0.115
MAP	0.282	0.241	0.307	0.34	0.295	0.315	0.358	0.362

Table 5.12: The Ratios (%) Of Reduction Of The Size Of The Index File Obtained From Its Original Index Size For Each Case in the Experiments.

Case Id	TF-IDF	TF-ATO
Ohsumed Case1	0.083%	0%
Ohsumed Case2	30.61%	30.65%
Ohsumed Case3	0.7%	0.75%
Ohsumed Case4	32.72%	32.76%
LATIMES case1	0.006%	0%
LATIMES case2	35.21%	35.22%
LATIMES case3	8.17%	8.16%
LATIMES case4	36.8%	36.78%
FBIS case1	9.12%	0%
FBIS case2	38.27%	33.7%
FBIS case3	30.22%	27.4%
FBIS case4	39.8%	39.6%
Cranfield case1	0.17%	0%
Cranfield case2	33.9%	33.83%
Cranfield case3	9.1%	9.4%
Cranfield case4	34.7%	34.5%
CISI case1	0.19%	0%
CISI case2	38.15%	38.4%
CISI case3	7.9%	7.5%
CISI case4	39%	38.9%

system measured in terms of the index file size. Results for this are presented in Table 5.12. From this table, we can see that when comparing cases 2 and 3 for each TWS on the five test collections, using stop-words removal (case 2) helps to reduce the index file size by 30.61-38.4% of the original index file (case 1). Whereas, the reduction when using DA only (case 3) is between 0.7-30.22%. Using both stop-words removal and DA (case 4) reduces the index file size between 32.72-39.8%. The positive effect of stop-words removal and DA is larger on TF-ATO than on TF-IDF. This is because IDF has already the ability to remove non-significant words.

Table 5.13: The Indexing Time in Minutes Using TF-IDF and TF-ATO For Each Case in the Experiments.

Case Id	TF-IDF	TF-ATO
Ohsumed Case1	36	22
Ohsumed Case2	40	25
Ohsumed Case3	38	23
Ohsumed Case4	43	27
LATIMES case1	39	21
LATIMES case2	43	24
LATIMES case3	41	25
LATIMES case4	46	29
FBIS case1	37	20
FBIS case2	43	23
FBIS case3	40	22
FBIS case4	45	29
Cranfield case1	7	5
Cranfield case2	10	7
Cranfield case3	8	6
Cranfield case4	12	9
CISI case1	6	4
CISI case2	8	6.5
CISI case3	7.5	7
CISI case4	11	9.5

From Table 5.13, the average computational indexing time for the TF-ATO was less than for the TF-IDF in the experimental case studies. Furthermore, the execution time for applying DA in each test collection is less than the execution time for applying Stop-

words removal. Thus, the TF-ATO outperformed the TF-IDF on computational time and also system effectiveness. The average running time of the TF-ATO was between 4 minutes and 29 minutes in the smallest and largest test collections, while the average computational time for the TF-IDF was between 6 minutes and 46 minutes. In general, the TF-ATO with DA outperformed the other approaches in terms of computation time and effectiveness. However, the TF-ATO with DA weighting scheme had lower reduction ratio values in the index size than the TF-IDF in some study cases. These experiments were conducted on a 3.60 GHz Intel (R) core(TM) i7-3820 CPU with 8GB RAM and the implementation was in Java NetBeans under Windows 7 Enterprise Edition.

5.6 Chapter Summary and Conclusion

From the study presented in this chapter, it is concluded that the proposed *Term Frequency - Average Term Occurrences (TF-ATO)* term-weighting scheme (TWS) can be considered competitive when compared to the widely used TF-IDF. The proposed TWS gives higher effectiveness in both cases of static and dynamic test collections. Moreover, the document centroid vector can act as a threshold in normalisation to discriminate between documents for better effectiveness in retrieving relevant documents. The variation and reduction in system effectiveness when using dynamic instead of static test collections were observed, plus there is additional cost for every update to the collection. The only adding very large number of documents to the test collections can have significant a negative impact on IR effectiveness, if the term re-weighting did not apply.

It was also observed that both stop-words removal and the DA have a positive effect on both TWS (TF-IDF and TF-ATO) for improving the IR performance and effectiveness. Furthermore, TF-IDF has a positive impact for removing some non-significant keywords from the test collections compared to TF-ATO. However, using stop-words removal and the DA have a larger impact on removing non-significant weights and keywords from the collection, more positive significantly on TF-ATO but also on TF-IDF. This means that it is beneficial to use the proposed DA as a heuristic method for improving IR effectiveness and performance with no information on the relevance judgement for the

collection. Our results showed that in general TF-ATO outperforms TF-IDF in terms of effectiveness and indexing time. Only when both stop-words removal and DA are not used, TF-IDF outperforms TF-ATO in terms of IR effectiveness. Chapter 6 argues a new methodology for evolving document representation weights and based on TF-ATO as local term-weights using a new technique called (1+1)-Evolutionary Gradient Strategy. This technique considers the limitation of the previous learning EML technique for evolving document representation weights.

Chapter 6

(1+1)-Evolutionary Gradient Strategy to Evolve Global Term Weights

6.1 Introduction

The effectiveness of an Information Retrieval (IR) system is measured by the quality of retrieving relevant documents responding to user information needs (queries). One of the common models used in IR is *Vector Space Model (VSM)*. Documents are represented in VSM as vectors of term weights. The VSM is the most well-known TVM category besides the probabilistic models. The term weight has a significant impact on the IR system effectiveness to retrieve relevant documents responding to user information needs. An IR system contains the document weight representations of the test collection in the form of an IR index file (Zobel and Moffat, July, 2006). For every index term in an IR index file, a term weight measures the information content or the importance of the term in the document. This term weight has two parts: the local and the global weights. The *Local Term Weight (LTW)* measures the contribution of the term within a given document. The *Global Term Weight (GTW)* measures the discrimination value of the term to represent the topic of the documents in the collection. GTW also indicates the importance of the term as a good discriminator between documents. Figure 6.1 shows the term weights structure in the *Index File* in an IR system.

Term weights can be improved for achieving better IR effectiveness if the users can

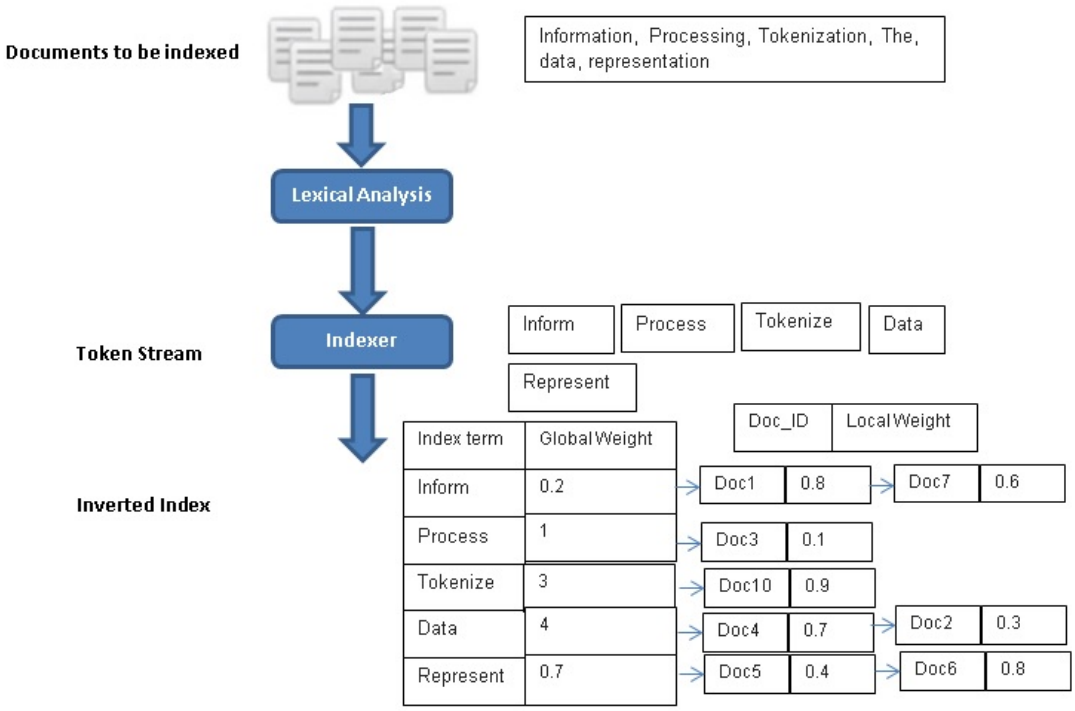

Figure 6.1: The Construction of the Index File (also called post file) which serves as an index for the IR system. It contains the global and local term weights for every term in each document and the document and term identifiers with the local term weight for each term.

identify examples of the relevant documents that they require for their current search. These examples of relevant documents and their corresponding user queries are stored into the relevance judgement file of the test collection. The relevance judgement of the IR test collection contains the group of relevant documents identified by users and their corresponding user information needs (queries). Evolutionary Computation (EC) techniques have been used extensively to improve IR effectiveness using the relevance judgement feedback from IR systems (Cordon et al., 2003; Cummins, 2008). Some of that previous research does not consider the problem size and the computational time that are required in order to achieve an improvement in IR effectiveness.

The related work on the *Term-Weighting Problem* can be divided into two categories: 1) evolving collection-based Term-Weighting Schemes (TWS) and 2) evolving term weights. These approaches have limited success to be used in real IR systems due to several reasons as explained below, which gives the motivation for the work presented in this chapter.

1. The TWS evolved by Genetic Programming (GP) rely on the relevance judgement (Cummins and O'Riordan, 2006; Cordon et al., 2003; Oren, 2002) to check the quality of the proposed weighting function. These approaches have the following limitations:

   - The IR test collections have not any relevance judgement values at the beginning of IR systems. However, the relevance judgement values are used in the objective functions to check the quality of the evolved solutions. In addition, the term-weighting scheme should be used first to collect the relevance judgement values.
   - Okapi-BM25 TWS outperformed the whole TWS evolved using small test collections different from the unseen large test collections in (Cummins and O'Riordan, 2006). This is because the evolved local term weighting schemes in small test collections did not perform well in large test collections.
   - The problem size of creating better collection-based weighting function using GP is large (Cummins and O'Riordan, 2006; Oren, 2002; Escalante et al., 2015). This is because the whole document space in the collection is considered in the evolving procedure of the global and local term-weighting functions.
   - The computational runtime required to create better collection-based weighting functions using GP is high (Cummins and O'Riordan, 2006; Oren, 2002; Escalante et al., 2015). In (Escalante et al., 2015), the computational runtime for evolving TWS in a subset of 20-Newsgroup collection (Rennie, 2015) using GP was 18 hours. Moreover, other GP approaches (Cummins and O'Riordan, 2006; Oren, 2002) applied on very small collections used a cluster of computers or took long computational runtime.

2. Evolving term weights of the document representations and evolving TWS using EC have resulted in better IR effectiveness regarding *Mean Average Precision (MAP)* and *Average Precision (AP)* on the same test collections used in the evolving procedure (Cummins, 2008; Cordon et al., 2003).

The main aim of this work is to propose a method to increase IR effectiveness by evolving better representations of documents in the collection for the trained queries with less computer memory usage. This is accomplished by evolving the Global Term Weights (GTWs) of the collection rather than evolving representations for the whole collection as is typically done with previous EC approaches in the literature. Hence, the main contribution of this chapter is the development of a *(1+1)-Evolutionary Gradient Strategy ((1+1)-EGS)* with *Adaptive Ziggurat Random Gaussian Mutation* to evolve GTWs. The Ziggurat random numbers is mentioned in (Kuo and Zulvia, 2015; Doornik, 2005; Loshchilov, 2014). The proposed methodology reduces the problem size, from evolving $(N \times M)$ document representation vectors to evolving $(1 \times M)$ vector, where N is the number of documents in the collection and M is the number of index terms in the collection. This chapter also examines a new meta-heuristic method ((1+1)-EGS) in IR with a new methodology for evolving document representation. This method considers the limitation of the relevance judgement of the test collections in EC (see subsection 3.1.3).

In order to evaluate the performance of the proposed method, experimental results are presented and discussed. The study compares results from using classical, fully evolved and partially evolved IR experiments. The proposed approach obtained improved MAP and improved AP compared to the Okapi-BM25 and TF-ATO weighting schemes (Robertson and Zaragoza, 2009; Ibrahim and Landa-Silva, 2016). In addition, the ratio of AP improvement obtained is larger than the one from evolving global term weighting function approaches in some related work (Cummins and O'Riordan, 2006; Fan et al., 2000; Cordon et al., 2003).

6.2 The Proposed Approach

This section presents the proposed approach to evolve Global Term Weights (GTWs) in information retrieval from test collections. The method uses Term Frequency-Average Term Occurrence (TF-ATO) that introduced in Chapter 5 and a (1+1)-Evolutionary

Gradient Strategy (EGS) for this purpose. The general Evolutionary Gradient algorithms was described in (Arnold and Salomon, 2007; Kuo and Zulvia, 2015). To the best of our knowledge, this approach is the first one that focuses on evolving the GTWs vector instead of evolving term-weighting functions or evolving term weights for the whole test collection, as discussed in the introduction. Experiments conducted here show that this approach achieves better MAP and AP compared to the other methods in the literature (Cummins and O'Riordan, 2006; Cordon et al., 2003; Oren, 2002; Fan et al., 2000).

An outline of the main steps in the method is given next. The first step is to obtain the corresponding vectors of local term weights for three sets of documents: the relevant document set, the irrelevant document set and their query set. These vectors contain TF-ATO values (see Chapter 5), of the index terms for every document in the three sets. Then, a (1+1)-EGS and Ziggurat random sampling (Doornik, 2005) is used to mutate the gradient steps. This method was selected because it has been shown that compared to other evolutionary strategies methods, Ziggurat random sampling has lower cost in terms of memory space or computational runtime (Loshchilov, 2014). The aim of the (1+1)-EGS is to optimise the residuals between relevance labels and labels produced from similarity matching between query set and document set. This can be accomplished by maximising cosine similarity (Baeza-Yates and Ribeiro-Neto, 2011) between the relevant document vectors and the query vectors. At the same time, it aims to minimise the cosine similarity between the irrelevant document vectors and the query vectors. The evolved GTWs will then be assigned to index-terms in the test collection. These GTWs are multiplied by TF-ATO to produce term weight vectors for each document in the collection.

The pseudo-code of the (1+1)-EGS is shown in Algorithm 2 and Table 6.1 lists the notations used in the pseudo-code. Steps 1 to 6 include two methods to initialise the parent GTW chromosome. The first method gives higher initialisation values and is applied to index terms that are good discriminators. An index term is a good discriminator when: 1) it exists in irrelevant documents only or 2) it exists with higher TF-ATO value in relevant documents than in irrelevant document and this index term

Table 6.1: The Notations Used in Algorithm 2.

Notation	Definition
RelDocSet	is the relevant document vector set of TF-ATO as the form of local term weight vectors.
IRRDocSet	is the irrelevant document vector set of TF-ATO as the local term weight representations.
QSet	is the query set of vectors in TF-ATO form.
ParentChromosomeGTW	is the current parent proposed of the evolved GTW vector chromosome for the index terms.
OffspringChromosomeGTW	is the current offspring of the evolved GTW vector of the index terms. This is the mutated (evolved) GTW parent chromosome (PG) of the current iteration.
ZGaussian(0,1)	is the Ziggurat random Gaussian number with 0 mean and 1 standard deviation and the value is between 0 and 1 Doornik (2005).
MutatPos	is the position of the gene that will undergo mutation.
MutatPosGood	is the array that saved the previous position of the gene that had mutations in the previous iteration.
NoMutations	is the number that indicates the number of genes (GTWs) that will be mutated.
NoMutationsGood	is the saved number from the previous generation that indicates the number of genes (GTWs) that had mutations.
MaxGTW	is the maximum GTW which is 1 in our case with using TF-ATO as a local weighting scheme.
Random(t1,t2)	is a function used to generate random number between t1 and t2

Algorithm 2: (1+1)-Evolutionary Gradient Strategy for Evolving GTWs

Data:
{**RelDocSet:**} is the Relevant Document Vector Set of TF-ATO weights.
{**IRRDocSet:**} is the Irrelevant Document Vector Set of TF-ATO weights.
{**QSet:**} is the Query Vector Set of TF-ATO weights.
{**MaxGTW:**} is equal 1 in case of using TF-ATO as a weighting scheme.
{**M:**} is equal to the number of index terms used to evolve their GTWs.
{**Good:**} has FALSE as an initialization value.
Result: Evolved GTWs of the Index Terms based on the relevance judgment values

```
1  Initialization for (IndexTerm Term_i ∈ M) do
2      if (Term_i is a good discriminator) then
3          ParentChromosomeGTW[i] = MaxGTW + ZGaussian(0,1);
4      else
5          ParentChromosomeGTW[i] = ZGaussian(0,1);
6      end
7      OffspringChromosomeGTW[i] = ParentChromosomeGTW[i];
8  end
9  while CosineSimilarity(RelDocSet,QSet,ParentChromosomeGTW) ≤ Maximum do
10     if (Good==TRUE) then
11         NoMutations=NoMutationsGood;
12     else
13         NoMutations = Random(0,M);
14         NoMutationsGood = NoMutations;
15     end
16     for i=1 → NoMutations do
17         if (Good==TRUE) then
18             MutatPos=MutatPosGood[i];
19         else
20             MutatPos = Random(0,M);
21             MutatPosGood[i]=MutatPos;
22         end
23         OffspringChromosomeGTW[MutatPos]=OffspringChromosomeGTW[MutatPos]+
             (ParentChromosomeGTW[MutatPos] - OffspringChromosomeGTW[MutatPos]) * ZGaussian(0,1);
24     end
       /* Keep the fitter evolved chromosome                                          */
25     if (CosineSimilarity(RelDocSet,QSet,ParentChromosomeGTW)
         <CosineSimilarity(RelDocSet,QSet,OffspringChromosomeGTW)) AND (CosineSimilarity(IRRDocSet,QSet,
         ParentChromosomeGTW) > CosineSimilarity(IRRDocSet,QSet,OffspringChromosomeGTW)) then
26         for i=1 → M do
27             ParentChromosomeGTW[i] = OffspringChromosomeGTW[i];
28         end
29         Good=TRUE;
30     else
31         for i=1 → M do
32             OffspringChromosomeGTW[i] = ParentChromosomeGTW[i];
33         end
34         Good=FALSE ;
35     end
36 end
```

exists in the queries. The second method gives lower initialisation values and is applied to index terms that are not good discriminators. Adding MaxGTW (a value of 1) to the initialisation for good discriminators, instead of only a Ziggurat random number, reduces the convergence runtime. The initialised parent chromosome is then copied as the offspring chromosome in step 7. Then, the main evolution cycle of the (1+1)-EGS is described in steps 9-36. The stopping criterion of the algorithm (step 9) indicates that the evolution will stop when the maximum similarity (a value of 1 as given by the cosine function) between relevant documents and user queries is achieved. Steps 10 to 24 show the procedure to control the mutation within the (1+1)-EGS. As shown in step 23, the actual mutation operator uses the genes gradient multiplied by Ziggurat random Gaussian number with mean equal to 0 and standard deviation equal to 1 as the step-size. Steps 10 to 22 show the strategy to control the number of gradient mutations and the position in the chromosome to mutate. Note that this strategy repeats the mutation settings when the mutated offspring chromosome improves upon the parent chromosome (this is indicated by the Boolean variable *Good*). The objective function that examines the quality of the offspring solution is shown in step 25. This objective function contains two conditions. The first condition is to increase the cosine similarity value between the relevant document vector set and the query vector set. The second condition is to reduce the cosine similarity between the irrelevant document vector set and the query vector set. That is, the offspring GTW chromosome is selected as the parent chromosome (line 27) for the next iteration if it increases the discrimination between the relevant and irrelevant document vector sets with the query vector set. In this case, the variable *Good* is set to TRUE so that the mutation settings are repeated in the next iteration. Otherwise, the offspring GTW chromosome is replaced by the parent GTW chromosome (line 32), and the variable *Good* is set to FALSE.

As explained above, the initialisation step in the above (1+1)-EGS distinguishes between index terms that are good discriminators and those that are not. This gives the proposed approach the ability to tackle *Polysemy*, one of the challenges in natural language. Polysemy happens when the same terms exists in both the relevant and the irrelevant document sets and the term has multiple different meanings in different contexts.

Hence, Polysemy words are not good discriminators because they have high TF-ATO values (LTWs) in relevant and irrelevant documents. However, with the proposed approach Polysemy words get lower GTWs than the good discriminator terms, which emphasises their non-discriminating nature. The Computational complexity of this algorithm is $\Omega(Q * N * n * log(R))$, where Q is the number of training queries, N is the number of training documents, n is the number of evolving iterations and R is the number of genes in the chromosome.

6.3 Experimental Study and Evaluation

6.3.1 Test Collections

Eight test collections were used in these experiments (Hersh et al., 1994; Glassgow, 2014; Smucker et al., 2012; TREC, 2004). Table 6.2 shows their main characteristics. In these experiments, four combination groups from the test collections were used to produce four test collections. Each test collection combination contains three textual materials: a set of documents, a set of queries, and relevance judgements between documents and queries. For each query, a list of relevant documents is associated with it. The first test collection consists of Ohsumed, CISI and CACM test collections (Hersh et al., 1994; Glassgow, 2014), containing 353226 documents and 233 queries. The second test collection consists of Cranfield, Medline and NPL test collections (Glassgow, 2014), containing 13862 documents and 348 queries. These two test collections were formed from sampled collections and they have been widely used for research such as in (Sebastiani, 2002; Cordon et al., 2003). The third and fourth collection combinations are from three test collections in the TREC Disks 4 & 5 with two different query sets and their relevance judgements. Crowdsourced and robust relevance evaluation were used with the queries and relevance judgements (Smucker et al., 2012; TREC, 2004). These third and fourth combinations contain FBIS, LA and FT test collections. The third test collection contains 472525 documents and 230 queries, while the fourth collection contains 18260 documents and 10 queries.

Table 6.2: Characteristics of the Test Collections Used in the Experiments.

ID	Description	No. of Docs	No. of Queries
Cranfield	Aeronautical engineering abstracts	1400	225
Ohsumed	Clinically-Oriented MEDLINE subset	348566	105
NPL	Electrical Engineering abstracts	11429	93
CACM	Computer Science ACM abstracts	3200	52
CISI	Information Science abstracts	1460	76
Medline	Biomedicine abstracts	1033	30
TREC Disks 4&5 (Robust 2004)	News and Broadcast WebPages	472525	230
TREC Disks 4&5 (Crowdsource 2012)	News and Broadcast WebPages	18260	10

6.3.2 Experimental Results

In this chapter, two term-weighting schemes were used. The first weighting scheme was the Okapi-BM25 probabilistic weighting scheme (see subsection 2.2.3). This weighting scheme has a good capability for estimating the term weights. It also outperformed the whole evolved term-weighting schemes produced by Genetic Programming (Cummins and O'Riordan, 2006). The second weighting scheme was TF-ATO with the Discriminative Approach (DA) (see Chapter 5), which is the only existing non-evolved approach that gives a good performance by discriminating documents without requiring any prior knowledge of the collection's relevance judgement. The number of index terms that were used in evolving their GTWs in the *Partially Evolved Experiment* in the test collections were 31658, 14679, 63091 and 6230 respectively. These terms are the keywords that exist in the relevant documents, the top-30 irrelevant documents using TF-ATO weighting scheme and their corresponding queries in the relevance judgement. In this experiment, the remaining non-evolved index terms in the test collections had values of 1s as GTWs. The number of index terms used in the *Fully Evolved Experiment* were 241450, 21600, 476850 and 18429 terms respectively. These terms constitute all the index terms in the collections. In this experiments, we normalised relevance labels between 0 and 1, where 0 is for irrelevant documents and 1 for relevant documents in the relevance judgement. Then, Residuals values can be represented by the subtraction between relevance labels and cosine similarity matching value between corresponding queries with documents. Figures 6.2 and 6.3 show the residuals values between normalised relevance labels

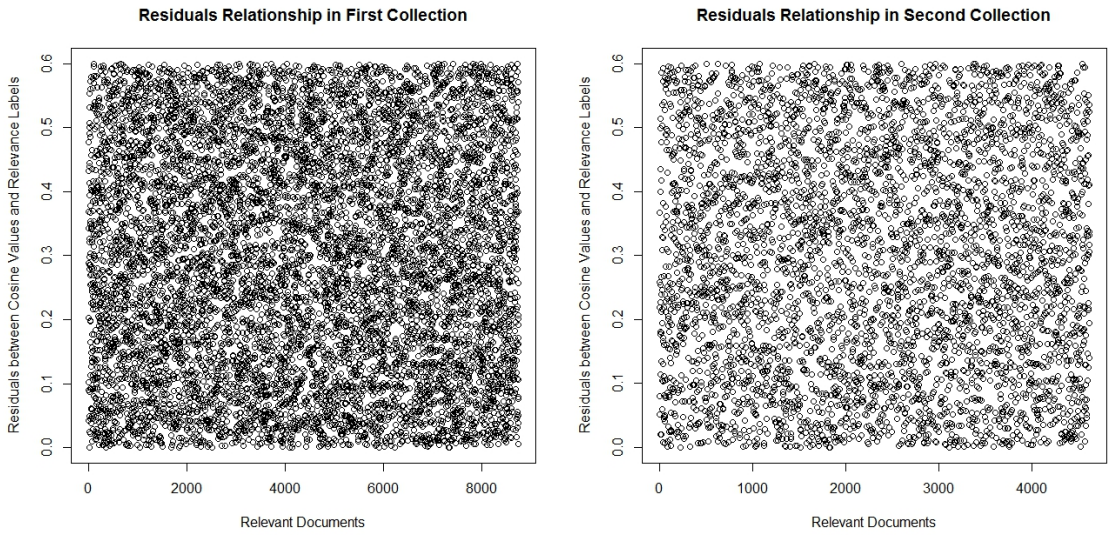

Figure 6.2: Shows Residuals between Relevant Documents/Queries Matching with Relevance Labels for First and Second Collection Combinations.

and cosine similarity values of the relevant documents with queries using the evolved GTW with TF-ATO as LTW. If the cosine similarity between a relevant document with a query, the residual value becomes closer to 0. From these figures, we can observe that the majority of relevant documents becomes more similar to their corresponding queries through optimising GTW using (1+1)-EGS. Thus, there are improvements in IR effectiveness through these evolving procedure.

Table 6.3: The NDCG@30 in the Four Collection Combinations of Using Okapi-BM25, TF-ATO with DA and the Proposed Approach.

Normalised Discounted Cumulative Gain for top-30 Documents Retrieved				
DocID	Okapi-BM25	TF-ATO with DA	Fully Evolved	Partially Evolved
1st Collection Combination	0.451	0.525	0.663	0.695
2nd Collection Combination	0.515	0.57	0.733	0.754
3rd Collection Combination	0.558	0.608	0.768	0.778
4th Collection Combination	0.519	0.569	0.729	0.739

Detailed results from our experiments are shown in Tables 6.5, 6.6, 6.7 and 6.8. Each table reports for one test collection, the average recall-precision values obtained with the four test collections. The last row in each of these tables shows the MAP values for each approach across 9-points recall values. Then, the MAP values are collated and presented in Table 6.3. Tables 6.3 and 6.4 show the average results of 10 runs of the proposed ap-

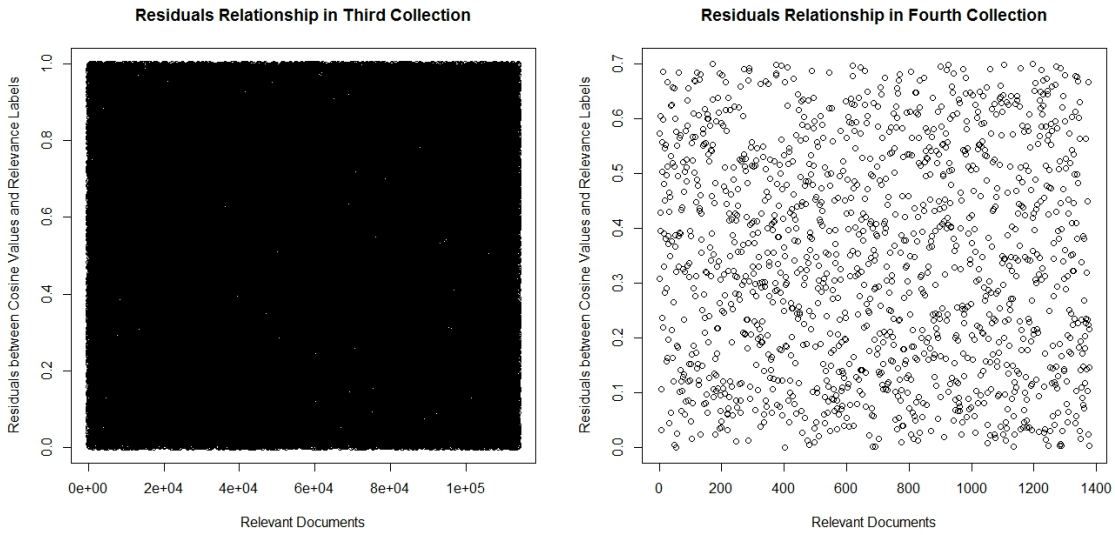

Figure 6.3: Shows Residuals between Relevant Documents/Queries Matching with Relevance Labels for Third and Fourth Collection Combinations.

Table 6.4: The Mean Average Precision in the Four Collection Combinations of Using Okapi-BM25, TF-ATO with DA and the Proposed Approach.

Mean Average Precision (MAP)				
DocID	Okapi-BM25	TF-ATO with DA	Fully Evolved	Partially Evolved
1st Collection Combination	0.29	0.364	0.4272	0.4779
2nd Collection Combination	0.345	0.4	0.4884	0.5157
3rd Collection Combination	0.3767	0.4243	0.5007	0.5245
4th Collection Combination	0.399	0.4512	0.5144	0.522

proach. These results are focused in the *MAP* and the *Normalised Discounted Cumulative Gain (NDCG@30)* for the experimental study. The *Partially Evolved Experiment* and the *Fully Evolved Experiment* in general outperformed the Okapi-BM25 and TF-ATO with DA approaches in terms of effectiveness. From Table 6.3, the *(NDCG@30)* values of the *Partially Evolved Experiment* were 0.695, 0.754, 0.778 and 0.739 for the test collection combinations, while the *NDCG@30* values of the *Fully Evolved Experiment* were 0.663, 0.733, 0.768 and 0.729. The ratios of improvement in *NDCG@30* regarding Okapi-BM25 in *Partially and Fully Evolved Experiments* were better than the improvement gained in evolving term-weighting functions in the literature (Cummins and O'Riordan, 2006; Fan et al., 2000). The ratios of improvement using the *Partially Evolved Experiments* with respect to Okapi-BM25 were 54.1%, 46.41%, 39.43% and 42.39% respectively in the four collections, while the improvement ratios in the *Fully Evolved Experiments* were 47.01%, 42.33%, 37.63% and 40.46%. From Table 6.4, the improvement ratios in the *MAP* values in the *Partially Evolved Experiments* were 64.8%, 49.5%, 39.24% and 30.83%, while the improvement ratios in the *MAP* values in the *Fully Evolved Experiments* were 47.31%, 41.57%, 32.92% and 28.92% respectively. From these results, the *Partially Evolved Experiments* outperformed *Fully Evolved Experiments*. The reason is that the training document set in *Partially Evolved Experiments* are only the relevant documents with top-30 irrelevant documents, while the document set in *Fully Evolved Experiments* are the whole relevant/irrelevant test collection. Thus, the convergence for better evolved global term-weights is slower in *Fully Evolved Experiments* than *Partially Evolved Experiments*.

Table 6.5: The improvement in MAP and AP on Partially Evolved and Fully Evolved Experiments of the first collection.

Recall	AP and MAP In The First Multi-topic Test Collection						
	Okapi-BM25	TF-ATO with DA	Fully Evolved Experiment	Partially Evolved Experiment	The ratio of improvement W.R.T. Okapi-BM25		
					DA Improvement (%)	Full Evolved Improvement (%)	Partially Evolved Improvement (%)
0.1	0.745	0.816	0.872	0.891	9.54	16.98	19.57
0.2	0.504	0.61	0.736	0.82	21.11	46.032	62.76
0.3	0.357	0.472	0.511	0.615	31.96	43.109	72.27
0.4	0.236	0.362	0.42	0.494	53.39	78.29	109.8
0.5	0.2	0.288	0.397	0.41	44.15	98.6	104.75
0.6	0.155	0.24	0.308	0.358	54.84	98.839	131.032
0.7	0.138	0.199	0.24	0.298	44.13	73.55	115.58
0.8	0.135	0.154	0.19	0.219	14.148	41.037	62.3
0.9	0.127	0.13	0.171	0.197	-3.64	34.65	55.12
MAP	0.289	0.364	0.427	0.478	25.57	39.067	50.1

Table 6.6: The improvement in MAP and AP on Partially Evolved and Fully Evolved Experiments of the second collection.

Recall	AP and MAP In The Second Multi-topic Test Collection						
	Okapi-BM25	TF-ATO with DA	Fully Evolved Experiment	Partially Evolved Experiment	The ratio of improvement W.R.T. Okapi-BM25		
					DA Improvement (%)	Full Evolved Improvement (%)	Partially Evolved Improvement (%)
0.1	0.62	0.765	0.857	0.874	23.487	38.354	41.001
0.2	0.509	0.655	0.698	0.715	28.740	37.271	40.554
0.3	0.479	0.526	0.610	0.657	9.969	27.544	37.283
0.4	0.396	0.408	0.575	0.595	3.183	45.275	50.379
0.5	0.348	0.361	0.482	0.496	3.741	38.705	42.647
0.6	0.281	0.293	0.391	0.428	4.018	39.011	52.312
0.7	0.214	0.226	0.342	0.392	5.621	60.000	83.607
0.8	0.146	0.192	0.248	0.281	31.712	70.068	92.123
0.9	0.118	0.173	0.192	0.205	46.480	63.020	73.537
MAP	0.345	0.4	0.488	0.516	17.44	46.583	57.049

Table 6.7: The improvement in MAP and AP on Partially Evolved and Fully Evolved Experiments on TREC Disk 4&5 Robust 2004 relevance feedback (TREC, 2004).

Recall	AP and MAP In The Third Multi-topic Test Collection						
	Okapi-BM25	TF-ATO with DA	Fully Evolved Experiment	Partially Evolved Experiment	The ratio of improvement W.R.T. Okapi-BM25		
					DA Improvement (%)	Fully Evolved Improvement (%)	Partially Evolved Improvement (%)
0.1	0.61	0.729	0.898	0.907	19.51	47.21	48.69
0.2	0.59	0.66	0.82	0.85	11.86	38.98	44.07
0.3	0.53	0.55	0.58	0.6	3.77	9.43	13.21
0.4	0.43	0.49	0.53	0.54	13.95	23.26	25.58
0.5	0.38	0.41	0.43	0.46	7.89	13.16	21.05
0.6	0.32	0.33	0.411	0.435	3.13	28.28	35.94
0.7	0.22	0.26	0.347	0.391	18.18	57.73	77.82
0.8	0.17	0.2	0.273	0.296	17.65	60.59	74.12
0.9	0.14	0.19	0.218	0.241	35.71	55.71	72.29
MAP	0.377	0.424	0.501	0.525	14.63	37.15	45.86

Figure 6.4 illustrates the bar chart for Fully and Partially Evolved experiments on the four test collections for the results reported in the tables mentioned above. In this figure, higher values correspond to better performance. From these figures it can be observed that the *Partially Evolved Experiments* exhibits the overall best performance. On the other hand, the p-values for paired t-test in the MAP results for Okapi-BM25 with TF-ATO, Okapi-BM25 with Partially Evolved, Okapi-BM25 with Fully Evolved, TF-ATO with Fully Evolved and TF-ATO with Partially Evolved are 0.0021, 0.00004, 0.000002, 0.00011 and 0.00002 respectively. The lower value indicates how the significant improvement between Partially or Fully Evolved comparing to Okapi-BM25 and TF-ATO with DA TWS.

Table 6.8: The improvement in MAP and AP Partially Evolved and Fully Evolved Experiments on TREC Disk 4&5 crowdsource 2012 relevance feedback (Smucker et al., 2012).

Recall	AP and MAP In The Fourth Multi-topic Test Collection						
	Okapi-BM25	TF-ATO with DA	Fully Evolved Experiment	Partially Evolved Experiment	The ratio of improvement W.R.T. Okapi-BM25		
					DA Improvement (%)	Fully Evolved Improvement (%)	Partially Evolved Improvement (%)
0.1	0.631	0.693	0.925	0.939	9.83	46.6	48.81
0.2	0.597	0.653	0.875	0.853	9.38	46.57	42.88
0.3	0.548	0.598	0.62	0.638	9.12	13.19	16.42
0.4	0.463	0.569	0.597	0.592	22.89	28.96	27.86
0.5	0.435	0.492	0.447	0.436	13.10	2.76	0.23
0.6	0.367	0.392	0.395	0.398	6.81	7.63	8.45
0.7	0.237	0.292	0.335	0.325	23.21	41.35	37.13
0.8	0.185	0.198	0.246	0.276	7.03	32.97	49.19
0.9	0.127	0.174	0.189	0.244	37.01	48.82	92.13
MAP	0.399	0.451	0.514	0.522	15.4	29.9	35.9

Table 6.9: The Average Computational runtime per a Document in the Four Collection Combinations of Using Okapi-BM25, TF-ATO with DA and the proposed Approach.

Average Computational Runtime in Seconds per an Instance				
DocID	Okapi-BM25	TF-ATO with DA	Fully Evolved	Partially Evolved
1st Collection Combination	17	15	300	180
2nd Collection Combination	19	17	430	120
3rd Collection Combination	18	15	600	230
4th Collection Combination	17	15	260	75

The computational runtime periods were computed for each instance (query with its relevant/irrelevant documents) in the test collections. From Table 6.9, the average computational run time for the *Partially Evolved Experiment* was less than for the *Fully Evolved Experiment* by 120 to 370 seconds depending on the number of evolved index terms in the GTWs vector. Thus, the *Partially Evolved Experiment* outperformed the *Fully Evolved Experiment* on computational time and also system effectiveness. The average running time of the *Partially Evolved Experiment* was between 75 seconds and 230 seconds in the smallest and largest collection combination, while the average computational time for the *Fully Evolved Experiment* was between 260 seconds and 600 seconds. In general, the TF-ATO with DA outperformed the other approaches in terms of computation time. However, the TF-ATO with DA weighting scheme had lower effectiveness values than the proposed approach. Thus, the next step in future research will be to reduce the com-

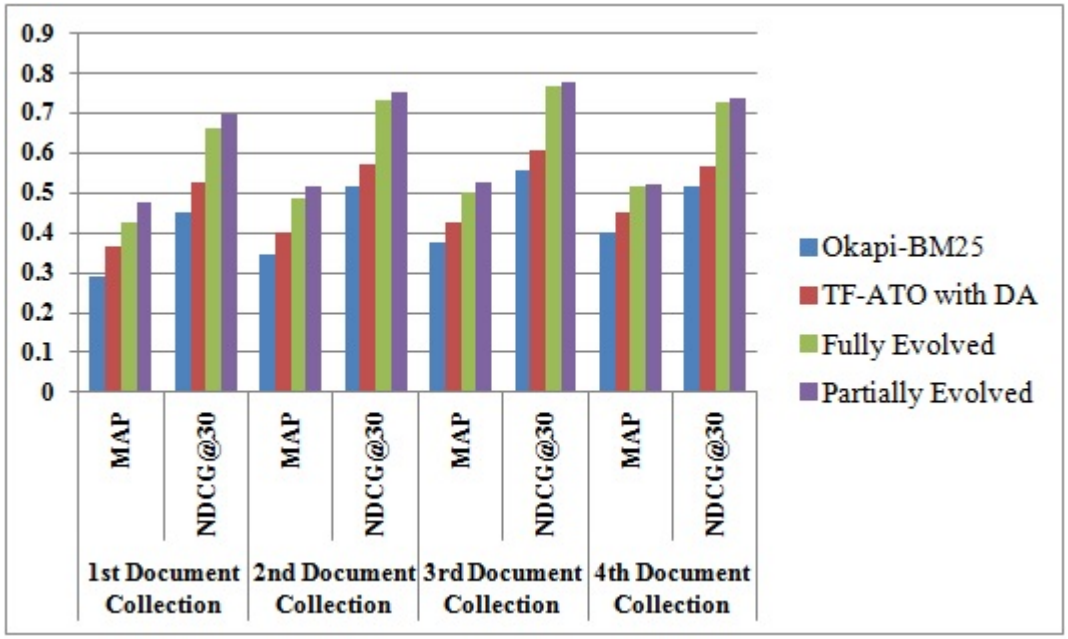

Figure 6.4: Illustrating the MAP and NDCG@30 performance for Okapi-BM25, TF-ATO with DA, Fully and Partially Evolved Experiments.

putational time using a combined machine learning technique with (1+1)-EGS. These experiments were conducted on a 3.60 GHz Intel (R) core(TM) i7-3820 CPU with 8GB RAM and the implementation was in Java NetBeans under Windows 7 Enterprise Edition.

6.4 Chapter Summary and Conclusion

This chapter proposes an approach based on a (1+1)-Evolutionary Gradient Strategy and on Term Frequency-Average Term Occurrence (TF-ATO), for evolving the Global Term Weights (GTWs) of the test collection in Information Retrieval (IR). By using (1+1)-chromosomes of M genes, the proposed method is less demanding in terms of computer memory, compared to other evolutionary computation approaches for IR used in the literature. Other approaches in the literature use non-adaptive evolutionary computation techniques and have large search spaces for evolving document vectors. In contrast, the technique described here optimised the document vectors through a GTW vector using the local weight vectors of the collection. This approach also has positive impacts on improving IR effectiveness. In addition, the *Partially Evolved Experiment* considers the

limitations of the relevance judgement of the collection. The index terms that did not exist in the *Partially Evolved Experiment* had values of 1 for GTWs and TF-ATO for LTWs. The *Partially Evolved Experiment* was used to evolve the GTWs of the index terms existing in the relevant document set and top-30 irrelevant document set rather than all the index terms existing in the collection. The remaining documents that did not have relevance judgement values only had TF-ATO representations. The *Partially Evolved Experiment* outperformed the *Fully Evolved Experiment* in IR system effectiveness. In addition, the two experimental methods had better effectiveness than the Okapi and TF-ATO weighting schemes. On the other hand, the *Fully Evolved Experiment* consumed more computational time than the *Partially Evolved Experiment* for evolving GTWs for the queries existing in the collection's relevance judgement.

Chapter 7

Learning to Rank Model Based on Feature Vector Model

7.1 Introduction

Ranking the search results responding to the user query is a vital research domain in IR system. In this research domain, the Evolutionary and Machine Learning (EML) techniques have been used intensively to achieve the best IR accuracy and performance. The contribution of this chapter is to investigate the importance of the initialisation procedure in (1+1)-Evolutionary Strategy ((1+1)-ES) to tackle the Learning to Rank (LTR) problem. It also introduces ES as a novel technique in LTR problem. The ES technique can be considered as a scalable alternative to Reinforcement Learning technique with which the optimisation problem converges to near optimal solutions in less runtime than other evolutionary computation techniques (Salimans et al., 2017; Beyer and Schwefel, 2002). Moreover, (1+1)-ES uses the lowest memory size comparing to other EML techniques in LTR problem. The proposed method is called ES-Rank and consists of evolving a vector of weights where each weight represents the importance value of a dataset feature. The initialisation procedure in ES-Rank has been tuned based on Linear Regression (LR) and Support Vector Machine (SVM) ranking models to produce IESR-Rank (Initialising ES-Rank with LR) and IESVM-Rank (Initialising ES-Rank with SVMRank). Details of the proposed method are presented in Section 7.2. In order to assess the performance of ES-Rank, *Mean Average Precision (MAP), Root Mean Square Error (RMSE), Normalised*

Discounted Cumulative Gain (NDCG@10), Reciprocal Ranking (RR@10) and Precision (P@10) at top-10 query-document pairs retrieved (Liu, 2011; Li, 2014) were used and a comparison is carried out against fourteen state-of-the-art LTR approaches from the literature. Experimental results in this chapter show that ES-Rank performs very well when compared to those other methods in terms of MAP, NDCG@10, RR@10, P@10 and RMSE. Furthermore, the better initialisation procedure using machine learning ranking model has a positive impact on improving ES-Rank in most cases. Furthermore, most of the other methods consumed very long computational 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 (Qin et al., 2010) while ES-Rank consumed only around 30 minutes on each fold. Another advantage 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). It is also observed that the appropriate initialisation values for ES-Rank can improve the accuracy of evolved ranking model. This chapter provides the evidence that the initialisation procedure based on LR can improve the accuracy of ES-Rank on LETOR datasets and introducing a new EML technique (ES-Rank) in LTR problem. Furthermore, this chapter introduces the first comparison between LTR techniques in terms of accuracy against computational time. The experimental results are presented in Section 7.3 which clarify the research finding of this chapter while the conclusion and the proposed future work are given in Section 7.4.

7.2 The Proposed Approaches

The proposed LTR methodology uses a (1+1)-Evolutionary Strategy (ES) for evolving the ranking function, due to the proven capability of evolutionary strategies to effectively and efficiently converge towards a better solution (Beyer and Schwefel, 2002). In addition, the list-wise approach have high performance values in terms of Mean Average Precision (MAP) and Normalised Discounted Cumulative Gain (NDCG) against pair-wise and point-wise approaches in literature (Cao et al., 2007). The proposed technique in this chapter is called Evolution Strategy Ranking (ES-Rank). The EGS-Rank ((1+1)-Evolutionary Gradient Strategy LTR) by mutating the chromosome under the

gradient of the fitness evaluation metric is another extension of ES-Rank. Unfortunately, EGS-Rank is not efficient than ES-Rank when used for LETOR datasets. The limitation of EGS-Rank relates to runtime and the accuracy when compared to the ES-Rank application. The chromosomes initialisation in ES-Rank are by assigning zero value for each gene (each query-document feature weight). It is well-known that choosing an appropriate initial solution in evolutionary techniques is an important issue (Diaz-Gomez and Hougen, 2007; Burke et al., 1998). Three ways to create the initial parent are investigated here. One is to set all weights to the same value of zero, another ones uses Linear Regression (LR), the third one uses Support Vector Machine. Experiments later in this chapter show that using Linear Regression or Support Vector Machine for parent initialisation helps ES-Rank to converge towards better solutions.

Algorithm 3 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 chapter) 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 7.2.1, 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.

$$Mutated\_Gene\_i = Gene\_i + Gaussian(0,1) * \exp(Cauchy(0,1)) \quad (7.2.1)$$

Algorithm 3: ES-Rank: (1+1)-Evolutionary Strategy Ranking Approach

Input : A training set $\phi(q,d)$ of query-document pairs of feature vectors. Weight Feature Vector $WLR = g(wlr_i)$ from applying LR or SVM on $\phi(q,d)$ set.

Output: A linear ranking function $F(q,d)$ that assigns a weight to every query-document pair indicating its relevancy degree.

```
Initialization
for (Gen_i ∈ ParentCh) do
    Gen_i = 0.0 or weight from LR or SVMRank ranking model;
end
Good=FALSE;
OffspringCh = ParentCh;
for G = 1 to MaxGenerations do
    if (Good==TRUE) then
        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;
    else
        Choose number of genes to mutate R at random from 1 to M;
        for j = 1 to R do
            Choose at random, Gen_i in OffSpringCh for mutation;
            Mutate Gene_i using MutationStep according to equation (7.2.1)
        end
    end
    if (Fitness(ParentCh,φ(q,d)) <Fitness(OffspringCh,φ(q,d))) then
        ParentCh = OffspringCh;
        Good=TRUE;
    else
        OffspringCh = ParentCh;
        Good=FALSE ;
    end
end
return the linear ranking function F(q,d) = ParentCh^T • φ(q,d) = W^T • φ(q,d),
 that is ParentCh at the end of the MaxGenerations contains the evolved vector
 W of M feature weights, T indicates the transpose
```

The mutation step defined by Equation 7.2.1 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 7.2.1 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 measured using MAP or NDCG. 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. The computational complexity of this algorithm is $\Omega(N * n * log(R))$, where N is the number of training query-document pairs, n is the number of evolving iterations and R is the number of genes in the chromosome.

Instead of the simple initialisation process in steps 1 to 4 of Algorithm 3, Linear Regression (LR) and Support Vector Machine (SVM-Rank) are used now (see Chapter 4). That is, the genes in the ParentCh vector take the weight values that result from the least square LR or SVM-Rank models (Dang, 2016; Joachims, 2016). Incorporating these machine learning techniques into an evolutionary approach is a novel idea within the LTR domain. The reason for choosing LR and SVMRank is as well as ES-Rank, they produce linear ranking models, while other techniques produce non-linear ranking models or they have high computational run-time.

The run-time efficiency of the proposed method also allows for all training instances to be used in each learning iteration. Most other LTR techniques do not do that and instead they use sampling methods for learning and checking the quality of the proposed ranking models. However, sampling methods such as bootstrap Bagging or Boosting cause over-fitting and under-fitting problems (Brownlee, 2017). The proposed method evolves better ranking models with smooth fitting and better performance regarding run-time and accuracy.

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 proposed approach is a novel approach in Learning to Rank domain. The following section shows the experimental study of these approaches as a comparative study with fourteen machine learning techniques which were mentioned in Chapter 4.

7.3 Implementation and Experimental Results

This section presents a comprehensive experimental study comparing the performance of the proposed LTR approach to fourteen other methods both in terms of accuracy and computational run-time. Accuracy is measured using five metrics described in Section 2.1.6: *Mean Average Precision (MAP)*, *Normalized Discounted Cumulative Gain (NDCG)*, *Precision (P)*, *Reciprocal Rank (RR)* and *Root Mean Square Error (RMSE)*. In order to assess the performance of a method for LTR, benchmark datasets containing training, validation and test sets are identified. The LTR approach is first applied to the training set in order to learn a ranking function. Then, the performance of the learned ranking function is assessed using the test set to measure the predictive performance of the LTR algorithm.

7.3.1 Dataset Benchmarks

The benchmark datasets used in the experiments of this chapter are MSLR-WEB10K, LETOR 4 (MQ2007 and MQ2008) and LETOR 3 (Ohsumed, TD2003, TD2004, HP2003, HP2004, NP2003 and NP2004) (Qin and Liu, 2013; Liu, 2011; Qin et al., 2010). Table 7.1 outlines the properties of these datasets. The number of query-document pairs and the number of features in the Microsoft Bing Search dataset (MSLR-WEB10K) are much larger than in the LETOR 4 (MQ2007 and MQ2008) or the LETOR 3 (Ohsumed and .Gov) datasets. Each query-document pair in the datasets contains low-level features

such as term frequency and inverse document frequency of the document terms existing in the queries. The low-level features were determined for all document parts (title, anchor, body and whole). There are also high-level features that indicate the similarity matching between the queries and the documents. Furthermore, hybrid features represent the recent IR models such as Language Model with Absolute Discounted Smoothing (LMIR.ABS), Language Model with Jelinek-Mercer smoothing (LMIR.JM), Language Model with Bayesian smoothing using Dirichlet priors (LMIR.DIR) and User Click features (Liu, 2011; Qin et al., 2010; Qin and Liu, 2013).

The largest number of queries (10000) is in the MSLR-WEB10K dataset. All the other datasets have less than 1000 queries with the exception of the MQ2007 dataset which has 1692. Each query has associated a number of relevant and irrelevant documents, i.e. query-document pairs for each query. The relevance label indicates the relevance degrees for the queries with the documents (query-document relationship). In most cases, the relevance labels include values of 0 (for irrelevant), 1 (for partially relevant) and 2 (totally relevant). The exception is for the MSLR-WEB10K dataset with values (created by the Bing search engine) from 0 (irrelevant) to 4 (perfectly relevant). The LETOR 3 and LETOR 4 datasets were constructed by several research groups in collaboration (Qin and Liu, 2013; Liu, 2011). To the best of our knowledge, this chapter is the first to conduct a comprehensive comparison between many LTR approaches considering several accuracy metrics and computational run-time on several very different benchmark datasets.

Table 7.1: Properties of the benchmark datasets used in the experimental study.

Dataset	Queries	Query-Document Pairs	Features	Relevance Labels	No. of Folds
MQ2007	1692	69623	46	{0, 1, 2}	5
MQ2008	784	15211	46	{0, 1, 2}	5
Ohsumed	106	16140	45	{0, 1, 2}	5
HP2003	150	147606	64	{0, 1, 2}	5
TD2003	50	49058	64	{0, 1, 2}	5
NP2003	150	148657	64	{0, 1, 2}	5
HP2004	75	74409	64	{0, 1, 2}	5
TD2004	75	74146	64	{0, 1, 2}	5
NP2004	75	73834	64	{0, 1, 2}	5
MSLR-WEB10K	10000	1200192	136	{0, 1, 2, 3, 4}	5

7.3.2 Results

In this study, *MAP*, *NDCG@10*, *P@10*, *RR@10* and *RMSE* were used (Baeza-Yates and Ribeiro-Neto, 2011; Li, 2014) as five separate fitness functions on the training sets. They also were used as the evaluation metrics for the ranking functions on the test sets. The variants of the proposed LTR method are called ES-Rank (baseline initialisation), IESR-Rank (linear regression initialisation) and IESVM-Rank (support vector machine initialisation). Tables 7.2, 7.3, 7.4, 7.5 and 7.6 show the overall results for all the methods tested. The other fourteen methods are implemented in the packages RankLib (Dang, 2016), Sofia-ml (Sculley, 2010), SVMRank (Joachims, 2016), Layered Genetic Programming for LTR (RankGP) (Lin et al., 2007*b*; Mick, 2016) and rt-rank for IGBRT (Mohan et al., 2011). IGBRT technique has not MAP, P@10 and RR@10 results due to the limitation of rt-rank package. The parameter values used for those other approaches are the default settings in these packages. Those settings produced the shortest computational run time and the lowest memory size requirements for each approach. The experimental results presented are the average scores of five runs on 5-folds cross validation. Each dataset fold consists of a training, a validation and a testing data. Experiments were conducted on a PC with 3.60 GHz Intel (R) core(TM) i7-3820 CPU and 8GB RAM. The implementation was in Java NetBeans under Windows 7 Enterprise Edition.

The results shown in Tables 7.2, 7.3, 7.4, 7.5 and 7.6 correspond to the predictive values of the average performance of five runs by the tested approaches. As mentioned above, the performance is measured with the evaluation metrics Average MAP, NDCG@10, P@10, RR@10 and RMSE. From these results, it can be seen that IESR-Rank is generally the best approach producing the best performance among all methods in 7 out of 10 average MAP, 6 out of 10 average NDCG@10, 2 out of 10 average P@10, 2 out of 10 RR@10 and 4 out of 10 RMSE. The second best approach is ES-Rank, producing the best performance in 2 out of 10 average MAP, 2 out of 10 average NDCG@10, 1 out of 10 average P@10, 2 out of 10 average RR@10 and 2 out of 10 RMSE. Random Forest comes in the third position with 3 out of 10 average P@10 and 3 out of 10 average RR@10, while IESVM-Rank is fourth with 4 out of 10 RMSE.

Table 7.2: Algorithms Average Performance Applied on 10 Datasets Using MAP Fitness Evaluation Metric

Algorithm	MSLR-WEB10K	MQ2008	MQ2007	Ohsumed	HP2003	TD2003	NP2003	HP2004	TD2004	NP2004
RankBoost	0.574	0.477	0.453	0.448	0.698	0.205	0.647	0.6259	0.218	0.553
SVMRank	0.457	0.4	0.408	0.383	0.419	0.081	0.433	0.351	0.124	0.378
ListNET	0.473	0.453	0.44	0.44	0.124	0.057	0.201	0.174	0.136	0.157
AdaRank	0.571	0.465	0.454	0.437	0.72	0.245	0.618	0.715	0.191	0.57
MART	0.58	0.473	0.459	0.427	0.746	0.188	0.665	0.5	0.204	0.519
Coordinate Ascent	0.586	0.481	0.46	0.446	0.748	0.237	0.663	0.658	0.225	0.653
LambdaMART	0.586	0.47	0.455	0.426	0.737	0.181	0.655	0.5	0.187	0.496
RankNET	0.486	0.452	0.448	0.435	0.737	0.224	0.65	0.619	0.186	0.647
Random Forest	0.598	0.47	0.459	0.433	0.769	0.285	0.708	0.63	0.254	0.603
Linear Regression	0.502	0.455	0.43	0.433	0.492	0.217	0.557	0.506	0.189	0.467
RankGP	0.467	0.427	0.414	0.399	0.564	0.215	0.581	0.526	0.21	0.514
CoRR	0.476	0.439	0.422	0.396	0.489	0.224	0.568	0.543	0.21	0.487
LambdaRank	0.476	0.348	0.34	0.307	0.717	0.131	0.645	0.367	0.172	0.644
ES-Rank	0.57	0.483	0.47	0.421	0.799	0.278	0.749	0.718	0.262	0.752
IESR-Rank	0.603	0.494	0.473	0.435	0.8	0.291	0.754	0.693	0.258	0.758
IESVM-Rank	0.457	0.473	0.456	0.443	0.637	0.254	0.663	0.575	0.193	0.52

Table 7.3: Algorithms Average Performance Applied on 10 Datasets Using NDCG@10 Fitness Evaluation Metric

Algorithm	MSLR-WEB10K	MQ2008	MQ2007	Ohsumed	HP2003	TD2003	NP2003	HP2004	TD2004	NP2004
RankBoost	0.335	0.5	0.433	0.439	0.745	0.275	0.681	0.677	0.309	0.63
SVMRank	0.222	0.432	0.365	0.333	0.442	0.107	0.479	0.347	0.197	0.434
ListNET	0.193	0.484	0.417	0.393	0.166	0.121	0.179	0.184	0.114	0.278
AdaRank	0.346	0.497	0.43	0.448	0.746	0.265	0.654	0.713	0.279	0.626
MART	0.395	0.504	0.44	0.428	0.787	0.276	0.708	0.548	0.245	0.587
Coordinate Ascent	0.402	0.507	0.443	0.452	0.777	0.319	0.74	0.772	0.315	0.702
LambdaMART	0.4	0.505	0.448	0.417	0.778	0.284	0.696	0.625	0.25	0.505
RankNET	0.191	0.484	0.424	0.441	0.767	0.22	0.686	0.663	0.259	0.722
Random Forest	0.4	0.497	0.439	0.438	0.798	0.362	0.755	0.665	0.349	0.64
Linear Regression	0.361	0.487	0.42	0.43	0.552	0.32	0.61	0.556	0.275	0.541
RankGP	0.354	0.441	0.415	0.414	0.593	0.254	0.586	0.67	0.273	0.712
CoRR	0.358	0.474	0.422	0.424	0.57	0.251	0.557	0.659	0.279	0.599
LambdaRank	0.196	0.313	0.276	0.28	0.722	0.151	0.686	0.295	0.130	0.539
IGBRT	0.394	0.518	0.457	0.444	0.808	0.306	0.756	NA	NA	NA
ES-Rank	0.382	0.507	0.451	0.446	0.826	0.362	0.754	0.778	0.358	0.792
IESR-Rank	0.415	0.517	0.455	0.454	0.829	0.376	0.758	0.778	0.356	0.79
IESVM-Rank	0.224	0.498	0.436	0.449	0.789	0.34	0.733	0.572	0.213	0.524

Table 7.4: Algorithms Average Performance Applied on 10 Datasets Using P@10 Fitness Evaluation Metric

Algorithm	MSLR-WEB10K	MQ2008	MQ2007	Ohsumed	HP2003	TD2003	NP2003	HP2004	TD2004	NP2004
RankBoost	0.587	0.274	0.372	0.504	0.102	0.144	0.088	0.083	0.233	0.085
SVMRank	0.405	0.25	0.332	0.401	0.065	0.072	0.068	0.053	0.148	0.053
ListNET	0.436	0.267	0.358	0.466	0.033	0.068	0.033	0.024	0.112	0.027
AdaRank	0.594	0.247	0.356	0.499	0.1	0.128	0.085	0.083	0.224	0.089
MART	0.631	0.275	0.379	0.476	0.104	0.146	0.084	0.082	0.237	0.081
Coordinate Ascent	0.627	0.273	0.378	0.483	0.103	0.158	0.092	0.097	0.249	0.092
LambdaMART	0.645	0.275	0.384	0.478	0.096	0.156	0.087	0.079	0.229	0.0747
RankNET	0.443	0.267	0.362	0.496	0.097	0.148	0.087	0.085	0.215	0.096
Random Forest	0.607	0.275	0.378	0.492	0.105	0.194	0.094	0.087	0.267	0.088
Linear Regression	0.457	0.274	0.372	0.481	0.087	0.18	0.083	0.08	0.225	0.081
RankGP	0.447	0.24	0.344	0.416	0.067	0.105	0.082	0.073	0.143	0.073
CoRR	0.441	0.243	0.355	0.418	0.077	0.08	0.068	0.059	0.111	0.056
LambdaRank	0.43	0.213	0.292	0.328	0.056	0.022	0.032	0.021	0.132	0.043
ES-Rank	0.634	0.27	0.377	0.494	0.097	0.184	0.096	0.096	0.257	0.091
IESR-Rank	0.643	0.276	0.372	0.498	0.104	0.19	0.089	0.099	0.259	0.091
IESVM-Rank	0.405	0.272	0.376	0.481	0.103	0.166	0.092	0.083	0.168	0.069

Table 7.5: Algorithms Average Performance Applied on 10 Datasets Using RR@10 Fitness Evaluation Metric

Algorithm	MSLR-WEB10K	MQ2008	MQ2007	Ohsumed	HP2003	TD2003	NP2003	HP2004	TD2004	NP2004
RankBoost	0.777	0.533	0.564	0.723	0.726	0.47	0.645	0.64	0.498	0.557
SVMRank	0.505	0.451	0.485	0.625	0.404	0.161	0.424	0.349	0.355	0.242
ListNET	0.581	0.513	0.552	0.67	0.155	0.169	0.104	0.096	0.163	0.257
AdaRank	0.803	0.533	0.548	0.739	0.752	0.41	0.576	0.698	0.51	0.568
MART	0.809	0.53	0.569	0.706	0.792	0.43	0.668	0.567	0.428	0.523
Coordinate Ascent	0.772	0.533	0.558	0.697	0.789	0.432	0.678	0.704	0.529	0.608
LambdaMART	0.812	0.53	0.57	0.737	0.775	0.397	0.672	0.561	0.424	0.481
RankNET	0.622	0.505	0.552	0.719	0.767	0.426	0.64	0.653	0.447	0.655
Random Forest	0.812	0.53	0.566	0.723	0.796	0.539	0.713	0.625	0.651	0.589
Linear Regression	0.594	0.513	0.55	0.741	0.519	0.471	0.554	0.506	0.504	0.464
RankGP	0.573	0.483	0.542	0.559	0.584	0.309	0.465	0.477	0.478	0.454
CoRR	0.586	0.455	0.543	0.541	0.581	0.295	0.466	0.464	0.461	0.452
LambdaRank	0.58	0.428	0.5	0.602	0.752	0.312	0.615	0.368	0.37	0.579
ES-Rank	0.764	0.537	0.546	0.727	0.801	0.524	0.674	0.724	0.573	0.629
IESR-Rank	0.819	0.535	0.569	0.725	0.792	0.506	0.69	0.729	0.57	0.629
IESVM-Rank	0.513	0.513	0.56	0.729	0.768	0.474	0.638	0.435	0.459	0.422

Table 7.6: Algorithms Performance Applied on 10 Datasets Using RMSE Fitness Evaluation Metric

Algorithms	MSLR-WEB10K	MQ2008	MQ2007	Ohsumed	HP2003	NP2003	TD2003	HP2004	NP2004	TD2004
RankBoost	0.232	0.097	0.102	0.191	0.051	0.041	0.04	0.041	0.036	0.05
SVMRank	0.182	0.08	0.082	0.154	0.032	0.028	0.027	0.043	0.16	0.067
ListNET	0.274	0.093	0.099	0.191	0.018	0.011	0.013	0.005	0.012	0.025
AdaRank	0.213	0.074	0.076	0.199	0.04	0.025	0.03	0.023	0.018	0.029
MART	0.23	0.098	0.105	0.196	0.057	0.042	0.037	0.03	0.034	0.046
Coordinate Ascent	0.228	0.098	0.102	0.2	0.052	0.042	0.041	0.046	0.043	0.055
LambdaMART	0.24	0.098	0.104	0.192	0.051	0.043	0.034	0.037	0.03	0.046
RankNET	0.265	0.093	0.098	0.188	0.049	0.04	0.027	0.042	0.044	0.045
Random Forest	0.295	0.096	0.105	0.194	0.053	0.045	0.048	0.041	0.039	0.063
Linear Regression	0.165	0.095	0.099	0.18	0.034	0.035	0.043	0.033	0.03	0.049
LambdaRank	0.291	0.072	0.062	0.124	0.031	0.017	0.017	0.012	0.012	0.019
IGBRT	0.397	0.183	0.193	0.162	0.016	0.014	0.021	NA	NA	NA
ES-Rank	0.095	0.03	0.028	0.056	0.001	0.002	0.003	0.001	0.00	0.001
IESR-Rank	0.068	0.028	0.029	0.05	0.00	0.001	0.002	0.00	0.003	0.002
IESVM-Rank	0.166	0.026	0.027	0.058	0.001	0.00	0.001	0.001	0.003	0.005

Table 7.7: Average run-time for the five evaluation fitness metrics in seconds of the algorithms on the datasets

Algorithm	MSLR-WEB10K	MQ2008	MQ2007	Ohsumed	HP2003	NP2003	TD2003	HP2004	NP2004	TD2004
RankBoost	3720	15	74	28	483	1153	460	493	597	604
SVMRank	32409	19	23	15	33	40	36	33	35	32
ListNET	18005	45	95	43	145	255	250	145	140	142
AdaRank	3600	11	20	16	228	453	486	227	123	240
MART	1200	8	11	12	12	23	11	13	15	19
CA	25200	37	240	28	580	940	396	460	480	460
LambdaMART	3720	9	11	8	24	89	21	23	25	27
RankNET	10800	33	96	98	55	119	130	110	117	298
RF	3660	27	55	17	72	168	71	72	70	80
LR	157	2	3	3	5	6	5	5	4	5
RankGP	26020	375	390	360	430	519	486	423	406	496
CoRR	10803	42	51	39	59	61	58	57	58	57
LambdaRank	18015	46	142	165	145	237	462	150	150	438
IGBRT	36750	274	253	197	393	389	386	NA	NA	NA
ES-Rank	1800	35	51	15	128	137	47	69	68	70
IESR-Rank	1957	37	54	18	133	143	52	74	72	75
IESVM-Rank	34209	54	74	30	161	177	83	102	103	102

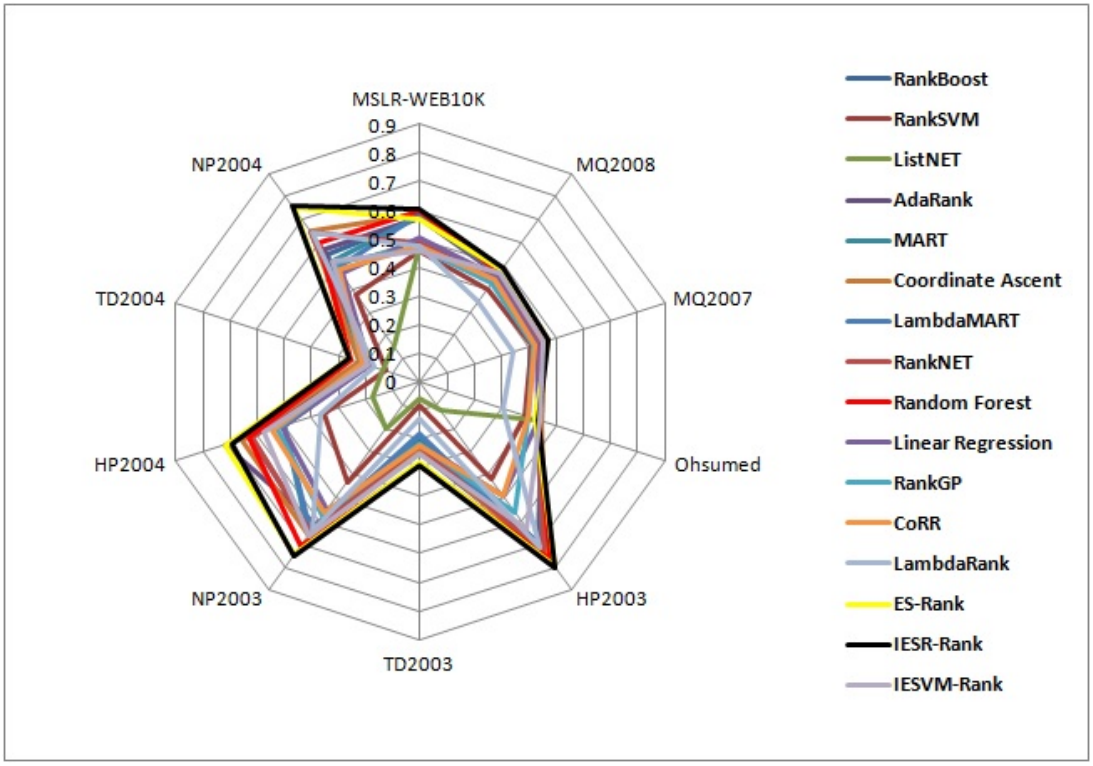

Figure 7.1: Illustrating the MAP performance for all LTR methods on the LETOR datasets.

The LambdaMART is fifth with 2 out of 10 average P@10 and 1 out of 10 average RR@10. The IGBRT and RankBoost are joint in the sixth position with 2 out of 10 average NDCG@10 for IGBRT, while RankBoost has 1 out of 10 average MAP and 1 out of 10 average P@10. Table 7.9 shows the detailed summary of the winner algorithm in each dataset.

Figures 7.1, 7.2, 7.3, 7.4 and 7.5 illustrate the radar chart for each fitness evaluation metric results reported in the tables mentioned above. In the first four figures, higher values correspond to better performance, while in the last figure lower values correspond to better performance. From these figures it can be observed that the IESR-Rank technique exhibits the overall best performance among all techniques.

The statistical Paired t-test of the results are presented in Table 7.8. This table shows how the degree of significant improvements between ES-Rank, IESR-Rank, IESVM-Rank against the Fourteen LTR techniques. The lower p-value indicates the algorithm

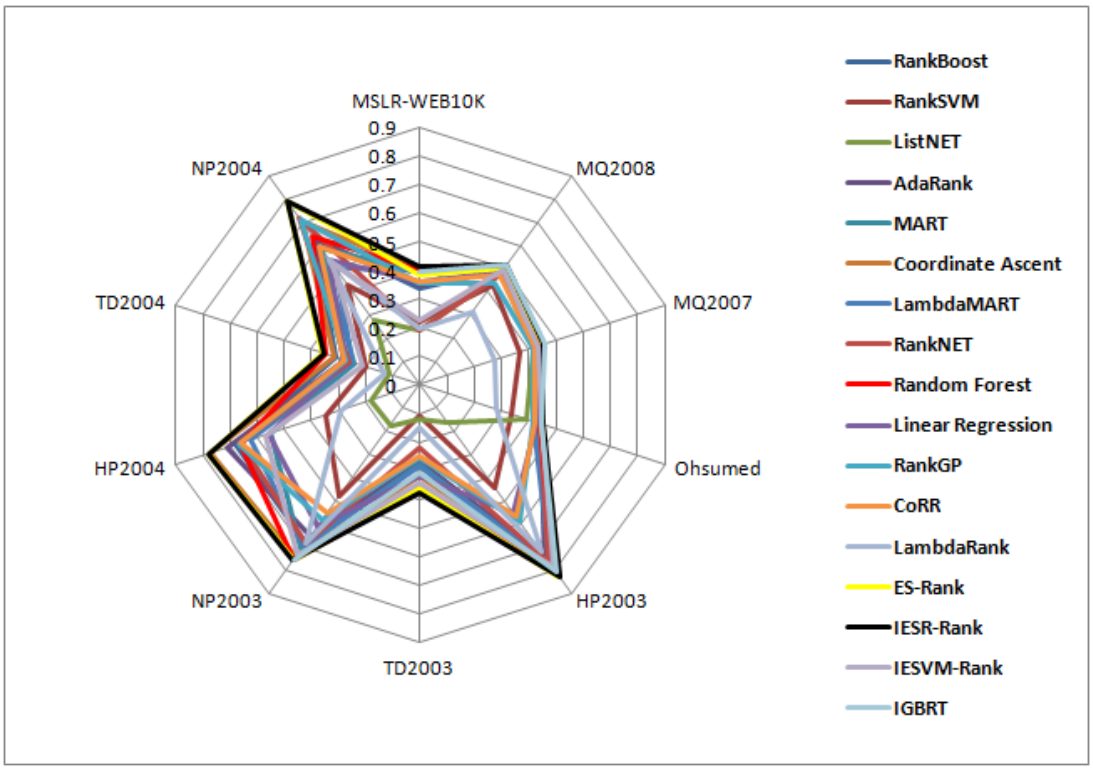

Figure 7.2: Illustrating the NDCG@10 performance for all LTR methods on the LETOR datasets.

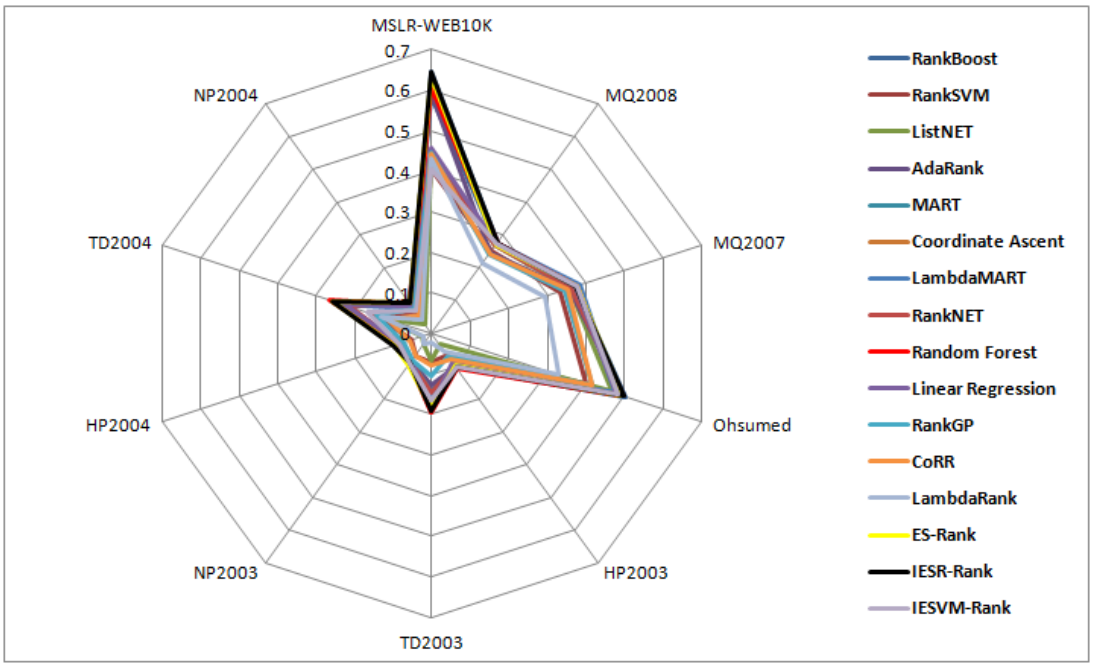

Figure 7.3: Illustrating the P@10 performance for all LTR methods on the LETOR datasets.

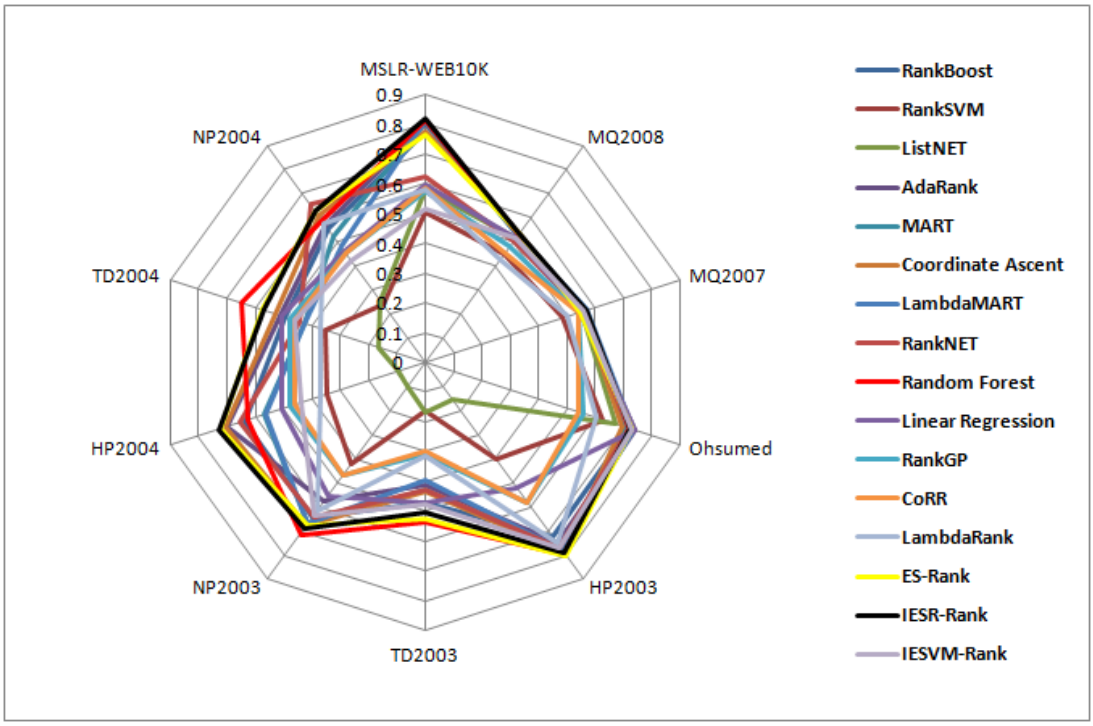

Figure 7.4: Illustrating the RR@10 performance for all LTR methods on the LETOR datasets.

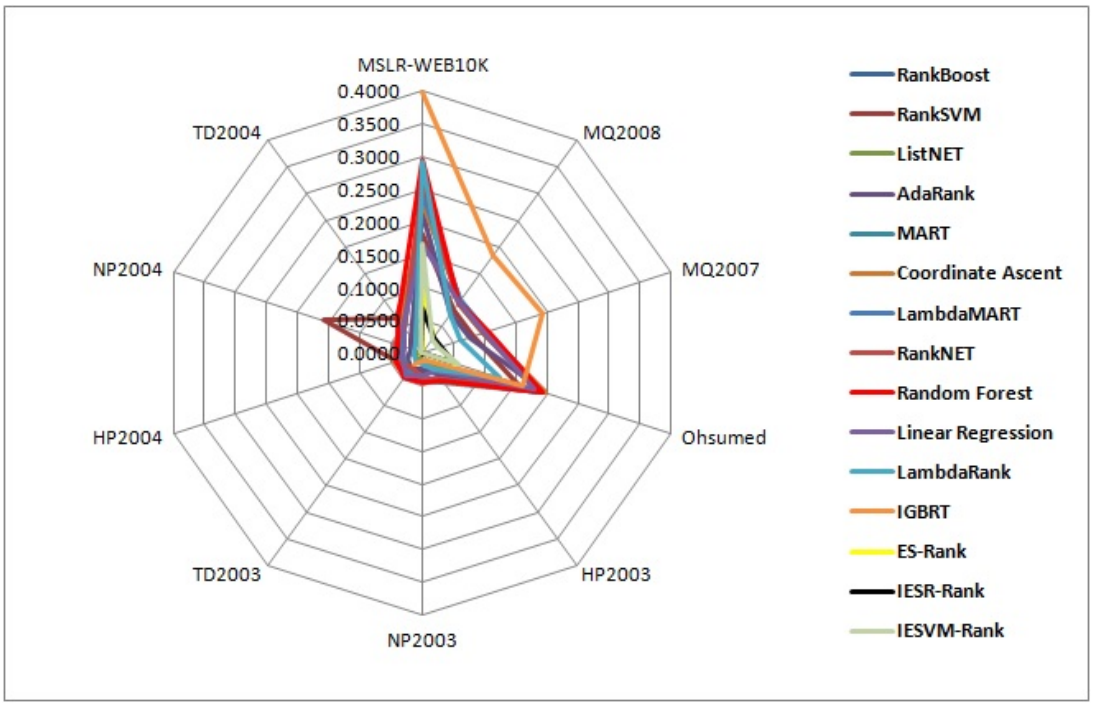

Figure 7.5: Illustrating the RMSE performance for all LTR methods on the LETOR datasets.

improvements is higher, while the p-value indicates the algorithm does not have a significant out-performance.

Table 7.8: P-values of Paired t-test between ES-Rank, IESR-Rank, IESVM-Rank and the Fourteen LTR Techniques

Algorithm	Paired t-test p-value		
	ES-Rank	IESR-Rank	IESVM-Rank
RankBoost	0.02419	0.01093	0.0001328
RankSVM	0.0000001	0.00000007	0.000011
ListNET	0.000002	0.0000013	0.000041
AdaRank	0.00745	0.003108	0.002079
MART	0.02	0.008526	0.002322
Coordinate Ascent	0.7094	0.3884	0.0000027
LambdaMART	0.01704	0.007375	0.005833
RankNET	0.003851	0.002967	0.003704
Random Forest	0.6615	0.9919	0.00000003
Linear Regression	0.0001257	0.000071	0.7654
RankGP	0.0000000001	0.00000000002	0.02346
CoRR	0.0000000002	0.00000000005	0.004306
LambdaRank	0.00000003	0.00000002	0.0006133
IGBRT	0.06411	0.1048	0.0103

The average computational run-times of the algorithms for each benchmark dataset are shown in Table 7.7. These results show that the variants of the proposed LTR method are still very efficient in terms of computational run-time. It can be seen that by incorporating linear regression into ES-Rank, the computational run-time of IESR-Rank increases just slightly over ES-Rank, but as discussed above, the accuracy results produced by IESR-Rank are much better.

7.4 Chapter Summary and Conclusion

This chapter presented a new LTR approach called ES-Rank which is based on a (1+1)-Evolutionary Strategy with a tailored mutation process and three initialisation techniques. The first technique uses zeros values to initialise the initial chromosomes and it is called ES-Rank. The second initialisation technique uses weights from LR ranking model to initialise the initial chromosome and it is called IESR-Rank. The third initialisation technique uses weights from Support Vector Machine model to initialise

Table 7.9: Winner Number of Accuracy Per Each Algorithm

Algorithm	Fitness and Evaluation Metric					
	P@10	MAP	NDCG@10	RR@10	RMSE	Total
RankBoost	1	1	0	0	0	2
SVMRank	0	0	0	0	0	0
ListNET	0	0	0	0	0	0
AdaRank	0	0	0	0	0	0
MART	0	0	0	0	0	0
Coordinate Ascent	0	0	0	0	0	0
LambdaMART	2	0	0	1	0	3
RankNET	1	0	0	1	0	2
Random Forest	3	0	0	3	0	6
Linear Regression	0	0	0	1	0	1
RankGP	0	0	0	0	0	0
CoRR	0	0	0	0	0	0
LambdaRank	0	0	0	0	0	0
IGBRT	0	0	2	0	0	2
ES-Rank	1	2	2	2	2	9
IESR-Rank	2	7	6	2	4	21
IESVM-Rank	0	0	0	0	4	4

the initial chromosome and it is called IESVM-Rank. The performance of the proposed approach was compared to fourteen EML approaches from the literature. The metrics *Mean Average Precision (MAP)*, *Normalized Discounted Cumulative Gain (NDCG@10)*, *Precision (P@10)*, *Reciprocal Ranking (RR@10)* and *Root Mean Square Error (RMSE)* were used as fitness functions within ES-Rank, IESR-Rank and IESVM-Rank. They also used for evaluating the performance of the LTR approaches in the comparison. The datasets used here are MSLR-WEB10K (Microsoft Bing ten thousand web queries) dataset, LETOR 4 (MQ2008, MQ2007 TREC Million queries datasets for years 2008 and 2007) and LETOR 3 (Ohsumed and 6 .Gov datasets).

From the experimental results, in general IESR-Rank exhibited an overall better performance than the other fourteen methods tested, achieving the best performance in 7 out of 10 average MAP, 6 out of 10 average NDCG@10, 2 out of 10 average P@10, 2 out of 10 average RR@10 and 4 out of 10 RMSE. The second best approach is ES-Rank, producing the best performance in 2 out of 10 average MAP, 2 out of 10 average NDCG@10, 1 out of 10 average P@10, 2 out of 10 average RR@10 and 2 out of 10 RMSE. Random Forest comes in the third position with 3 out of 10 average P@10 and 3 out of 10 average

RR@10, while IESVM-Rank is the fourth best performance with 4 out of 10 RMSE. The LambdaMART is in the fifth position with 2 out of 10 average P@10 and 1 out of 10 average RR@10. The IGBRT and RankBoost are joint in the sixth position with 2 out of 10 average NDCG@10 for IGBRT, while RankBoost has 1 out of 10 average MAP and 1 out of 10 average P@10. Thus, the (1+1)-Evolutionary Strategy is a competitive approach to tackle the LTR problem.

Chapter 8

ESClick: Learning to Rank Using Evolutionary Strategy Based Click Model

8.1 Introduction

Most search engines save user clicks behaviours (historical interaction data) in their logs. This data is called Click-through data. This data in search engine logs is a highly valuable source for embedded user satisfaction and implicit relevance labels. Numerous research such as in (Joachims, 2002; Liu, 2009; Schuth et al., 2013) proposed Click-though data as replacement resources for the relevance labels. This is because Click-through data costs are lower than Pooling and Cranfield paradigms for IR dataset creation. However, Click-through data undergoes several criteria of bias by user decisions. Examples of these criteria are: 1) The user education levels, intelligence and his familiarity with IR system which affect the decision of user satisfaction and IR system performance (Al-Maskari and Sanderson, 2011). This causes click bias based user education level and familiarity with IR system. 2) User clicks are known to be biased, noisy and difficult to interpret for the position of the document in the retrieved ranked list (Li, 2014; Liu, 2009; Kharitonov, 2016). One of the bias criteria is the user preference to click the top ranked documents regardless if these documents are relevant or not. Thus, research has been established to simulate the actual Click-through data without noise and bias. This

research has been used to propose several Click models based on various assumptions for removing the noise from Click-through data. These Click models were used in online LTR problems. In this research, the relevance labels on the benchmarks were used to check the performance of the click ranking models on the training set. The performance metrics for online Click models can be classified into two categories. The first category is offline and the second is online evaluation metrics. The offline and online evaluation metrics such as MAP, NDCG, P@K, RR@K,Err@K, A/B testing and Interleaving respectively (Hofmann, 2013; Schuth, 2016; Kharitonov, 2016).

The offline evaluation metrics are MAP, NDCG, P@K, RR@K and Err@K metrics (see subsection 2.1.6) without A/B testing and without Interleave user interactions, while the online evaluation metrics are these metrics with A/B testing or with interleave user interactions. The A/B testing and Interleave procedures simulate the user interaction with the search result retrieved to the user, while user is replaced by Click models. In A/B testing, user Click model chooses (clicks or picks) the preferred documents from two search result list produced by ranking models A and B. Then, fitness evaluation metric such as MAP evaluates the two ranking models A and B based on the clicked (picked) documents in the search ranked lists. The ranking model that has search ranked list containing more clicked (picked) of relevant documents based on evaluation metric is selected for the next learning iteration. The Interleave procedure is different from A/B testing by combining the two search result ranked list into one search result ranked list and then Click model is selected from one list and finally the preference between the two ranking models is based on the number of clicked documents from each ranking model. According to Kharitonov (Kharitonov, 2016; Hofmann, 2013) Probabilistic Interleave is more effective to represent the user interaction with the IR system than A/B test in online LTR. The offline and online evaluation metrics are based on the relevance labels existing in the training set. In addition, offline evaluation metrics are the most reliable metrics for comparing between CI and ML techniques due to online evaluation metric limitations (Al-Maskari and Sanderson, 2011; Kharitonov, 2016). This chapter shows the capability of ES-Rank to optimise online LTR click models using five evaluation metrics (MAP, NDCG, P@10, RR@10 and Err@10). Firstly, the LTR model is initialised from

online LTR model based on the Dependent Click Model (DCM), while Probabilistic Interleave is used to modify the ranked query-document pairs list based on the DCM model before evaluation. Then, the ranking model weights of the training set features, produced by online LTR Lerot package, were used as the initial chromosome vector for (1+1)-Evolutionary Strategy.

For online LTR, there is a randomised procedure to produce another LTR model from the current learning one. Then, two ranking models were introduced to Probabilistic Interleave to merge them into one ranked list. Then, the DCM is used to re-order its query-document pairs. This procedure is repeated until a better ranking model produced using DCM model and Probabilistic Interleave fitness evaluation. Then, the weights of features in the final online ranking model are assigned as an initial chromosome vector for ES-Rank. This chapter begins with the definition of the used click model in section 8.2. Then, the results are discussed in section 8.3.

8.2 Dependent Click Model

The Cascade Click Model (CCM) assumes that a user scans the ranked retrieved document list from top to bottom until she/he finds one relevant document and click it (Craswell et al., 2008). However, the user may be required to browse more than one document from the retrieved list until she/he is satisfied. Thus, (Guo et al., 2009) proposed a more sophisticated click model which is an extension for CCM. This model is called Dependent Click Model (DCM). The DCM is CCM with multiple clicks until the user is satisfied with the browsed relevant clicked documents from the ranked retrieved list. The DCM considers the position of the clicked documents (instances), the average of user clicks and the document examination parameters in its calculation. Assume μ_i is the position-dependent parameter that is used to reflect the probability that the user would like to see more documents after a click at position i. If the user does not click the document, the next document will be examined with probability one. The parameters μ are a set of shared behaviour parameters for a user over multiple query sessions. The Click and the Examination probabilities in DCM can be demonstrated in the following recursive process

($1 \leq i \leq M$, where M is the number of documents in the search result):

$$\begin{aligned} E_{d_{1,1}} &= 1, \\ C_{d_{i,i}} &= E_{d_{i,i}} \cdot r_{d_i}, \\ E_{d_{i+1,i+1}} &= \mu_i \cdot C_{d_{i,i}} + (E_{d_{i,i}} - C_{d_{i,i}}). \end{aligned} \tag{8.2.1}$$

where $E_{d_{i,i}}$ is the probability of examination of document d_i, while $C_{d_{i,i}}$ is the click probability for document d_i, r_{d_i} is the relevance degree of document d_i and μ_i is the user behaviour parameter for browsing (examining) document i.

The DCM model can be explained as follows. The click probability of the document in the search result is determined by its relevance, after the user examination. Then, after clicking on a document, the probability for next document examination is determined by μ_i. From equation 8.2.1, the examination and click formula for document $d_{i,i}$ are as follows:

$$\begin{aligned} E_{d_{i,i}} &= \prod_{j=1}^{i-1}(1 - r_{d_j} + \mu_j \cdot r_{d_j}), \\ C_{d_{i,i}} &= r_{d_i} \cdot \prod_{j=1}^{i-1}(1 - r_{d_j} + \mu_j \cdot r_{d_j}). \end{aligned} \tag{8.2.2}$$

If the actual click event use $\{C_1, C_2, ..., C_M\}$ in query session, while M is the number of documents in the search result (ranked retrieved list) and the r_{d_i} is the document relevance of document d_i. The log-likelihood for DCM clicks in each query session before position l is given by (Guo et al., 2009):

$$\begin{aligned} L_{DCM} = \sum_{i=1}^{l-1} \Big(C_i * (log(r_{d_i}) + log(\mu_i)) + (1 - C_i) * log(1 - r_{d_i}) \Big) \\ + C_l * log(r_{d_l}) + (1 + C_l) * log(1 - r_{d_l}) + log\Big(1 - \mu_l + \mu_l * \prod_{j=l+1}^{n} (1 - r_{d_j})\Big), \end{aligned} \tag{8.2.3}$$

$$L_{DCM} \geq \sum_{i=1}^{l} C_i * log(r_{d_i}) + (1 + C_i) * log(1 - r_{d_i}) + \sum_{i=1}^{l-1} C_i * log(\mu_i) + log(1 - \mu_l) \quad (8.2.4)$$

If the query session does not receive any clicks, then L_{DCM} is a particular case with $l = M, C_l = \mu_l = 0$. A learning approach is carried for DCM learning by maximising the lower bound of L_{DCM} in equation 8.2.4. The user behaviour parameter is estimated by:

$$\mu_i = 1 - \frac{No.\ of\ query\ sessions\ when\ last\ clicked\ position\ is\ i}{No.\ of\ query\ sessions\ when\ position\ i\ is\ clicked} \quad (8.2.5)$$

For $1 \leq i \leq M - 1$, the equation 8.2.5 gives the empirical probability of position i being a non-last-clicked position for all query sessions in the training set. Finally, the sampling procedure of DCM model for examination variables E_i and click variables C_i, while the users examine the search result list one-by-one from the top position until the end is given by:

1: $E_1 \leftarrow 1$;
2: **if** $E_i == 0$, **then**
3: $\quad C_i \leftarrow 0$,
4: $\quad E_{i+1} \leftarrow 0$,
5: **else**
6: $\quad C_i \approx Bernoulli(r_{d_i})$,
7: $\quad E_{i+1} \approx Bernoulli(1 - C_i + \mu_i * C_i)$
8: **end if**

where $Bernoulli()$ is the Bernoulli probabilistic distribution (Teugels, 1990). For the most state-of-the-art click models, (Chuklin et al., 2015) surveyed their details.

Table 8.1: Average of the Training Data Results of Nine LETOR Datasets

Average Training Data Results of the Datasets										
Evaluation Metric	Approach	MQ2008	MQ2007	HP2004	TD2004	NP2004	HP2003	TD2003	NP2003	Ohsumed
MAP	ES-Click	0.49	0.467	0.822	0.247	0.23	0.817	0.336	0.729	0.457
	DCM Model	0.309	0.329	0.112	0.027	0.044	0.044	0.055	0.232	0.327
NDCG@10	ES-Click	0.512	0.442	0.833	0.357	0.366	0.823	0.402	0.773	0.474
	DCM Model	0.34	0.266	0.122	0.0313	0.057	0.048	0.062	0.251	0.252
P@10	ES-Click	0.277	0.382	0.101	0.284	0.277	0.111	0.201	0.095	0.537
	DCM Model	0.211	0.266	0.019	0.028	0.048	0.008	0.034	0.035	0.327
RR@10	ES-Click	0.55	0.575	0.819	0.628	0.681	0.806	0.672	0.744	0.735
	DCM Model	0.345	0.39	0.108	0.075	0.12	0.042	0.094	0.226	0.594
Err@10	ES-Click	0.023	0.024	0	0	0	0	0	0.00	0.036
	DCM Model	0.058	0.061	0.00	0.001	0.007	0.003	0.007	0.014	0.113

Table 8.2: Average of the Test Data Results of Nine LETOR Datasets

Average Test Data Results of the datasets										
Evaluation Metric	Approach	MQ2008	MQ2007	HP2004	TD2004	NP2004	HP2003	TD2003	NP2003	Ohsumed
MAP	ES-Click	0.473	0.457	0.712	0.223	0.197	0.736	0.256	0.66	0.446
	DCM Model	0.298	0.326	0.117	0.025	0.048	0.045	0.055	0.21	0.332
NDCG@10	ES-Click	0.493	0.43	0.691	0.292	0.312	0.78	0.286	0.714	0.452
	DCM Model	0.325	0.262	0.131	0.033	0.059	0.049	0.079	0.233	0.262
P@10	ES-Click	0.271	0.38	0.09	0.249	0.215	0.102	0.156	0.085	0.496
	DCM Model	0.207	0.268	0.021	0.029	0.045	0.009	0.05	0.034	0.343
RR@10	ES-Click	0.526	0.552	0.699	0.452	0.535	0.761	0.353	0.669	0.628
	DCM model	0.329	0.381	0.114	0.082	0.124	0.046	0.167	0.206	0.623
Err@10	ES-Click	0.025	0.026	0.00	0.00	0.00	0.00	0.003	0.00	0.056
	DCM Model	0.054	0.06	0.00	0.00	0.01	0.003	0.014	0.013	0.118

8.3 Implementation and Experimental Results

In this section, the summary of the results obtained from applying this approach on nine dataset benchmarks were presented. Lerot and then ES-Rank library packages (Schuth et al., 2013) were used for the implementation of this experiments. The Lerot package is an online LTR package that is used for producing ranking models for the LTR datasets. The experimental settings of ES-Rank and Lerot are the default of these packages. The number of evolving iteration is 1300 iteration in ES-Rank, while configuration setting in Lerot is the default one provided with the package. The ES-Rank takes the ranking model produced by Lerot package using DCM model as initial parent chromosome. Then, it evolves a better ranking model for each training dataset using various evaluation fitness metrics. The p-value of the null hypothesises of paired t-test is 10^{-15} for these results. This indicates the high degree of confidence in the result improvements for the various experimental settings between ESClick and DCM.

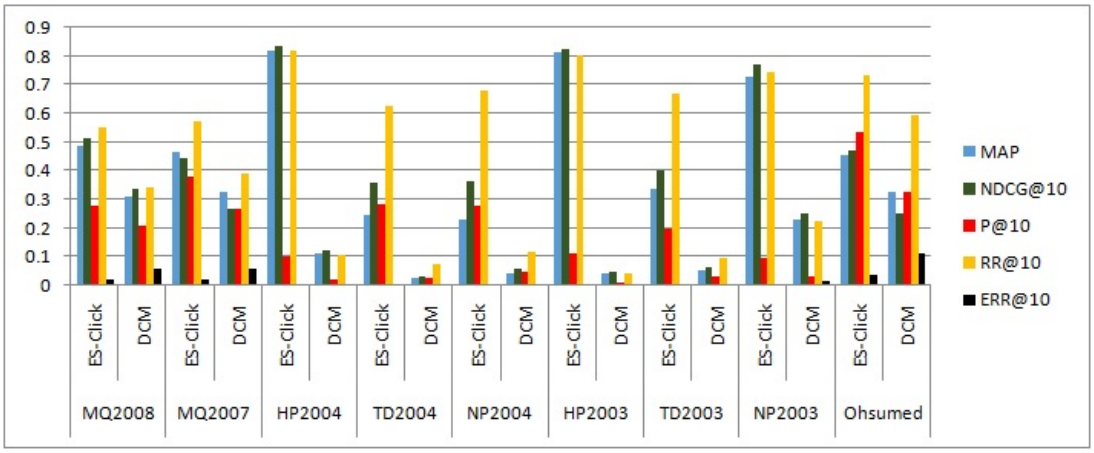

Figure 8.1: Illustrating the Table 8.1 performance for on the LETOR datasets.

From Table 8.1 and 8.2, the ES-Rank can improve the DCM ranking model when applied on the training and test data of nine well-known LETOR dataset benchmarks. This approach used five fitness evaluation metrics to show the optimisation capability of ES-Rank. The MAP, NDCG@10, P@10, RR@10 and Err@10 are the evaluation fitness metrics used in this experiment. The highest values of each DCM and ES-Click in the MAP, NDCG@10, P@10 and RR@10 results are the best accuracy results, while the lowest value of each DCM and ES-Click on Err@10 is the best accuracy results. For example, the training ES-Click result value in MQ2008 is 0.4896, while the corresponding DCM value on the same training set (MQ2008) is 0.3094. Thus, ES-Click outperforms DCM model on MQ2008. Similarly, the other datasets have similar accuracy on the rest of benchmarks on the training and test data values. Thus, ES-Click in general outperformed DCM model. The LETOR dataset benchmarks used on this experiment are MQ2008, MQ2007, HP2004, TD2004, NP2004, HP2003, TD2003, NP2003 and Ohsumed benchmarks. The details of these results on the five folds in the nine benchmarks are shown on the Appendix C in Tables C.1 to C.18.

Figures 8.1 and 8.2 illustrates the bar chart for each fitness evaluation metric results reported in the Tables 8.1 and 8.2. In the figures, higher values correspond to better performance, while in the last figure lower values correspond to better performance. Figure 8.1 represents the performance on the training sets, while Figure 8.2 represents the predictive performance on testing sets. From these figures it can be observed that the ES-Click technique exhibits the overall best performance among all techniques.

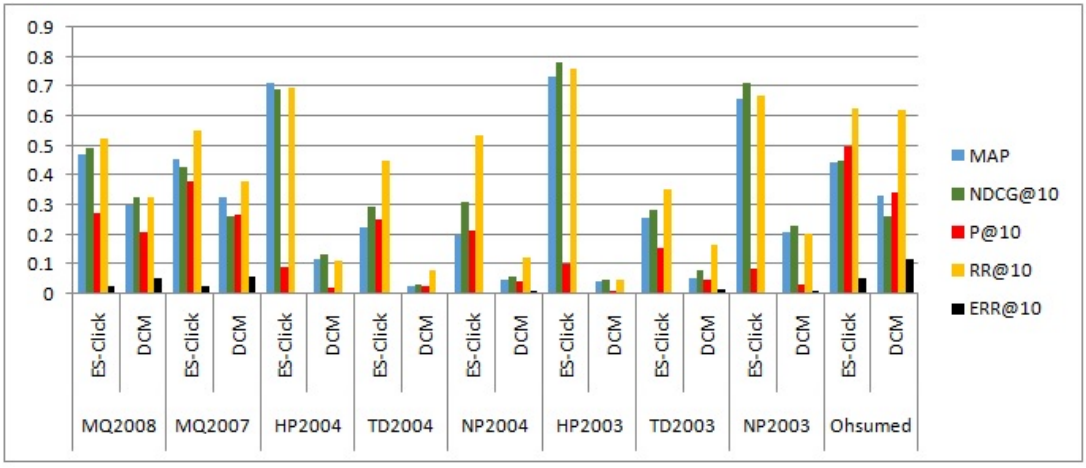

Figure 8.2: Illustrating the Table 8.2 performance for on the LETOR datasets.

8.4 Chapter Summary and Conclusion

To sum up, the ES-Rank can improve the performance of DCM ranking model using five evaluation fitness metrics. This approach is called ES-Click (Evolutionary Strategy Dependent Click Model). It started with learning DCM ranking model from the benchmarks and then the ES-Rank used the DCM ranking models to evolve better ranking models. In general, the ES-Click ranking model outperformed the DCM model in all cases. The Lerot and ES-Rank packages are used in this chapter for producing DCM ranking model and improving it using ES-Rank. Nine LETOR datasets are used in this experiment which show the powerful of ES-Rank to optimise a DCM model. Thus, Evolutionary Strategy can improve the online LTR models as it showed its capability to improve the offline LTR models.

Chapter 9

Conclusions and Future Work

9.1 Introduction

This chapter provides the answers to the research questions presented in Chapter 1. It also presents the conclusion and the summary of contributions of the whole thesis. Moreover, it discusses the possible extensions for the future related research work in the field.

9.2 Answers to Research Questions

This section summarises the answers for the main research questions that were presented in Chapter 1. These questions and their answers are as follows:

1. Is there a limitation in Evolutionary and Machine Learning (EML) techniques on IR systems for TVM representation (Bag-of-Words) ? and What is the need for mathematical and non-learning term-weighting schemes? What is the importance of relevance judgement and relevance label in evolutionary and machine learning approaches?

   Yes, there is a limitation for using EML techniques at the beginning of the IR system. This limitation applies for partially judged and non-judged test collections. The reason for this limitation is that the fitness and loss functions use the relevance labels to check the quality of the solutions in every learning iteration of any EML technique. Thus, the non-existence of relevance judgement at the beginning of IR

system restricts the use of EML technique and its quality for optimising the IR accuracy (effectiveness). In Chapter 5, the limitation of applying EML was identified in the well-known test collections by the IR community. In addition, TF-ATO has been proposed as a new mathematical term-weighting scheme which is more efficient than the well-known TF-IDF weighting scheme.

2. What is the limitation of static collection characteristics on different IR weighting functions on Term Vector Models? How will this parameter affect on dynamic variation in test collection?

   The static test collection characteristics have been used in various MM (Mathematical Models such as TF-ATO and TF-IDF) and EML term-weighting schemes such as TF-IDF, Okapi, evolving term-weighting scheme using genetic programming and local search techniques. They usually use the number of documents in the collection as a parameter. However, this parameter should be a variable number that changes by adding and removing documents from the collection. The high variations in the test collection size causes a negative effect on the IR system. This impact is illustrated by the dynamic variation using TF-IDF and TF-ATO in Chapter 5. The only very large dynamic variation in the test collections affects on the IR accuracy.

3. What is the impact of the pre-processing procedure (stop-word removal) in term-weighting functions?

   The stop-word removal has a positive impact for improving the IR system. The non-use of stop-word removal has a strong negative effect on TF-ATO more than TF-IDF. The reason for this issue is that the TF-IDF weighting scheme removes some stop-words from the collection. The word (term) has 0 value as TF-IDF weight when the term is repeated in all documents in the collection at least one time. Furthermore, the term-weighting scheme used for creating TREC pooled collections are usually TF-IDF or its variations. This causes another bias in the collections related to the chosen term-weighting scheme. However, TF-ATO performs better with the Discriminative Approach (DA) which can be considered as collection domain of stop-words removal. The DA increases the accuracy (effectiveness) using TF-IDF

or TF-ATO, because its capability for removing some noisy keywords. Chapter 5 demonstrated the impact of the stop-words removal and the DA for increasing the performance of the IR systems.

4. What is the importance of EML techniques in IR to overcome the pre-processing (stop-words removal and stemming) impact for creating effective IR system?

   Usually, the use of stop-words removal or stemmer from different topic domains has a negative effect of the performance of the IR system. This is because some discriminative terms were removed or stemmed which were indicated as the topic of the document. Previous studies proved that the EML techniques can adjust the the similarity matching between the relevant documents and the queries. Consequently, EML techniques can improve the accuracy of the IR system using the relevance judgement. Chapter 4 states the capability of EML techniques to improve IR systems.

5. What are the limitations of applying EML techniques on IR systems for FVM representation (Bag-of-Features)?

   The previous studies did not indicate fixed settings to evaluate the performance and effectiveness of various EML techniques. Furthermore, some EML techniques such as RankNET, ListNET among others did not consider the over-fitting and under-fitting problems in the sampling process in each learning iteration. Moreover, pairwise approaches have a limitation for producing an accurate ranking model for graded relevance labels. Furthermore, there is a limitation for creating FVM datasets and using EML techniques at the beginning of IR systems. On the other hand, most of EML techniques consumes large computational runtime. These issues motivate to propose ES-Rank application as an effective EML technique in Learning to Rank problem in IR.

6. How is the adaptive (1+1)-evolutionary techniques can be used to improve the IR system with the lowest problem size and the lowest computational time?

   The (1+1)-evolutionary techniques are similar to the other population-based EML techniques for improving the IR accuracy based on the relevance judgement.

The (1+1)-evolutionary techniques use one parent chromosome and one offspring (child) chromosome to evolve a better solution. These techniques are using less memory than the population-based EML techniques. Chapters 6, 7 and 8 proposed (1+1)-Evolutionary Gradient Strategy and (1+1)-Evolutionary Strategy to optimise IR systems with various novel methods. To the best of my knowledge, these methods have not been used before in the literature of the IR research field. These techniques outperformed the TF-IDF, Okapi and fourteen EML techniques in TVM and FVM approaches.

7. What is the importance of the initialisation procedure in (1+1)-Evolutionary Strategy technique? Chapter 7 showed the importance of the initialisation procedure in ES-Rank. The appropriate initialisation procedure improves the performance and accuracy of the ES-Rank. In Chapter 7, the zero values, the ranking models produced by the linear regression and the support vector machine have been used as initialisation values in ES-Rank. The best performance and the best accuracy produced by linear regression as initialisation procedure.

8. Can (1+1)-Evolutionary Strategy improve user simulation click ranking model?

   Yes, The linear ranking model from *Dependent Click Model (DCM)* has been used as an initialisation procedure in ES-Rank application. We called this technique as ES-Click. Chapter 8 illustrated the ability of ES-Rank ((1+1)-Evolutionary Strategy) to improve DCM model in both training and testing dataset.

9.3 Review of Contributions

During the progression of this thesis there was a set of achievements in terms of developing new techniques and publications for optimising IR systems. The following is a list of contributions that identified these achievements:

1. Based on an analysis of commonly used test collections in Chapter 5, we provided an argument in favour of using heuristic mathematical TWS at the beginning of the IR system. Then, the IR system can be used for gathering the relevance judgement

from the users through users historical data. Then, the IR system can adapt its performance using EML techniques.

2. A new non-learning TWS is proposed which is called *Term Frequency With Average Term Occurrence (TF-ATO)* with a *discriminative approach* to remove less significant weights from the documents. Chapter 5 described and studied the performance of TF-ATO compared to the widely used TF-IDF approach using various types of test collections such as sampled and pooled (Soboroff, 2007; Hersh et al., 1994). Our experimental results showed that the proposed TF-ATO gives higher effectiveness in both cases of static and dynamic test collections. Thus, TF-ATO can be used at the early IR system stage for gathering the user interaction feedback data.

3. The impact of our discriminative approach and the stop-words removal process on the IR system effectiveness and performance when using TF-ATO and TF-IDF have been demonstrated in Chapter 5. These two processes have a positive impact on both TWSs for improving the IR performance and effectiveness. Thus, they are recommended to be used at the early stage of the IR system to improve the accuracy of the IR system without prior knowledge about relevance judgement values.

4. This thesis proposed a new method for evolving better document representations in the collection through global term weights. This method is an efficient EML technique in terms of computer memory usage. This is accomplished by evolving only the Global Term Weights (GTWs) of the collection rather than evolving representations for the whole collection as is typically done with previous EC approaches in the literature. Hence, the main contribution of Chapter 6 is the development of a *(1+1)-Evolutionary Gradient Strategy ((1+1)-EGS)* with *Adaptive Ziggurat Random Gaussian Mutation* (Kuo and Zulvia, 2015; Doornik, 2005; Loshchilov, 2014) to evolve GTWs. The proposed methodology reduces the problem size, from evolving $(N \times M)$ document representation vectors to evolving $(1 \times M)$ vector, where N is the number of documents in the collection and M is the number of index terms in the collection. This method considers the limitation of the relevance judgement of the test collections that illustrated in subsection 3.1.3.

5. Chapter 7 presented an Evolutionary Strategy (ES) technique to tackle the LTR problem. The proposed technique is called ES-Rank and includes evolving a vector of weights where each weight represents the importance of a desirable feature. The mutation step-size in ES-Rank has been tuned based on preliminary experimentation. Details of the proposed method have been presented in section 7.2. In order to assess the performance of ES-Rank, MAP, P@10, RR@10, NDCG@10, RMSE (Qin et al., 2010) were used in the comparison carried out against fourteen state-of-the-art LTR approaches from the literature. Experimental results in Chapter 7 show that ES-Rank outperforms other methods in terms of MAP, P@10, RR@10, NDCG@10 and RMSE. Furthermore, most of the other techniques 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 (Qin et al., 2010) 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).

6. The importance of the initialisation procedure in Evolutionary Strategy (ES) in ES-Rank is presented to tackle the LTR problem. The initialisation procedure in ES-Rank has been tuned based on Linear Regression and Support Vector Machine ranking model. In order to assess the performance of ES-Rank, MAP, RMSE, NDCG@10, RR@10 and P@10 at top-10 query-document pairs retrieved (Liu, 2011; Li, 2014) were used. The comparison has been carried out against fourteen state-of-the-art LTR approaches from the literature. Experimental results in Chapter 7 show that the use of machine learning ranking model as an initialisation procedure has a positive impact on ES-Rank technique for Learning to Rank problem. Furthermore, the better initialisation procedure using *Linear Regression (LR)* machine learning ranking model has a better positive impact on improving ES-Rank in most cases than *Support Vector Machine (SVM)*.

7. ES-Rank technique has been used to optimise Click Dependent Model to tackle the Click Ranking Models problem. The experimental results in Chapter 8 show the

improvements of ES-Rank in DCM model on training and testing LETOR datasets.

9.4 Conclusions

This thesis has argued that using EML techniques in evolving the document representations and the learning to rank models improves the effectiveness of IR systems. However, the usage of EML techniques undergo the limitation of the existing of relevance judgement at the early stage of creating the IR system. Thus, there is a need for proposing an effective mathematical term-weighting scheme at the beginning of the IR system to gather user relevance judgements from historical user interactions. Then, it is possible to propose a partially evolved technique for optimising the document weight representation for obtaining a better accuracy for partially judged test collections. This thesis proposed a novel partially evolved technique with efficient mathematical model to improve the IR system effectiveness. This technique uses (1+1)-Evolutionary Gradient Strategy ((1+1)-EGS) and the proposed mathematical TWS in this thesis is TF-ATO (Term Frequency-Average Term Occurrence). The partially evolved technique used less memory than Genetic Programming techniques in the literature and it outperformed TF-IDF, Okapi-BM25 and TF-ATO with DA (Mathematical and Probabilistic Models) in terms of MAP and NDCG.

Furthermore, this thesis also proposed ES-Rank technique to optimise the IR systems through FVM approach in LTR problem. This technique uses lower problem size than other EML techniques in the literature. It is also faster and more efficient than other EML techniques. In this thesis, fourteen EML techniques have been used in a comparison with ES-Rank to learn the best ranking model for the LETOR datasets. From the results of this comparison, ES-Rank outperformed the fourteen EML techniques in the accuracy and the computational run-time. The accuracy in this comparison includes five fitness evaluation metrics which are: MAP, NDCG@10, P@10, RR@10 and Err@10. This comparison is detailed in Chapter 7. This Chapter also discussed the importance of the initialisation procedure in ES-Rank. Two machine learning techniques have been used to create initial weight representations for the initial parent chromosome in ES-Rank. A

comparison between these two initialisation procedures and the zero value initialisation procedure has been carried out. This comparison has been used to examine the impact of the initialisation procedure on the accuracy. Additional strength for the ES-Rank is the evolving iteration in this technique considers all training data instance for checking the quality of the evolved ranking model, while most of EML techniques consider a sample of the training data instance in each learning iteration. This strength increases the accuracy and the convergence for better evolved ranking model than EML techniques.

9.5 Practical Implications and Future Work

The work introduced in this thesis can be regarded as a step of research findings for using EML techniques in IR systems. Thus, this research can be considered the start point of exploring a new perspective of establishing the IR systems based on EML techniques. Future extensions of the thesis work can be divided in the following research points:

1. One of the research limitations are based on the significant runtime for evolving GTWs using cumulative Cosine similarity fitness function. This can be reduced by using less complex similarity measures. An extension research work for evolving GTW can involve the examination of the proposed approach using city block function, distance function, MAP and NDCG@10 as objective functions. This extended work will investigate these objective functions for better performance. Moreover, combining machine learning techniques with (1+1)-EGS for evolving the GTWs may reduce computational run-time and may give better IR effectiveness than when using an (1+1)-Evolutionary Gradient Strategy.

2. Another research limitation is the method of evolving GTWs has not been tested with machine learning techniques. Learning to Weight using EML techniques can be another extension for evolving GTW. However, the textual test collections are partially judged with small number of fully judged documents. Thus, this research costs money and effort to apply it regardless of the capability of re-using the same Learning to Rank packages in Learning to Weight.

3. The ES-Rank can be improved using the acceptance of weak offspring under spe-

cific cooling temperature. This technique may converge the evolved ranking model to global optima faster than ES-Rank itself. This technique will be developed as a future work. This technique is called Simulated Annealing Strategy.

4. ES-Rank can use a different ranking model from different EML technique than LR and SVM as an initial chromosome. This initialisation may improve the accuracy of ES-Rank.

5. ES-Rank can be extended to multi-objective optimisation problem by using three or more evaluation fitness metrics (such as MAP, NDCG@10 with Err@10) as objective functions. This will be the first multi-objective technique in LTR problem. However, the computational run-time will increase.

6. Various online LTR click models can use ES-Rank technique for improving the accuracy in their online LTR techniques. The click models simulates the user behaviours on IR systems.

Bibliography

Al-Maskari, A. and Sanderson, M. (2011), 'The effect of user characteristics on search effectiveness in information retrieval', *Information Processing & Management* **47**(5), 719 – 729. Managing and Mining Multilingual Documents.

Arnold, D. V. and Salomon, R. (2007), 'Evolutionary gradient search revisited', *IEEE Transactions on Evolutionary Computation* **11**(4), 480–495.

Back, T., Foussette, C. and Krause, P. (2013), *Contemporary Evolution Strategies*, Natural Computing Series, Springer.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999), *Modern Information Retrieval*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (2011), *Modern Information Retrieval - the concepts and technology behind search, Second edition*, Pearson Education Ltd., Harlow, England.

Belkin, N. J., Cool, C., Croft, B. and Callan, J. P. (1993), The effect multiple query representations on information retrieval system performance, *in* 'Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '93, ACM, New York, NY, USA, pp. 339–346.

Beyer, H.-G. and Schwefel, H.-P. (2002), 'Evolution strategies - A comprehensive introduction', *Natural Computing* **1**, 3–52.

Bollegala, D., Noman, N. and Iba, H. (2011), RankDE: Learning a ranking function for information retrieval using differential evolution, *in* 'Proceedings of the 13th An-

nual Conference on Genetic and Evolutionary Computation', GECCO '11, ACM, New York, NY, USA, pp. 1771–1778.

Bookstein, A. (1980), 'Fuzzy requests: An approach to weighted boolean searches', *Journal of the American Society for Information Science* **31**(4), 240–247.

Bordogna, G., Carrara, P. and Pasi, G. (1995), Fuzzy approaches to extend boolean information retrieval, *in* P. Bosc and J. Kacprzyk, eds, 'Fuzziness in Database Management Systems', Vol. 5 of *Studies in Fuzziness*, Physica-Verlag HD, pp. 231–274.

Bordogna, G. and Pasi, G. (1993), 'A fuzzy linguistic approach generalizing boolean information retrieval: A model and its evaluation', *Journal of the American Society for Information Science* **44**, 70–82.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32.

Brownlee, J. (2017), 'Overfitting and underfitting with machine learning algorithms'. **URL:** *https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/*

Buckley, C., Dimmick, D., Soboroff, I. and Voorhees, E. (2007), 'Bias and the limits of pooling for large collections', *Information Retrieval* **10**(6), 491–508.

Buettcher, S., Clarke, C. L. A. and Cormack, G. V. (2010), *Information Retrieval: Implementing and Evaluating Search Engines*, The MIT Press.

Burges, C. J. C. (2010), From RankNet to LambdaRank to LambdaMART: An overview, Technical report, Microsoft Research.

Burges, C. J. C., Ragno, R. and Le, Q. V. (2006), Learning to rank with nonsmooth cost functions, *in* 'Proceedings of the 19th International Conference on Neural Information Processing Systems', NIPS'06, MIT Press, Cambridge, MA, USA, pp. 193–200.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G. (2005), Learning to rank using gradient descent, *in* 'Proceedings of the 22Nd International Conference on Machine Learning', ICML '05, ACM, New York, NY, USA, pp. 89–96.

Burke, E. K., Newall, J. P. and Weare, R. F. (1998), 'Initialization strategies and diversity in evolutionary timetabling', *Evolutionary Computation* **6**(1), 81–103.

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

Chai, T. and Draxler, R. R. (2014), 'Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature', *Geoscientific Model Development* **7**(3), 1247–1250.

Chang, C. H. and Hsu., C. C. (1999), The design of an information system for hypertext retrieval and automatic discovery on WWW, PhD thesis, National Taiwan University.

Chapelle, O. and Chang, Y. (2011), Yahoo! learning to rank challenge overview, *in* 'Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010', pp. 1–24.

Chen, S. F. and Goodman, J. (1996), An empirical study of smoothing techniques for language modeling, *in* 'Proceedings of the 34th Annual Meeting on Association for Computational Linguistics', ACL '96, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 310–318.

Chuklin, A., Markov, I. and Rijke, M. d. (2015), 'Click models for web search', *Synthesis Lectures on Information Concepts, Retrieval, and Services* **7**(3), 1–115.

Cleverdon, C. W. (1960), Aslib cranfield research project: Report on the first stage of an investigation into the comparative efficiency of indexing systems, Technical report, College of Aeronautics, Cranfield.

Cooper, W. S. (1991), Some inconsistencies and misnomers in probabilistic information retrieval, *in* 'Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', ACM New York, Inc. New York, NY, USA, pp. 57–61.

Cooper, W. S. (1994), The formalism of probability theory in ir: a foundation or an encumbrance?, *in* 'SIGIR '94 Proceedings of the 17th annual international ACM SIGIR

conference on Research and development in information retrieval', Springer-Verlag New York, Inc. New York, NY, USA, pp. 242 – 247.

Cooper, W. S. (1995), 'Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval', *ACM Trans. Inf. Syst.* **13**(1), 100–111.

Cooper, W. S., Gey, F. C. and Dabney, D. P. (1992), Probabilistic retrieval based on staged logistic regression, *in* 'SIGIR '92 Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval', ACM New York, NY, USA, pp. 198–210.

Cordon, O., Herrera-Viedma, E., Lopez-Pujalte, C., Luque, M. and Zarco, C. (2003), 'A review on the application of evolutionary computation to information retrieval', *International Journal of Approximate Reasoning* **34**(2-3), 241–264.

Cover, T. M. and Thomas, J. A. (1991), *Elements of Information Theory*, Wiley-Interscience, New York, NY, USA.

Craswell, N., Zoeter, O., Taylor, M. and Ramsey, B. (2008), An experimental comparison of click position-bias models, *in* 'Proceedings of the 2008 International Conference on Web Search and Data Mining', WSDM '08, ACM, New York, NY, USA, pp. 87–94.

Cummins, R. (2008), The Evolution and Analysis of Term-Weighting Schemes in Information Retrieval, PhD thesis, National University of Ireland, Galway.

Cummins, R. (2016), A study of retrieval models for long documents and queries in information retrieval, *in* 'Proceedings of the 25th International Conference on World Wide Web', WWW '16, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, pp. 795–805.

Cummins, R. and O'Riordan, C. (2006), 'Evolving local and global weighting schemes in information retrieval', *Information Retrieval* **9**(3), 311–330.

Cummins, R., Paik, J. H. and Lv, Y. (2015), 'A polya urn document language model for improved information retrieval', *ACM Trans. Inf. Syst.* **33**(4), 21:1–21:34.

Dang, V. (2016), 'RankLib, http://www.cs.umass.edu/ṽdang/ranklib.html'.

Diaz-Gomez, P. A. and Hougen, D. F. (2007), Initial population for genetic algorithms: A metric approach., *in* 'Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods GEM', pp. 43–49.

Doornik, J. A. (2005), 'An improved ziggurat method to generate normal random samples'.

Escalante, H. J., Garcia-Limon, M. A., Morales-Reyes, A., Graff, M., Montes-Y-Gomez, M., Morales, E. F. and Martinez-Carranza, J. (2015), 'Term-weighting learning via genetic programming for text classification', *Knowledge-Based Systems* **83**, 176 – 189.

Fan, W., Gordon, M. D. and Pathak, P. (1999), Automatic generation of matching functions by genetic programming for effective information retrieval, *in* W. D. Haseman and D. L. Nazareth, eds, 'Proceedings of the 1999 Americas Conference on Information Systems', Milwaukee, WI, USA, pp. 49–51.

Fan, W., Gordon, M. D. and Pathak, P. (2000), Personalization of search engine services for effective retrieval and knowledge management, *in* 'Proceedings of the Twenty First International Conference on Information Systems', ICIS '00, Association for Information Systems, Atlanta, GA, USA, pp. 20–34.

Fan, W., Gordon, M. and Pathak, P. (2004), 'Discovery of context-specific ranking functions for effective information retrieval using genetic programming', *Knowledge and Data Engineering, IEEE Transactions on* **16**(4), 523–527.

Fernandez-Reyes, F. C., Valadez, J. H. and Suarez, Y. G. (2015), Term dependence statistical measures for information retrieval tasks, *in* G. Sidorov and S. N. Galicia-Haro, eds, 'Advances in Artificial Intelligence and Soft Computing: 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, October 25-31, 2015, Proceedings, Part I', Springer International Publishing, pp. 83–94.

Fox, C. (1989), 'A stop list for general text', *SIGIR Forum* **24**(1-2), 19–21.

Fox, C. (1992), Lexical analysis and stoplists, *in* W. B. Frakes and R. Baeza-Yates, eds, 'Information Retrieval', Prentice-Hall, Inc., Upper Saddle River, NJ, USA, pp. 102–130.

Fox, E. A. and Shaw, J. A. (1994), Combination of multiple searches, *in* 'The Second Text REtrieval Conference (TREC-2)', Vol. 500-215 of *NIST Special Publication*, NIST, pp. 243–252.

Freund, Y., Iyer, R., Schapire, R. E. and Singer, Y. (2003), 'An efficient boosting algorithm for combining preferences', *Journal of Machine Learning Research* **4**, 933–969.

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

Ganjisaffar, Y., Caruana, R. and Lopes, C. V. (2011), Bagging gradient-boosted trees for high precision, low variance ranking models, *in* 'Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '11, ACM, New York, NY, USA, pp. 85–94.

Glassgow, I. G. (2014), 'SMART System Document Collections'.
URL: *http://ir.dcs.gla.ac.uk/resources/test_collections/*

Gordon, M. (1988), 'Probabilistic and genetic algorithms in document retrieval', *Communication of ACM* **31**(10), 1208–1218.

Greengrass, E. (2000), Information retrieval : A survey, Technical report, University of Maryland, USA.
URL: *http://www.csee.umbc.edu/csee/research/cadip/readings/IR.report.120600.book.pdf*

Guo, F., Liu, C. and Wang, Y. M. (2009), Efficient multiple-click models in web search, *in* 'Proceedings of the Second ACM International Conference on Web Search and Data Mining', WSDM '09, ACM, New York, NY, USA, pp. 124–131.

Habernal, I., Sukhareva, M., Raiber, F., Shtok, A., Kurland, O., Ronen, H., Bar-Ilan, J. and Gurevych, I. (2016), New collection announcement: Focused retrieval over the web, *in* 'Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '16, ACM, New York, NY, USA, pp. 701–704.

Hansen, N. (2016), 'CMA-ES: Co-variance Matrix Adaptation Evolutionary Strategy'.
URL: *https://www.lri.fr/ hansen/cmaesintro.html*

Harman, D. (1993), Overview of the first trec conference, *in* 'Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '93, ACM, New York, NY, USA, pp. 36–47.

Harter, S. P. (1986), *Online Information Retrieval: Concepts, Principles, and Techniques*, Academic Press, Inc., Orlando, FL, USA.

Hersh, W., Buckley, C., Leone, T. J. and Hickam, D. (1994), Ohsumed: An interactive retrieval evaluation and new large test collection for research, *in* 'Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '94, Springer-Verlag New York, Inc., New York, NY, USA, pp. 192–201.

Hofmann, K. (2013), 'Fast and reliable online learning to rank for information retrieval', *SIGIR Forum* **47**(2), 140–140.

Ibrahim, M. and Carman, M. (2016), 'Comparing pointwise and listwise objective functions for random-forest-based learning-to-rank', *ACM Transaction of Information System* **34**(4), 20:1–20:38.

Ibrahim, O. A. S. and Landa-Silva, D. (2017), *(1+1)-Evolutionary Gradient Strategy to Evolve Global Term Weights in Information Retrieval*, Springer International Publishing, Cham, pp. 387–405.

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

Igel, C., Heidrich-Meisner, V. and Glasmachers, T. (2008), 'Shark', *Journal of Machine Learning Research* **9**, 993–996.

Igel, C., Suttorp, T. and Hansen, N. (2006), A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies, *in* 'Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation', GECCO '06, ACM, New York, NY, USA, pp. 453–460.

Islam, M. A. (2013), 'RankGPES: Learning to rank for information retrieval using a hybrid genetic programming with evolutionary strategies'.

Jarvelin, K. and Kekalainen, J. (October 2002), 'Cumulated gain-based evaluation of ir techniques', *ACM Transaction of Information System* **20**(4), 422–446.

Joachims, T. (2002), Optimizing search engines using clickthrough data, *in* 'Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '02, ACM, New York, NY, USA, pp. 133–142.

Joachims, T. (2015), 'Svmlight: Support vector machine for classification and ranking'. **URL:** *http://svmlight.joachims.org/*

Joachims, T. (2016), 'SVM-Rank: Support vector machine for ranking'. **URL:** *https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html*

Jones, K. S. (2004), 'A statistical interpretation of term specificity and its application in retrieval', *Journal of Documentation* **60**(5), 493–502.

Jones, K. S. and Willett, P., eds (1997), *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kharitonov, E. (2016), Using interaction data for improving the offline and online evaluation of search engines, PhD thesis, University of Glasgow.

Kraft, D. H. and Buell, D. A. (1983), 'Fuzzy sets and generalized boolean retrieval systems.', *International Journal of Man-Machine Studies* **19**, 45–56.

Kuo, R. and Zulvia, F. E. (2015), 'The gradient evolution algorithm', *Information Sciences* **316**, 246–265.

Kwok, K. L. (1997), 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', ACM, New York, NY, USA, pp. 34–41.

Le, K. N. (2011), A study of evolutionary multiobjective algorithms and their application to knapsack and nurse scheduling problems, PhD thesis, The University of Nottingham.

Le, Q. V. and Smola, A. J. (2007), 'Direct optimization of ranking measures', *CoRR* **abs/0704.3359**.
URL: *http://arxiv.org/abs/0704.3359*

Lee, J. H. (1994), Properties of extended boolean models in information retrieval, *in* 'Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '94, Springer-Verlag New York, Inc., New York, NY, USA, pp. 182–190.

Lee, J. H. (1995), 'Combining multiple evidence from different properties of weighting schemes', *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '95* pp. 180–188.

Lemur (2016*a*), 'http://boston.lti.cs.cmu.edu/clueweb12/'.

Lemur (2016*b*), 'http://lemurproject.org/clueweb12/specs.php'.

Lewis, D. D. (1997), 'Reuters-21578 text categorization test collection v1.0'.
URL: *http://kdd.ics.uci.edu/databases/reuters21578*

Li, H. (2014), *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*, Morgan and Claypool Publishers.

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

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

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

Liu, T.-Y. (2009), 'Learning to rank for information retrieval', *Foundations and Trends in Information Retrieval* **3**(3), 225–331.

Liu, T.-Y. (2011), *Learning to Rank for Information Retrieval*, Springer Berlin Heidelberg, Berlin, Heidelberg, chapter The LETOR Datasets, pp. 133–143.

Liu, T.-Y., Xu, J., Qin, T., Xiong, W. and Li, H. (2007), Letor: Benchmark dataset for research on learning to rank for information retrieval, *in* 'In Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval'.

Lo, R. T.-w., He, B. and Ounis, I. (2005), 'Automatically building a stopword list for an information retrieval system', *Digital Information Management: special issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR 2005)* **3**(1), 3–8.

Lorigo, L., Pan, B., Hembrooke, H., Joachims, T., Granka, L. and Gay, G. (2006), 'The influence of task and gender on search and evaluation behavior using google', *Information Processing & Management* **42**(4), 1123 – 1131.

Loshchilov, I. (2014), A computationally efficient limited memory cma-es for large scale optimization, *in* 'Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation', GECCO '14, ACM, New York, NY, USA, pp. 397–404.

Lovins, J. B. (1968), Development of a stemming algorithm, Technical report, Massachusetts Institute of Technology, Electronic Systems Lab.

Luhn, H. (1957), 'A statistical approach to mechanized encoding and searching of literary information', *IBM Journal of Research and Development* **1**(4), 309–317.

Makrehchi, M. and Kamel, M. S. (2008), Automatic extraction of domain-specific stopwords from labelled documents, *in* 'ECIR'08 Proceedings of the IR research, 30th European conference on Advances in information retrieval', Springer-Verlag Berlin, Heidelberg, pp. 222–233.

Manning, C. D., Raghavan, P. and Schutze, H. (2008), *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA.

Metzler, D. and Croft, W. B. (2007), 'Linear feature-based models for information retrieval', *Information Retrieval* **10**(3), 257–274.

Mick, J.-Y. L. (2016).
URL: *http://people.cs.nctu.edu.tw/ jylin/lagep/lagep.html*

Miller, S. J. (2006), 'The method of least squares'.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.710.4069*

Mohan, A. (2010), An empirical analysis on point-wise machine learning techniques using regression trees for web-search ranking, Master's thesis, School of Engineering and Applied Science, Washington University In ST. Louis, USA.

Mohan, A., Chen, Z. and Weinberger, K. (2011), Web-search ranking with initialized gradient boosted regression trees, *in* 'Journal of Machine Learning Research, Workshop and Conference Proceedings', Vol. 14, pp. 77–89.

Nopiah, Z. M., Khairir, M. I., Abdullah, S., Baharin, M. N. and Arifin, A. (2010), Time complexity analysis of the genetic algorithm clustering method, *in* 'Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation', ISPRA'10, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 171–176.

Oren, N. (2002), Reexamining tf.idf based information retrieval with genetic programming, *in* 'Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology', SAICSIT 2002, South African Institute for Computer Scientists and Information Technologists, Republic of South Africa, pp. 224–234.

Page, L., Brin, S., Motwani, R. and Winograd, T. (1998), The pagerank citation ranking: Bringing order to the web, *in* 'Proceedings of the 7th International World Wide Web Conference', Brisbane, Australia, pp. 161–172.

Paice, C. D. (1984), 'Soft evaluation of boolean search queries in information retrieval systems', *Information Technology Research Development Application* **3**(1), 33–41.

Pohl, S., Moffat, A. and Zobel, J. (2012), 'Efficient extended boolean retrieval', *IEEE Transactions on Knowledge and Data Engineering* **24**(6), 1014–1024.

Pohl, S., Zobel, J. and Moffat, A. (2010), Extended boolean retrieval for systematic biomedical reviews, *in* 'Proceedings of the Thirty-Third Australasian Conferenc on Computer Science - Volume 102', ACSC '10, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 117–126.

Ponte, J. M. and Croft, W. B. (1998), A language modeling approach to information retrieval, *in* 'Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '98, ACM, New York, NY, USA, pp. 275–281.

Porter, M. F. (1997), An algorithm for suffix stripping, *in* K. S. Jones and P. Willett, eds, 'Readings in Information Retrieval', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 313–316.

Qin, T. and Liu, T.-Y. (2013), 'Introducing LETOR 4.0 datasets', *CoRR* **abs/1306.2597**.
URL: *http://arxiv.org/abs/1306.2597*

Qin, T. and Liu, T.-Y. (2016), 'Introducing microsoft bing search engine datasets'.
URL: *https://www.microsoft.com/en-us/research/project/mslr/*

Qin, T., Liu, T.-Y., Xu, J. and Li, H. (2010), 'Letor: A benchmark collection for research on learning to rank for information retrieval', *Information Retrieval* **13**(4), 346–374.

Radwan, A. A. A., Abdel Latef, B. A., Ali, A. M. A. and Sadek, O. A. (2006), Using genetic algorithm to improve information retrieval systems, *in* C. Ardil, ed., 'Proceedings of World Academy of Science, Engineering and Technology'.

Reed, J., Jiao, Y., Potok, T., Klump, B., Elmore, M. and Hurson, A. (2006*a*), Tf-icf: A new term weighting scheme for clustering dynamic data streams, *in* '2006 5th International Conference on Machine Learning and Applications (ICMLA'06)', IEEE, pp. 258–263.

Reed, J. W., Jiao, Y., Potok, T. E., Klump, B. A., Elmore, M. T. and Hurson, A. R. (2006*b*), Tf-icf: A new term weighting scheme for clustering dynamic data streams, *in*

'Proceedings of the 5th International Conference on Machine Learning and Applications', ICMLA '06, IEEE Computer Society, Washington, DC, USA, pp. 258–263.

Rennie, J. (2015), '20 newsgroups document collection'.
URL: *http://qwone.com/ jason/20Newsgroups/*

Rijsbergen, C. J. V. (1979), *Information Retrieval*, Butterworth.

Robertson, S. E. (1991), 'On term selection for query expansion', *Journal of Documentation* **46**(4), 359–364.

Robertson, S. E. and Walker, S. (1994), Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval, *in* 'Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '94, Springer-Verlag New York, Inc., New York, NY, USA, pp. 232–241.

Robertson, S. E. and Walker, S. (2000), Microsoft cambridge at trec-9: Filtering track., *in* 'Text REtrieval Conference (TREC)'.

Robertson, S. E., Walker, S. and Beaulieu, M. (1998), Okapi at trec-7: Automatic ad hoc, filtering vlc and interactive track, *in* E. M. Voorheer and D. K. Harman, eds, 'NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)', NIST, Gaithersburg, MD, pp. 253–264.

Robertson, S. E., Walker, S., Hancock-Beaulieu, M. M., Jones, S. and Gatford, M. (1995), Okapi at (trec-3), *in* D. Harman, ed., 'Proceeding of Third Text REtrieval Conference TREC3', Gaithersburg, pp. 109–126.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. and Gatford, M. (1994), Okapi at (trec-3), *in* 'Proceedings of the Third Text REtrieval Conference', TREC-3, pp. 109–126.

Robertson, S., Walker, S., Beaulieu, M. M., Gatford, M. and Payne, A. (1995), Okapi at (trec-4), *in* 'NIST Special Publication 500-236: The Fourth Text REtrieval Conference (TREC-4)', pp. 73–96.

Robertson, S., Walker, S., Beaulieu, M. M., Gatford, M. and Payne, A. (1996), Okapi at (trec-4), *in* D. Harman, ed., 'Proceeding of Fourth Text REtrieval Conference TREC-4', Gaithersburg, pp. 73–96.

Robertson, S. and Zaragoza, H. (2009), 'The probabilistic relevance framework: Bm25 and beyond', *Foundation Trends Information Retrieval* **3**(4), 333–389.

Robertson, S., Zaragoza, H. and Taylor, M. (2004), Simple bm25 extension to multiple weighted fields, *in* 'Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management', CIKM '04, ACM, New York, NY, USA, pp. 42–49.

Saif, H., Fernandez, M., He, Y. and Alani, H. (2014), On stopwords, filtering and data sparsity for sentiment analysis of Twitter, *in* 'In LREC 2014, Ninth International Conference on Language Resources and Evaluation', Reykjavik, Iceland, pp. 810–817.

Salimans, T., Ho, J., Chen, X. and Sutskever, I. (2017), 'Evolution strategies as a scalable alternative to reinforcement learning', *CoRR* **abs/1703.03864**.

Salton, G. (1988), 'A simple blueprint for automatic boolean query processing', *Information Processing and Management* **24**(3), 269–280.

Salton, G. (2013), 'SMART System'.
URL: *ftp://ftp.cs.cornell.edu/pub/smart*

Salton, G. and Buckley, C. (1988), 'Term weighting approaches in automatic text retrieval', *Information Processing and Management: an International Journal* **24**(5), 513–523.

Salton, G. and Buckley, C. (1997), Improving retrieval performance by relevance feedback, *in* K. Sparck Jones and P. Willett, eds, 'Readings in Information Retrieval', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 355–364.

Salton, G., Fox, E. A. and Wu, H. (1983), 'Extended boolean information retrieval', *Communication of the ACM* **26**(11), 1022–1036.

Salton, G. and McGill, M. J. (1986), *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, NY, USA.

Salton, G., Wong, A. and Yang, C. S. (1975), 'A vector space model for automatic indexing', *Communication of the ACM* **18**(11), 613–620.

Sanderson, M. (2010), 'Test collection based evaluation of information retrieval systems', *Foundations and Trends in Information Retrieval* **4**(4), 247–375.

Schuth, A. (2016), Search Engines that Learn from Their Users, PhD thesis, Informatics Institute, University of Amsterdam.

Schuth, A., Hofmann, K., Whiteson, S. and de Rijke, M. (2013), Lerot: an online learning to rank framework, *in* 'Proceedings of the 2013 workshop on Living labs for information retrieval evaluation CIKM 2013, San Francisco, California, USA, November 1, 2013', pp. 23–26.

Schwefel, H. (1965), Kybernetische evolution als strategie der experimentellen forschung in die strömungstechnik, Diplomarbeit, TU Berlin. Diplomarbeit 233.

Schwefel, H.-P. (1981), *Numerical Optimization of Computer Models*, John Wiley & Sons, Inc., New York, NY, USA.

Sculley, D. (2010), Combined regression and ranking, *in* 'Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '10, ACM, New York, NY, USA, pp. 979–988.

Sculley, D. and Inc, G. (2009), Large scale learning to rank, *in* 'NIPS 2009 Workshop on Advances in Ranking'.

Sebastiani, F. (2002), 'Machine learning in automated text categorization', *ACM Computing Survey* **34**(1), 1–47.

Shaw, W. M., Wood, J. B., Wood, R. E. and Tibbo, T. R. (1991), 'The cystic fibrosis database: Content and research opportunities', *Library and Information Science Research* **13**(4), 347–366.

Sinka, M. P. and Corne, D. W. (2003*a*), 'Evolving better stoplists for document clustering and web intelligence', *Design and application of hybrid intelligent systems* pp. 1015–1023.

Sinka, M. P. and Corne, D. W. (2003*b*), Towards modernised and web-specific stoplists for web document analysis, *in* 'Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence', WI '03, IEEE Computer Society, Washington, DC, USA.

SMART (2014), 'SMART System Stop-words List'.
URL: *http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop*

Smith, M. E. (1990), Aspects of the P-Norm Model of Information Retrieval: Syntactic Query Generation, Efficiency, and Theoretical Properties, PhD thesis, Cornell University, Ithaca, NY, USA.

Smucker, M. D., Kazai, G. and Lease, M. (2012), Overview of the trec 2012 crowdsourcing track, Technical report, DTIC Document.

Soboroff, I. (2007), A comparison of pooled and sampled relevance judgments, *in* 'Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '07, ACM, New York, NY, USA, pp. 785–786.

Soboroff, I., Voorhees, E. and Craswell, N. (2003), 'Summary of the sigir 2003 workshop on defining evaluation methodologies for terabyte-scale test collections', *SIGIR Forum* **37**(2), 55–58.

Talbi, E.-G. (2009), *Metaheuristics: From Design to Implementation*, Wiley Publishing.

Tax, N., Bockting, S. and Hiemstra, D. (2015), 'A cross-benchmark comparison of 87 learning to rank methods', *Information Processing & Management* **51**(6), 757 – 772.

Teugels, J. L. (1990), 'Some representations of the multivariate bernoulli and binomial distributions', *Journal of Multivariate Analysis* **32**(2), 256 – 268.

Toman, M., Tesar, R. and Jezek, K. (2006), Influence of word normalization on text classification, *in* 'In Proceedings of the 1st international conference on multidisciplinary information sciences and technologies', Merida, Spain, pp. 354–358.

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

Toraman, C., Can, F. and Kocberber, S. (2011), Developing a text categorization template for turkish news portals, *in* 'In International symposium on innovations in intelligent systems and applications (INISTA)', IEEE, Istanbul, pp. 379–383.

TREC (2004), 'Text Retrieval Conference 2004 Robust Track'.
URL: *http://trec.nist.gov/data/t13_robust.html*

TREC (2016*a*).
URL: *http://trec.nist.gov/data/reljudge_eng.html*

TREC (2016*b*).
URL: *http://trec.nist.gov/data/qrels_eng/index.htm*

TREC (2016*c*).
URL: *http://trec.nist.gov/data/topics_eng/index.html*

TREC (2016*d*), 'Text Retrieval Conference 2014 Web Track'.
URL: *http://trec.nist.gov/data/web2014.html*

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

Uysal, A. K. and Gunal, S. (2014), 'The impact of preprocessing on text classification', *Information Processing & Management* **50**(1), 104–112.

Van Rijsbergen, C. (1975), *Information Retrieval*, Butterworths.
URL: *http://www.dcs.gla.ac.uk/Keith/Preface.html*

Van Rijsbergen, C. (1977), 'A theoretical basis for the use of co-occurrence data in information retrieval', *Journal of Documentation* **33**(2), 106–119.

Voorhees, E. M. (2000), 'Variations in relevance judgments and the measurement of retrieval effectiveness', *Information Processing and Management* **36**(5), 697–716.

Voorhees, E. M. (2004), Overview of the trec 2004 robust retrieval track, *in* 'In Proceedings of the Thirteenth Text REtrieval Conference (TREC-2004)', p. 13.

Vrajitoru, D. (1998), 'Crossover improvement for the genetic algorithm in information retrieval', *Information Processing & Management* **34**(4), 405 – 415.

Walker, S., Robertson, S. E., Boughanem, M., Jones, G. J. F. and Jones, K. S. (1997), Okapi at trec-6 automatic ad hoc, vlc, routing, filtering and qsdr, *in* E. Voorhees and D. Harman, eds, 'TREC-6', pp. 125–136.

Waller, W. G. and Kraft, D. H. (1979), 'A mathematical model of a weighted boolean retrieval system', *Information Processing and Management* **15**(5), 235–245.

Xu, J. and Li, H. (2007), 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, ACM, New York, NY, USA, pp. 391–398.

Yan, H., Ding, S. and Suel, T. (2009), Inverted index compression and query processing with optimized document ordering, *in* 'WWW '09 Proceedings of the 18th international conference on World wide web', ACM, New York, NY, USA, pp. 401–410.

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

Yeh, J. Y., Lin, J. Y., Ke, H. R. and Yang, W. P. (2007), Learning to rank for information retrieval using genetic programming, *in* T. Joachims, H. Li, T. Y. Liu and C. Zhai, eds, 'SIGIR 2007 workshop: Learning to Rank for Information Retrieval'.

Zhai, C. and Lafferty, J. (2004), 'A study of smoothing methods for language models applied to information retrieval', *ACM Trans. Inf. Syst.* **22**(2), 179–214.

Zipf, G. K. (1949), *Human Behavior and the Principle of Least Effort*, Addison-Wesley (Reading MA).

Zobel, J. and Moffat, A. (July, 2006), 'Inverted files for text search engines', *ACM Computing Survey* **38**(2).

Appendix A

Dataset Collections

Textual Document Collections

```
.I 1
.T
experimental investigation of the aerodynamics of a
wing in a slipstream .
.A
brenckman,m.
.B
j. ae. scs. 25, 1958, 324.
.W
experimental investigation of the aerodynamics of a
wing in a slipstream .
  an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios .  the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with
supporting evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or
boundary-layer-control effect .  the integrated remaining lift
increment, after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .
```

Figure A.1: One document from the Cranfield collection

```
.I 001
.W
what similarity laws must be obeyed when constructing aeroelastic models
of heated high speed aircraft .
```

Figure A.2: Example of one query from the Cranfield collection

Feature List of Feature Model Benchmarks (LETOR Datasets)

Microsoft Bing Search Engine Dataset Features

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
1	Covered query term frequency	body	
2		anchor	
3		title	
4		url	
5		whole document	
6	Covered query term ratio	body	
7		anchor	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
8		title	
9		url	
10		whole document	
11	stream length	body	
12		anchor	
13		title	
14		url	
15		whole document	
16	IDF(Inverse document frequency)	body	
17		anchor	
18		title	
19		url	
20		whole document	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
21	sum of term frequency	body	
22		anchor	
23		title	
24		url	
25		whole document	
26	min of term frequency	body	
27		anchor	
28		title	
29		url	
30		whole document	
31	max of term frequency	body	
32		anchor	
33		title	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
34		url	
35		whole document	
36	mean of term frequency	body	
37		anchor	
38		title	
39		url	
40		whole document	
41	Variance of term frequency	body	
42		anchor	
43		title	
44		url	
45		whole document	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
46	Sum of stream length normalized term frequency	body	
47		anchor	
48		title	
49		url	
50		whole document	
51	Min of stream length normalized term frequency	body	
52		anchor	
53		title	
54		url	
55		whole document	
56	Max of stream length normalized term frequency	body	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
57		anchor	
58		title	
59		url	
60		whole document	
61	Mean of stream length normalized term frequency	body	
62		anchor	
63		title	
64		url	
65		whole document	
66	Variance of stream length normalized term frequency	body	
67		anchor	
68		title	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
69		url	
70		whole document	
71	Sum of tf*idf	body	
72		anchor	
73		title	
74		url	
75		whole document	
76	Min of tf*idf	body	
77		anchor	
78		title	
79		url	
80		whole document	
81	Max of tf*idf	body	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
82		anchor	
83		title	
84		url	
85		whole document	
86	Mean of tf*idf	body	
87		anchor	
88		title	
89		url	
90		whole document	
91	Variance of tf*idf	body	
92		anchor	
93		title	
94		url	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
95		whole document	
96	Boolean model	body	
97		anchor	
98		title	
99		url	
100		whole document	
101	Vector space model	body	
102		anchor	
103		title	
104		url	
105		whole document	
106	BM25	body	
107		anchor	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
108		title	
109		url	
110		whole document	
111	LMIR.ABS	body	Language model approach for information retrieval (IR) with absolute discounting smoothing
112		anchor	
113		title	
114		url	
115		whole document	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
116	LMIR.DIR	body	Language model approach for IR with Bayesian smoothing using Dirichlet priors
117		anchor	
118		title	
119		url	
120		whole document	
121	LMIR.JM	body	Language model approach for IR with Jelinek-Mercer smoothing
122		anchor	
123		title	
124		url	

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
125		whole document	
126	Number of slash in URL		
127	Length of URL		
128	Inlink number		
129	Outlink number		
130	PageRank		
131	SiteRank		Site level PageRank
132	QualityScore		The quality score of a web page. The score is outputted by a web page quality classifier.

Table A.1: The MSLR-WEB10K and MSLR-WEB30K feature list as in (Qin and Liu, 2016)

Feature List in Query-Document Pair of Microsoft Bing Datasets			
feature id	**Feature Description**	**Stream**	**Comments**
133	QualityScore2		The quality score of a web page. The score is outputted by a web page quality classifier, which measures the badness of a web page.
134	Query-url click count		The click count of a query-url pair at a search engine in a period
135	url click count		The click count of a url aggregated from user browsing data in a period
136	url dwell time		The average dwell time of a url aggregated from user browsing data in a period

LETOR 4 Datasets Feature List

Table A.2: The MQ2007 and MQ2008 feature list as in (Qin and Liu, 2013).

Feature List in Query-Document Pair of MQ2007 and MQ2008	
FeatureID	Description
1	TF(Term frequency) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	IDF(Inverse document frequency) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
16	DL(Document length) of body
17	DL of anchor

Table A.2: The MQ2007 and MQ2008 feature list as in (Qin and Liu, 2013).

Feature List in Query-Document Pair of MQ2007 and MQ2008	
FeatureID	Description
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of whole document

Table A.2: The MQ2007 and MQ2008 feature list as in (Qin and Liu, 2013).

Feature List in Query-Document Pair of MQ2007 and MQ2008	
FeatureID	Description
36	LMIR.JM of body
37	LMIR.JM of anchor
38	LMIR.JM of title
39	LMIR.JM of URL
40	LMIR.JM of whole document
41	PageRank
42	Inlink number
43	Outlink number
44	Number of slash in URL
45	Length of URL
46	Number of child page

LETOR 3 Datasets Feature List

Table A.3: The TREC .Gov LETOR datasets feature list as in (Qin et al., 2010).

Feature List for TREC .Gov LETOR Datasets	
FeatureID	**Description**
1	Term frequency (TF) of body

Table A.3: The TREC .Gov LETOR datasets feature list as in (Qin et al., 2010).

Feature List for TREC .Gov LETOR Datasets	
FeatureID	**Description**
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	Inverse document frequency (IDF) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
16	Document length (DL) of body
17	DL of anchor
18	DL of title
19	DL of URL

Table A.3: The TREC .Gov LETOR datasets feature list as in (Qin et al., 2010).

Feature List for TREC .Gov LETOR Datasets	
FeatureID	**Description**
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of whole document
36	LMIR.JM of body
37	LMIR.JM of anchor

Table A.3: The TREC .Gov LETOR datasets feature list as in (Qin et al., 2010).

Feature List for TREC .Gov LETOR Datasets	
FeatureID	**Description**
38	LMIR.JM of title
39	LMIR.JM of URL
40	LMIR.JM of whole document
41	Sitemap based term propagation
42	Sitemap based score propagation
43	Hyperlink base score propagation: weighted in-link
44	Hyperlink base score propagation: weighted out-link
45	Hyperlink base score propagation: uniform out-link
46	Hyperlink base feature propagation: weighted in-link
47	Hyperlink base feature propagation: weighted out-link
48	Hyperlink base feature propagation: uniform out-link
49	HITS authority
50	HITS hub
51	PageRank
52	HostRank
53	Topical PageRank
54	Topical HITS authority
55	Topical HITS hub

Table A.3: The TREC .Gov LETOR datasets feature list as in (Qin et al., 2010).

Feature List for TREC .Gov LETOR Datasets	
FeatureID	**Description**
56	Inlink number
57	Outlink number
58	Number of slash in URL
59	Length of URL
60	Number of child page
61	BM25 of extracted title
62	LMIR.ABS of extracted title
63	LMIR.DIR of extracted title
64	LMIR.JM of extracted title

Table A.4: The Ohsumed LETOR datasets feature list as in (Qin et al., 2010).

Feature List for Ohsumed dataset in LETOR 3	
1	Term frequency (TF) of title
2	logarithm(TF+1) of title
3	Normalized TF of title
4	logarithm(Normalized TF+1) of title
5	Inverse document frequency (IDF) of title
6	logarithm(IDF) of title

Table A.4: The Ohsumed LETOR datasets feature list as in (Qin et al., 2010).

Feature List for Ohsumed dataset in LETOR 3	
7	logarithm(Collection Size(C)/Query Term Frequency(QTF)+1) of title
8	logarithm(Normalized TF * IDF+1) of title
9	TF*IDF of title
10	logarithm(Normalized TF*C/QTF + 1) of title
11	BM25 of title
12	log(BM25) of title
13	LMIR.DIR of title
14	LMIR.JM of title
15	LMIR.ABS of title
16	TF of abstract
17	logarithm(TF+1) of abstract
18	Normalized TF of abstract
19	logarithm(Normalized TF+1) of abstract
20	Inverse document frequency (IDF) of abstract
21	logarithm(IDF) of abstract
22	logarithm(Collection Size(C)/Query Term Frequency(QTF)+1) of abstract
23	logarithm(Normalized TF * IDF+1) of abstract
24	TF*IDF of abstract

Table A.4: The Ohsumed LETOR datasets feature list as in (Qin et al., 2010).

Feature List for Ohsumed dataset in LETOR 3	
25	logarithm(Normalized TF*C/QTF + 1) of abstract
26	BM25 of abstract
27	log(BM25) of abstract
28	LMIR.DIR of abstract
29	LMIR.JM of abstract
30	LMIR.ABS of abstract
31	Term frequency (TF) of title+abstract
32	logarithm(TF+1) of title+abstract
33	Normalized TF of title+abstract
34	logarithm(Normalized TF+1) of title+abstract
35	Inverse document frequency (IDF) of title+abstract
36	logarithm(IDF) of title+abstract
37	logarithm(Collection Size(C)/Query Term Frequency(QTF)+1) of title+abstract
38	logarithm(Normalized TF * IDF+1) of title+abstract
39	TF*IDF of title+abstract
40	logarithm(Normalized TF*C/QTF + 1) of title+abstract
41	BM25 of title+abstract
42	log(BM25) of title+abstract

Table A.4: The Ohsumed LETOR datasets feature list as in (Qin et al., 2010).

Feature List for Ohsumed dataset in LETOR 3	
43	LMIR.DIR of title+abstract
44	LMIR.JM of title+abstract
45	LMIR.ABS of title+abstract

Appendix B

Learning to Rank Model Based on Feature Vector Model

B.1 The Predictive results on test data for Mean Average Precision (MAP) as a Fitness and an Evaluation Function

Table B.1: MAP results for ranking models using evolutionary and machine learning techniques on MSLR-WEB10K

Algorithm	MAP Results in MSLR-WEB10K Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5746	0.5716	0.5743	0.5739	0.5741
RankSVM	0.4583	0.456	0.4543	0.4598	0.4584
ListNET	0.4723	0.4745	0.4731	0.4735	0.4739
AdaRank	0.5711	0.5714	0.5713	0.5711	0.571
MART	0.5796	0.5795	0.5791	0.5798	0.5796
Coordinate Ascent	0.5869	0.5845	0.5876	0.5887	0.5837
LambdaMART	0.5849	0.5861	0.5859	0.5857	0.5861
RankNET	0.4857	0.4865	0.4859	0.4856	0.4855
Random Forest	0.5997	0.5967	0.5984	0.5986	0.5975
Linear Regression	0.5016	0.5023	0.5019	0.5025	0.5017
RankGP	0.4687	0.4634	0.4674	0.4665	0.4706
Combined Regression & Ranking	0.4758	0.4741	0.4787	0.4736	0.4785
LambdaRank	0.4763	0.4755	0.4743	0.4735	0.4791
ES-Rank	0.5661	0.576	0.5689	0.5706	0.57033
IESR-Rank	0.5975	0.5993	0.5995	0.6173	0.6
IESVM-Rank	0.4586	0.4564	0.4546	0.4599	0.4576

Table B.2: MAP results for ranking models using evolutionary and machine learning techniques on MQ2008

Algorithm	MAP Results in MQ2008 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.4673	0.4348	0.4525	0.528	0.5035
RankSVM	0.4002	0.3514	0.3656	0.4471	0.4349
ListNET	0.4406	0.4107	0.4486	0.4906	0.4723
AdaRank	0.3962	0.4149	0.4525	0.5392	0.5237
MART	0.4657	0.4309	0.4506	0.5171	0.5019
Coordinate Ascent	0.4651	0.4318	0.4572	0.5379	0.5134
LambdaMART	0.4522	0.4206	0.4456	0.5257	0.5079
RankNET	0.4453	0.4114	0.428	0.4966	0.4786
Random Forest	0.4599	0.4353	0.4462	0.5067	0.5014
Linear Regression	0.4378	0.4166	0.4236	0.5035	0.4935
RankGP	0.4373	0.4309	0.4016	0.4279	0.4359
Combined Regression & Ranking	0.4406	0.4382	0.4396	0.4407	0.4382
LambdaRank	0.4031	0.3472	0.2622	0.3586	0.3707
ES-Rank	0.4898	0.5042	0.5177	0.477	0.4275
IESR-Rank	0.4921	0.5198	0.5121	0.4815	0.4628
IESVM-Rank	0.4517	0.4314	0.4583	0.5215	0.5039

Table B.3: MAP results for ranking models using evolutionary and machine learning techniques on MQ2007

Algorithm	MAP Results in MQ2007 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.4627	0.4531	0.4498	0.4304	0.4714
RankSVM	0.4474	0.4141	0.4236	0.3748	0.3793
ListNET	0.4614	0.4375	0.4417	0.4157	0.4419
AdaRank	0.4776	0.4434	0.4515	0.4336	0.4631
MART	0.478	0.4616	0.4585	0.4342	0.4624
Coordinate Ascent	0.4867	0.4518	0.4516	0.4398	0.4689
LambdaMART	0.4707	0.4523	0.4532	0.4392	0.4607
RankNET	0.4685	0.4357	0.4516	0.4283	0.4563
Random Forest	0.4772	0.4608	0.458	0.4377	0.4596
Linear Regression	0.4317	0.4378	0.4429	0.4235	0.4128
RankGP	0.4094	0.4173	0.4176	0.4201	0.4068
Combined Regression & Ranking	0.4197	0.4274	0.4186	0.4164	0.4261
LambdaRank	0.3533	0.2833	0.3788	0.278	0.4056
ES-Rank	0.4687	0.4672	0.4648	0.4758	0.4737
IESR-Rank	0.4842	0.4656	0.4679	0.4725	0.4754
IESVM-Rank	0.4828	0.4436	0.4515	0.4337	0.4659

Table B.4: MAP results for ranking models using evolutionary and machine learning techniques on Ohsumed

Algorithm	MAP Results in Ohsumed Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3458	0.447	0.4623	0.514	0.4701
RankSVM	0.2685	0.4073	0.4304	0.4741	0.3355
ListNET	0.3357	0.4146	0.4635	0.5114	0.4753
AdaRank	0.3228	0.4529	0.4374	0.5182	0.4515
MART	0.3325	0.4316	0.4342	0.4835	0.4527
Coordinate Ascent	0.3455	0.4414	0.4707	0.5145	0.4581
LambdaMART	0.3212	0.4265	0.4307	0.4909	0.4597
RankNET	0.3198	0.4446	0.4347	0.5047	0.4717
Random Forest	0.3405	0.4405	0.4352	0.4907	0.4557
Linear Regression	0.3208	0.4359	0.4494	0.5065	0.4539
RankGP	0.3169	0.4175	0.4273	0.4165	0.4175
Combined Regression & Ranking	0.3017	0.4045	0.4249	0.4235	0.4256
LambdaRank	0.1673	0.2656	0.4248	0.4062	0.2725
ES-Rank	0.3369	0.4515	0.4514	0.4406	0.4277
IESR-Rank	0.3593	0.4897	0.4426	0.4522	0.4302
IESVM-Rank	0.3397	0.451	0.4506	0.5138	0.4609

Table B.5: MAP results for ranking models using evolutionary and machine learning techniques on HP2003

Algorithm	MAP Results in HP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.6043	0.7737	0.7085	0.7847	0.6207
RankSVM	0.6871	0.626	0.5233	0.1298	0.1301
ListNET	0.0625	0.0327	0.1798	0.3306	0.0151
AdaRank	0.6391	0.7927	0.7325	0.7651	0.6727
MART	0.6988	0.7984	0.7235	0.7217	0.7877
Coordinate Ascent	0.7122	0.8218	0.8022	0.7411	0.6612
LambdaMART	0.7135	0.8116	0.7426	0.7256	0.6932
RankNET	0.7268	0.8274	0.7552	0.7454	0.6299
Random Forest	0.7294	0.8476	0.7689	0.7583	0.7386
Linear Regression	0.5366	0.5078	0.4235	0.5061	0.4883
RankGP	0.5682	0.5726	0.5795	0.5623	0.5363
Combined Regression & Ranking	0.5125	0.4462	0.5013	0.4876	0.4968
LambdaRank	0.7266	0.7885	0.7497	0.6853	0.6331
ES-Rank	0.74	0.7949	0.7845	0.8192	0.8565
IESR-Rank	0.7455	0.8117	0.8023	0.7903	0.8512
IESVM-Rank	0.7301	0.7259	0.6654	0.4994	0.5635

Table B.6: MAP results for ranking models using evolutionary and machine learning techniques on TD2003

Algorithm	MAP Results in TD2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.1499	0.2177	0.3243	0.2097	0.1249
RankSVM	0.0529	0.091	0.1562	0.0348	0.0723
ListNET	0.0897	0.01	0.0087	0.1399	0.0382
AdaRank	0.1652	0.2449	0.3126	0.3577	0.1456
MART	0.149	0.229	0.2393	0.1965	0.1249
Coordinate Ascent	0.1071	0.3092	0.3356	0.259	0.1747
LambdaMART	0.0966	0.1676	0.2952	0.2647	0.0785
RankNET	0.124	0.2296	0.3069	0.3294	0.1312
Random Forest	0.1848	0.3487	0.3381	0.3733	0.1784
Linear Regression	0.1231	0.1939	0.2615	0.2658	0.2407
RankGP	0.1262	0.1709	0.2665	0.2658	0.245
Combined Regression & Ranking	0.1409	0.1824	0.2602	0.2936	0.2449
LambdaRank	0.1071	0.115	0.2791	0.0534	0.1
ES-Rank	0.1913	0.2226	0.3395	0.3738	0.2648
IESR-Rank	0.217	0.2353	0.3442	0.3856	0.2747
IESVM-Rank	0.1786	0.3104	0.3919	0.2252	0.1615

Table B.7: MAP results for ranking models using evolutionary and machine learning techniques on NP2003

Algorithm	MAP Results in NP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5406	0.6185	0.6551	0.6963	0.7251
RankSVM	0.4829	0.5366	0.5701	0.4074	0.1688
ListNET	0.1729	0.1412	0.2674	0.2453	0.1789
AdaRank	0.5246	0.5415	0.6538	0.6541	0.7168
MART	0.6372	0.6146	0.6343	0.6811	0.7591
Coordinate Ascent	0.6497	0.6278	0.628	0.6792	0.7294
LambdaMART	0.5788	0.5738	0.6952	0.7303	0.6944
RankNET	0.5339	0.6271	0.6729	0.6942	0.7205
Random Forest	0.6682	0.6874	0.6864	0.7271	0.7706
Linear Regression	0.4939	0.4953	0.6375	0.6099	0.546
RankGP	0.5942	0.5879	0.5649	0.6069	0.5529
Combined Regression & Ranking	0.5644	0.5612	0.5924	0.5685	0.5528
LambdaRank	0.5523	0.6204	0.6832	0.6839	0.686
ES-Rank	0.7457	0.8053	0.7563	0.7133	0.7241
IESR-Rank	0.7507	0.784	0.754	0.7569	0.7266
IESVM-Rank	0.6377	0.6468	0.6467	0.7213	0.6604

Table B.8: MAP results for ranking models using evolutionary and machine learning techniques on HP2004

Algorithm	MAP Results in HP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5612	0.5496	0.7262	0.6404	0.6519
RankSVM	0.3207	0.3307	0.3997	0.3439	0.3615
ListNET	0.4156	0.0051	0.3317	0.0014	0.1175
AdaRank	0.7354	0.6676	0.7827	0.6701	0.7208
MART	0.3531	0.4873	0.664	0.3879	0.6052
Coordinate Ascent	0.6178	0.6946	0.7141	0.6498	0.6112
LambdaMART	0.5699	0.6131	0.4225	0.3826	0.5142
RankNET	0.5875	0.6157	0.7454	0.6174	0.5313
Random Forest	0.5838	0.6273	0.7663	0.545	0.6273
Linear Regression	0.4806	0.5043	0.5591	0.4132	0.5743
RankGP	0.5207	0.507	0.5997	0.439	0.5615
Combined Regression & Ranking	0.5256	0.5744	0.4871	0.5686	0.5616
LambdaRank	0.6786	0.15	0.4391	0.1	0.4659
ES-Rank	0.7206	0.7187	0.7007	0.7161	0.7336
IESR-Rank	0.6568	0.6447	0.6972	0.7334	0.7311
IESVM-Rank	0.5729	0.5638	0.5729	0.5892	0.5749

Table B.9: MAP results for ranking models using evolutionary and machine learning techniques on TD2004

Algorithm	MAP Results in TD2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.1971	0.2143	0.1958	0.2518	0.2302
RankSVM	0.1153	0.1263	0.1273	0.1244	0.1272
ListNET	0.0344	0.1585	0.1604	0.1562	0.1688
AdaRank	0.1324	0.2051	0.1991	0.2161	0.2041
MART	0.1353	0.2136	0.2114	0.2484	0.2119
Coordinate Ascent	0.2103	0.2313	0.2212	0.1991	0.2604
LambdaMART	0.1439	0.2772	0.1731	0.1477	0.1952
RankNET	0.1867	0.1507	0.1937	0.1995	0.2009
Random Forest	0.1895	0.2582	0.2785	0.2707	0.2736
Linear Regression	0.1725	0.2141	0.1628	0.1615	0.2321
RankGP	0.1825	0.2117	0.2196	0.1951	0.2431
Combined Regression & Ranking	0.2078	0.2161	0.2149	0.1999	0.2128
LambdaRank	0.196	0.1619	0.1289	0.1909	0.1827
ES-Rank	0.2511	0.2792	0.273	0.2598	0.2446
IESR-Rank	0.2605	0.2637	0.279	0.2532	0.2355
IESVM-Rank	0.1937	0.1896	0.1925	0.1956	0.1948

Table B.10: MAP results for ranking models using evolutionary and machine learning techniques on NP2004

Algorithm	MAP Results in NP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3639	0.4748	0.6884	0.6526	0.5846
RankSVM	0.3685	0.3918	0.3837	0.3754	0.3713
ListNET	0.0501	0.1713	0.1	0.3246	0.1382
AdaRank	0.3564	0.5948	0.6383	0.6304	0.6304
MART	0.4012	0.6846	0.6502	0.4386	0.4196
Coordinate Ascent	0.5247	0.6546	0.8111	0.624	0.6507
LambdaMART	0.472	0.4034	0.595	0.514	0.4943
RankNET	0.5489	0.6425	0.7389	0.6146	0.692
Random Forest	0.5744	0.6196	0.6902	0.6097	0.5212
Linear Regression	0.5027	0.4245	0.5746	0.3869	0.4467
RankGP	0.4685	0.4918	0.5837	0.554	0.473
Combined Regression & Ranking	0.5142	0.5066	0.5498	0.3702	0.4917
LambdaRank	0.4512	0.6501	0.7889	0.6244	0.7066
ES-Rank	0.7896	0.774	0.7138	0.7327	0.7513
IESR-Rank	0.7797	0.7838	0.7298	0.7325	0.7638
IESVM-Rank	0.5281	0.5277	0.5195	0.5153	0.5072

B.2 The Predictive results on test data for Normalized Discounted Cumulative Gain (NDCG@10) as a Fitness and an Evaluation Function

Table B.11: NDCG@10 results for ranking models using evolutionary and machine learning techniques on MSLR-WEB10K

Algorithm	NDCG@10 Results in MSLR-WEB10K Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3345	0.3349	0.3356	0.334	0.3375
RankSVM	0.2195	0.2196	0.2183	0.2235	0.2269
ListNET	0.1949	0.1923	0.1915	0.1927	0.1923
AdaRank	0.3467	0.3465	0.3455	0.3463	0.3462
MART	0.3946	0.3947	0.3949	0.3948	0.3945
Coordinate Ascent	0.4015	0.4023	0.4019	0.4016	0.4005
LambdaMART	0.4015	0.3997	0.3993	0.3995	0.3998
RankNET	0.1895	0.1923	0.1915	0.1918	0.1923
Random Forest	0.4012	0.3995	0.399	0.3985	0.4
Linear Regression	0.3598	0.3623	0.3619	0.3601	0.3618
RankGP	0.3543	0.3523	0.3546	0.3529	0.3543
Combined Regression & Ranking	0.3563	0.3579	0.3572	0.3582	0.3583
LambdaRank	0.1924	0.1952	0.1959	0.1975	0.1986
ES-Rank	0.3638	0.3887	0.3849	0.3867	0.3876
IESR-Rank	0.415	0.4133	0.4174	0.4109	0.4186
IESVM-Rank	0.2229	0.2234	0.2207	0.226	0.2272
IGBRT	0.39355	0.3901	0.39327	0.39507	0.39921

Table B.12: NDCG@10 results for ranking models using evolutionary and machine learning techniques on MQ2008

Algorithm	NDCG@10 Results in MQ2008 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.4868	0.4449	0.4818	0.553	0.5351
RankSVM	0.4254	0.3827	0.4021	0.4822	0.4687
ListNET	0.4647	0.4312	0.4779	0.532	0.5142
AdaRank	0.4835	0.4365	0.4728	0.5491	0.5412
MART	0.4931	0.4579	0.4769	0.5527	0.5375
Coordinate Ascent	0.4871	0.4517	0.4965	0.5568	0.5413
LambdaMART	0.4891	0.4576	0.4821	0.5504	0.5473
RankNET	0.4726	0.432	0.4644	0.5365	0.514
Random Forest	0.4897	0.4537	0.4708	0.5381	0.5315
Linear Regression	0.4725	0.4358	0.4599	0.5356	0.5318
RankGP	0.4658	0.4692	0.4125	0.4197	0.4359
Combined Regression & Ranking	0.4647	0.4723	0.4759	0.4815	0.4759
LambdaRank	0.3747	0.2776	0.2638	0.2963	0.3501
ES-Rank	0.51	0.5417	0.5346	0.4984	0.4481
IESR-Rank	0.5114	0.5411	0.5397	0.5048	0.4875
IESVM-Rank	0.4792	0.4416	0.4923	0.5504	0.5246
IGBRT	0.5073	0.5186	0.4995	0.5167	0.5486

Table B.13: NDCG@10 results for ranking models using evolutionary and machine learning techniques on MQ2007

Algorithm	NDCG@10 Results in MQ2007 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.4611	0.4221	0.4325	0.4054	0.4429
RankSVM	0.4068	0.3724	0.3914	0.3306	0.3217
ListNET	0.4301	0.4045	0.4298	0.3906	0.4295
AdaRank	0.4542	0.4202	0.4448	0.3985	0.4322
MART	0.4667	0.4317	0.4496	0.4109	0.4401
Coordinate Ascent	0.4635	0.4268	0.4448	0.4134	0.4646
LambdaMART	0.4721	0.4351	0.4609	0.4234	0.4473
RankNET	0.447	0.4129	0.4337	0.3941	0.4337
Random Forest	0.4723	0.429	0.4454	0.4189	0.4313
Linear Regression	0.4217	0.4298	0.4263	0.3987	0.4221
RankGP	0.4278	0.4281	0.4042	0.4169	0.4003
Combined Regression & Ranking	0.4289	0.4229	0.4201	0.4194	0.4211
LambdaRank	0.3272	0.3083	0.2392	0.2167	0.2865
ES-Rank	0.4452	0.4479	0.4565	0.4561	0.4474
IESR-Rank	0.4673	0.4464	0.4538	0.4537	0.4541
IESVM-Rank	0.4593	0.4237	0.4394	0.4139	0.4416
IGBRT	0.4326	0.5042	0.4293	0.4375	0.4825

Table B.14: NDCG@10 results for ranking models using evolutionary and machine learning techniques on Ohsumed

Algorithm	NDCG@10 Results in Ohsumed Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3766	0.4433	0.4143	0.4677	0.4907
RankSVM	0.2449	0.4294	0.3945	0.3739	0.2221
ListNET	0.3274	0.4029	0.3638	0.4536	0.415
AdaRank	0.3701	0.4487	0.4441	0.4764	0.4997
MART	0.3766	0.4485	0.3748	0.4581	0.4831
Coordinate Ascent	0.3678	0.4607	0.4368	0.4831	0.5126
LambdaMART	0.3212	0.4511	0.4038	0.4499	0.4566
RankNET	0.3547	0.4532	0.4193	0.4658	0.5139
Random Forest	0.3727	0.4665	0.4144	0.4655	0.4694
Linear Regression	0.3347	0.4584	0.4144	0.4659	0.4778
RankGP	0.3574	0.4613	0.4275	0.4189	0.4035
Combined Regression & Ranking	0.3368	0.4593	0.4461	0.4418	0.4375
LambdaRank	0.2754	0.2319	0.3975	0.3674	0.1285
ES-Rank	0.382	0.4527	0.4729	0.4699	0.453
IESR-Rank	0.3873	0.4939	0.473	0.4726	0.4452
IESVM-Rank	0.3581	0.4785	0.4607	0.4511	0.4943

Table B.15: NDCG@10 results for ranking models using evolutionary and machine learning techniques on HP2003

Algorithm	NDCG@10 Results in HP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.6755	0.8107	0.7264	0.8289	0.6821
RankSVM	0.7193	0.7031	0.5793	0.0595	0.1506
ListNET	0.0851	0.4328	0.014	0.2289	0.0688
AdaRank	0.6849	0.8095	0.7648	0.7681	0.7045
MART	0.7337	0.8465	0.7668	0.7775	0.8129
Coordinate Ascent	0.7235	0.8316	0.8329	0.7762	0.7219
LambdaMART	0.7304	0.7463	0.818	0.7856	0.8072
RankNET	0.7487	0.8267	0.8061	0.7839	0.669
Random Forest	0.7416	0.8509	0.8157	0.7951	0.7849
Linear Regression	0.5776	0.5802	0.4882	0.5573	0.5588
RankGP	0.5924	0.5975	0.5978	0.5889	0.5883
Combined Regression & Ranking	0.5638	0.5386	0.5623	0.5888	0.5943
LambdaRank	0.6957	0.8045	0.7759	0.6641	0.6706
ES-Rank	0.8056	0.8299	0.8156	0.8185	0.8612
IESR-Rank	0.8116	0.8303	0.8093	0.8294	0.8627
IESVM-Rank	0.7193	0.8819	0.7896	0.8094	0.7423
IGBRT	0.7988	0.8327	0.7988	0.8153	0.7952

Table B.16: NDCG@10 results for ranking models using evolutionary and machine learning techniques on TD2003

Algorithm	NDCG@10 Results in TD2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.1498	0.2648	0.4201	0.2564	0.2823
RankSVM	0.0504	0.1087	0.2253	0.0434	0.1051
ListNET	0.0479	0.1848	0.1574	0.073	0.1434
AdaRank	0.1587	0.3739	0.3596	0.2397	0.1939
MART	0.2107	0.3133	0.3301	0.2418	0.2823
Coordinate Ascent	0.2222	0.3489	0.4018	0.2882	0.3335
LambdaMART	0.186	0.2445	0.3986	0.3518	0.2396
RankNET	0.1498	0.1659	0.4156	0.1925	0.1766
Random Forest	0.2447	0.452	0.4218	0.3984	0.2931
Linear Regression	0.175	0.3837	0.3386	0.3468	0.3576
RankGP	0.2182	0.2938	0.2788	0.2463	0.2344
Combined Regression & Ranking	0.2063	0.2873	0.2514	0.2453	0.2624
LambdaRank	0.1479	0.2597	0.0835	0.1909	0.0723
ES-Rank	0.2514	0.4024	0.4189	0.3673	0.3722
IESR-Rank	0.2619	0.4378	0.4272	0.3743	0.3793
IESVM-Rank	0.2635	0.3438	0.4901	0.3417	0.2619
IGBRT	0.2156	0.3051	0.4223	0.316	0.272

Table B.17: NDCG@10 results for ranking models using evolutionary and machine learning techniques on NP2003

Algorithm	NDCG@10 Resullts in NP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.6017	0.6514	0.7004	0.7238	0.7268
RankSVM	0.5252	0.6125	0.6391	0.4439	0.1737
ListNET	0.0796	0.1257	0.1308	0.2791	0.2786
AdaRank	0.5707	0.608	0.6833	0.6487	0.7616
MART	0.6564	0.692	0.6983	0.7263	0.7693
Coordinate Ascent	0.7148	0.7042	0.7235	0.7682	0.7914
LambdaMART	0.6779	0.6385	0.7	0.7596	0.7015
RankNET	0.6017	0.6519	0.7175	0.7039	0.754
Random Forest	0.7167	0.7599	0.7327	0.7652	0.7995
Linear Regression	0.5319	0.5376	0.7006	0.6704	0.6127
RankGP	0.5909	0.5839	0.5471	0.6003	0.6073
Combined Regression & Ranking	0.5659	0.5798	0.5839	0.5366	0.5172
LambdaRank	0.5972	0.6889	0.7361	0.69	0.7179
ES-Rank	0.74246	0.7209	0.7826	0.7584	0.7646
IESR-Rank	0.7033	0.7205	0.7894	0.8027	0.7717
IESVM-Rank	0.7149	0.6788	0.7333	0.757	0.7788
IGBRT	0.71009	0.74667	0.73266	0.8353	0.75691

Table B.18: MAP results for ranking models using evolutionary and machine learning techniques on MSLR-WEB10K

Algorithm	NDCG@10 Results in HP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.602	0.6199	0.7496	0.7122	0.7023
RankSVM	0.3631	0.3233	0.3754	0.3299	0.3423
ListNET	0.01	0.308	0.0409	0.0667	0.4927
AdaRank	0.746	0.7259	0.6831	0.697	0.714
MART	0.4373	0.5607	0.69	0.4038	0.6499
Coordinate Ascent	0.7725	0.7735	0.7821	0.7829	0.7508
LambdaMART	0.5941	0.6641	0.7784	0.4041	0.6842
RankNET	0.7095	0.6584	0.7247	0.6537	0.5697
Random Forest	0.6727	0.6477	0.7667	0.5562	0.6792
Linear Regression	0.4917	0.5746	0.6351	0.4506	0.6259
RankGP	0.693	0.6778	0.6187	0.6522	0.7062
Combined Regression & Ranking	0.6696	0.693	0.6677	0.6187	0.6455
LambdaRank	0.5376	0.1	0.2	0.19	0.4459
ES-Rank	0.7733	0.7319	0.8114	0.8206	0.752
IESR-Rank	0.8076	0.7317	0.8274	0.7503	0.7733
IESVM-Rank	0.5531	0.5793	0.5816	0.5729	0.5725

Table B.19: NDCG@10 results for ranking models using evolutionary and machine learning techniques on TD2004

Algorithm	NDCG@10 Results in TD2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3162	0.2907	0.3049	0.3225	0.3093
RankSVM	0.1967	0.1959	0.1958	0.1948	0.2031
ListNET	0.0488	0.1371	0.1368	0.1301	0.1165
AdaRank	0.1775	0.3182	0.2636	0.2943	0.3432
MART	0.1861	0.292	0.2795	0.166	0.3003
Coordinate Ascent	0.2875	0.3202	0.3028	0.329	0.3368
LambdaMART	0.2453	0.1834	0.3075	0.2549	0.2603
RankNET	0.2756	0.2075	0.2904	0.2412	0.2794
Random Forest	0.2765	0.3564	0.3842	0.3538	0.375
Linear Regression	0.2456	0.3194	0.252	0.2362	0.3217
RankGP	0.2969	0.2922	0.2385	0.2362	0.2992
Combined Regression & Ranking	0.2996	0.2378	0.2778	0.2822	0.2956
LambdaRank	0.1	0.113	0.1427	0.1222	0.1729
ES-Rank	0.3284	0.3399	0.3854	0.3717	0.3642
IESR-Rank	0.3164	0.3496	0.3933	0.3786	0.3406
IESVM-Rank	0.2	0.2156	0.2149	0.2175	0.2183

Table B.20: NDCG@10 results for ranking models using evolutionary and machine learning techniques on NP2004

Algorithm	NDCG@10 Results in NP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.4975	0.5719	0.7438	0.7017	0.6374
RankSVM	0.4066	0.4853	0.4299	0.4174	0.4287
ListNET	0.1345	0.5857	0.4224	0.1	0.1483
AdaRank	0.3795	0.6582	0.7019	0.6946	0.6946
MART	0.456	0.7131	0.7192	0.5415	0.5072
Coordinate Ascent	0.4952	0.7277	0.877	0.7525	0.6587
LambdaMART	0.4192	0.4023	0.5684	0.5242	0.612
RankNET	0.6094	0.7019	0.8595	0.7279	0.7128
Random Forest	0.6456	0.655	0.7279	0.5991	0.572
Linear Regression	0.5834	0.4655	0.6628	0.5147	0.4782
RankGP	0.7381	0.7872	0.698	0.6418	0.6948
Combined Regression & Ranking	0.712	0.6159	0.7003	0.4896	0.4756
LambdaRank	0.4997	0.1954	0.6412	0.6533	0.707
ES-Rank	0.8299	0.8312	0.7637	0.7368	0.7996
IESR-Rank	0.8269	0.8127	0.7813	0.7445	0.7841
IESVM-Rank	0.5368	0.5183	0.5275	0.5182	0.5173

B.3 The Predictive results on test data for Precision (P@10) as a Fitness and an Evaluation Function

Table B.21: P@10 results for ranking models using evolutionary and machine learning techniques on MSLR-WEB10K

Algorithm	P@10 Results in MSLR-WEB10K Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5907	0.5874	0.5834	0.5863	0.5861
RankSVM	0.4063	0.405	0.4002	0.4045	0.4067
ListNET	0.4519	0.4352	0.4276	0.4286	0.436
AdaRank	0.5944	0.5789	0.5848	0.5854	0.6273
MART	0.6152	0.5929	0.6128	0.6658	0.6692
Coordinate Ascent	0.624	0.6308	0.6026	0.6366	0.6401
LambdaMART	0.6215	0.6486	0.6314	0.6571	0.6656
RankNET	0.4381	0.4441	0.4527	0.4421	0.4371
Random Forest	0.4528	0.6302	0.6514	0.6458	0.656
Linear Regression	0.4579	0.4569	0.4511	0.4611	0.4596
RankGP	0.4517	0.4536	0.4472	0.4427	0.4419
Combined Regression & Ranking	0.4483	0.4423	0.4391	0.4372	0.4387
LambdaRank	0.4317	0.4327	0.4289	0.4308	0.4267
ES-Rank	0.6116	0.6319	0.6394	0.6575	0.6319
IESR-Rank	0.6214	0.6491	0.6481	0.6547	0.6436
IESVM-Rank	0.4074	0.4069	0.4002	0.4048	0.4075

Table B.22: P@10 results for ranking models using evolutionary and machine learning techniques on MQ2008

Algorithm	P@10 Results in MQ2008 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.2694	0.2496	0.2512	0.3257	0.2729
RankSVM	0.2443	0.2291	0.2289	0.2951	0.2531
ListNET	0.2611	0.2362	0.2512	0.3155	0.271
AdaRank	0.247	0.2043	0.2385	0.306	0.2411
MART	0.2681	0.2457	0.2576	0.3244	0.2793
Coordinate Ascent	0.2613	0.2457	0.255	0.3251	0.2793
LambdaMART	0.2694	0.2489	0.2563	0.3251	0.2755
RankNET	0.2643	0.2355	0.2518	0.3136	0.2691
Random Forest	0.2694	0.2496	0.2531	0.3206	0.2806
Linear Regression	0.2694	0.2464	0.2537	0.3257	0.2723
RankGP	0.2519	0.2367	0.2275	0.2368	0.2472
Combined Regression & Ranking	0.2582	0.2415	0.2149	0.2473	0.2519
LambdaRank	0.2168	0.1948	0.1697	0.2455	0.2366
ES-Rank	0.2611	0.2432	0.2575	0.3181	0.271
IESR-Rank	0.2662	0.2508	0.2563	0.3264	0.2768
IESVM-Rank	0.2661	0.2381	0.2569	0.3289	0.2691

Table B.23: P@10 results for ranking models using evolutionary and machine learning techniques on MQ2007

Algorithm	P@10 Results in MQ2007 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3884	0.3855	0.3595	0.3513	0.3735
RankSVM	0.3607	0.3467	0.3267	0.3109	0.3147
ListNET	0.3708	0.3708	0.3509	0.3392	0.3593
AdaRank	0.3961	0.3808	0.3521	0.3147	0.3342
MART	0.3926	0.3956	0.3742	0.3569	0.3743
Coordinate Ascent	0.3902	0.3888	0.3722	0.3637	0.3735
LambdaMART	0.4012	0.3862	0.3775	0.3705	0.3835
RankNET	0.3789	0.3752	0.3521	0.3401	0.3628
Random Forest	0.394	0.3873	0.3725	0.3628	0.3746
Linear Regression	0.3845	0.3894	0.3627	0.3499	0.3755
RankGP	0.3356	0.3415	0.3583	0.3365	0.3478
Combined Regression & Ranking	0.3716	0.3574	0.3579	0.3465	0.3394
LambdaRank	0.3539	0.2655	0.281	0.2437	0.3177
ES-Rank	0.3958	0.3894	0.371	0.356	0.372
IESR-Rank	0.3845	0.3876	0.363	0.3498	0.3755
IESVM-Rank	0.4027	0.3846	0.3624	0.3551	0.374

Table B.24: P@10 results for ranking models using evolutionary and machine learning techniques on Ohsumed

Algorithm	P@10 Results in Ohsumed Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3682	0.4524	0.5143	0.5762	0.6095
RankSVM	0.2636	0.4143	0.5	0.519	0.3095
ListNET	0.3182	0.4571	0.4429	0.5429	0.5667
AdaRank	0.3545	0.4762	0.5476	0.5714	0.5429
MART	0.3682	0.4381	0.4762	0.5571	0.5381
Coordinate Ascent	0.3636	0.4852	0.5	0.5238	0.5429
LambdaMART	0.3045	0.4476	0.481	0.5619	0.5952
RankNET	0.3545	0.4524	0.5	0.5619	0.6095
Random Forest	0.3682	0.4667	0.4952	0.5762	0.5524
Linear Regression	0.3182	0.4619	0.5	0.5619	0.5619
RankGP	0.3315	0.3607	0.4163	0.5127	0.4563
Combined Regression & Ranking	0.3381	0.4172	0.4498	0.4362	0.4479
LambdaRank	0.1273	0.4048	0.4143	0.5	0.1952
ES-Rank	0.3727	0.4857	0.5095	0.5619	0.5381
IESR-Rank	0.3864	0.4476	0.519	0.5921	0.5429
IESVM-Rank	0.35	0.4619	0.5333	0.5571	0.5047

Table B.25: P@10 results for ranking models using evolutionary and machine learning techniques on HP2003

Algorithm	P@10 Results in HP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0833	0.1067	0.1167	0.097	0.1067
RankSVM	0.09	0.106	0.09	0.01	0.0267
ListNET	0.05	0.02	0.0767	0.017	0.0033
AdaRank	0.08	0.0967	0.1167	0.103	0.1033
MART	0.09	0.1067	0.11	0.103	0.11
Coordinate Ascent	0.0767	0.1033	0.12	0.107	0.11
LambdaMART	0.0833	0.0933	0.1	0.103	0.1
RankNET	0.0833	0.1067	0.1167	0.087	0.0933
Random Forest	0.0867	0.1067	0.1167	0.107	0.11
Linear Regression	0.0733	0.0967	0.09	0.083	0.09
RankGP	0.05	0.0533	0.0767	0.087	0.07
Combined Regression & Ranking	0.0533	0.0767	0.09	0.1	0.0667
LambdaRank	0.01	0.05	0.1067	0.047	0.0667
ES-Rank	0.08	0.107	0.1133	0.107	0.08
IESR-Rank	0.09	0.107	0.12	0.1	0.1033
IESVM-Rank	0.0766	0.11	0.1167	0.103	0.1067

Table B.26: P@10 results for ranking models using evolutionary and machine learning techniques on TD2003

Algorithm	P@10 Results in TD2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.14	0.14	0.18	0.11	0.15
RankSVM	0.05	0.06	0.13	0.03	0.09
ListNET	0.04	0.1	0.1	0.06	0.04
AdaRank	0.12	0.13	0.13	0.12	0.14
MART	0.14	0.21	0.13	0.11	0.14
Coordinate Ascent	0.16	0.2	0.1	0.12	0.21
LambdaMART	0.13	0.23	0.14	0.12	0.16
RankNET	0.13	0.17	0.17	0.12	0.15
Random Forest	0.17	0.3	0.14	0.15	0.21
Linear Regression	0.13	0.26	0.14	0.15	0.22
RankGP	0.1	0.12	0.1	0.1033	0.1033
Combined Regression & Ranking	0.05	0.1	0.09	0.06	0.1
LambdaRank	0.01	0.02	0.05	0.02	0.01
ES-Rank	0.17	0.2	0.19	0.15	0.21
IESR-Rank	0.16	0.26	0.15	0.16	0.22
IESVM-Rank	0.18	0.18	0.15	0.14	0.18

Table B.27: P@10 results for ranking models using evolutionary and machine learning techniques on NP2003

Algorithm	P@10 Results in NP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0867	0.08	0.0933	0.0867	0.0933
RankSVM	0.07	0.0866	0.0933	0.0614	0.0266
ListNET	0.0001	0.0367	0.01	0.0848	0.0333
AdaRank	0.0833	0.0833	0.0867	0.0833	0.09
MART	0.08	0.09	0.09	0.08	0.08
Coordinate Ascent	0.0833	0.0867	0.1	0.0881	0.1
LambdaMART	0.0833	0.09	0.0867	0.0881	0.0867
RankNET	0.0833	0.08	0.1	0.0833	0.09
Random Forest	0.09	0.0933	0.0967	0.0914	0.0967
Linear Regression	0.07	0.07	0.0967	0.0914	0.0867
RankGP	0.08	0.07	0.09	0.08	0.09
Combined Regression & Ranking	0.07	0.05	0.07	0.07	0.08
LambdaRank	0.0667	0.0233	0.0633	0.0081	0.001
ES-Rank	0.0867	0.09	0.11	0.0915	0.1
IESR-Rank	0.0867	0.08	0.097	0.0881	0.0933
IESVM-Rank	0.0833	0.0966	0.0866	0.0947	0.0966

Table B.28: P@10 results for ranking models using evolutionary and machine learning techniques on HP2004

Algorithm	P@10 Results in HP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0833	0.0867	0.0933	0.0667	0.0867
RankSVM	0.06	0.05	0.04	0.05	0.0667
ListNET	0.001	0.0333	0.02	0.0267	0.04
AdaRank	0.08	0.1067	0.08	0.06	0.0867
MART	0.09	0.0933	0.0733	0.0733	0.08
Coordinate Ascent	0.0933	0.1	0.1	0.1	0.0933
LambdaMART	0.06	0.0867	0.0933	0.06	0.0933
RankNET	0.0833	0.0933	0.08	0.0933	0.0733
Random Forest	0.08	0.0933	0.0933	0.08	0.0867
Linear Regression	0.06	0.1	0.0867	0.0667	0.0867
RankGP	0.0733	0.0867	0.0733	0.0733	0.06
Combined Regression & Ranking	0.06	0.05	0.06	0.0733	0.05
LambdaRank	0.0333	0.0133	0.04	0.0133	0.0067
ES-Rank	0.0867	0.1133	0.0933	0.0933	0.0933
IESR-Rank	0.1	0.107	0.0867	0.1	0.1
IESVM-Rank	0.0933	0.1	0.0667	0.0733	0.08

Table B.29: P@10 results for ranking models using evolutionary and machine learning techniques on TD2004

Algorithm	P@10 Results in TD2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.2467	0.2267	0.26	0.2	0.2333
RankSVM	0.13	0.1053	0.1383	0.1772	0.19
ListNET	0.1	0.2267	0.0533	0.1067	0.0733
AdaRank	0.2	0.2333	0.2133	0.2067	0.2667
MART	0.18	0.2267	0.2133	0.28	0.2867
Coordinate Ascent	0.24	0.2933	0.2333	0.2133	0.2667
LambdaMART	0.1933	0.26	0.1867	0.2533	0.2533
RankNET	0.2	0.1933	0.2467	0.1867	0.2467
Random Forest	0.2067	0.2933	0.26	0.2467	0.3267
Linear Regression	0.2133	0.26	0.1733	0.1933	0.2867
RankGP	0.125	0.129	0.109	0.17	0.18
Combined Regression & Ranking	0.1197	0.1019	0.115	0.1021	0.117
LambdaRank	0.1067	0.1667	0.04	0.1667	0.18
ES-Rank	0.2533	0.3	0.22	0.2333	0.28
IESR-Rank	0.207	0.287	0.2333	0.2399	0.327
IESVM-Rank	0.17	0.11	0.19	0.19	0.18

Table B.30: P@10 results for ranking models using evolutionary and machine learning techniques on NP2004

Algorithm	P@10 Results in NP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0667	0.0933	0.0933	0.0867	0.0867
RankSVM	0.0667	0.04	0.04	0.0667	0.0533
ListNET	0.04	0.02	0.04	0.02	0.0133
AdaRank	0.08	0.0867	0.1	0.0867	0.0933
MART	0.0667	0.0867	0.0867	0.08	0.0867
Coordinate Ascent	0.0867	0.1	0.0933	0.0867	0.0933
LambdaMART	0.0667	0.0667	0.0733	0.08	0.0867
RankNET	0.0867	0.1067	0.1	0.0933	0.0933
Random Forest	0.0867	0.0933	0.0933	0.0733	0.0933
Linear Regression	0.0867	0.0667	0.0933	0.0933	0.0667
RankGP	0.07	0.0667	0.07	0.0867	0.0733
Combined Regression & Ranking	0.0667	0.0467	0.0667	0.0467	0.0533
LambdaRank	0.0467	0.0333	0.004	0.0533	0.08
ES-Rank	0.09	0.09	0.0933	0.08	0.1
IESR-Rank	0.08	0.1	0.0933	0.09	0.09
IESVM-Rank	0.07	0.0667	0.0667	0.07	0.07

B.4 The Predictive results on test data for Reciprocal Rank (RR@10) as a Fitness and an Evaluation Function

Table B.31: RR@10 results for ranking models using evolutionary and machine learning techniques on MSLR-WEB10K

Algorithm	RR@10 Results in MSLR-WEB10K Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5746	0.5716	0.5743	0.5739	0.5741
RankSVM	0.4583	0.456	0.4543	0.4598	0.4584
ListNET	0.4723	0.4745	0.4731	0.4735	0.4739
AdaRank	0.5711	0.5714	0.5713	0.5711	0.571
MART	0.5796	0.5795	0.5791	0.5798	0.5796
Coordinate Ascent	0.5869	0.5845	0.5876	0.5887	0.5837
LambdaMART	0.5849	0.5861	0.5859	0.5857	0.5861
RankNET	0.4857	0.4865	0.4859	0.4856	0.4855
Random Forest	0.5997	0.5967	0.5984	0.5986	0.5975
Linear Regression	0.5016	0.5023	0.5019	0.5025	0.5017
RankGP	0.4687	0.4634	0.4674	0.4665	0.4706
Combined Regression & Ranking	0.4758	0.4741	0.4787	0.4736	0.4785
LambdaRank	0.4763	0.4755	0.4743	0.4735	0.4791
ES-Rank	0.5661	0.576	0.5689	0.5706	0.57033
IESR-Rank	0.5975	0.5993	0.5995	0.6173	0.6
IESVM-Rank	0.4586	0.4564	0.4546	0.4599	0.4576

Table B.32: RR@10 results for ranking models using evolutionary and machine learning techniques on MQ2008

Algorithm	RR@10 Results in MQ2008 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5192	0.4844	0.5141	0.5979	0.5499
RankSVM	0.4525	0.3965	0.4061	0.5032	0.4943
ListNET	0.4867	0.4717	0.5106	0.5731	0.525
AdaRank	0.5021	0.4765	0.5122	0.6181	0.557
MART	0.5131	0.4839	0.5057	0.5888	0.556
Coordinate Ascent	0.5004	0.4684	0.5195	0.5921	0.587
LambdaMART	0.5106	0.4831	0.5207	0.586	0.5477
RankNET	0.482	0.46	0.4955	0.5798	0.5059
Random Forest	0.5083	0.5028	0.5124	0.5729	0.5559
Linear Regression	0.4867	0.4552	0.4947	0.5785	0.5517
RankGP	0.4723	0.4835	0.4952	0.4719	0.4931
Combined Regression & Ranking	0.4419	0.4572	0.4581	0.4625	0.4571
LambdaRank	0.4166	0.3773	0.4258	0.4976	0.4225
ES-Rank	0.5093	0.4901	0.5212	0.5989	0.5662
IESR-Rank	0.5024	0.47	0.5235	0.6049	0.5752
IESVM-Rank	0.4884	0.4524	0.4815	0.5682	0.5739

Table B.33: RR@10 results for ranking models using evolutionary and machine learning techniques on MQ2007

Algorithm	RR@10 Results in MQ2007 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5938	0.5508	0.5569	0.5433	0.5742
RankSVM	0.5166	0.5057	0.5217	0.4459	0.4368
ListNET	0.5596	0.5422	0.5537	0.5301	0.5731
AdaRank	0.5696	0.5147	0.5454	0.5308	0.5811
MART	0.5921	0.5627	0.585	0.5351	0.5706
Coordinate Ascent	0.5766	0.5532	0.5648	0.5248	0.5694
LambdaMART	0.5817	0.5636	0.583	0.5452	0.5779
RankNET	0.5659	0.5541	0.561	0.5209	0.5592
Random Forest	0.589	0.5591	0.5736	0.5406	0.5682
Linear Regression	0.5768	0.5654	0.5315	0.5306	0.5479
RankGP	0.5485	0.5368	0.5415	0.5437	0.5385
Combined Regression & Ranking	0.5375	0.5417	0.5465	0.5492	0.5418
LambdaRank	0.5295	0.475	0.5119	0.4735	0.5106
ES-Rank	0.5867	0.4887	0.5357	0.5448	0.5754
IESR-Rank	0.5768	0.5727	0.5736	0.5497	0.57
IESVM-Rank	0.5756	0.5505	0.5789	0.5261	0.5687

Table B.34: RR@10 results for ranking models using evolutionary and machine learning techniques on Ohsumed

Algorithm	RR@10 Results in Ohsumed Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5922	0.6984	0.718	0.7728	0.8322
RankSVM	0.4876	0.7261	0.7739	0.7132	0.4256
ListNET	0.6001	0.5524	0.7198	0.7183	0.7576
AdaRank	0.5947	0.7579	0.7659	0.7817	0.7952
MART	0.6103	0.6973	0.7401	0.6993	0.7833
Coordinate Ascent	0.5899	0.6845	0.6981	0.7143	0.7996
LambdaMART	0.658	0.7119	0.666	0.7782	0.869
RankNET	0.549	0.7103	0.8092	0.7579	0.7698
Random Forest	0.5947	0.7341	0.7445	0.6996	0.8413
Linear Regression	0.6125	0.8135	0.7206	0.7825	0.7778
RankGP	0.5245	0.5639	0.5783	0.5369	0.5923
Combined Regression & Ranking	0.5174	0.5278	0.5435	0.5513	0.5639
LambdaRank	0.5028	0.5417	0.7496	0.5992	0.6167
ES-Rank	0.6137	0.719	0.7687	0.7837	0.7476
IESR-Rank	0.5709	0.7857	0.7298	0.773	0.7635
IESVM-Rank	0.5428	0.7794	0.7703	0.7436	0.8095

Table B.35: RR@10 results for ranking models using evolutionary and machine learning techniques on HP2003

Algorithm	RR@10 Results in HP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5991	0.8053	0.8603	0.719	0.6456
RankSVM	0.6983	0.6305	0.4862	0.047	0.1577
ListNET	0.0333	0.0289	0.1318	0.515	0.0637
AdaRank	0.6484	0.8194	0.8511	0.736	0.7066
MART	0.7203	0.8317	0.8164	0.77	0.8222
Coordinate Ascent	0.6959	0.8444	0.8542	0.783	0.7667
LambdaMART	0.7437	0.802	0.8575	0.728	0.7417
RankNET	0.731	0.8417	0.8417	0.735	0.6844
Random Forest	0.7333	0.8411	0.8472	0.779	0.78
Linear Regression	0.5469	0.5095	0.4787	0.538	0.523
RankGP	0.5974	0.5817	0.5845	0.574	0.5839
Combined Regression & Ranking	0.5853	0.5809	0.5739	0.58	0.5842
LambdaRank	0.7042	0.802	0.8594	0.726	0.6694
ES-Rank	0.7589	0.8492	0.8742	0.778	0.7453
IESR-Rank	0.7375	0.8833	0.8444	0.756	0.7383
IESVM-Rank	0.7704	0.85	0.8219	0.75	0.6472

Table B.36: RR@10 results for ranking models using evolutionary and machine learning techniques on TD2003

Algorithm	RR@10 Results in TD2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.2425	0.5333	0.4833	0.5225	0.5667
RankSVM	0.095	0.1867	0.3055	0.0225	0.1952
ListNET	0.085	0.3643	0.26	0.025	0.1111
AdaRank	0.445	0.5167	0.3047	0.4093	0.375
MART	0.3783	0.5908	0.3944	0.325	0.46
Coordinate Ascent	0.3403	0.4583	0.4844	0.4533	0.4243
LambdaMART	0.3992	0.3226	0.3625	0.395	0.506
RankNET	0.5236	0.55	0.4593	0.2319	0.3644
Random Forest	0.4093	0.6394	0.52	0.5292	0.595
Linear Regression	0.3136	0.525	0.4144	0.4268	0.675
RankGP	0.3389	0.3194	0.2959	0.3158	0.2762
Combined Regression & Ranking	0.3045	0.3076	0.2874	0.2782	0.2995
LambdaRank	0.0343	0.485	0.5208	0.1726	0.345
ES-Rank	0.3792	0.5725	0.6311	0.4267	0.61
IESR-Rank	0.356	0.5983	0.55	0.4293	0.595
IESVM-Rank	0.4694	0.6125	0.4333	0.3366	0.5194

Table B.37: RR@10 results for ranking models using evolutionary and machine learning techniques on NP2003

Algorithm	RR@10 Results in NP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5495	0.6098	0.6458	0.6881	0.73
RankSVM	0.4681	0.5299	0.5797	0.3956	0.1459
ListNET	0.1903	0.0033	0.1428	0.1731	0.0083
AdaRank	0.5179	0.537	0.5383	0.6517	0.6344
MART	0.6389	0.6133	0.6329	0.6886	0.7667
Coordinate Ascent	0.6525	0.6356	0.6403	0.6923	0.7672
LambdaMART	0.6648	0.6522	0.6428	0.7093	0.6931
RankNET	0.5134	0.6139	0.6936	0.6725	0.7053
Random Forest	0.6717	0.7089	0.6983	0.737	0.7511
Linear Regression	0.4825	0.4844	0.6454	0.6064	0.5491
RankGP	0.4391	0.4783	0.4759	0.4672	0.4662
Combined Regression & Ranking	0.4486	0.4617	0.4783	0.4853	0.4573
LambdaRank	0.5278	0.6111	0.6537	0.5995	0.6825
ES-Rank	0.6279	0.6725	0.6597	0.7164	0.6929
IESR-Rank	0.6771	0.5742	0.7256	0.7006	0.7736
IESVM-Rank	0.5431	0.5944	0.6361	0.7123	0.7056

Table B.38: RR@10 results for ranking models using evolutionary and machine learning techniques on HP2004

Algorithm	RR@10 Results in HP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5991	0.5747	0.7241	0.6585	0.6433
RankSVM	0.2683	0.3791	0.3685	0.3685	0.3585
ListNET	0.3207	0.0407	0.05	0.03	0.04
AdaRank	0.6484	0.7022	0.7833	0.5889	0.7667
MART	0.7203	0.5194	0.6467	0.35	0.6
Coordinate Ascent	0.7648	0.725	0.7667	0.6744	0.5872
LambdaMART	0.5911	0.6133	0.5207	0.3889	0.6911
RankNET	0.7281	0.71	0.7262	0.5778	0.5206
Random Forest	0.6	0.6333	0.7639	0.5067	0.6222
Linear Regression	0.4889	0.5111	0.5611	0.3963	0.575
RankGP	0.4758	0.4723	0.4753	0.4815	0.4823
Combined Regression & Ranking	0.4683	0.4672	0.4658	0.4593	0.4579
LambdaRank	0.6079	0.0956	0.6556	0.4778	0.001
ES-Rank	0.7467	0.75	0.7833	0.6889	0.65
IESR-Rank	0.6689	0.7778	0.6933	0.7389	0.7667
IESVM-Rank	0.4359	0.4381	0.4277	0.4385	0.4365

Table B.39: RR@10 results for ranking models using evolutionary and machine learning techniques on TD2004

Algorithm	RR@10 Results in TD2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.5706	0.4973	0.5067	0.5115	0.4022
RankSVM	0.3485	0.3395	0.3638	0.3633	0.3582
ListNET	0.04	0.0333	0.412	0.0863	0.2439
AdaRank	0.4133	0.5534	0.5429	0.478	0.5623
MART	0.2889	0.4859	0.5963	0.4112	0.358
Coordinate Ascent	0.5056	0.6111	0.5963	0.4374	0.4945
LambdaMART	0.3329	0.4783	0.438	0.4402	0.4303
RankNET	0.5185	0.3489	0.4939	0.4278	0.4444
Random Forest	0.5556	0.5511	0.7389	0.7652	0.6444
Linear Regression	0.3972	0.4967	0.6856	0.4423	0.4984
RankGP	0.4791	0.5075	0.4762	0.4618	0.4636
Combined Regression & Ranking	0.4486	0.4793	0.4573	0.4562	0.4615
LambdaRank	0.4229	0.0467	0.4408	0.3741	0.5633
ES-Rank	0.4345	0.7017	0.6244	0.5689	0.5363
IESR-Rank	0.5733	0.5692	0.5596	0.6017	0.5467
IESVM-Rank	0.4583	0.4592	0.4581	0.4553	0.4617

Table B.40: RR@10 results for ranking models using evolutionary and machine learning techniques on NP2004

Algorithm	RR@10 Results in NP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.3636	0.5074	0.6817	0.6484	0.5833
RankSVM	0.2173	0.2572	0.2475	0.2376	0.2512
ListNET	0.1	0.2263	0.1667	0.4733	0.3162
AdaRank	0.3278	0.6069	0.5373	0.7278	0.6417
MART	0.3911	0.7278	0.6484	0.4362	0.4094
Coordinate Ascent	0.4661	0.5917	0.7278	0.6472	0.6095
LambdaMART	0.4204	0.4524	0.5056	0.5306	0.4984
RankNET	0.5614	0.6579	0.68	0.6944	0.6833
Random Forest	0.5804	0.604	0.7	0.5667	0.4951
Linear Regression	0.498	0.4206	0.5722	0.3869	0.4411
RankGP	0.4753	0.4486	0.4386	0.4467	0.4597
Combined Regression & Ranking	0.4618	0.4519	0.4219	0.4387	0.4852
LambdaRank	0.4589	0.5562	0.65	0.6056	0.6229
ES-Rank	0.4229	0.64	0.689	0.731	0.6611
IESR-Rank	0.4984	0.5861	0.7317	0.6833	0.647
IESVM-Rank	0.4279	0.4131	0.4183	0.4271	0.4219

B.5 The Predictive results on test data for Error Rate (ERR@10) as Fitness and Evaluation Functions

Table B.41: ERR@10 results for ranking models using evolutionary and machine learning techniques on MSLR-WEB10K

Algorithm	ERR@10 Results in MSLR-WEB10K Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.229	0.2369	0.2289	0.2378	0.2265
RankSVM	0.185	0.1808	0.1767	0.1882	0.1793
ListNET	0.2744	0.2759	0.2718	0.2689	0.2763
AdaRank	0.2097	0.2093	0.2187	0.2165	0.2092
MART	0.2312	0.2275	0.2259	0.2287	0.2365
Coordinate Ascent	0.2219	0.2243	0.2263	0.2284	0.2361
LambdaMART	0.2375	0.2384	0.2476	0.2366	0.2397
RankNET	0.2514	0.2682	0.2837	0.2531	0.2657
Random Forest	0.2919	0.2965	0.2907	0.2975	0.2972
Linear Regression	0.1621	0.1659	0.1606	0.1678	0.1701
LambdaRank	0.2972	0.2855	0.2913	0.2957	0.2849
ES-Rank	0.0846	0.1101	0.1098	0.0823	0.0854
IESR-Rank	0.0645	0.0642	0.0666	0.0727	0.0721
IESVM-Rank	0.1634	0.1672	0.1655	0.1681	0.1631

Table B.42: ERR@10 results for ranking models using evolutionary and machine learning techniques on MQ2008

Algorithm	ERR@10 Results in MQ2008 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0984	0.0813	0.0878	0.1177	0.0973
RankSVM	0.0803	0.0675	0.0702	0.0948	0.0819
ListNET	0.0912	0.0821	0.0862	0.1134	0.0908
AdaRank	0.0801	0.0702	0.0655	0.0701	0.0836
MART	0.0986	0.0842	0.0908	0.1181	0.095
Coordinate Ascent	0.0961	0.08	0.0916	0.1158	0.1004
LambdaMART	0.0979	0.0851	0.0926	0.1142	0.1
RankNET	0.0952	0.0784	0.0849	0.1115	0.0921
Random Forest	0.0971	0.0827	0.0898	0.1139	0.0955
Linear Regression	0.0956	0.0808	0.0879	0.1137	0.095
LambdaRank	0.0605	0.047	0.0781	0.099	0.0634
ES-Rank	0.0294	0.0269	0.0249	0.0337	0.0317
IESR-Rank	0.0284	0.0296	0.0214	0.0308	0.0306
IESVM-Rank	0.026	0.0251	0.0225	0.0289	0.0273

Table B.43: ERR@10 results for ranking models using evolutionary and machine learning techniques on MQ2007

Algorithm	ERR@10 Results in MQ2007 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.1107	0.0978	0.0979	0.0988	0.1031
RankSVM	0.0914	0.0852	0.0853	0.0748	0.0714
ListNET	0.1044	0.0954	0.0975	0.092	0.1027
AdaRank	0.0793	0.0699	0.0847	0.0689	0.0734
MART	0.1141	0.1004	0.104	0.0988	0.1045
Coordinate Ascent	0.108	0.0987	0.1009	0.0984	0.1023
LambdaMART	0.1102	0.1011	0.1037	0.1015	0.1053
RankNET	0.1038	0.0942	0.0976	0.0929	0.1028
Random Forest	0.1134	0.1011	0.1049	0.0996	0.1036
Linear Regression	0.1064	0.099	0.0954	0.095	0.1007
LambdaRank	0.0752	0.0551	0.0548	0.0638	0.0588
ES-Rank	0.0331	0.0304	0.0282	0.0206	0.0272
IESR-Rank	0.0308	0.032	0.0266	0.0241	0.0315
IESVM-Rank	0.0286	0.0312	0.0248	0.02	0.0268

Table B.44: ERR@10 results for ranking models using evolutionary and machine learning techniques on Ohsumed

Algorithm	ERR@10 Results in Ohsumed Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.1553	0.1766	0.1809	0.2048	0.2267
RankSVM	0.1033	0.1994	0.1811	0.1603	0.0954
ListNET	0.1509	0.2009	0.1783	0.2004	0.2185
AdaRank	0.1548	0.1892	0.2121	0.2108	0.2186
MART	0.172	0.201	0.1873	0.1972	0.2189
Coordinate Ascent	0.1499	0.1924	0.2026	0.2006	0.2378
LambdaMART	0.1553	0.1841	0.1837	0.2085	0.2197
RankNET	0.1576	0.1809	0.1755	0.1972	0.2242
Random Forest	0.1474	0.1941	0.1896	0.2032	0.2256
Linear Regression	0.1424	0.1424	0.1797	0.2013	0.2183
LambdaRank	0.0264	0.1688	0.0963	0.1329	0.1429
ES-Rank	0.0412	0.0411	0.0519	0.0775	0.0603
IESR-Rank	0.0243	0.0545	0.053	0.0558	0.0569
IESVM-Rank	0.0333	0.0464	0.0475	0.0813	0.0687

Table B.45: ERR@10 results for ranking models using evolutionary and machine learning techniques on HP2003

Algorithm	ERR@10 Results in HP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0528	0.0525	0.0586	0.046	0.0439
RankSVM	0.0444	0.0419	0.0378	0.003	0.0099
ListNET	0.0275	0.007	0.0226	0.019	0.0026
AdaRank	0.0497	0.0365	0.0448	0.032	0.035
MART	0.0681	0.0556	0.0536	0.05	0.055
Coordinate Ascent	0.0469	0.0598	0.0536	0.052	0.0476
LambdaMART	0.0467	0.0541	0.0537	0.05	0.0516
RankNET	0.039	0.0549	0.0564	0.049	0.0446
Random Forest	0.0466	0.0571	0.059	0.051	0.0518
Linear Regression	0.0342	0.0351	0.0322	0.035	0.0333
LambdaRank	0.007	0.0388	0.001	0.041	0.0398
ES-Rank	0	0	0.0026	0	0
IESR-Rank	0	0	0	3E-04	0.0005
IESVM-Rank	0.0012	0	0	0	0

Table B.46: ERR@10 results for ranking models using evolutionary and machine learning techniques on TD2003

Algorithm	ERR@10 Results in TD2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0239	0.0465	0.0383	0.029	0.0534
RankSVM	0.001	0.0142	0.0117	0.0317	0.0473
ListNET	0.0158	0.0054	0.0189	0.0078	0.011
AdaRank	0.0235	0.0379	0.0295	0.017	0.0373
MART	0.0311	0.0544	0.0317	0.0266	0.033
Coordinate Ascent	0.0306	0.0451	0.0314	0.0349	0.0561
LambdaMART	0.0317	0.0314	0.0372	0.0331	0.038
RankNET	0.0217	0.0225	0.034	0.0181	0.0353
Random Forest	0.0344	0.0643	0.0443	0.0375	0.0517
Linear Regression	0.0271	0.0546	0.0327	0.0313	0.0581
LambdaRank	0.0163	0.006	0.0332	0.002	0.0063
ES-Rank	0	0.0062	0	0.0007	0.0006
IESR-Rank	0.001	0	0	0.0002	0.0041
IESVM-Rank	0	0.001	0.001	0	0

Table B.47: ERR@10 results for ranking models using evolutionary and machine learning techniques on NP2003

Algorithm	ERR@10 Results in NP2003 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0335	0.0381	0.042	0.043	0.0466
RankSVM	0.0292	0.033	0.0367	0.0247	0.0091
ListNET	0.0233	0.0033	0.001	0.0089	0.0005
AdaRank	0.01	0.0348	0.0005	0.003	0.041
MART	0.0399	0.0383	0.0408	0.0422	0.0482
Coordinate Ascent	0.0384	0.0381	0.042	0.0424	0.0495
LambdaMART	0.0401	0.0363	0.0437	0.0469	0.0477
RankNET	0.0354	0.0366	0.0427	0.0414	0.0451
Random Forest	0.0425	0.0425	0.0455	0.0453	0.047
Linear Regression	0.0302	0.0303	0.0416	0.0381	0.0343
LambdaRank	0.0173	0.0105	0.0244	0.0146	0.0132
ES-Rank	0	0	0.0023	0.0029	0
IESR-Rank	0	0	0	0.0004	0
IESVM-Rank	0	0	0	0.0002	0

Table B.48: ERR@10 results for ranking models using evolutionary and machine learning techniques on HP2004

Algorithm	ERR@10 Results in HP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0362	0.0376	0.0472	0.0418	0.0409
RankSVM	0.0453	0.0391	0.0457	0.0436	0.0429
ListNET	0.001	0.0079	0.0067	0.0021	0.0004
AdaRank	0.0334	0.0306	0.0192	0.0124	0.0027
MART	0.0213	0.0342	0.0308	0.0211	0.0375
Coordinate Ascent	0.0448	0.0498	0.0479	0.0383	0.0469
LambdaMART	0.0343	0.0383	0.0439	0.0258	0.0415
RankNET	0.04	0.0441	0.045	0.045	0.0335
Random Forest	0.0411	0.0415	0.0492	0.031	0.0397
Linear Regression	0.0306	0.0341	0.0351	0.0248	0.0364
LambdaRank	0.0019	0.0097	0.0052	0.0089	0.0225
ES-Rank	0	0.002	0	0	0
IESR-Rank	0	0.0006	0	0	0
IESVM-Rank	0.0007	0.002	0.001	0	0

Table B.49: ERR@10 results for ranking models using evolutionary and machine learning techniques on TD2004

Algorithm	ERR@10 Results in TD2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0547	0.0538	0.0512	0.0493	0.0366
RankSVM	0.0617	0.0661	0.0673	0.0694	0.0686
ListNET	0.003	0.0231	0.031	0.0116	0.0368
AdaRank	0.0168	0.001	0.049	0.0228	0.0306
MART	0.0294	0.0485	0.0545	0.0438	0.0494
Coordinate Ascent	0.0502	0.0605	0.0584	0.0564	0.051
LambdaMART	0.029	0.0501	0.0449	0.0483	0.0526
RankNET	0.0499	0.0411	0.0483	0.034	0.0498
Random Forest	0.0574	0.0586	0.0707	0.064	0.0651
Linear Regression	0.0419	0.054	0.0529	0.0396	0.0548
LambdaRank	0.0069	0.0072	0.0327	0.0244	0.0021
ES-Rank	0	0.0024	0.0016	0.001	0.001
IESR-Rank	0.0005	0.0041	0	0.0028	0
IESVM-Rank	0.004	0.005	0.007	0.006	0.001

Table B.50: ERR@10 results for ranking models using evolutionary and machine learning techniques on NP2004

Algorithm	ERR@10 Results in NP2004 Dataset Folds				
	Fold1	Fold2	Fold3	Fold4	Fold5
RankBoost	0.0227	0.0331	0.0426	0.0405	0.0384
RankSVM	0.0352	0.0429	0.0391	0.0495	0.347
ListNET	0.0173	0	0.0203	0.0063	0.0046
AdaRank	0.0042	0.0035	0.0008	0.0306	0.027
MART	0.0244	0.0455	0.0405	0.0273	0.0269
Coordinate Ascent	0.0302	0.0483	0.0507	0.0427	0.0396
LambdaMART	0.0293	0.0228	0.0297	0.0348	0.0319
RankNET	0.0344	0.0436	0.05	0.0433	0.0453
Random Forest	0.0355	0.0402	0.0441	0.0385	0.0349
Linear Regression	0.0311	0.0263	0.0358	0.0242	0.0295
LambdaRank	0.0093	0.0136	0.0025	0.0045	0.021
ES-Rank	0	0	0	0	0
IESR-Rank	0.0041	0	0.0042	0	0
IESVM-Rank	0.0062	0	0.0037	0	0

Appendix C

Detailed Results of the ESClick Experimental Study

Table C.1: The training data results for DCM and ES-Click on MQ2008

Training data results for MQ2008						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.4848	0.51238	0.5136	0.4741	0.463
	DCM Model	0.2849	0.4488	0.2824	0.2547	0.276
NDCG	ES-Click	0.509	0.5393	0.5238	0.5037	0.4847
	DCM Model	0.3149	0.4857	0.31264	0.2815	0.3052
P@10	ES-Click	0.2776	0.2896	0.2922	0.2652	0.2586
	DCM Model	0.2016	0.2748	0.206	0.1827	0.1916
RR@10	ES-Click	0.5538	0.5909	0.5757	0.5297	0.50216
	DCM Model	0.3331	0.5064	0.3	0.2635	0.3197
ERR@10	ES-Click	0.023	0.0233	0.0245	0.0237	0.0226
	DCM Model	0.0546	0.0917	0.04899	0.041	0.05299

Table C.2: The test data results for DCM and ES-Click on MQ2008

Test data results for MQ2008						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.4497	0.436	0.4491	0.524	0.5045
	DCM Model	0.2811	0.3976	0.2326	0.2918	0.2879
NDCG	ES-Click	0.4821	0.4253	0.4662	0.5535	0.5402
	DCM Model	0.3082	0.4119	0.2583	0.3015	0.3439
P@10	ES-Click	0.26169	0.2457	0.2492	0.3257	0.2704
	DCM Model	0.2014	0.2317	0.1767	0.2111	0.2131
RR@10	ES-Click	0.4995	0.4621	0.5271	0.5935	0.54815
	DCM Model	0.3424	0.431	0.2429	0.2896	0.3391
ERR@10	ES-Click	0.0258	0.0265	0.0209	0.0288	0.0251
	DCM Model	0.05582	0.0722	0.0395	0.04635	0.05397

Table C.3: The training data results for DCM and ES-Click on MQ2007

Training data results for MQ2007						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.4565	0.4652	0.4674	0.4747	0.4697
	DCM Model	0.3326	0.3409	0.2859	0.3703	0.31514
NDCG	ES-Click	0.4235	0.4464	0.4519	0.4367	0.4535
	DCM Model	0.2771	0.2877	0.2004	0.32496	0.2409
P@10	ES-Click	0.3752	0.3723	0.3812	0.3957	0.3844
	DCM Model	0.2743	0.2829	0.2162	0.3064	0.2519
RR@10	ES-Click	0.5579	0.5726	0.5842	0.5856	0.5746
	DCM Model	0.3925	0.4168	0.3193	0.4574	0.36469
ERR@10	ES-Click	0.0218	0.0226	0.02486	0.02597	0.02483
	DCM Model	0.06278	0.06512	0.047	0.07597	0.05534

Table C.4: The test data results for DCM and ES-Click on MQ2007

Test data results for MQ2007						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.4847	0.4537	0.4489	0.4273	0.4683
	DCM Model	0.36	0.3551	0.2738	0.3356	0.3076
NDCG	ES-Click	0.4523	0.4275	0.4407	0.3915	0.4398
	DCM Model	0.3	0.3014	0.2029	0.2793	0.2265
P@10	ES-Click	0.4033	0.3903	0.3645	0.3584	0.3811
	DCM Model	0.2988	0.3115	0.2123	0.2711	0.24484
RR@10	ES-Click	0.5658	0.55	0.5563	0.5133	0.57624
	DCM Model	0.41933	0.4344	0.3069	0.3999	0.3459
ERR@10	ES-Click	0.0285	0.02568	0.02656	0.02178	0.02764
	DCM Model	0.0715	0.0685	0.04633	0.06251	0.0495

Table C.5: The training data results for DCM and ES-Click on HP2004

Training data results for HP2004						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.8568	0.8554	0.8131	0.7738	0.8094
	DCM Model	0.0036	0.0361	0.1299	0.38638	0.006
NDCG	ES-Click	0.8437	0.8366	0.8224	0.8204	0.8435
	DCM Model	0	0.0403	0.1543	0.4086	0.0064
P@10	ES-Click	0.107	0.1	0.0911	0.1022	0.1044
	DCM Model	0	0.007	0.029	0.0578	0.0022
RR@10	ES-Click	0.82	0.8556	0.7819	0.8217	0.8174
	DCM Model	0	0.0389	0.1125	0.3876	0.0022
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0	0	0	0	0

Table C.6: The test data results for DCM and ES-Click on HP2004

Test data results for HP2004						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.695	0.75	0.812	0.7253	0.58
	DCM Model	0.01797	0.0723	0.036	0.4554	0.0038
NDCG	ES-Click	0.7084	0.7579	0.6841	0.7438	0.5588
	DCM Model	0.0237	0.07	0.0433	0.5201	0
P@10	ES-Click	0.1	0.107	0.08	0.0933	0.07
	DCM Model	0.007	0.007	0.0133	0.08	0
RR@10	ES-Click	0.5661	0.775	0.8167	0.7778	0.5578
	DCM Model	0.011	0.07	0.028	0.4595	0
ERR@10	ES-Click	0.0007	0	0	0	0
	DCM Model	0.0007	0	0	0	0

Table C.7: The training data results for DCM and ES-Click on TD2004

Training data results for TD2004						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.2452	0.27113	0.25298	0.2279	0.237
	DCM Model	0.0384	0.0418	0.0225	0.01726	0.01418
NDCG	ES-Click	0.3678	0.3651	0.3576	0.38404	0.31014
	DCM Model	0.0549	0.0518	0.02254	0.01519	0.01198
P@10	ES-Click	0.2689	0.3178	0.2956	0.2733	0.2622
	DCM Model	0.0444	0.0467	0.02444	0.01333	0.0111
RR@10	ES-Click	0.6986	0.6754	0.5869	0.571	0.6107
	DCM Model	0.136	0.1313	0.0347	0.03833	0.0333
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0	0	0	0.0027	0

Table C.8: The test data results for DCM and ES-Click on TD2004

Test data results for TD2004						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.1979	0.2308	0.22719	0.1949	0.2643
	DCM Model	0.0331	0.0208	0.0335	0.01027	0.0276
NDCG	ES-Click	0.2728	0.348	0.2008	0.3222	0.31659
	DCM Model	0.05546	0.0235	0.0447	0	0.0408
P@10	ES-Click	0.2267	0.2733	0.2333	0.2467	0.2667
	DCM Model	0.0533	0.02667	0.03333	0	0.0333
RR@10	ES-Click	0.4392	0.5452	0.3579	0.3475	0.5719
	DCM Model	0.1303	0.0495	0.1083	0	0.1233
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0	0	0	0	0

Table C.9: The training data results for DCM and ES-Click on NP2004

Training data results for NP2004						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.2591	0.26197	0.2599	0.1296	0.2379
	DCM Model	0.01754	0.07488	0.0859	0.0196	0.02
NDCG	ES-Click	0.3482	0.4074	0.3647	0.385	0.3229
	DCM Model	0.01869	0.1126	0.1159	0.0192	0.02096
P@10	ES-Click	0.2689	0.2578	0.3044	0.3089	0.2467
	DCM Model	0.0156	0.0956	0.09556	0.0111	0.0222
RR@10	ES-Click	0.7334	0.6837	0.6973	0.6481	0.6428
	DCM Model	0.04157	0.2594	0.2331	0.03095	0.03259
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0.0019	0.0125	0.0189	0	0

Table C.10: The test data results for DCM and ES-Click on NP2004

Test data results for NP2004						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.2192	0.21956	0.2058	0.0813	0.2614
	DCM Model	0.0113	0.04863	0.1242	0.0131	0.0406
NDCG	ES-Click	0.2916	0.3709	0.2371	0.3099	0.3508
	DCM Model	0	0.0661	0.17269	0.0107	0.04578
P@10	ES-Click	0.2133	0.2467	0.1467	0.2467	0.22
	DCM Model	0	0.0467	0.1333	0.01333	0.0333
RR@10	ES-Click	0.4467	0.5889	0.5673	0.4973	0.5735
	DCM Model	0	0.175	0.3417	0.01667	0.0889
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0.0028	0.0169	0.0287	0	0

Table C.11: The training data results for DCM and ES-Click on HP2003

Training data results for HP2003						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.8626	0.8077	0.76656	0.8098	0.838
	DCM Model	0.00822	0.1509	0.03151	0.0254	0.0028
NDCG	ES-Click	0.8246	0.8437	0.7615	0.8148	0.8696
	DCM Model	0.008	0.1672	0.03514	0.02821	0
P@10	ES-Click	0.12	0.1156	0.1056	0.108	0.108
	DCM Model	0.0022	0.027	0.007	0.0056	0
RR@10	ES-Click	0.8592	0.8493	0.7946	0.7798	0.7448
	DCM Model	0.0038	0.1511	0.0337	0.02196	0
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0.00024	0.0097	0.0021	0.00137	0

Table C.12: The test data results for DCM and ES-Click on HP2003

Test data results for HP2003						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.71065	0.8671	0.72791	0.7049	0.66765
	DCM Model	0.04224	0.1554	0.02047	0.00239	0.0049
NDCG	ES-Click	0.7169	0.87	0.78	0.785	0.7492
	DCM Model	0.0436	0.1805	0.0217	0	0
P@10	ES-Click	0.077	0.107	0.12	0.107	0.1
	DCM Model	0.0067	0.03	0.007	0	0
RR@10	ES-Click	0.7423	0.8678	0.8472	0.7492	0.5992
	DCM Model	0.0444	0.1706	0.017	0	0
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0.0028	0.0109	0.00139	0	0

Table C.13: The training data results for DCM and ES-Click on TD2003

Training data results for TD2003						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.3653	0.373	0.2938	0.2845	0.3649
	DCM Model	0.0059	0.211	0.0093	0.0115	0.03485
NDCG	ES-Click	0.4452	0.4927	0.3516	0.3711	0.3479
	DCM Model	0	0.2582	0.0055	0.0133	0.0345
P@10	ES-Click	0.21	0.2133	0.207	0.1833	0.19
	DCM Model	0	0.137	0.007	0.01	0.017
RR@10	ES-Click	0.5831	0.715	0.6062	0.725	0.7306
	DCM Model	0	0.3289	0.0125	0.0483	0.0822
ERR@10	ES-Click	0	0	0	0	0
	DCM Model	0	0.02778	0.0008	0.003	0.0051

Table C.14: The test data results for DCM and ES-Click on TD2003

Test data results for TD2003						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.1624	0.2497	0.3833	0.2599	0.2257
	DCM Model	0.0181	0.1696	0.0291	0.0048	0.0548
NDCG	ES-Click	0.2286	0.3826	0.323	0.3301	0.1641
	DCM Model	0.0214	0.2279	0.0412	0	0.103
P@10	ES-Click	0.1499	0.18	0.18	0.13	0.14
	DCM Model	0.01	0.15	0.02	0	0.07
RR@10	ES-Click	0.1968	0.6	0.177	0.3458	0.4472
	DCM Model	0.05	0.4292	0.1	0	0.2536
ERR@10	ES-Click	0.003125	0	0	0	0.009375
	DCM Model	0.003125	0.03844	0.0082	0	0.01975

Table C.15: The training data results for DCM and ES-Click on NP2003

Training data results for NP2003						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.6874	0.7769	0.7555	0.7283	0.6971
	DCM Model	0.0047	0.2994	0.5611	0.2908	0.0025
NDCG	ES-Click	0.7289	0.8108	0.8024	0.7998	0.7223
	DCM Model	0.006	0.3396	0.5839	0.3244	0.0033
P@10	ES-Click	0.096	0.0983	0.0949	0.097	0.088
	DCM Model	0.00159	0.0527	0.0694	0.049	0.0011
RR@10	ES-Click	0.7553	0.7765	0.7433	0.75	0.6958
	DCM Model	0.0037	0.29182	0.55686	0.2755	0.0012
ERR@10	ES-Click	0.00009	0.00009	0.00009	0	0
	DCM Model	0.00023	0.0182	0.0348	0.01722	0.00007

Table C.16: The test data results for DCM and ES-Click on NP2003

Test data results for NP2003						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.6019	0.6114	0.5857	0.75132	0.7478
	DCM Model	0.001	0.1765	0.4702	0.39801	0.0026
NDCG	ES-Click	0.6315	0.6848	0.7144	0.7804	0.7575
	DCM Model	0	0.1991	0.528	0.4358	0
P@10	ES-Click	0.08	0.087	0.09	0.0881	0.08
	DCM Model	0	0.0333	0.0733	0.0614	0
RR@10	ES-Click	0.6057	0.63	0.6298	0.7139	0.7631
	DCM Model	0	0.158	0.47948	0.3906	0
ERR@10	ES-Click	0	0	0.0021	0.00029	0
	DCM Model	0	0.0098	0.03	0.02441	0

Table C.17: The training data results for DCM and ES-Click on Ohsumed

Training data results for Ohsumed						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.4868	0.4912	0.441	0.4324	0.4324
	DCM Model	0.2952	0.3228	0.33	0.4006	0.2844
NDCG	ES-Click	0.4823	0.4978	0.4731	0.4667	0.448
	DCM Model	0.161	0.1884	0.2514	0.425	0.2355
P@10	ES-Click	0.5587	0.5921	0.5438	0.5031	0.4891
	DCM Model	0.2667	0.2873	0.3438	0.4391	0.2984
RR@10	ES-Click	0.8075	0.4772	0.8375	0.8142	0.7402
	DCM Model	0.4147	0.8276	0.4944	0.7647	0.4705
ERR@10	ES-Click	0.0312	0.0396	0.04	0.0353	0.03486
	DCM Model	0.0696	0.0924	0.1059	0.1934	0.1038

Table C.18: The test data results for DCM and ES-Click on Ohsumed

Test data results for Ohsumed						
Evaluation Metrics	Approcahes	Fold1	Fold2	Fold3	Fold4	Fold5
MAP	ES-Click	0.3587	0.4426	0.4594	0.5127	0.4555
	DCM Model	0.1839	0.2748	0.3302	0.5043	0.3648
NDCG	ES-Click	0.3722	0.4496	0.44	0.4924	0.5051
	DCM Model	0.1277	0.1967	0.224	0.4852	0.274
P@10	ES-Click	0.3727	0.4905	0.519	0.5762	0.5238
	DCM Model	0.1455	0.2714	0.3143	0.5762	0.4095
RR@10	ES-Click	0.6315	0.3845	0.7966	0.7476	0.5822
	DCM Model	0.3697	0.8056	0.5152	0.8095	0.6143
ERR@10	ES-Click	0.0488	0.0428	0.0404	0.0827	0.06367
	DCM Model	0.0657	0.083	0.1052	0.2142	0.1237