Investigating the Case of Weak Baselines in Ad-hoc Retrieval and Question Answering

Master's Thesis

Francisco Morales Martínez

Delft University of Technology

Investigating the case of weak baselines in Ad-hoc Retrieval and Question Answering

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
DATA SCIENCE & TECHNOLOGY TRACK

by

Francisco Morales Martínez
born in Quito - Ecuador

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

Investigating the case of weak baselines in Ad-hoc Retrieval and Question Answering

Author: Francisco Morales Martínez
Student id: 4723589
Email: fjmoralesmartinez@hotmail.com

Abstract

Weak baselines have been present in Information Retrieval (IR) for decades. They have been associated with IR progress stagnation, baseline selection bias to publish results more readily, and models' effectiveness reproducibility issues that hinder the validation of results by independent research teams. Weak baselines have been studied by the IR community; however, the focus has been almost exclusive on ad-hoc retrieval, the most popular IR task, leaving outside other IR tasks and datasets recently developed. Current deep neural IR research is particularly vulnerable to the issues with weak baselines due to the hype surrounding deep learning.

In this thesis we investigate the cases of weak baselines in ad-hoc retrieval and question answering (QA), two representative IR tasks among 13 cases of weak baselines we found in current deep neural IR research from EMNLP 2018 conference. In particular, we study whether the recently introduced deep neural IR models are actually significantly more effective than the reported IR baselines or than LambdaMART, the Learning to Rank (LTR) model we propose plus hyperparameter optimization (HPO). We also benchmark two HPO methods: RS and BOHB, to determine which method is more efficient to retrieve a good hyperparameter configuration.

Throughout our experiments we show that the effectiveness of the novel deep neural IR models can be difficult to replicate, it might be lower than reported, and that it is not necessarily significantly higher than the baseliness. Furthermore, we demonstrate that BOHB is more efficient than RS, but the HPO process not always improves the effectiveness of LambdaMART significantly.

Thesis Committee:

Chair:	Dr. C. Hauff, Faculty EEMCS, TUDelft, supervisor
Committee Member:	Dr. ir. C. C.S. Liem, Faculty EEMCS, TUDelft
Committee Member:	Prof. dr. ir. S. Verwer, Faculty EEMCS, TUDelft

Preface

With this thesis I conclude my master studies, the first step towards countless dreams that I will wholeheartedly pursue after. The last two years seem to have happened as a flash before my eyes, but they have left a profound and indelible mark inside me. I have become a more knowledgeable professional, a better researcher, but most importantly, I have grown as a person in ways that would not have been possible without the people I encountered alongside this path.

I want to extend my gratitude to my supervisor, Claudia Hauff; her insightful feedback and encouraging words helped me to improve the quality of my work every time. Likewise, thanks to Arthur Câmara and Wanning Yang, without their collaboration this work would not have been possible. My esteem to Cynthia Liem and Sicco Verwer, who enthusiastically accepted being part of my defense committee. Additionally, I want to acknowledge the members of the Lambda-Lab from TU Delft for the discussions we had about top-notch IR and NLP research. Those were exciting and humorous sessions that expanded our researchers' minds.

Special thanks to my parents, sister, and brother for their unconditional love and support. I owe them this accomplishment just as all the other achievements in my file. Thanks to the many friends I have made at Uni, for sharing their amazing personalities with me.

My gratefulness to Andra H., who has not stopped cheering me up for a second during this process, and with whom the future always looks brighter.

Last but not least, I want to thank the Ecuadorian government institutions SENESCYT and IFTH, for funding my studies at TU Delft through the *Convocatoria Abierta 2016* scholarship.

Francisco Morales Martínez
Delft, the Netherlands
February 14, 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep neural networks have significantly improved the state of the art (SOTA for brevity) of fields like computer vision [106] or Natural Language Processing (NLP) [123]. Although this is also becoming the case for IR with the most novel developments such as the Bidirectional Encoder Representations from Transformers (BERT) [34], the improvements brought by many other deep neural network approaches to Information Retrieval (IR) are still modest compared to non-deep neural networks methods and might be unwarranted according to a recent study performed by Lin [71]. Lin [71] warns the IR community about the dangers of using weak baselines in current IR research, specially in deep neural IR.

1.1 Weak Baselines in IR research

In an empirical research field such as IR the effectiveness improvement of a new model is determined comparing it to the effectiveness achieved by previous models referred as *baselines*. Each model is tested on a given IR task, document collection, and a set of evaluation metrics.

Recently, Lin [71] warns the IR community that *weak* baselines are being employed in current IR research. Weak IR baselines are characterized as non-competitive models [11] that lack the latest discoveries in the field [11] and are inadequately-tuned [71]. Other researchers [11, 60, 140] have demonstrated that the use of weak baselines in IR is not new but still a prevalent problem, particularly in the most popular retrieval task, ad-hoc retrieval, which consists in determining the relevance rank order of given documents with respect to a query. The use of weak baselines in IR involves several issues: (1) progress stagnation, i.e. the effectiveness of new proposed models has reached a plateau instead of showing an upward trend throughout the years [11, 143]; (2) baseline selection bias, meaning that weak baselines might be chosen because they are easy to beat and show statistical significance upon them, facilitating the publication of these kind of results [11, 108, 71]; and (3) models' effectiveness reproducibility issues, concerning the practices that hinder the verification of results by independent investigators [11, 73]. Deep neural IR is particularly vulnerable to these issues due to the hype surrounding deep learning (DL) [71].

Researchers have addressed the issues with weak baselines in IR from different perspectives. Firstly, Armstrong et al. [11], Yang et al. [143], and Kharazmi et al. [60] gathered a large evidence about the presence of weak baselines in ad-hoc retrieval by the means of meta-analyses surveying the IR literature over two decades until recent years. Secondly, Armstrong et al. [10] built a public online system to keep a register of the effectiveness of the best IR models such that these models could be easily queried and considered as baselines. However, this system is not active anymore; apparently, it did not gain traction among the IR practitioners [71]. Thirdly, an entire SIGIR workshop has been devoted for the effectiveness replicability of different IR baselines: the Open-Source IR Replicability Challenge, OSIRRC 2019 [73]. Fourthly, most researchers [11, 71, 60, 143] agree on the need to employ *strong* baselines, which are competitive, mature, and adequately tuned models (i.e. the exact opposite to weak baselines), an objective that is difficult to reach due to the unavailability of open implementations for the SOTA models. Lastly, the IR community advocates for good and reproducible research practices when comparing IR models; for instance, it encourages the release of the code when new models are proposed [96]. Some of these measures are taking effect, since we see that comparisons to weak baselines are being held from publication in recent conferences related to IR [41].

1.2 Reducing the gap

The IR community has made important contributions to tackle the problem of weak baselines; nonetheless, there are still gaps to be filled. In this work, we focus in four aspects to address weak baselines: the IR tasks, candidate strong IR baselines, hyperparameter optimization (HPO), and reproducibility.

IR tasks. To the best of our knowledge, past research has focused exclusively on the ad-hoc retrieval task for which we already know many prominent cases of weak baselines. The main reason being the availability of hundreds of studies about this task (on the same dataset) that allows to establish an anlysis over baselines. However, we still lack this kind of information for other IR tasks. We conduct an examination of the IR studies presented at the latest edition of the Empirical Methods for Natural Language Processing (EMNLP[1]) conference (an influential research venue with >2000 paper submissions that hosts IR and other related research topics [7]) and find that weak baselines are employed in diverse IR tasks (e.g. [80, 68, 112]).

In this work we focus on two IR tasks: ad-hoc retrieval and question-answering (QA). Ad-hoc retrieval is a classic IR task and has been largely studied [78]. It consists of an IR system that returns a ranked list of documents given a query representing the user's information needs [78]. Usually, queries and documents are heterogeneous, meaning that documents are longer than queries. On the other hand, questions and answers are both short texts expressed in natural language. In contrast to ad-hoc queries, questions convey a more defined information need [120, 45]. Moreover, while ad-hoc retrieval is

[1] https://www.emnlp2018.org

concerned with how relevant a document is with respect to a query, QA aims to find the *right* answer for a question [120]. To exemplify the difference between ad-hoc retrieval and QA, let's imagine we search for *"European countries"* on the internet and the returned documents include the whole Wikipedia[2] pages of The Netherlands and Spain but also web pages advertising tours to those countries; both groups of retrieved documents from the web could be seen as relevant depending on the user *intent*. On the other hand if we submit the question *"What are the names of the European countries?"*, we expect a specific list with the countries' names. Furthermore, if instead we ask *"Are Angola and Morocco European countries?"*, possible answers are a simply *"No"* or *"No, they are African countries"*.

The nuances in the way questions can be asked and how the answers are tailored make more difficult to evaluate a QA model regarding how effectively it retrieves the correct answer [120]. This also makes the design of standard QA tasks, document collections, and evaluation metrics more complex [126]. We suspect that QA has passed unnoticed in the research related to weak baselines because of the difficulty to find enough studies performed on standard QA tasks and dataset collections, at least to the extent we have seen in ad-hoc retrieval. For example, Armstrong et al. [11] and Yang et al. [143] altogether have compared over 200 ad-hoc retrieval papers that employ the same dataset. Since QA has a years-long research trajectory [131], it is a common task in current deep neural IR research [45], and it has not been taken into account in previous weak baseline research, we include it in our current investigation.

Candidate strong IR baselines. Opposed to weak baselines, *strong* baselines are competitive, mature, and adequately-tuned models [11, 71]. Ideally, the effectiveness of new proposed models would always be compared to that of the SOTA models [60], but for practical constraints such as the unavailability of a public implementation of the required model, this might not be feasible [72]. Past IR literature have proposed different types of IR models as baselines, from traditional ones generally considered as weak baselines [71, 60] to learning to rank (LTR) approaches that have been regarded as strong baselines [60]. However, we have not seen LTR methods proposed as strong baselines for deep neural IR models; probably because of the need for feature engineering, i.e. tailoring the features for the particular IR task and dataset.

Unlike previous probabilistic models (e.g. BM25 or language models), LTR is a machine learning (ML) approach based on the combination of features representing the documents as vectors, and also ground truth labels indicating the relevance relationship between documents and queries [74]. Common features combined through the LTR approach include term frequencies and BM25 scores. Other important features are discovered by the means of feature engineering [74]. LTR is precisely a step before deep neural IR. Deep neural IR models actually aim to learn these kind of relevant features automatically, easing the feature engineering process which is time-consuming and requires expert knowledge [82]. In this thesis we propose to examine the effectiveness

[2] https://www.wikipedia.org/

of a LTR model, LambdaMART [137], as our candidate for strong baseline for deep neural IR methods.

Hyperparameter optimization. Several IR models have *hyperparameters*, i.e. the settings that control the model's behavior and are not learnt when the ranking function is built (e.g. during the training phase of a LTR approach). Hyperparameters affect the effectiveness of an IR model. It has been previously established that different hyperparameter values work best for different datasets [63]. Choosing the right hyperparameter *configuration* is not a trivial task. A model can contain several hyperparameters of different types, what can result in a large and complex configuration search space [54]. Moreover, the size of the document collection and the complexity of the execution of the IR algorithm could render HPO a time-consuming process [54]. The most conventional HPO technique, grid search, is not efficient because attempts to explore the entire configuration space in search for the best hyperparameter configuration; on the other hand, random search might take a long time until it converges to the best solution [37]. Recently, many HPO studies have been devoted to overcome these shortcomings with sophisticated techniques Hutter et al. [54]. In this thesis, we employ a HPO method that combines Bayesian optimization and HyperBand (two recent advances on this field) into one algorithm: BOHB. BOHB has been successfully applied for optimizing the hyperparameters of complex machine learning models and deep neural networks on large datasets [37], outperforming previous approaches (for more details, see section 2.5.4). We also compare BOHB versus random search, a common HPO baseline Falkner et al. [37].

In the context of weak baselines, the use of HPO presents two advantages: (1) it is easier to reproduce the results of a model when an automatic open-source HPO tool is employed, and (2) researchers usually spent a considerable amount of time and effort to optimized their recently proposed deep neural IR models; thus, applying HPO to the proposed IR baseline is a step towards a fair comparison of models [71, 108, 54].

Reproducibility. Craswell et al. [32] and Lin [71] denounce the reproducibility issues of deep neural IR models. Reproducing research results is key in empirical research to eventually accept the drawn conclusions of previous studies as facts [96, 93]. Potthast et al. [96] point out several important aspects for reproducible research in computer science, such as code and data availability, as well as self-contained papers with sufficient details. The IR community has made available standard document collections [127, 97] and shared IR toolkits with several IR models' implementations (e.g. Anserini [142]) to facilitate IR research. With this thesis we build more evidence that validates or challenges the findings of current deep neural IR research.

1.3 Research objectives

The aim of this work is to investigate the cases of weak baselines in recent deep neural IR research, with a focus on four aspects: (1) broaden the weak baseline analysis to include the QA task from a NLP conference with an IR track, instead of a pure IR conference; (2) replicate and reproduce the effectiveness results for recently introduced deep neural IR models and (weak) IR baselines for ad-hoc retrieval and QA tasks with different datasets, (3) propose a mature LTR model as candidate strong baseline for these two IR tasks, and (4) optimize the hyperparameters of the LTR model with a more efficient HPO method. Consequently, the main research question of this work is:

> ***RQ:*** *Does the effectiveness advantage of new deep neural IR models proposed for ad-hoc retrieval and QA tasks still hold when compared to a SOTA LTR model with hyperparameter optimization?*

1.4 Approach

Our approach to investigate the case of weak baselines in current deep neural IR research is divided in three parts: (1) exploratory analysis of weak baselines in recent deep neural IR studies from EMNLP 2018 conference, (2) reproduction of effectiveness results of most effective deep neural IR model and most effective weak baseline for an ad-hoc retrieval task and a QA retrieval task, and (3) application of the optimized LTR baseline to the ad-hoc retrieval and QA tasks. Figure 1.1 summarizes the steps of our investigation approach.

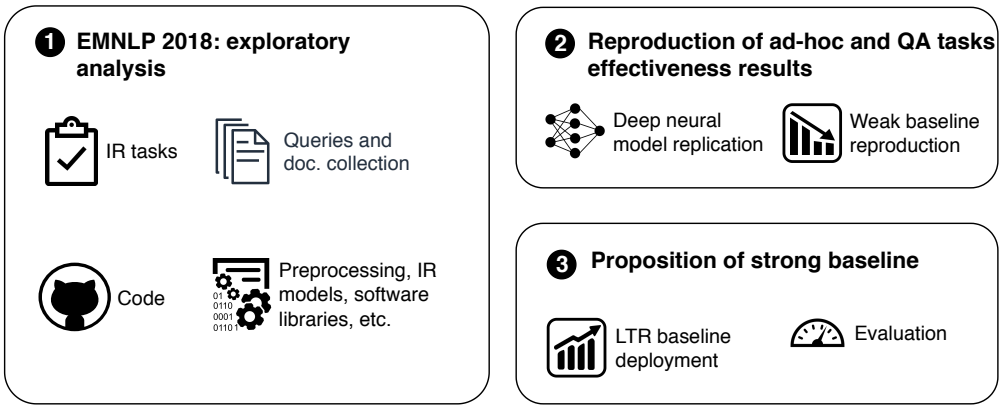


Figure 1.1: Approach for investigating weak baselines in current deep IR research.

1.4.1 EMNLP 2018: exploratory analysis

Previous weak IR baseline research has focused on premier IR conferences such as SIGIR or CIKM [11]. In contrast, EMNLP, the conference we examine in

this work, covers empirical methods applied in different disciplines such as machine translation, language modeling, IR, among others. This opens the spectrum of the weak baseline analysis to non-traditional (IR) tasks and datasets. We examine the papers from EMNLP 2018 where deep neural IR models and IR baselines have been applied to identify the characteristics of the IR tasks, datasets, effectiveness evaluation, reproducibility, and other relevant details. For each one of these papers we determine how the task at hand is modeled as an IR task, particularly distinguishing ad-hoc retrieval and QA tasks, as well as checking which data elements are regarded as queries/questions and which ones as documents/answers. Statistics about the datasets such as the average length and standard deviation of documents and queries are collected. Moreover, the results for the best proposed deep neural IR model and the best weak baseline are highlighted according to the evaluation metric used in each paper besides any claim of statistical significance. In contrast to the meta-analyses performed in [11, 143] that focus in ad-hoc retrieval exclusively, we explore all IR related tasks where deep neural IR models and weak baselines have been employed[3].

For reproducibility purposes, the availability of datasets and code (specially for newly proposed deep neural networks models), preprocessing steps, and other details are inspected. The outcome of this exploratory analysis is a list of papers relevant to our work, suitable for our reproducibility experiments explained in the next section.

1.4.2 IR effectiveness results reproducibility

In this phase we propose the replication and reproduction of the effectiveness results for deep neural IR models and weak baselines respectively, for two reasons: (1) confirm the effectiveness results reported for these two types of IR models, and (2) break the habit of simply copying effectiveness results tables from previous studies, which is a practice observed in the NLP community [71]. Copying previous results tables is dangerous because new research then focuses on beating previous effectiveness results instead of the science behind their fruition [115]. Moreover, simply copying previous table results is in a way accepting those are facts, without being scrutinized, validated, or challenged.

Our reproduction approach is closely related to that of Potthast et al. [96]; however, it differs in three aspects: (1) we do not re-implement any paper's model from scratch, but reuse the resources made available by the authors, such as code and datasets; (2) Potthast et al. [96] recruited one volunteer for each paper they reproduced, whereas our work is an individual effort that obliges to limit the number of papers to reproduce, and (3) we propose the deployment of a candidate strong IR baseline that is not part of the original paper. Consequently, from the previously identified list of papers suitable for reproduction, we pick two: one related to ad-hoc retrieval [80] and one related to QA [67]. We choose these two papers because they cover two representative IR tasks, ad-hoc retrieval and QA [45, 120], the two main IR baselines we encounter in the

[3]The exploratory analysis we make is the outcome of a joint effort of four members of the Lambda-Lab at TU Delft: Arthur Câmara, Wanning Yang, Claudia Hauff, and Francisco Morales (author of this thesis).

exploratory analysis (tf-idf, BM25), and because they provide the required artifacts to run our reproducibility experiments. The work of McDonald et al. [80] in particular includes a dataset (Robust04) that lets us establish comparisons with previous weak IR baseline research.

Subsequently, the reproduction of effectiveness results is split in two cases: replication of the most effective proposed deep neural IR model and reproduction of the most effective weak IR baseline for each task. Following the terminology from the Association for Computing Machinery (ACM)[4], we talk about results *replication* when we use the same artifacts that the original researchers employed to obtain their effectiveness results. This is the case for the deep neural IR models of our selected papers, for which the code and datasets are available. On the other hand, we refer to results *reproduction* when the same artifacts are not accessible and the independent research team has to come up with their own. We face this circumstance with the weak IR baselines because their implementations are not provided by the authors of the papers; thus, we opt to employ an IR toolkit instead.

For our reproducibility approach we follow the same IR model evaluation pipeline established in each selected paper. Broadly speaking, this pipeline comprises: preprocessing of queries and documents, IR model execution, and effectiveness evaluation (see figure 1.2). The preprocessing prepares the queries and documents before any model is applied. During IR model execution we respect the hyperapameters set by the papers' authors, or use default values if they are not specified. Additionally, five-fold cross-validation is carried out in the evaluation pipeline of paper [80] for Robust04 exclusively, due to its relative small size. Cross-validation helps to obtain an stable and confident estimate of the supervised models' effectiveness, specially when training data is scarce [101]; it provides a better approximation of the true error [133]; and also helps to avoid overfitting [101]. Since our goal is to emulate the same conditions under the target IR models have been tested, we do not assume other feats, e.g. hyperparameter optimization, that might distort the reproduced effectiveness results we get when compared to the effectiveness results reported in the papers. Finally, each model is evaluated with the same set of evaluation metrics and the same evaluation tools.

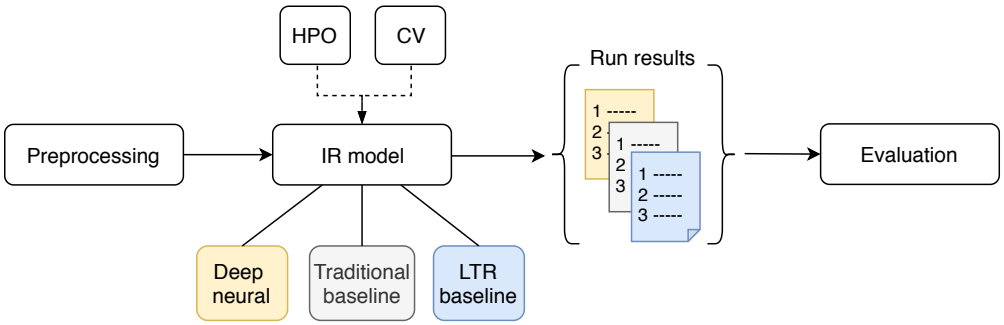


Figure 1.2: IR model effectiveness evaluation pipeline.

[4] https://www.acm.org/publications/policies/artifact-review-badging

1.4.3 Candidate strong LTR baseline application and optimization

This section details how we apply a LTR model as a baseline for the aforementioned ad-hoc retrieval and QA tasks. Likewise the reproduction phase for the deep neural IR and weak baseline models of a study, it is necessary to adapt the LTR model to the same IR evaluation pipeline (see figure 1.2). LTR models come in three categories: pointwise, pairwise, and listwise. LambdaMART is the SOTA LTR (pairwise) method we choose as our candidate for strong IR baseline. Likewise other machine learning approaches, LambdaMART requires to be trained with a set of features and ground truth labels that capture the documents' relevance with respect to the queries. Common features are extracted directly from other ranking models, such as BM25 scores or term frequencies [74], and are combined by LambdaMART to yield better effectiveness results. In this work we consider the features identified in [74].

Following the advice of Robertson [104] and Lin [71], we employ HPO to optimize our candidate for strong LTR baseline. We compare two HPO techniques: random search and BOHB. BOHB is a guided method that tends to be more efficient than RS [37]. HPO and CV affect the IR model execution and are optional components of the pipeline as depicted in figure 1.2.

1.5 Scientific Contributions

This work builds evidence of the cases of weak baselines in ad-hoc retrieval and QA, two representative IR tasks in recent deep neural IR research. It also tests whether the effectiveness advantage of new proposed deep neural IR models over weak baselines holds when compared to a SOTA LTR baseline. The additional contributions of this study are:

- A literature survey on recent cases of weak baselines in deep neural IR from EMNLP 2018.
- An independent reproduction of recent deep neural IR research for the ad-hoc retrieval and QA tasks.
- A benchmark of two different HPO methods for tuning our candidate for strong LTR baseline model, including BOHB, a SOTA HPO technique.
- The release of the code accompanying the experiments performed in this work for reproducibility purposes[5].

1.6 Outline

The remainder of this thesis is organized as follows. In Chapter 2, we summarize the literature related to our work and the key ideas we adopt. Chapter 3 elaborates on the approach of our investigation of weak baselines in current

[5] https://github.com/fj-morales/investigating

empirical deep neural IR research. Chapter 4 describes the experiments we perform to replicate the recent deep neural IR models and to reproduce the weak IR baselines; it also describes how we deploy and optimize our candidate LTR baseline, followed by a discussion of the experimental results. In Chapter 5 we present the conclusions for our research and discuss lines of future work.

Chapter 2

Related Work

The purpose of this chapter is to establish the context in which this thesis is developed by describing related work. Section 2.1 introduces basic IR concepts, types of IR tasks, traditional models, and evaluation. Section 2.2 describes how LTR formulates the ranking problem as a ML problem that depends on extracted features to learn a ranking function. We briefly review the latest LTR models, with emphasis on LambdaMART, the SOTA pairwise model we choose for our experiments in chapter 4. An overview of neural IR is given in section 2.3, comprising the most common deep neural IR categories and architectures.

Subsequently, section 2.4 presents the state of affairs regarding weak IR baselines in current neural IR research, its main issues, and the approaches that have been proposed to address them. Lastly, section 2.5 surveys the two HPO mechanisms we compare in this work: random search, and BOHB.

2.1 Information Retrieval

In essence, IR deals with how to satisfy the information's need of a user by retrieving a ranked list of documents. For example, a new TU Delft student might be interested in finding the *TU Delft faculty buildings*. The student then submits this query to the search engine from TU Delft website[1], which responds with a list of text snippets ordered by their relevance w.r.t. the user's query. These text snippets represent the documents in this scenario. Internally, the search engine relies on a retrieval model to compute and assign the relevance score to each document contained in its index. The index comprises all the documents that might be searched, and can be web pages or other media files in this hypothetical scenario. Figure 2.1 illustrates the general information retrieval case[2].

[1] https://www.tudelft.nl/

[2] A comprehensive introduction to IR and its main concepts is given in [78]

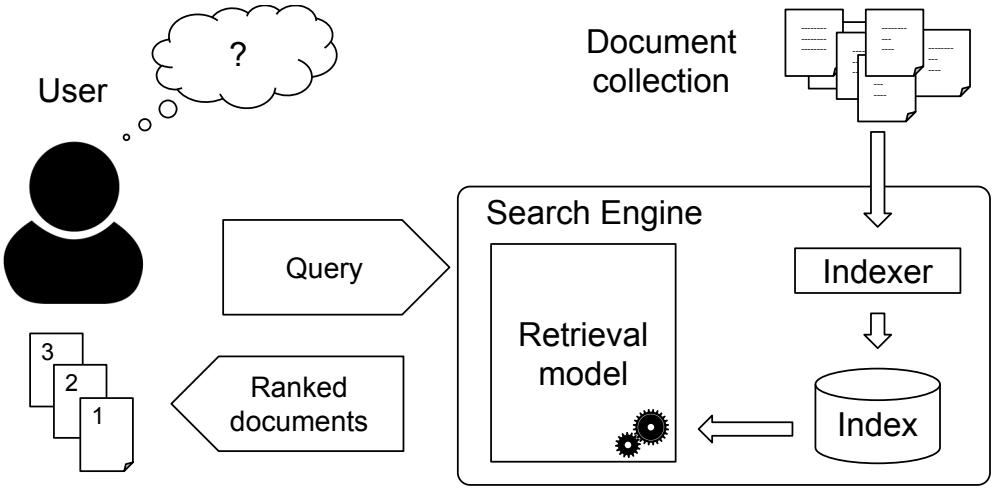


Figure 2.1: General information retrieval case, adapted from [46].

2.1.1 Tasks

Although an IR task might involve different types of media such as text [104], image [140], audio [77], or video [62] depending on the user needs, this thesis deals with text retrieval exclusively, where both queries and documents are expressed in natural language.

We find several IR tasks in literature: ad-hoc retrieval [24], QA [59], automatic search conversation [45], known-item retrieval [48], passage retrieval [85], among others. Ad-hoc retrieval and QA are two representative IR tasks that have been studied by the IR community for more than two decades (ad-hoc retrieval [11], QA [130]), and are still active research topics (ad-hoc [144], QA [29]). Therefore, we will focus on these two IR tasks along this thesis, and we describe them as follows.

Ad-hoc retrieval

Ad-hoc retrieval can be seen as the standard IR task and it has been one of the main tasks at the Text Retrieval Conference (TREC) [128]. It consists on retrieving a list of ranked documents (from a large document collection) that are relevant to an arbitrary user's query reflecting his information needs [78]. The term *ad-hoc* means that once the user submits the query to the system, he or she cannot refine it. Ad-hoc retrieval also presumes that the document collection does not require frequent addition of new documents, in other words the collection may be considered static [78, 120]. Since the document is the minimum unit in the collection to be retrieved, we talk about ad-hoc document retrieval, the most basic and largely studied form of IR [120]. Queries are typically made of a few keywords or a phrase written in natural language, while the document's body text is longer and spans through several sentences or paragraphs [82]. This *heterogeneity* might cause vocabulary mismatch problems [45]. For example, if we search for the word *laptop* but the documents have only the term *notebook*, the query will be unsuccessful [78]. The vocabulary mismatch

can be addressed with semantic matching at the word or sentence level, but exact matching is still needed for rare terms [45].

Question Answering

QA consists in returning a piece of text as an answer for a question that has been formulated in natural language [120]. Questions express well-defined information needs, but enclose more information than a list of keywords due to the syntactic and semantic relationships among the question terms [64]. Common question types include: *factoids*, which start with Wh-interrogated words like What, When, Where, Who, and require a fact in the body text; *list*, asking to list/name a set of items; and *definition*, asking for the definition of a term and typically starts with *"What is"* [64]. QA tasks vary from choosing the right answer from multi-answer candidates [103], ranking passages with the answer [98], or synthetizing the response as a human would do by gathering evidence from multiple sources [87]. In contrast to ad-hoc retrieval, where the entire document is regarded as relevant for the user's information need, in QA the user is interested in a concise piece of information holding the right answer [64]. For instance, the answer for the question: *What is the capital of Kosovo?* might be as short as a single word: *Prishtina*, or a whole sentence like: *The capital of Kosovo is Prishtina* [131]. Although questions and answers show less length *heterogeneity* and the notion of relevance is clearer, vocabulary mismatch is still a basic problem in QA. The evaluation of QA systems is also a challenge [45].

We employ terms such as *relevance*, *query*, *document* in general to explain several IR concepts alongside this thesis, whilst task specific terms like *question* and *answer* are employed when the distinction between ad-hoc retrieval and QA is needed.

2.1.2 Traditional IR Models

Many IR models have been proposed in the literature [12], and can be categorized as query-dependent and query-independent models [74]. Query-dependent models judge the relevance of the document with respect to the query (e.g. Boolean model [78]). On the other hand, query-independent models judge the importance of documents regardless of the query (e.g. PageRank) [74]. This thesis deals with query-dependent models only, and more precisely, with IR traditional models that typically appear as baselines in recent neural IR research. This includes vector space models, probabilistic models, language models, and pseudo relevance feedback models. These models predict a relevance degree of a document w.r.t. a query (e.g. by applying tf-idf weights), in contrast to boolean models that can only judge if a document is relevant or not. This relevance degree is useful to generate a list of relevance ordered documents.

This section does not pretend to thoroughly analyze each model, but to cover their intuitions and main aspects, allowing the reader to understand their general functioning and differences. LTR and neural IR query-dependent models are described in sections 2.2 and 2.3 respectively.

We first introduce tf-idf since it is the basic term weighting scheme employed in combination with many traditional, LTR, and neural IR models.

Tf-idf weighting

An early established and relatively computing-efficient way of judging the relevance of a document with respect to a query term is by counting the number of occurrences of term t in document d. Thus, the more frequent a term is in a document, the more relevant the document is. Using term occurrence to asses document relevance results in a term weighting scheme known as *term frequency*, denoted as $\text{tf}_{t,d}$ [78]. Term frequency ignores the order of terms, for which is also called *bag of words*. Highly repeated terms throughout the collection might hinder the discriminating power of term frequency. To counteract this effect, the weight of a common term is scaled down by its *inverse document frequency* (idf), defined in equation 2.1 [78].

$$\text{idf}_t = log\frac{D}{\text{df}_t} \tag{2.1}$$

Here, D represents the total number of documents in the collection and df_t is the number of documents in which term t appears. Thus, if a term is very rare in the collection, its idf weight will be very high, but very low if the term is common [78].

Tf and idf are combined in a single term weighting scheme by multiplying the two values (equation 2.2).

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \tag{2.2}$$

As a result, tf-idf weight is higher when the term is frequent in the document but uncommon in the collection, and lower when the term seldom occurs in the document but usually appears in the collection [78]. The tf-idf weight for a term seen in the collection but not in the current document is set to zero [78]. It is important to note that this is not the only definition of tf-idf, but there are also other variants that follow a similar principle [114].

Vector Space model

Vector Space model was first introduced by Salton et al. [107] in 1975. This model consists in representing documents as vectors, with one vector component for each term seen in the collection [78]. The term order does not influence the document vector representation, thus it is effectively seen as *bag of words*. Since queries can also be expressed as vectors in the same vector space of documents, queries and documents can be compared by, for instance, *cosine similarity*. Figure 2.2, illustrates a two-axes vector space representation of a query q and document d.

Cosine similarity (equation 2.3) is used to score documents according to how close they are to the queries in vector space. The numerator in equation 2.3 is the inner product between the query and the document vectors, while the denominator serves as a length-normalization term to account for the length

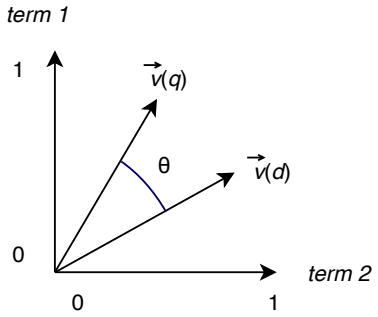


Figure 2.2: Cosine similarity illustrated in two-term axes vector space, adapted from [78].

difference of two similar documents [78]. Vectors might be binary, i.e. each vector component is a one or a zero representing the presence or absence of a term, or it might be weighted by tf-idf [78].

$$\text{sim}(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|} \tag{2.3}$$

Probabilistic models

IR systems are uncertain about the relevance of a document towards a query [78]. The Binary Independence Model (BIM) uses probability theory as a foundation to reason about such uncertainty [78]. BIM treats queries and documents as binary term incidence vectors and assumes that terms are independent of each other [104]. Although these assumptions are strong, BIM is able to estimate the probability of a document being relevant to a query and provide a ranked list of documents in descending order, with the estimated most relevant documents on top. BIM can be extended to the popular BM25 model [105] by incorporating term counts and other statistics, as we will see below.

BM25

BM25 stands for *best match* 25, a family of ranking models firstly introduced in the Okapi retrieval system by Robertson et al. [105] in 1995. BM25 extends the BIM model by adding document weights and query term weights [46]. Each variant of a BM25 model depends on the chosen components and parameters. A common BM25 instance is shown in equation 2.4[3] [74].

$$\text{BM25}(d,q) = \sum_{i=1}^{M} \frac{\text{idf}_{t_i} \cdot \text{tf}_{t_i,d} \cdot (k_1+1)}{\text{tf}_{t_i,d} + k_1 \cdot (1 - b + b \cdot \frac{dl}{avdl})} \tag{2.4}$$

Equation 2.4 shows the BM25 score of document d for a query q with query terms $t_1, ..., t_M$. We see the familiar term frequency $\text{tf}_{t_i,d}$ of t in d and inverse document frequency idf_{t_i} of t weights; dl is the document length and $avdl$

[3]Other BM25 models can be found in [104].

id the average document length in the collection. Additionally, we have the hyperparameters k_1 and b that need to be determined manually or through an automated process before the IR system is ready for the users. $k_1 > 0$ controls the contributions of the term frequencies, while $0 \leq b \leq 1$ is the document length normalization component; $b = 0$ means no normalization and $b = 1$ full normalization [104].

Language models for IR

Language models for IR (LMIR) lead to effective retrieval functions with less heuristic design, as opposed to vector space or BM25 models [146]. LMs have been successfully applied to speech recognition [56] and machine translation [19]. In IR, a language model is a statistical approach that estimates what is the probability that the query terms are generated from a given document [94]. Then these probabilities serve as relevance scores to rank documents w.r.t. to the query [146]. One of the first proposed IR language models is query likelihood, introduced by Ponte and Croft [94]. In general, query likelihood can be expressed as the conditional probability of a query q given the language model of document d (equation 2.5).

$$\text{score}(q, d) = prob(q|LM_d) \tag{2.5}$$

In this equation, LM_d represents the language model of document d. Different retrieval functions can be obtained depending on how the language model is learned from d [146]. For example, in [94] this function is a Bernoulli model with the vocabulary words of the language model represented as binary variables, where a value of one means the presence of a word in a query and zero its absence [146]. One common approach to learn a document's language model is to use a maximum likelihood method [74]; however, it requires some kind of smoothing to avoid assigning zero probability to unseen words [146]. In language models it is also possible to relax the strong assumption of term independence to model some limited dependency with bigram language models, as in [146]. Zhai [146] covers other forms of language models in more detail.

Pseudo relevance feedback

Considering that it is difficult to infer the user's intent behind a query, the underlying idea of *relevance feedback* is to get some input signal directly from the user to refine the query [78]. In general terms, the user indicates which documents he or she perceives as relevant and non-relevant from an initial ranked list. With this new information, the system refines the query and returns an adjusted list of documents that better reflects the user's information need.

Pseudo relevance feedback (PRF) performs the process described above automatically, i.e. without the user's intervention. The system performs a first retrieval round as usual, the top U documents are assumed to be relevant, then the query is refined according to these documents, and a new list of documents is generated. This process improves the effectiveness over the queries that do not perform well in the first retrieval round. However, this improvement comes

at the risk that the refined query *drifts* towards the most common topic of the first round top U documents [78]. Relevance model 3 (RM3) [8] is regarded as one of the most successful automatic *query expansion* (QE) models that follows the PRF idea [30]. QE is effective against the vocabulary mismatch problem between query and relevant documents [30]. The effectiveness of RM3 depends on three hyperparameters: the number of feedback documents, the number of feedback terms, and the weight of the original query (a decimal in the range [0,1]) [76]. More details about about the formulation of PRF and relevance models can be found in [65, 49, 76].

2.1.3 IR evaluation

At the beginning of this chapter we pointed out that the purpose of an IR system is to satisfy the user's information need by retrieving a set of *relevant* documents in ad-hoc retrieval or the *correct answer* in QA. However, relevance and answer correctness are difficult to assess. Relevance depends on the perspective of the user and the situation; thus, it is a subjective concept [78]. For example, a user looking for the word "cat" on a web search engine might be referring to the feline or perhaps to the abbreviation of the Caterpillar corporation. On the other hand, in QA a question might have multiple right answers that vary in quality [120]. The TREC QA track has had some difficulties setting an evaluation procedure over different track editions due to the varied types of questions and answers. For example, it has been restricted to factual questions, excluding opinions [120]. Moreover, the answer returned by any participant system has been constrained to be 250 bytes long (approximately a paragraph) at the beginning, then to 50 bytes (approximately a sentence), and lately to exact answers [120]. In addition to the answer, the systems had also to return the supporting document where the answer must be present [120].

Due to these challenges, IR evaluation is needed to measure the effectiveness of different models applied to ad-hoc retrieval and QA tasks. In the context of this thesis, IR evaluation allows us to differentiate weak IR baselines from strong baselines and SOTA models.

The standard approach to IR evaluation compromises three elements: a document collection, a test set of information needs (expressed as queries or questions), and a set of relevance (or *correct answer*) judgments [78]. In large modern collections, only a subset of the entire document collection is employed to assess the relevance of every document with respect to each query. This subset is formed with the top U documents returned by a number of IR systems, in a process known as *pooling* [78]. Each pair of query-document (question-answer) in this subset is given a relevant (correct) or non-relevant (wrong) judgment label, which is known as the *gold standard* or *ground truth* [78]. Collecting these ground truth labels for ad-hoc retrieval and QA tasks is expensive because it involves humans experts as annotators to assess the relevance or correctness of each query-document or question-answer pair respectively. This issue has been acknowledged in TREC ad-hoc retrieval [132] and QA [130] tasks. The effectiveness of an IR system is measured comparing its results against the ground truth labels in the test collection. However, when the IR models have param-

eters to tune, we also require a *development* (also called *validation*) set apart from the test set [78].

Standard collections

Due to the highly empirical nature of IR, the use of standard collections has been recognized as a necessity to evaluate and compare the effectiveness among models [32]. GOV2 is an example of standard collection for ad-hoc retrieval and QA tasks [78]. TREC collections are known for the rigorous process of manually judging relevance documents and right answers [120], which has lead to the construction of high quality but also expensive document collections [126]. TREC collections are also designed with the goal of being reusable for the evaluation of future IR systems [120]. Several collections have appeared in the last years to address both ad-hoc retrieval (e.g. Yahoo! Learning to rank challenge [27] with 36k queries and 883k documents) and diverse forms of QA(e.g. MS Marco [87] with 1M questions and 8.8M passages, or SQuAD [98] with 100k question-answer pairs). Although these collections have made their contribution and have encouraged more research in the field, most of them are not regarded as *standard* collections [32] and usually resort to crowdsourcing to build the ground truth labels (e.g. through Amazon Mechanical Turk[4]) [98, 99]. Building a standard collection for QA is particularly problematic because it is important to decide what is the collection unit that must be judged [129]. In ad-hoc retrieval this unit is a document with a unique identifier [129]; however, in a QA task the minimum unit is the answer string returned by the IR system and it generally differs among IR systems [129].

Evaluation metrics

Given a test collection with the ground truth judgments, we can compare the effectiveness of different IR models making use of evaluation metrics. Several metrics have been proposed in the literature and are problem-dependent [78, 120]. In this section we describe the metrics employed to compare different IR models in the studies related to weak baselines in neural IR research we investigate in this thesis.

Precision (P) is defined as the fraction of retrieved documents that are relevant (equation 2.6) [78]. Precision is a set-based measure that does not take the ranking position of retrieved relevant or non-relevant documents into account. In applications where speed is an important factor for user satisfaction, e.g. web search, Precision is reported over the top k retrieved documents ($P@k$), instead of the whole list, at the cost of ignoring relevant documents at the bottom of the list. P@k tends to be unstable and does not average well. Precision reported at the test collection-level is the average over the set of queries.

$$\text{Precision} = \frac{\#(\textit{relevant documents retrieved})}{\#(\textit{retrieved documents})} \tag{2.6}$$

[4] https://www.mturk.com/

Mean Average Precision (MAP) is a composite measure that generalizes over different queries [120] which considers the ranking position of relevant and non-relevant documents. MAP does take the order into account. Firstly, precision is computed at each point when a new relevant document is retrieved, with a value of zero if the retrieved document is non-relevant. Secondly, the average over these precision values for a query is taken. Lastly, the mean over all queries is calculated. MAP definition is expressed in equation 2.7 where $d_1, ..., d_{m_j}$ is the set of relevant documents of a query $q_j \in Q$ and R_{jk} is the set of ranked retrieval results from the top result until document d_k is retrieved. MAP has shown good discrimination and stability.

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{2.7}$$

Normalized Discounted Cumulative Gain (NDCG) is popular with machine learning approaches to ranking [74]. It is designed for cases of multiple ordered categories, not only binary judgments, and has an explicit position discount factor [78, 74]. DCG can be computed with a cut-off at the top k retrieved documents (likewise precision) and the normalization by its maximum value gives the Normalized DCG metric (NDCG). For a set of queries Q, with $R(j,d)$ as the relevance judgments given to document d for query j, NDCG is shown in equation 2.8. It has been argued that NDCG is an stable metric for the evaluation of document retrieval.

$$\text{NDCG}(Q,k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_k j \sum_{k=1}^{k} \frac{2^{R(j,m)} - 1}{log_2(1+m)} \tag{2.8}$$

Mean Reciprocal Ranking (MRR). Given a query $q \in Q$ and r_q the position of the first relevant retrieved document for query q, Reciprocal ranking is denoted as $RR_q = \frac{1}{r_q}$ [120]. MRR is the average of the reciprocal ranking over all Q queries (equation 2.9). Observe that MRR is not influenced by documents in lower positions than the first relevant document, which would be the *correct* answer in a QA setting.

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} RR_q \tag{2.9}$$

Accuracy is the measure of choice in classification tasks [78]. IR can be seen as binary classification of relevant and non-relevant documents. Accuracy is defined as the ratio between all the documents that have been correctly classified as relevant and non-relevant over all the classified documents (equation 2.10). The contingency table 2.1 is presented to better understand accuracy and its relationship with relevance in IR. Note that accuracy sees relevant and non-relevant documents equally regardless their position in the ranked list of documents, when we already know that to satisfy the user's information need it is more important to focus on the position of the relevant documents, as MAP or NDCG do. As a result, accuracy is not a good measure for relevance because most documents are non-relevant w.r.t. a query; hence, it would be easier for

an IR system to classify every document as non-relevant to achieve a higher accuracy. However, the average accuracy of picking the right answer over a set of questions is a common metric in QA tasks [120].

	Relevant	Non-relevant
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

Table 2.1: Contingency table

$$\text{Accuracy} = \frac{tp+tn}{tp+fp+fn+tn} \tag{2.10}$$

The values of the five described metrics range between [0,1]. Other metrics not discussed in this thesis, such as Recall or the 11-point interpolated average precision, are described in the works of Manning et al. [78] and Teufel [120].

Ideally, we would like an IR model that considers any of these metrics directly in the optimization process; however, this is problematic since the metrics are discontinuous or flat when viewed as functions of the models' scores [22], and require sorting by score, which is a non-differentiable operation [137].

2.2 Learning to Rank

In contrast to the previous IR methods that we have seen in section 2.1.2, LTR is a different approach that models the ranking function as a machine learning problem [74]. The LTR framework has four components: (1) *input space*, comprising the objects under investigation, in this case the documents represented by feature vectors extracted by different mechanisms; (2) *output space*, that contains the learning target corresponding to the input space and can be a continuous in the space of real numbers or discrete categories; 3) *hypothesis space*, defining the functions that map the input to the output space; and (4) *loss function*, that measures to what extent the prediction made by the hypothesis is in line with the ground truth label. In the previous section (2.1.3) we explained that we need the ground truth labels in the test set to evaluate an IR model; apart from this, we also require a training set in LTR. Train and test data sets must come from the same data distribution and contain ground truth labels to be used with an LTR approach.

LTR methods are ML approaches that learn the optimal way of combining extracted *features* from query-document pairs by the means of *discriminative* training [74]. All the documents are represented by feature vectors encoding the relevance of the documents w.r.t. the query. This is, for a given pair of query q and document d, the feature vector $x = \Phi(d,q)$, where Φ is a feature extraction function. Common feature extractors are retrieval models such as BM25, where the resulting feature vector is composed by the BM25 scores [74]. The advantage of LTR is that it is able to combine these and many more features in one model, such that any new advance can be readily incorporated as another component in the feature vector [74].

Because LTR algorithms tend to be expensive to run over the whole document collection [124] (as well as deep neural IR models [144]), LTR is usually employed as a second ranking step, or a *re-ranking* of documents (see figure 2.3). Often, re-ranking is done over the top $N << |D|$ documents retrieved in the first ranking step, where |D| is the size of the document collection, . The re-ranking step returns the top U ranked documents as the final output of the IR system.

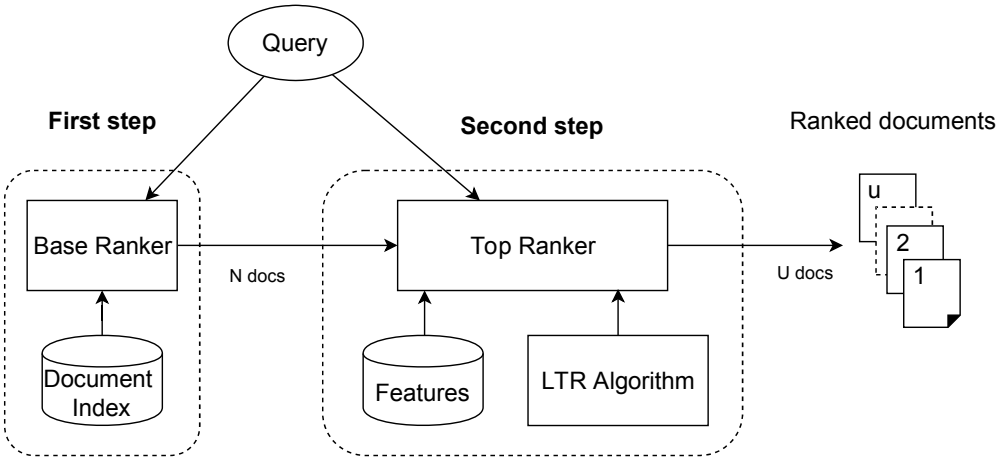


Figure 2.3: Generic LTR re-ranking approach, adapted from [26]

Discriminative training is a supervised learning process where training data is needed [74]. Training data is generated in the same way as test data. It has n queries and a feature vector for each query-document pair with the respective set of relevance labels. The next step is to apply a learning algorithm that combines the features in such a way that it is able to predict the training ground truth labels according to the loss function [74]. To test the learned model, an unseen query is fed to the system, the model then sorts the documents according to their relevance w.r.t. to the incoming query and returns a ranked list of documents.

2.2.1 Unified model formulation

Before start describing specific LTR models, we introduce the unified model formulation[5] proposed in [45] that will serve to understand the commonalities between LTR models, and generalizes to tasks such as ad-hoc retrieval and QA [45]. Moreover, the same formulation will be used to study neuralIR in section 2.3.

Let Q be a query set, representing either queries or natural language questions, and D the generalized document set, which could be the set of documents or answers. Suppose that $Y = \{1, 2, ..., l\}$ is the label set, with each label representing grades. There is a total order among the grades $l \succ l-1 \succ ... \succ 1$, where $\succ$ denotes the order relationship. Let $q_i \in Q$ be the i-th query, $D_i = \{d_{i,1}, d_{i,2}, ..., d_{i,n_i}\} \in D$ be the set of documents associated with query q_i, and $\mathbf{y}_i = \{y_{i,1}, y_{i,2}, ..., y_{i,n_i},\}$ be the set of labels associated with query q_i, where n_i

[5] We slightly modify the model formulation to fit the notation used in this thesis

represents the size of D_i and y_i and $\mathbf{y}_i$ and $y_{i,j}$ denotes the relevance degree of $d_{i,j}$ with respect to q_i. Let F be the function class and $f(q_i, d_{i,j}) \in F$ be a ranking function which assigns a relevance score to a query-document pair. Let $L(f; q_i, d_{i,j}, \mathbf{y}_{i,j})$ be the loss function defined on prediction of f over the query-document pair and their corresponding label. Thus, a generalized LTR problem is to find the optimal ranking function $f*$ by minimizing the loss function over some labeled dataset as expressed in equation 2.11.

$$f* = \arg\min \sum_i \sum_j L(f; q_i, d_{i,j}, y_{i,j}) \tag{2.11}$$

The ranking function f can also be abstracted as equation 2.12

$$f(q,d) = g(\psi(q), \phi(d), \eta(q,d)) \tag{2.12}$$

q and d are two input texts, ψ, ϕ are representation functions which extract features from q and d respectively, η is the interaction function which extracts features from the (q,d) pair, and g is the evaluation function which calculates the relevance score based on the feature representations.

Observe that In LTR approaches, functions ψ, ϕ, and η are commonly manually defined feature functions. The evaluation function g can be any ML model, e.g. logistic regression or support vector machines, to be trained with the data. Moreover, the inputs q and d are usually raw texts. We will see later that all functions ψ, ϕ, η, and g are encoded in network structures such that all of them are learned from the training data, and the inputs could be either raw text or word embeddings [45].

2.2.2 LTR approaches

There are three types of LTR approaches that follow the ML scheme and may vary in terms of their inputs, outputs, learning models, and loss function. These approaches are: pointwise, pairwise, and listwise [74].

Pointwise approaches address the ranking problem as a standard classification task or a regression task [74]. These methods expect that each query-document pair has either a numerical or ordinal associated rank score, or a relevance label representing one, two, or more labels. In the case of a numerical score, the goal is to find an ordinary linear model able to predict such score correctly; in the case of the relevance label, this can be solved with a simple classifier, such as logistic regression [9]. The pointwise approach does not consider the position of a document in a ranked list, nor makes use of the fact that some documents are associated with the same query [74]. The general loss function of the pointwise approach is defined in equation 2.13.

$$L(f; Q, D, Y) = \sum_i \sum_j L(y_{i,j}, f(q_i, d_{i,j})) \tag{2.13}$$

Pairwise approaches aim to learn a preference between pairs of documents instead of an absolute rank, as opposed to pointwise approaches [9].

Thus, the pair of documents, represented by their features vectors, receive a binary preference label {+1, -1}. The positive value is assigned to the document with the highest ranking compared to the other document. This results in a binary classification problem, which aims to predict those labels, by minimizing the number of incorrectly ordered pairs. A ranked list of documents is inferred from the predicted pairwise preference labels [9]. Considering that most metrics are position-based, the pairwise approach is not able to capture this type of information in the loss function [74]. Some pairwise model examples are FRank [121], SVMRank [57], and RankNet [**?**]. equation 2.14 represents the general loss function of the pairwise approach, where document $d_{i,j}$ is preferred over $d_{i,k}$ for query q_i (i.e. $y_{i,j} \succ y_{i,k}$).

$$L(f;Q,D,Y) = \sum_{i} \sum_{(j,k),y_i,j \succ y_i,k} L(f(q_i,d_{i,j}) - f(q_i,d_{i,k})) \tag{2.14}$$

Listwise approaches operate on the entire list of documents associated with a query. In contrast to the pointwise or the pairwise approaches, in the listwise approach a direct loss function is minimized between the ground truth and predicted ranked lists. This is possible due to the loss function of the listwise approach is able to accommodate an explicit evaluation metric, e.g. NDCG, specified by the user. These methods outperform the other two categories in IR tasks [23]. Examples of listwise approaches are LambdaRank [23], ListNet [25], and AdaRank [139]. Most listwise approaches follow the loss function in equation 2.15, where D_i is the set of candidate documents for query q_i.

$$L(f;Q,D,Y) = \sum_{i} L(\{y_{i,j}, f(q_i,d_{i,j}) | d_{i,j \in D_i}\}) \tag{2.15}$$

2.2.3 Feature extraction

One of the advantages of the LTR approach over traditional IR methods is the ability to combine the outcomes of multiples rankers into a single model with higher effectiveness than its individual constituents [74]. This is possible when features are extracted from each ranker (e.g. BM25 scores) and concatenated in feature vectors that are fed to the LTR model. Features are not limited to be the outputs of other rankers, but can also be other document representations that encode the relevance of the document with respect to the query, such as the tf and idf values [74]. In most scenarios it is not feasible to extract features for all the documents in the corpora, but only for a reduced number of documents that have been highly ranked previously [74]. Consider figure 2.3, the first ranking step may be performed via BM25 to retrieve the top N documents, and it is only for these N retrieved documents that features are extracted.

A comprehensive list of extracted features are presented in [97] and are roughly categorized as: low-level content (tf, idf, document length or their combinations), high-level content (outputs of BM25, LMIR, and LMIR variants), hyperlink (PageRank [89], HITS [61], and their variants), and hybrid features

(content plus hyperlink information). Hyperlink, HITS, and their variants are web-specific features.

2.2.4 LambdaMART: a state-of-the-art LTR model

Many LTR models have been proposed in the IR literature [6]. Among them, LambdaMART is a pairwise LTR model built on top of precedent IR and ML research ideas into a model with high effectiveness that has been able to outperform other LTR approaches and traditional IR models in general [21, 45]. LambdaMART has been considered a SOTA model before the application of deep neural networks in IR [137]. Consequently, LambdaMART is the model we choose as our candidate for a strong IR baseline model in this thesis. We give an overview of LambdaMART and its main characteristics as follows.

LambdaMART [137] takes advantage of two different models: MART [40] and LambdaRank [23]. MART (Multiple Additive Regression Trees) is a gradient boosted tree algorithm that sees the cost C as a function of the model output (i.e. function F), this is $C = C_0 + \frac{\partial C}{\partial F}\delta F$. As in ordinary gradient descent, the model reduces the cost by choosing $\delta F \propto -\frac{\partial C}{\partial F}$ for a suitable step size. Given that the gradient $\frac{\partial C}{\partial F}$ cannot be evaluated everywhere, but only at the training data points, the trees allow to estimate a smooth regression to the gradients at any point. Each tree in MART can be practically considered as a small step δF in function space, where the step size turns into the weight associated to that tree. Trees in MART are built as standard regression trees and the best split is calculated with the least square error[6] [137].

LambdaMART directly optimizes the desired IR evaluation metric, in this case NDCG [137], by employing the idea of λ-gradients, introduced by the LambdaRank model [23]. NDCG cost function is either non-continuous or flat everywhere. To overcome this issue, LambdaRank uses the so-called λ-gradients, which are an approximation to the cost gradients. For a ranked set of documents associated with a given query during model training, a particular document is assigned a scalar λ-gradient value calculated with contributions of all the pairs of documents in which this document is a member, and for which the second member of the pair has been generated for the same query but has a different label. Consequently, the λ-gradient associated with a particular document depends both on its own position in the sorted list as well as the other documents' positions. This λ-gradient results from the multiplication of two factors: 1) RankNet's pairwise cost function, and 2) the NDCG variation resulting from exchanging the document pair positions, Δ_{NDCG}. RankNet's pairwise cost function only considers information at the pair level, while Δ_{NDCG} adds information about the whole query structure and the IR evaluation measure. Following this formulation, LambdaRank has been able to optimize other IR metrics such as MAP or MRR by simply replacing NDCG with the new target

[6] A coherent explanation of standard regression trees and the least square error is given in [22].

metric [35]. The λ-gradients may be expressed as in equation 2.16

$$\lambda_{ij} \equiv S_{ij} \left| \Delta_{NDCG} \frac{\partial C_{ij}}{\partial (s_i - s_j)} \right| \tag{2.16}$$

Here, s_i and s_j are the ranking scores for a pair of documents for a given query (where $s = F(x)$), C_{ij} is the cross-entropy cost (equation 2.17), and Δ_{NDCG} is the NDCG gained when the two documents have been swapped (after all documents were ordered by their scores), and $S_{ij} \in \{+1, -1\}$ is the positive value if document d_i is preferred over d_j, and the negative if d_j is preferred instead.

$$C_{ij} \equiv C(s_i - s_j) = s_j - s_i + \log(1 + e^{s_i - s_j}) \tag{2.17}$$

We sum up LambdaMART in Algorithm 1 [22], where the **number of trees** T (also named *boosting* operations), the **number of leaves** per tree K, and the **learning rate** (or *shrinkage* coefficient) ρ, are user-defined **parameters**; while the number of training samples m is the size of the training dataset split. The scores $F_0(x)$ can be initialized with a BaseModel(x_i) for model adaptation [137], otherwise the scores are set to zero. During training, T rounds of boosting stages (trees) are performed, with a regression tree built and trained on all documents for all queries at each boosting stage.

Algorithm 1 LambdaMART algorithm(taken from [22])

set number of trees T, number of training samples m, number of leaves per tree K, learning rate ρ.
for $i = 0$ to m **do**
    $F_0(x_i)$ = BaseModel(x_i) //If BaseModel is empty, set $F_0(x_i) = 0$
end for
for $j = 1$ to T **do**
    for $i = 0$ to m **do**
        $y_i = \lambda$ //Calculate λ-gradient for sample i
        $w_i = \frac{\partial y_i}{\partial F_{j-1}(x_i)}$ //Calculate derivative of λ-gradient for sample i
    end for
    $\{R_{kj}\}_{k=1}^{K}$ //Create K-terminal node tree on $\{x_i, y_i\}_{i=1}^{m}$
    $\gamma_{kj} = \frac{\sum_{x_i \in R_{kj}} y_i}{\sum_{x_i \in R_{kj}} w_i}$ //Assign leaf values based on Newton step
    $F_j(x_i) = F_{j-1}(x_i) + \rho \sum_k \gamma_{kj} 1(x_i \in R_{kj})$ //Take step with learning rate ρ
end for

Limitations. Based on the pairwise RankNet cost function, LambdaMART cannot encode information about the position of all the documents in the ranked list, but only the pairwise relative preference of one document over another, in contrast to listwise approaches (see section 2.2.2. Additionally, working with the ML framework, LambdaMART requires the crafting of the IR features as a preprocessing step.

2.3 Neural IR

In the last years there have been dramatic improvements in several fields such as computer vision, speech recognition, and machine translation that have been possible thanks to the recent advances in deep neural networks plus the availability of large datasets [83]. The IR community has also started applying deep neural networks to move the state of the art forward and possibly achieve performance breakthroughs comparable to the other fields. This has led to an increase in the the number of studies related to deep learning in IR (see figure 2.4). In this section we examine the main characteristics of deep neural IR and the common architectures we can find in recent research. For a comprehensive introduction to neural IR, we recommend the works of [83], and [45].

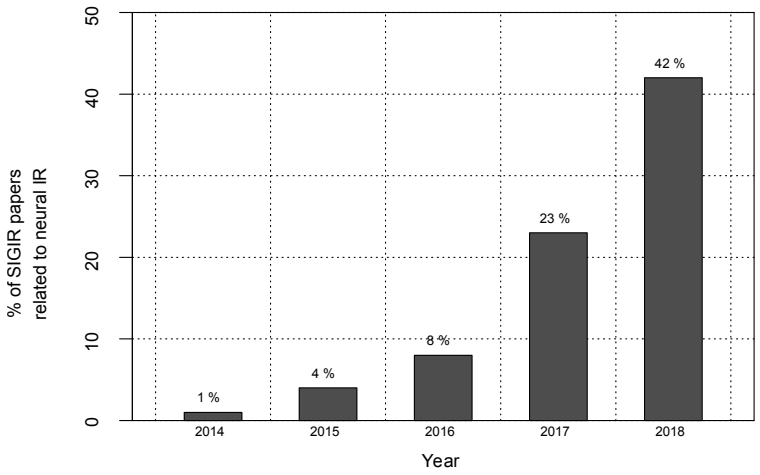


Figure 2.4: Percentage of neural IR papers at the ACM SIGIR conference, determined by manual inspection of the papers, copied from [83].

2.3.1 Deep neural IR

According to [83], neural IR is defined as the application of *shallow* or *deep* neural networks to IR tasks. This can be done to learn query or document representations, match query and document representations to estimate the relevance of the document w.r.t. the query, or a combination of both. Shallow neural IR approaches usually depend on hand-crafted features to achieve good effectiveness. Such is the case of RankNet [20], a LTR pairwise approach, in which a shallow neural network is employed to build the ranking function. On the other hand, deep neural IR approaches have the capacity to learn good features directly from the raw query and document texts besides the relevance estimation function [45]. Following the unified model formulation we introduced in section 2.2.1, this means that deep neural IR approaches can learn the functions $\psi(q)$ (query representation), $\phi(d)$ (document representation), and $\eta(q,d)$ (query-document pair feature extraction function), instead of requiring them to be manually defined [45]. In deep neural IR approaches, we could use raw

query and document texts or word embeddings[7] as a basic input layer, before the application of ψ, ϕ, and η [45].

2.3.2 Model architectures

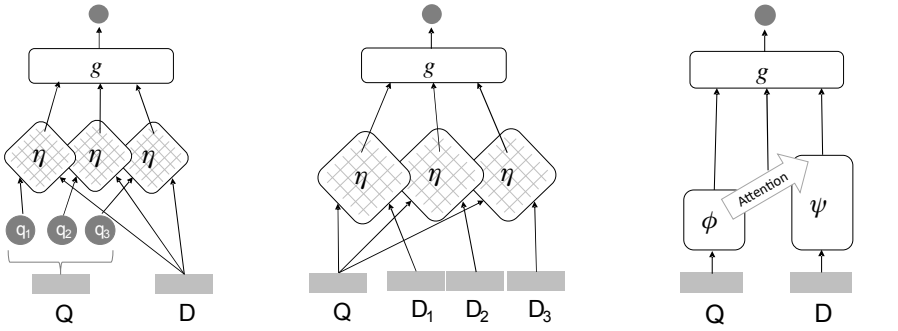
Deep neural IR architectures can be split in different categories [83, 45]. Here, we briefly describe the architecture types distinguished by Guo et al. [45], which are divided in three groups: symmetric vs asymmetric, representation-focused vs interaction-focused, and single-granularity vs multi-granularity architectures. Notation is in accordance with the unified model formulation of section 2.2.1, where the relevance is computed by a function g over the representation functions of the query ($\phi(q)$), the document ($\psi(d)$), or the interaction representation between them ($\eta(q, d)$).

Symmetric vs asymmetric architectures

Symmetric architectures assume that the query and document are homogeneous (e.g. QA with natural language short texts); changing their positions in the input layer does not affect the final output. Symmetric architectures can be further split in: *siamese networks* and *symmetric interaction networks*. *Siamese networks* encode the query and document representations through the same function (i.e. $\psi = \phi$), for example with two identical multi-layer perceptrons (MLP) (as in the Deep Structured Semantic Model, DSSM [52]) or two identical long-short term memory networks (as in LSTM-RN [90]). *Symmetric interaction networks* make use of a symmetric interaction function η for input representation. For instance, Arc-II model [51] computes the similarity between every n-gram pair from query and document.

Asymmetric architectures presume that query and document are heterogeneous (e.g. short query and long document in ad-hoc retrieval); hence, they are processed with asymmetric structures. If the query and the document exchange positions in the input layer, the final output differs completely. Asymmetric architectures with the heterogeneity between the query and the document by the means of three strategies: (1) *query split*, which considers the query made of keywords to be matched against the document (e.g. Deep Relevance Ranking Model -DRRM- [44]); (2) *document split*, where the documents are divided into passages that might be relevant to the query instead of the whole document, for instance HiNT model [39]; and, (3) *joint split*, that combines the two previous assumptions, with DeepRank [91] and PACRR [53] as representative models. For QA, another popular strategy is the *one-way attention mechanism*, which highlights candidate answer words based on the query representation to enhance the answer representation (e.g. IARNN [135]). Figure 2.5 depicts the asymmetric architecture strategies.

[7] Word embeddings are word vector representations produced by models like Word2Vec [81].



(a) Query split (b) Document split (c) One-way attention

Figure 2.5: Neural asymmetric architecture strategies, adapted from [45].

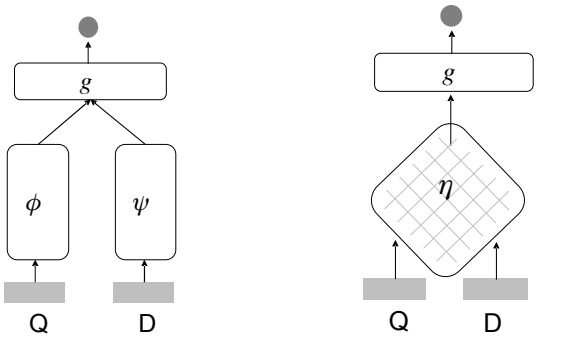
Representation-focused vs interaction-focused architectures

Depending if the representations are learned independently, i.e. through functions ϕ and ψ, or jointly, i.e. through an interaction function η, we have representation-focused or interaction-focused architectures (see figure 2.6):

Representation-focused architectures assume that relevance depend on the semantics of the input texts [45]. These architectures model complex representation functions ϕ and ψ with deep neural networks, but no interaction function η. The high-level representations obtained are matched with a less complicated function (e.g. cosine similarity) to assess the relevance score. Representation-based architectures include: (1) fully-connected networks, such as DSSM; (2) convolutional networks, where the high-level representations for q and d are produced by applying 1D convolutional layers and max pooling layers to the input texts, as in the ARC-I model [50]; and, (3) recurrent networks, with e.g. one-directional [90] or bi-directional [134] LSTMs, in which the top-k strong signals from both high-level representations are matched via a MLP to determine the relevance score. Representation-focused architectures are more appropriate for tasks with short input texts.

Interaction-focused architectures adopt the idea that it is more effective to compute relevance out of the relationships between the input texts rather than their individual representations, i.e. these models define a function $\eta(q, d)$ instead of the representation functions ϕ and ψ. The matching function g becomes more complex and thus is modeled with deep neural networks too [45] to calculate the relevance score. Two categories of interaction functions are distinguished: (1) *non-parametric interaction functions*, which measure the distance among inputs without parameters to learn, some are computed between each pair of input vectors (e.g. cosine similarity [141]) and others over an individual word vector and a group of word vectors (e.g. matching histogram mapping of DRMM [44]); and (2) *parametric interaction functions*, that learn the similarity function from the data when there is enough data for training (e.g. Arc-II learns the interaction between two phrases through a 1D convolu-

tional layer [51]). Interaction-focused architectures suit most IR tasks, as they fit better specific matching patterns (e.g. exact word matching) or adapt to heterogeneous inputs [45].



(a) Representation-focused (b) Interaction-focused

Figure 2.6: Representation-focused and interaction-focused architectures, adapted from [45].

Single-granularity vs multi-granularity architectures

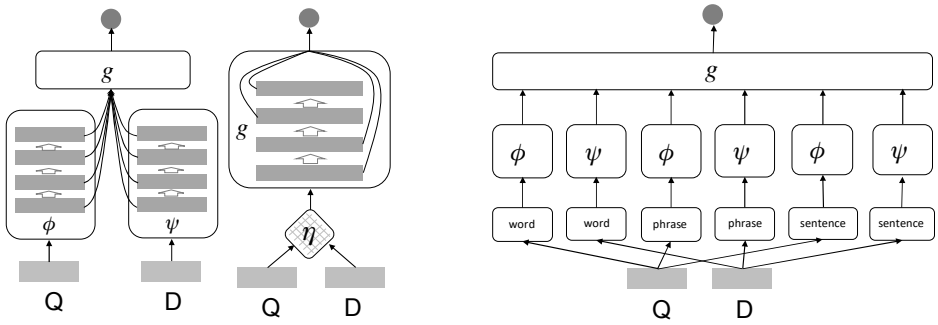
Single-granularity architectures assume that relevance estimation requires only the high-level representations from the functions ϕ, ψ, and η; thus, the input texts are only streams of words or word embeddings from which no other type of language structure is extracted. This is a basic idea that several models adopted, e.g. DSSM, DRMM, ARC-I, among others.

Multi-granularity architectures rely on multiple levels of abstraction or diverse language units of the inputs to estimate relevance. There are two types of multi-granularity: vertical and horizontal. *Vertical multi-granularity* exploits the hierarchical structure of deep network to extract features at multiple levels of abstraction, for example MACM model [88] aggregates the layer scores of its convolutional neural network with MLP to compute the final relevance score. *Horizontal multi-granularity* considers that relevance is better estimated when the inherent structures of language present in phrases or sentences are taken into account instead of isolated words. The inputs for these models are then extended to n-grams, phrases or sentences, where single-granularity is applied individually and the results are aggregated to compute the final relevance score.

The two types of multi-granularity architectures are shown in figure 2.7. They are expected to perform better on ad-hoc and QA tasks, but are more complex models at the same time [45].

2.4 Weak baselines in deep neural IR

IR heavily relies on empirical methods where it is common practice to compare the effectiveness of new models versus the bar set by previous ones, also called *baselines*, for a given combination of IR task, document collection, and set



(a) Vertical multi-granularity (b) Horizontal multi-granularity

Figure 2.7: Multi-granularity architectures, adapted from [45].

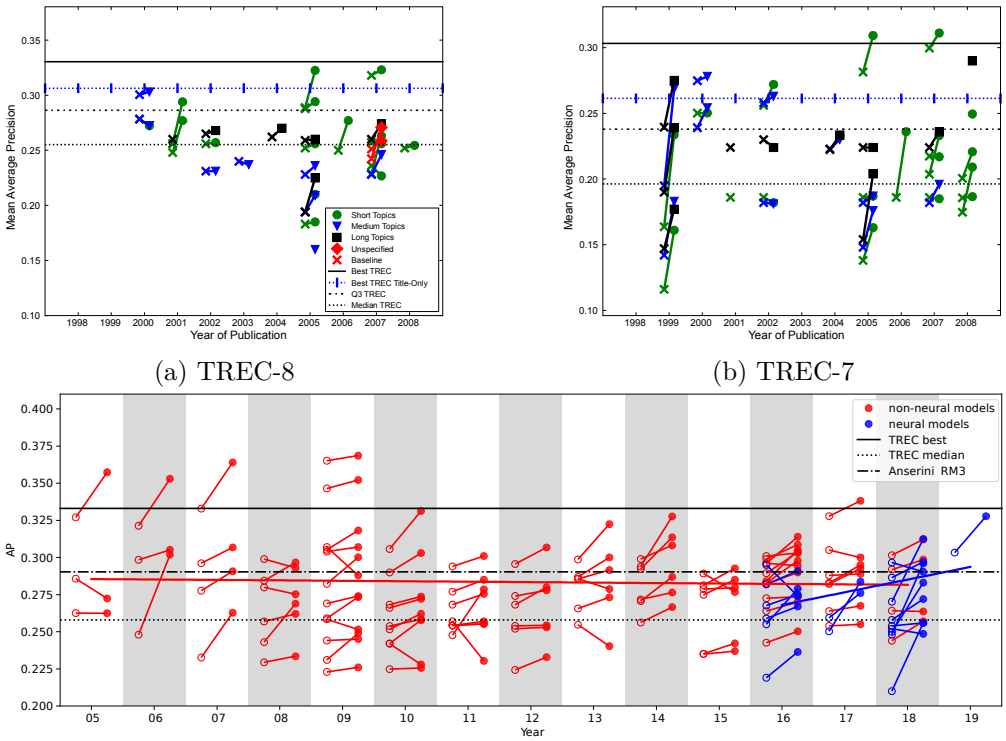
of evaluation metrics (as explained in 2.1.3). In section 1.1 we have already mentioned the risks that weak baselines in IR pose to current deep neural IR research such as progress stagnation or reproducibility issues. Nevertheless, the problem of weak baselines in IR is not new [11]. Throughout the years, researchers coincided on the need of carrying out meta-analyses [11, 60, 143], proposed [11, 60, 71, 143] and encouraged [41] the use of *strong* baselines, as well as relying on IR toolkits and standard document collections that facilitate further comparisons. While surveying the literature we not only noticed that the problem of weak baselines is indeed prevalent in ad-hoc retrieval, but also that, as a community, we lack the knowledge about the state of affairs with weak baselines regarding other IR tasks. Next we give an overview of the issues of weak baselines and the approaches that have been followed to address them.

2.4.1 Issues

The main issues we identify in the IR literature concerning weak baselines are: ad-hoc retrieval progress stagnation, weak baseline selection bias, and reproducibility.

Ad-hoc progress retrieval stagnation. A decade ago, Armstrong et al. [11] made a worrying observation about the state of progress in ad-hoc retrieval. After scrutinizing all the ad-hoc retrieval papers present in the premier IR conferences SIGIR and CIKM from 1998 through 2008, they reached to the conclusion that there is no upward trend in effectiveness in ad-hoc retrieval despite many studies that claim (statistical significant) improvements. This lack of progress is attributed to the use of under-performing baselines that do not contain the latest discoveries in the field, what they called *weak baselines*. Armstrong et al. [11] warned the IR community to avoid their use. Ten years later, Lin [71] hypothesized weak baselines should be non-existent or very difficult to find in current IR research. However he showed two easy-to-find papers containing cases of weak baselines in recent studies. Following this trace, Yang et al. [143] performed the same analysis as [11], but in an updated period, from 2005 through 2019, and confirmed that weak baselines are still present in IR

and no upward trend is evidenced. Figure 2.8 shows the MAP results found by Armstrong et al. [11] on TREC-8/7 and Yang et al. [143] in TREC Robust04 collections (Robust04 for short) . It clearly shows how baselines and the accompanying new models are positioned below the best known runs within the analyzed periods. Examples of weak baselines include BM25 or tf-idf models [71, 143]. These models are decades old, do not include new discoveries in the field, and usually under-perform when compared to other IR approaches such as LTR [60].



(a) TREC-8

(b) TREC-7

(c) Robust04

Figure 2.8: Published MAP scores for ad-hoc retrieval on TREC collections from 1998 through 2019.Baselines and new proposed models are connected by solid lines. (a) and (b) are copied from [11] and (c) from [143].

Weak baseline selection bias. Armstrong et al. [11] noticed that many papers lacked enough comparison and reference to the best runs for a determined dataset. They suspected that weak baselines might be purposely being selected and compared with new techniques because it is easier to beat them, show statistical significant improvements over them, and readily have favorable results that are more likely to be published. A similar situation is observed more recently by Sculley et al. [108] in other empirical fields such as ML and NLP. This selection bias towards weak baselines is one of the risks Lin [71] warns the IR community to be aware of in current deep neural IR research, where the hype that surrounds deep learning could lead to pick non-competitive baselines. This

unveils a research culture that rewards *wins* over the pursue scientific knowledge, which lets us understand not only the strengths but also the weaknesses of the proposed models [108].

Reproducibility. Only knowing what are the best performing baselines and having the intention to use them might not be enough. Researches also face difficulties at the moment of reproducing someone else's models to use them as baselines. Some would re-implement a paper's model from scratch, which would be error-prone due to bugs in the code or missing details in the original paper (e.g. preprocessing steps, parameter settings, or software library versions) [96]. This is the less attractive alternative for reproduction because it consumes time that might be better spent on developing new models [73]. Another alternative is that researchers also share working implementations of their models. However, authors may be reluctant to share their software because of the time needed to polish their code, give support, or due to intellectual property constraints [117]. As a consequence, the best baseline for a given IR task and document collection might not be available and researches would have not other option than to resort to a less effective baseline. We suspect this is the case in [71] and [143]. Both studies identified two of the most effective IR models in terms of MAP score for ad-hoc retrieval on the Robust04 dataset: entity query feature expansion [33] and reciprocal rank fusion [31], but Lin [71] and Yang et al. [143] ended up employing a less effective model (BM25+RM3), available in the Anserini IR toolkit. We speculate they made this decision because neither [33] nor [31] are accompanied with a public implementation of their models.

2.4.2 Addressing approaches

There have been several approaches in IR literature to address the issues with weak baselines described before. These include: meta-analyses [11, 60, 143], proposition of strong baselines [71, 143], and reproducibility guidelines mostly related with the implementation of strong baselines [11, 71, 73, 60, 143].

Meta-analyses. The weak baseline analyses performed by Armstrong et al. [11], Kharazmi et al. [60], and Yang et al. [143] have examined more than 200 ad-hoc retrieval papers spanning over more than two decades, covering different types of IR models, ranging from traditional, LTR, and deep neural IR methods. They form the largest evidence about the status of weak baselines in ad-hoc retrieval. Here the focus is on collecting evidence about the scores obtained by baselines, new proposed models, and best runs. Armstrong et al. [11] are the first to conduct this type of analysis and advocate for more rigorous standards in empirical IR research to prevent weak baselines. In particular, Armstrong et al. [11] introduced the hypothesis of *additive improvements*. According to this hypothesis, the improvements introduced by a new model C, which are (statistically) demonstrated over a weak baseline A, would be considered meaningful only if they also happen to be improvements over a strong baseline B.

Later, [60] and [143] continued with similar meta-analyses and testing the additive hypothesis with models from different points in time. [60] conclude that measuring statistical significant improvements over weak baselines is not a good predictor of similar improvements over strong baselines. Yang et al. [143] also examine several deep neural ranking models and find that only DRMM was able to improve over a strong baseline.

Accessible baseline registers. Armstrong et al. [10] built an online system named EvaluatIR [10]. Its purpose was to register the best effectiveness results achieved in IR tasks across standard collections as well as the evaluation metrics used, such that researchers, reviewers, and readers could easily find the models and compare them. Unfortunately such repository is not active anymore, apparently it did not gained traction among IR practitioners [71]. Moreover, Lin [71] acknowledges that the NLP community, for instance, actually keeps a record of their best performing models with *leaderboards* which is useful to keep track of the state-of-the-art in the NLP community.

Strong baselines are generally described as competitive models that contain the latest discoveries in IR [11] and are adequately tuned [71] (more details in section 2.5); thus, strong baselines should be preferred to weak baselines in order to show the improvements of newly proposed models. For example, Lin [71] and [143] recently used the combination of BM25+RM3 as a strong model, due to the maturity of RM3, and showed that the combined model outperforms not only BM25 but also some current deep neural IR models such as DSSM. Table 2.2 summarizes the results obtained by Lin [71] and Yang et al. [143] for the ad-hoc retrieval task on the Robust04 dataset. In contrast, Kharazmi et al. [60] determine that LTR approaches (with a pool of features) are stronger baselines for ad-hoc retrieval than other techniques, including BM25, LM, QE, among others. Kharazmi et al. [60] are also of the opinion that new techniques should not be compared with anything less than the SOTA models; however, this might not be feasible due to reproducibility issues and constraints. Of note, a new study from the NAACL[8] [41] showed that reviewers actually regard the lack of strong baselines as a weakness in recently submitted papers, which signals a positive change in the tendency of publishing *wins*.

Reproducible baselines. For reproducible research in IR, most researchers recommend to use an open-source IR toolkit and standard collections [11, 71]. IR toolkits such as Indri, Anserini, or RankLib[9] provide ready-to-use implementations of several known IR models that, in most cases, are well-debugged thanks to active user communities. However, a model varies across implementations due to differences in tokenization, stemming, stopwords, and so on [71]. Of note, the open-source reproducibility challenge [73] aims to become an actual IR baseline repository with the goal of reproducing the submitted runs for a given collection; thus, going goes the *leaderboard* proposed in [10], which.

[8] https://naacl2019.org/
[9] https://sourceforge.net/p/lemur/wiki/RankLib/

Condition	MAP	P@20	NDCG@20
2-fold results from Paper 1			
Paper 1 QL baseline	0.2499	0.3556	-
Paper 1 best proposed neural variant	0.2971	0.3948	-
Lin's QL	0.2496	0.3543	-
Lin's BM25+RM3 (joint)	0.2973	0.3871	-
Yang's BM25+RM3	0.2987	-	0.4398
Yang's BM25+RM3+DRMM	0.3126	-	0.4646
5-fold results from Paper 2			
Paper 2 BM25 baseline	0.2380	0.3540	-
Paper 2 best proposed neural variant	0.2720	0.3860	-
Lin's BM25	0.2528	0.3598	-
Lin's BM25+RM3 (independent)	0.2991	0.3901	-
Yang's BM25+RM3	0.3033	-	0.4514
Yang's BM25+RM3+DRMM	0.3152	-	0.4718
Best run at TREC 2004 Robust (pircRB04t3) [127]	0.3330	-	-
Best reported result (Reciprocal rank fusion [31])	**0.3686**	-	-

Table 2.2: Results of the experiments with weak baselines, deep neural IR models, and strong baselines, presented in [71] and [143].

Standard document collections are also important for reproducibility, otherwise even if the models are the same the effectiveness results would differ. Data splits (e.g. train, test) and cross-validation folds should also be the same to have comparable results [71, 143]. As an illustration of how sensitive is the matter of reproducibility, we refer to the MAP scores for the BM25+RM3 model by Lin [71] and Yang et al. [143] in table 2.2. Both research teams carefully aim to reproduce the different models' scores using the same settings, including: preprocessing steps (tokenization, stemming, etc.), IR toolkit, cross-validation folds, among others. In spite of all these precautions, there are still noticeable variations. According to Yang et al. [143], this discrepancy is due to improvements in the code[10]. Other approaches for reproducible research are based on software containers like Docker[11] that enable portability and better software versioning control, among other advantages [17].

2.4.3 Current cases of weak baselines in deep neural IR research

Gao et al. [41] and Lin [71] point out the existence of weak baselines cases in current deep neural IR research after the admonishments of Armstrong et al. [11]. Based on this trace, we take a look at the papers published at the last

[10] We assume Yang et al. [143] refer to improvements to Anserini's source code.

[11] https://www.docker.com/

edition of the Empirical Methods in Natural Language Processing [7]. EMNLP is an influential conference on empirical studies regarding IR, QA, dialog systems, machine translation, and other NLP related topics [2]. Its last edition received 2231 submissions and accepted more than 550 research papers [5]. In this conference there is a mix of traditional and non-traditional IR tasks. Our team found thirteen papers comparing deep neural models to IR baselines such as BM25 and tf-idf. These baselines fit the description of weak baselines we presented in section 1.1. In particular, a paper [80] employs a BM25 baseline on the Robust04 with a low MAP score of 0.272 compared to the best TREC run score of 0.333 [132] (see section 2.4.1). Table 2.3 summarizes our findings.

Paper	**IR Task**	**IR baseline**
[80]	Ad-hoc retrieval	BM25
[68]	Ad-hoc retrieval	BM25
[112]	Question similarity	BM25
[43]	Question similarity	tf-idf
[92]	Ad-hoc retrieval	tf-idf
[28]	Keyphrase generation	tf-idf
[67]	QA	tf-idf
[36]	QA	tf-idf
[109]	QA	tf-idf
[79]	Automatic conversation	tf-idf
[119]	Reading comprehension	tf-idf
[95]	Fact extraction	tf-idf
[66]	Fact checking	tf-idf

Table 2.3: Weak IR baselines compared to recently proposed deep neural IR models found in EMNLP 2018.

2.5 Hyperparameter Optimization

Hyperparameters are the settings of a model that control its behaviour, rendering it less or more effective depending on the chosen hyperparameter values and the target dataset [54]. Different types of models include hyperparameters. For example, we have already seen that BM25 depends on b and $k1$ (section 2.1.2) or LambdaMART on K, ρ, and T (section 2.2.4). Moreover, the hyperparameters placed in ML models, such as the regularization parameter in Support Vector Machines (SVM), help the model to generalize better across datasets and avoid overfitting [138]. We also find hyperparameters in (deep) neural network models, the learning rate or the number of hidden layers, for instance [54]. Choosing the best hyperparameter values for a given model and dataset is not a trivial problem [13]. This problem is known as *hyperparameter optimization* (HPO) or *hyperparameter tuning*, and it has been attempted both manually and automatically in past research [54]. HPO is commonly performed

in a separate process from ML and DL model's weights learning or training, and with a distinct data split (the validation split) to avoid overfitting [54].

With the increasing complexity of ML models and their many hyperparmeters [54], HPO is recommended as a standard practice in empirical research (e.g. ML, NLP), specially when different models are subject of comparison [108]. HPO prevents unfair model comparisons[12], where the parameters of newly-proposed models are highly optimized (e.g. deep neural models), while baselines are used in their vanilla forms [108]. Therefore, we recognize that it is important to perform appropriate hyperparameter-tuning[13] of a model to build a competitive, strong baseline as stated in the preceding section.

HPO is useful but far from trivial. Next we present a brief formulation of the HPO problem and the main challenges it faces. Then, we review three HPO techniques we cover in this thesis, namely: grid search and random search which are two basic but widely used techniques [13], and Bayesian Optimization HyperBand (BOHB) [37], an effective state-of-the-art method that outperforms the former techniques. For more details on the Bayesian Optimization framework, we refer the reader to [113] and for recent advances and techniques on HPO to [54].

2.5.1 Problem statement and challenges

The HPO problem is defined as follows [54]. Let $\mathcal{A}$ denote a (ML) algorithm with J hyperparameters. The domain of the j-th hyperparameter is represented by $\mathcal{X}_j$ and the entire *hyperparameter configuration space* by $\boldsymbol{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2 \times ...\mathcal{X}_J$. A vector of hyperparameters is denoted by $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$, and $\mathcal{A}_{\boldsymbol{x}}$ is an instance of algorithm $\mathcal{A}$ with the specific set of hyperparameters $\boldsymbol{x}$.

A hyperparameter domain can contain real values, integer values, binary values, categories, and also supports conditionality (i.e. meaning that a hyperparameter is relevant only if other(s) hyperparameter(s) take(s) a given value) [54]. In the context of this thesis, we deal with real and integer hyperparameter domain values exclusively.

Then, the performance of a ML algorithm can be modelled as a (loss[14]) function $f : \boldsymbol{\mathcal{X}} \to \mathbb{R}$ of its hyperparameters $\boldsymbol{x}$. Thus, we aim to find the optimal set of hyperparameters $\boldsymbol{x}^*$, according to equation 2.18. It is important to note that we evaluate $f(\boldsymbol{x})$ on a validation (development) data set different from the training data set to avoid overfitting [74], usually with a hold-out or cross-validation protocol [16].

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} f(\boldsymbol{x}) \tag{2.18}$$

In general, we assume that we can access $f(\boldsymbol{x})$ plus some noise ϵ, this is, we observe $y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon$ [37].

[12] Whenever open HPO tools are available.

[13] This contrasts with the practice of employing the default parameter values of a particular model implementation without acknowledging that different datasets require different settings to achieve higher effectiveness [63].

[14] Considering an IR evaluation metric such as MAP, the loss function results from subtracting the MAP score from one $\mathcal{L} = 1 - MAP$.

HPO presents various challenges, including: large and complex hyperparameter configuration spaces, as well as functions that are expensive to evaluate [54]. The hyperparameter configuration space could be large (i.e. millions of configurations to try), high dimensional (multiple hyperparameters), and contain heterogeneous hyperparameter domains (different types of hyperparameters). For instance, the combined model BM25+RM3 proposed in [71] has five hyperparamters: k_1, b, and the weight of original query are real positive values, the number of feedback documents and the number of feedback terms are integer positive values. The ranges and resolution defined for those hyperparameters in [71] account for twenty million different hyperparameter combinations. On the other hand, The HPO process is guided by a loss function which result is obtained after following an evaluation pipeline. This pipeline can grow in complexity depending on the number of steps it comprises and where the hyperparameters to tune are involved (in an IR setup, we expect the hyperparamters to be involved in the retrieval phase, but they could also appear in other steps such as the indexing phase if we consider the use of stopwords as a binary hyperparameter for example). Additionally, the size of the dataset also influences the amount of resources needed to accomplish the function evaluation [54]. Below we introduce two basic HPO techniques: grid search and random search, and describe how they address these challenges.

2.5.2 Grid search

Grid search is the simplest HPO method. The user has to define a set of values for each hyperparameter. Grid search evaluates each hyperparameter combination among those sets. The number of function evaluations grows exponentially with the number of hyperparameters and, substantially, by increasing the discretization resolution of the hyperparameter values [54]. Despite these shortcomings, grid search is easy to implement, its parallelization is trivial, and is reliable in low dimensional spaces [13].

2.5.3 Random search

Random search samples random configurations until a search budget is consumed. Random search enjoys the same benefits of grid search, but in addition, it is able to get better results when some hyperparameters are more important than others [13]. If the budget is limited to a fixed number of function evaluations B, and the number of hyperparameters is N, grid search could afford to evaluate only $B^{1/N}$ different values for each hyperparameter, while random search would explore B different values. Figure 2.9 illustrates a case of nine optimization trials, both with grid search and random search, where random search hits more points of the important parameter than grid search (this being the rule more than the exception, according to Bergstra and Bengio [13]). A further advantage of random search is its flexible resource allocation, i.e. since every search trial is independent from each other, search trials can be run asynchronously, new random trials can be added on the fly, and if one trial fails, it can be abandoned, all of this without affecting the search design [13].

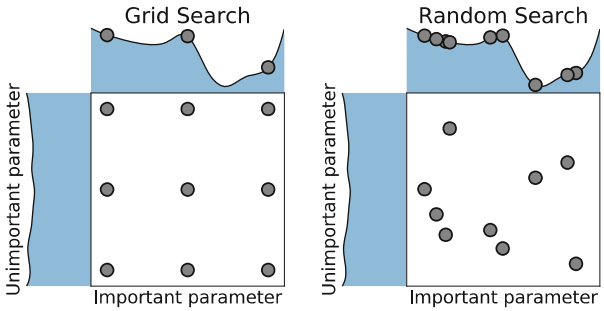


Figure 2.9: Comparison of grid search and random search for minimizing a function with one important and one unimportant parameter., copied from [54].

With enough resources, random search would find a hyperparamter configuration close to the optimum [54]. Moreover, combined with more sophisticated approaches, it serves to initialize the search process [54]. Notwithstanding all the benefits of random search, it usually takes longer than guided searches [54]. In the following section we introduce the Bayesian Optimization HyperBand (BOHB) algorithm, a guided HPO approach.

2.5.4 Bayesian Optimization HyperBand

BOHB is a SOTA guided HPO method that searches for the next hyperparameter configuration based on the already seen configurations and the loss with respect to a target metric that they present [54]. BOHB outperforms grid search, random search, and other methods in problems such as high-dimensional toy functions, SVM, feed-forward neural networks, Bayesian neural networks, deep reinforcement learning, and convolutional neural networks [37]. BOHB is built upon the Bayesian Optimization framework which grants it fast convergence to optimal configurations and the bandit configuration of an algorithm named HyperBand, which gives BOHB strong anytime performance. We first overview the two underlying strategies of BOHB and then highlight the relevant characteristics of the resulting BOHB algorithm.

Bayesian optimization (BO) is a framework for the global optimization of expensive functions [54]. It has been successfully applied to HPO, achieving SOTA results in deep neural network settings [116]. BO is an iterative algorithm based on two components: a surrogate probabilistic model $p(f|D)$ and an acquisition function $a(\boldsymbol{x})$ [54]. The surrogate model represents the objective function f based on the already observed data points $D = \{(x_0, y_0), ..., (x_{i-1}, y_{i-1})\}$. In each iteration BO performs three steps: (1) pick the hyperparameter point $\boldsymbol{x}_{new}$ that maximizes $a(\boldsymbol{x})$, i.e. $\boldsymbol{x}_{new} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} a(\boldsymbol{x})$; (2) evaluate the objective function $\boldsymbol{y}_{new} = f(\boldsymbol{y}_{new}) + \epsilon$; and, (3) append the new data point $)(\boldsymbol{x}_{new}, \boldsymbol{y}_{new})$ to D and refit the model. A common choice for $a(\boldsymbol{x})$ is the expected improvement (EI) [54, 37]. In contrast to most BO methods that use Gaussian processes to model the objective function [54, 113], BOHB relies on the Tree Parzen Estimator (TPE) [14]. TPE uses a kernel density estimator that allows it to support

mixed continuous and discrete configuration spaces, as well as linear scalability with the number of data points [37].

Hyperband (HB) [69] is a HPO bandit method that dynamically assigns resources to a set of random configurations with a technique called successive halving (SH) [55]. Since function evaluation can be expensive, we could approximate the objective function $f(\cdot)$ to a cheap-to-evaluate version $\tilde{f}(\cdot, b)$, where $b \in [b_{min}, b_{max}]$ is a limited *budget*. The quality of the model increases with b, until we reach $b = b_{max}$, where we have that $\tilde{f}(\cdot, b) = f(\cdot)$. The budget can represent the number of (validation) data samples, number of epochs in a neural network algorithm, wallclock time, etc. [37, 54]. Making use of these budgets, HB calls SH successively to find the best configuration among n randomly sampled configurations. The main idea behind SH is that for an initial budget all configurations are evaluated; then, a $1/\eta$ fraction of the top performing configurations are kept, the budget is increased in η times, and the process continues until $b|max$ is reached [37]. Parameter η is set to 3 in BOHB.

Compared to most common BO approaches, HB displays strong anytime performance and scalability to higher-dimensional spaces. HB also beats BO and random search on small to medium budgets. On the downside, it samples configurations randomly with no learning from previously seen configurations [54] which limits its convergence to the global optimum. If large budgets are set, HB practically does not have any advantage against random search.

BOHB algorithm. BOHB determines how many configurations to evaluate and with which budget according to HB, but instead of selecting configurations randomly, the search is guided by the BO component. SH begins when the the desired number of configurations per iteration is reached. Thus, the combined algorithm overcomes the limitations of its individual components. At the end the combined algorithm, BOHB, results in a robust HPO technique that outperforms its individual constituents. Figure 2.10 shows that BOHB starts quickly and converges to the global optimum faster compared to BO, HB, and random search.

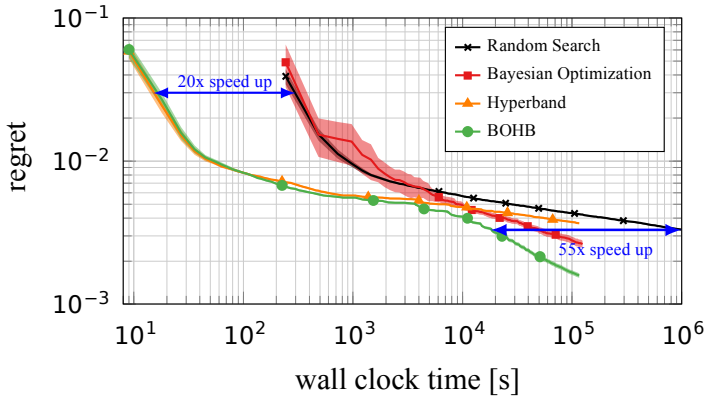


Figure 2.10: Results of optimizing six hyperparameters of a neural network with four different HPO methods., taken from [37].

Chapter 3

Approach

Our research approach consists of three parts: (1) an exploratory analysis of IR literature that let us determine to what extent weak IR baselines are still an issue in recent deep neural IR research, (2) the effectiveness results reproducibility for an ad-hoc and a QA tasks to validate the findings of previous research, and (3) the proposition of a LTR baseline to contrast how the latest advances in the field (before deep neural IR) compare with the recently proposed neural IR models.

3.1 EMNLP 2018: exploratory analysis

The goal of this exploratory analysis is to determine to what extent weak IR baselines are still an issue in current deep neural IR research. We start by looking at EMNLP 2018[1], a very influential venue which in its 2018 edition accepted more than 500 papers and received over 2000 submissions. In contrast to other venues where IR studies can be found (e.g. SIGIR, CIKM, WWW, ICTIR, ECIR, and so on), EMNLP focuses more on empirical work, where effectiveness comparisons among models play a major role. Here we examine the papers where deep neural IR models have been presented and what IR models they have been compared to. We also describe these papers in terms of the types of IR tasks and datasets that have been employed, plus other reproducibility characteristics including dataset and code availability.

We inspected the EMNLP 2018 proceedings[2] and looked for papers that: (a) contain keywords related to the use of IR models as baselines, including *IR baseline, IR system, BM25, Okapi, tf-idf, tfidf* (66 papers), (b) present tables summarizing the effectiveness results of the proposed deep neural IR model and baselines (25 papers), (c) model the main paper task as an IR task (13 papers), and (d) are accompanied by both a publicly available code repository for the proposed deep neural IR models and the respective dataset(s) (8 papers). Table 3.1 shows all the papers we found with an IR task. We observe different kinds of taks, with ad-hoc retrieval and QA tasks being around half of them.

[1] EMNLP 2018 was the venue with the most recent deep neural IR published papers by the time we performed this exploratory analysis

[2] https://www.aclweb.org/anthology/events/emnlp-2018/

In most tasks query- and document-like elements tend to be short (i.e tens of words), with a few exceptions that are long (i.e. hundreds of words, see table 3.2). Interestingly, the most used IR baseline is tf-idf, followed by BM25, two decades-old models regarded as weak baselines [71, 60]. A cross (x) indicates that either the code or dataset were not available because the resource was not provided by the authors or, if a link was given, this was not online (see more details in table 3.3). In table 3.2 we collected word length statistics for query and document elements for the available datasets.

A variety of deep neural IR models and datasets are employed among the analyzed papers. Of note, the main purpose of some papers is not the introduction of an innovative deep neural IR model, but the release of a new task (or *challenge*) [36, 67, 110]. These papers can also come with a new tailored dataset (e.g. [36, 67]) or reuse an already existing one (e.g. [110]). In contrast, only two papers [80, 68] present experiments with standard TREC collections for ad-hoc retrieval. The studies that do introduce a new deep neural IR model (e.g. [95, 28]) experiment with two or more similar datasets (e.g. both have short queries and long documents) to justify results generalization.

Paper	IR task	Query	Document	IR baseline	Deep neural IR model	Code	Dataset	
							Name	Avail?
[80]	Ad-hoc retrieval	Queries (short)	Documents (long)	BM25	POSIT-DRMM+MV	✓	BioASQ TREC Robust04	✓ ✓
[68]	Ad-hoc retrieval	Queries (short)	Documents (long)	BM25	NPRFds-DRMM	✓	TREC 1-3 TREC Robust04	✓ ✓
[112]	Question similarity	Question (short)	Question (short)	BM25	ADA	✓	AskUbuntu SuperUser Apple Android Sprint Quora	✓ ✓ ✓ ✓ ✓ ✓
[43]	Question similarity	Question (short)	Question (short)	tf-idf	STL-MLP	✓	SemEval 2016 and 2017 subtask B MultiNLI Fake News challenge dataset	✓ ✓ ✓
[92]	Ad-hoc retrieval	Query (short)	Web page element (long)	tf-idf	Embeddings	✓	Phrasenode	✓
[28]	Keyphrase generation	Document (long)	Keyword (short)	tf-idf	CorrRNN	x	KP20K SemEval2010 NUS Krapivin	✓ ✓ x ✓
[67]	QA	Question (short)	Answer (short)	tf-idf	(LSTM based)	✓	TVQA	✓
[36]	QA	Question sequence (3 short questions)	A paragraph from a document (short)	tf-idf	DrQA	x	QBLink	✓
[109]	QA	Question (short)	Phrase (short)	tf-idf	LSTM+SA+ELMo	✓	SQuAD 1.1	✓
[79]	Automatic conversation	Comment (short)	Response (short)	tf-idf	Transformer	x	Reddit	x
[119]	Reading comprehension	Question (long) Text (long)	Answers (short) Span (short)	tf-idf	DCU	✓	RACE SearchQA NarrativeQA	✓ ✓ ✓
[95]	Fact extraction	Text (long)	Facts (short)	tf-idf	SALIE	✓	New York Times	x
[66]	Fact checking	Claim (short) Claim (short)	Wikipedia document (long) Evidence (short)	tf-idf	DArank+NER	x	FEVER.AI	x

Table 3.1: EMNLP 2018 papers with IR tasks, deep neural IR models, and IR baselines.

Dataset	# queries	# docs	Avg q length (words)	Median query length (words)	Avg doc length (words)	Median document length (words)	Avg relevant document per topic	Median relevant documents per topic
BioASQ	2251	17741805	9.42 (± 3.39)	9	205.36 (± 84.5)	205	7.73 (± 7.55)	6
TREC Robust04	250	528155	2.16 (± 1.28)	3	541.08 (± 1637.55)	347	18.97 (± 0.18)	19
AskUbuntu	257173	257173	75.93 (± 26.42)	82	75.93 (± 26.42)	82	0.17 (± 1.10)	0
SuperUser	343033	343033	80.49 (± 24.12)	90	80.49 (± 24.12)	90	0.064 (± 26.42)	0
Apple	80466	80466	78.55 (± 24.61)	87	78.55 (± 24.61)	87	0.06 (± 0.43)	0
Android	42970	42970	77.74 (± 24.65)	86	77.74 (± 24.65)	86	0.11 (± 0.86)	0
Sprint	31768	31768	27.06 (± 9.95)	26	27.06 (± 9.95)	26	0.70 (± 0.74)	1
Quora	537211	537211	11.65 (± 6.46)	10	11.65 (± 6.46)	10	0.56 (± 1.82)	0
SemEval 2016 and 2017 subtask B	408	4079	33.99 (± 20.03)	30	47.42 (± 23.25)	48	4.10 (± 2.63)	4
Phrasenode	51663	1835 pages 1079.05 (± 1121.92) elements/page	4.15 (± 2.01)	4	37.92 (± 2040.24)	20	1.46 (± 1.07)	1
Krapivin	2304	13926 (± 1121.92)	8039 (± 2237.91)	7814.5	2.41 (± 13.81)	2.0	6.34 (± 2.77)	6
TVQA	137274	724575	13.45 (± 3.6)	13	4.74 (± 3.46)	4	1	1
SQuAD 1.1	98169	490	10.24 (± 3.6)	10	120.16 (± 51.43)	110	1.08 (± 0.34)	1
RACE	97687 (questions) 492489 (sentences)	27933 (articles) 492489 (sentences)	9.74 (± 3.40) (q) 16.30 (± 11.83) (s)	9 (q) 14 (s)	287.31 (± 95.69) (a) 16.30 (± 11.83) (s)	291 (a) 14 (s)	1 (± 0)	1
NarrativeQA	46765 (questions) 1572 stories	93530 (answers) 1572 (summaries) 27.64 (± 11.18) (sentences/summary)	9.82 (± 3.41)	9	4.71 (± 3.88) (answer) 665.03 (± 223.06) (summary) 23.73 (± 11.54) (sentences)	3 (answer) 682.5 (summary) 22 (sentence)	1.86 (± 11.83)	2
FEVER.AI	185445	5416537 (docs) 4.16 (± 4.76) (sentence/doc)	8.36 (± 3.32)	8	85.60 (± 100.97) 20.57 (± 16.61)	54 18	5 (± 1)	5 (± 1)

Table 3.2: Word length statistics for available datasets.

Paper	Code repository URL	Dataset	
		Name	**URL**
[80]	https://github.com/nlpaueb/deep-relevance-ranking	BioASQ	http://bioasq.org/
		TREC Robust04	https://trec.nist.gov
[68]	https://github.com/ucasir/NPRF	TREC 1-3	https://trec.nist.gov
		TREC Robust04	
[112]	https://github.com/darsh10/qra_code	AskUbuntu	https://drive.google.com/file/d/1blckd0P8KX0FHCA8tBlkJ5hmYhrO5DcX/view?usp=sharing
		SuperUser	
		Apple	
		Android	
		Sprint	
		Quora	
[43]	https://github.com/anavaleriagonzalez/FAQ_rank	SemEval 2016 and 2017 subtask B	http://alt.qcri.org/semeval2016/task1/
		MultiNLI	https://www.nyu.edu/projects/bowman/multinli/
		Fake News challenge	https://github.com/FakeNewsChallenge/fnc-1
[92]	https://github.com/stanfordnlp/phrasenode	Phrasenode	https://nlp.stanford.edu/projects/phrasenode/
[28]	Error 404	KP20K	https://github.com/memray/seq2seq-keyphrase
		SemEval2010	Error 404
		NUS	N/A
		Krapivin	http://disi.unitn.it/~krapivin/keyphrases/data-to-learn-keyphrases/all_docs_abstacts_refined.zip
[67]	https://github.com/jayleicn/TVQA	TVQA	http://tvqa.cs.unc.edu/index.html#download
[36]	N/A	QBLink	https://sites.google.com/view/qanta/projects/qblink
[109]	https://github.com/uwnlp/piqa	SQuAD 1.1	https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/
[79]	N/A	Reddit	N/A
[119]	https://github.com/vanzytay/EMNLP18_DCU	RACE	http://www.cs.cmu.edu/~glai1/data/race/
		SearchQA	https://github.com/nyu-dl/dl4ir-searchqA
		NarrativeQA	https://github.com/deepmind/narrativeqa
[95]	https://github.com/mponza/SalIE	N/A	N/A
[66]	Error 404	FEVER.AI	http://fever.ai/data.html

Table 3.3: Code repository and dataset URLs.

3.2 Reproducibility experiments

Reproducibility allows us to validate the results obtained in past studies; hence, it is an effort that helps us to build trust about the conclusions derived from previous scientific research or to identify the need for collecting more evidence before accepting a given claim [93]. Here we describe the papers we select for reproducibility, and the approaches we follow to replicate and reproduce the effectiveness of the deep neural IR models and IR baselines respectively.

3.2.1 Paper selection criteria for reproducibility

Since reproducing the results for all the papers identified in table 3.1 would take a considerable amount of human effort and computing resources (as stated in section 1.4.2), in this thesis we have limited the number of papers for our reproducibility experiments to [80] and [67]. We have chosen these papers based on the following reasons:

- To cover two representative IR tasks that appear in our paper analysis: ad-hoc retrieval and QA. Given that these two IR tasks are older and more studied than the others, we argue that the selection of baselines should be more rigorous in these papers.
- To cover the two different types of IR baselines in our list: BM25 and tf-idf. These models are the most effective traditional IR baselines used for model comparison in the selected papers.
- Both papers meet the basic criteria for our reproducibility experiments (more details in chapter 4). These criteria are: (a) each paper extensively describes the its novel deep neural IR models (see sections 3.2.2 and 3.2.3), (b) the IR baselines are clearly identified, and (c) the code repositories and datasets are publicly available. If we add the fact that we can find the IR baselines implementations in an IR toolkit, we have the main components to carry out our reproducibility experiments, avoiding common shortcomings such as code debugging during the time-consuming endeavour of implementing any model from scratch [96]
- In particular, [80] presents experiments with the well-known Robust04 dataset which let us establish comparisons with earlier weak IR baseline research [11, 71, 143]. On the other hand, [67] is an interesting case where a new QA task and dataset are released; thus, no previous baselines exist in the literature for this specific combination of task and dataset.

3.2.2 Paper A: deep relevance ranking using enhanced document-query interactions

Task. This paper is developed in the context of the 6th BioASQ biomedical challenge question answering task (BioASQ task 6b [86]). This task is divided in two phases. The first phase consists in retrieving the documents that potentially contain the required answers, while the second phase employs sophisticated NLP

methods to extract snippets of text or sentences with more precise answers [122]. This paper presents a document retrieval approach for the first phase.

Approach. The authors investigate deep neural IR interaction based architectures (recall section 2.3.2), in particular DRMM [44] and PACRR [53] model variations. The authors introduce their most effective deep neural IR: POSIT-DRMM+MV, which stands for POoled SImilariTy DRMM with *multiple views*. The main enhancements over DRMM are: (i) context-sensitive representation enrichment, and (ii) the use of multiple views of terms (i.e. POSIT-DRMM encodings view, pre-trained embeddings view, and one-hot vector representations view). They apply POSIT-DRMM+MV to re-rank the top N documents retrieved by BM25 following the re-ranking scheme explained in section 2.2.

Datasets. The datasets employed by this paper are: TREC Robust04 [1] and BioASQ [122]. Robust04 is a standard TREC collection designed for ad-hoc retrieval. It is composed by 249 topics[3], where 200 topics where collected throughout the past ad-hoc TREC tracks and 50 new topics were added in TREC Robust 2004. Figure 3.1 shows the structure of a sample topic. Robust04 distinguishes the topic (information need) from the actual query (e.g. a query can be the title only or the concatenation of title plus description). Robust04 includes 528,155 documents from TREC disks 4 and 5. This document set is commonly used for TREC tasks and includes material from the Financial Times Limited, the Federal Register, the Foreign Broadcast Information Service, and the Los Angeles Times -excluding the Congressional Record. Each document is structured and tagged using SGML or XML, and uniquely identified with a DOCNO tag. The text in the documents remain as close as possible to the original in that spelling and alike errors have not been removed [132]. Robust04's construction was expensive due to the large human effort required to create the topics and to produce the corresponding document relevance judgements [132]. This dataset was not created with predefined train, validation, and test data splits or cross-validation folds (in contrast to, e.g. LETOR [97]).

```
<num> Number:  656
<title> lead poisoning children
<desc>
How are young children being protected against lead poi-
soning from paint and water pipes?
<narr>
Documents describing the extent of the problem, including
suits against manufacturers and product recalls, are rel-
evant.  Descriptions of future plans for lead poisoning
abatement projects are also relevant.  Worker problems
with lead are not relevant.  Other poison hazards for
children are not relevant.
```

Figure 3.1: A topic sample from Robust04.

[3]Originally there are 250, but one topic does not have any relevant document.

BioASQ is the dataset for the document ranking task of the BioASQ challenge, years 1-5[4]. It contains 2,251 English biomedical questions[5], formulated by biomedical experts, who looked for relevant documents through PubMed[6]. The document collection contains approximately 18M articles (i.e. concatenated titles and abstracts) from the PubMed Baseline 2018. In contrast to Robust04, BioASQ questions are simpler structures made of a question identifier and a single sentence. Figure 3.2 depicts a question example from BioASQ. In the 6th year of the BioASQ challenge, questions from year 1-5 are for training, while the questions added on the 6th year are held for testing, and consequently, their ground truth labels are kept secret for fair evaluation. This dataset does not include an official validation split.

```
"id": "58a2e5f760087bc10a000007"
"body": "Which is the primary protein component of Lewy
bodies?"
```

Figure 3.2: A question example from BioASQ dataset.

Experiments. Their experiments consist in comparing their deep neural IR variants to BM25 and BM25+extra (a linear combination of four features that includes the BM25 scores, but achieves a better effectiveness than vanilla BM25) on the Robust04 and BioASQ datasets. The metrics reported in this study are MAP, P@20, and NDCG@20. The authors distributed the Robust04 queries in five folds for cross-validation due to the reduced size of this dataset, where each fold contains 150 train, 50 validation, and 50 test queries. For BioASQ the authors decided to employ the first 4 years questions (1,751) for training, while the 5th year questions were split in validation (100), and test (400). To account for the variation introduced by the random seeds in the deep neural IR model, they performed their experiments five times and reported the best resulting model. The BM25 baselines are referred as *very strong* in this paper; however, when BM25+extra is applied to Robust04 it obtains a MAP of 0.25, 0.12 points below the best known MAP score for this dataset; thus, this model is clearly a weak IR baseline as discussed in section 2.4.1. The authors report statistical significance in their model comparisons, where their most effective baseline (BM25+extra) is at most 0.05 points below their most effective deep neural IR model (POSIT-DRMM+MV) across all the reported metrics and datasets.

3.2.3 Paper B: TVQA: localized, compositional video question answering

Task. Here, the task is to choose the right answer among five choices for a natural language question based on visual and textual information. The visual

[4] https://bioasq.org

[5] Although the original task actually addresses questions instead of queries, the task is about retrieving relevant documents instead of short correct answers.

[6] https://www.ncbi.nlm.nih.gov/pubmed/

information is extracted from video frames from TV shows and the textual information (apart from the questions) comes from the video scenes subtitles [67]. Although our experiments regarding this paper concern only the textual components of the task (section 4.1.3), we describe the multimodal characteristics of this paper for a holistic understanding of the problem it addresses.

Approach. The main purpose of this paper is to launch a new QA challenge[7], TVQA, and release its associated dataset (description in the next paragraph). The authors also introduced a multi-stream neural network with multiple layers for multimodal Video Question Answering. The inputs are video clip fragments, a subtitle, a question, and five candidate answers. GloVe word embeddings are used as inputs instead of raw text and video fragments are processed by a faster RCNN [102] for object and region detection. The inputs pass by several layers with LSTM, context matching, fusion, max-pooling, among them, to be finally combined with a softmax layer that outputs the predicted answer. The whole network architecture is illustrated in figure 3.3. This model can operate with different data streams or inputs (besides the answers), e.g. subtitles plus questions (S+Q), video features plus questions (V+Q), or subtitles plus videos plus questions (S+V+Q) [67].

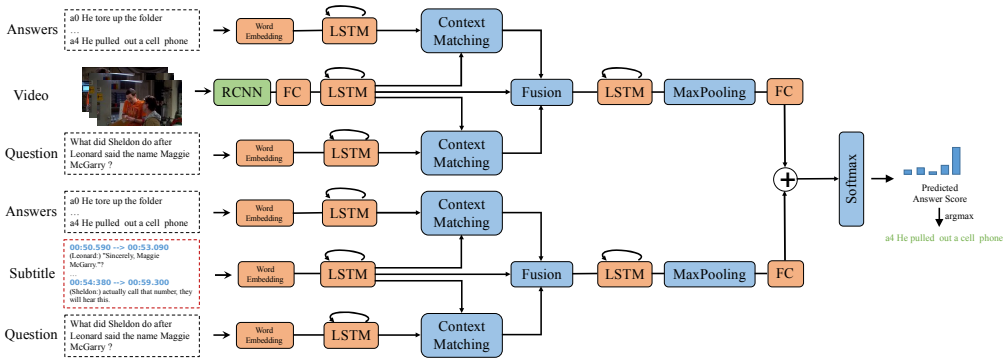


Figure 3.3: TVQA multi-stream model for multimodal Video QA, copied from [67].

Dataset. This paper introduces TVQA, a dataset created to evaluate video-QA models' multimodal semantic understanding. The dataset contains 152,545 human-written QA pairs, 21,973 video clips, and the corresponding subtitles from six popular TV shows about three genres: medical drama, sitcoms, and crime shows. The dataset was collected to reflect the content of rich dynamics and realistic social interactions, where the human annotators wrote QA pairs that require visual and language understanding to answer. For each question there are five answer candidates including only one correct answer. This dataset contains several types of questions, including: abstract (what), person (who), reasoning (why), method (how), among others. Some questions could be an-

[7] http://tvqa.cs.unc.edu/

swered based on subtitles or video only, while other questions require both modalities as illustrated in figure 3.4.

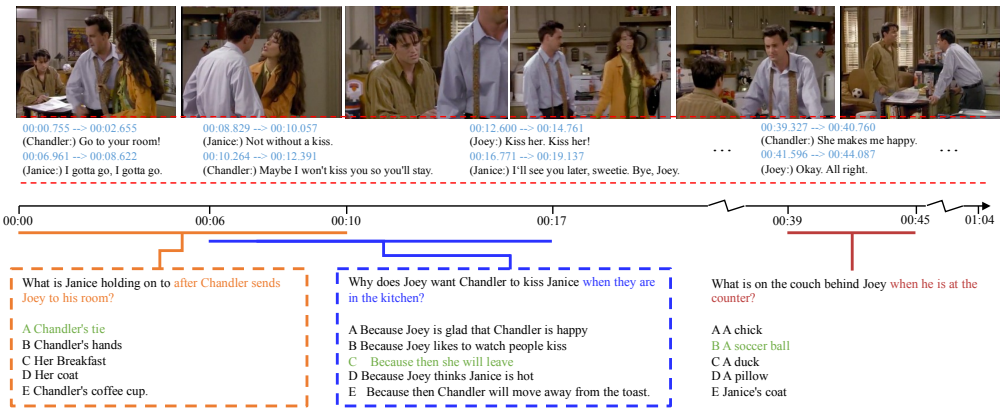


Figure 3.4: TVQA question examples, copied from [67].

Experiments. The authors propose several experiments with different models and different stream subsets. Among the proposed baselines, their most effective model is a combination of nearest neighbor search with tf-idf, where the vectors of question-answer, or subtitle-answer are compared via cosine similarity to predict the right answer. While their baselines use textual information only, their deep neural IR model exploits either question, question-video, subtitle-question, or subtitle-video-question features. Their most effective deep neural IR model is based on the three stream sources, followed by the model where subtitle-question features are employed. Another model variant is the one that uses localized subtitles, i.e. subtitles with answer localization timestamp annotation. The evaluation measure of choice is accuracy and no details of statistical significance testing are reported. The accuracy of the most effective baseline is around 0.13 points below the most effective deep neural model (text-based only).

3.2.4 Models for reproducibility experiments

In this thesis we are interested in building evidence about the effectiveness difference between the new proposed deep neural IR models and the IR baselines employed in the two previously identified ad-hoc and QA papers, therefore we focus our reproducibility efforts in the most effective proposed models of each type, namely: BM25+extra and POSIT-DRMM+MV for paper A, and tf-idf and TVQA S+Q for paper B. Following our discussion about reproducibility in section 1.4.2 we divide our effort in two approaches: replication of the most effective deep neural IR model and reproduction of the most effective IR baseline for each paper. Below, we explain the main decisions and steps for the two reproducibility approaches.

3.2.5 Replication of deep neural IR models

Besides the datasets, the artifact that enables the replication of effectiveness results for our selected papers is the available code for the proposed deep neural IR models (see table 3.3). The replication steps we follow for each paper are: (i) prepare an execution environment with the requirements specified by the authors, meaning that we get the required software packages (e.g. required Python version) to run on compatible hardware[8]; (ii) get a copy of the dataset(s) and code for deep neural IR models, (iii) run any preprocessing steps (e.g. enabling the right stemming options for indexing or running preprocessing scripts that have been already included), (iv) run the model's code, and (v) contrast the effectiveness outcomes with the results reported in the paper. Note that the ideal scenario for the replication approach is to obtain the results as reported in the original papers through the automated artifacts provided by the authors with minimal intervention from our side. We expect that the decisions made by the paper's authors regarding preprocessing (e.g. stemmer type) or model's behaviour (e.g. hyperparameters) are already set in code or are clearly specified in the paper's details or code documentation.

3.2.6 Reproduction of IR baselines.

The main difference between our reproduction approach and the replication procedure explained above is that the code for the IR baseline is not provided with the papers. Paper A mentions their selected IR toolkit (Galago v3.10 [9]), but paper B does not. To overcome this shortcoming, we decide to employ Indri v5.13, a well known IR toolkit, for both cases. Indri already includes the baseline implementations we require. Moreover, we rely on common libraries like Python Scikit-learn linear regression model[10] for the BM25+extra baseline from paper A. We aim to obtain similar effectiveness results although we employ different artifacts than those of the original papers. In general, our reproduction approach requires the following steps to accommodate either IR baseline into the model's evaluation pipeline: (i) parse documents and queries to TREC format[11], (ii) extract qrels from ground truth data, (iii) index documents, (iv) retrieve relevant documents, (v) format effectiveness results for specific evaluation, (vi) run evaluation and contrast the effectiveness results with the paper. For the reproduction phase we assume the use of the default values for preprocessing steps such as tokenization or the baseline hyperparameters unless specific values have been specified in the papers. As noted in [96, 71, 143], we anticipate that using different implementations of the same baseline can lead to different effectiveness results than those reported in the original studies.

[8] All our experiments are carried out on a dedicated physical server with Intel processor, SSD storage, GPU, and GNU/Linux (Ubuntu) operating system

[9] http://www.lemurproject.org/galago.php

[10] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[11] TREC is one of the formats supported by Indri for indexing and retrieval

3.3 Deployment of LTR baseline

In this section we propose to use LamdaMART as a baseline for the IR tasks from paper A and B to overcome the weak IR baselines' shortcomings we examined in the previous chapter (section 2.4.1). In short, we choose the LTR approach because it is the latest advance in IR prior to deep neural IR models (see section 2.2.1), and LambdaMART, in particular, because it is considered a SOTA LTR model (as argued in section 2.2.4) and the LTR approach has been applied in ad-hoc [38, 100] and QA [125, 147] tasks previously.

Here below we describe the decisions we make regarding the implementation, feature extraction, and hyperparameter optimization of LambdaMART.

3.3.1 Implementation

In the same way that we prefer an already available implementation for traditional IR baselines, we choose to use an available LambdaMART implementation. Among the available implementations of LambdaMART [75, 4, 3], we choose the one included in the popular RankLib package [75]. Similar to Indri, RankLib is part of the lemur project, and it is a well-maintained LTR model library that has been extensively used in past IR research [91, 136, 33]. RankLib is available as a java package that can be directly executed in our linux environment. RankLib's LambdaMART implementation exposes an API that allows us to specify the model's hyperparameters (number of trees, number of leaves, shrinkage or learning rate).

3.3.2 Feature extraction

For the feature extraction phase, we follow the work of Qin et al. [97] in that we extract similar features and employ the same output format. Since the datasets described in sections 3.2.2 and 3.2.3 do not contain hyperlink information, we exclude the features derived from this type of information, but extract low- and high-level content-based features only (recall section 2.2.3).

We wrote a customized C++ feature extractor program[12] reusing Indri's open source code[13] such that we access the same tokenization, stemming, indexing, and other functions used by our IR baselines; thus, preventing effectiveness variations that can be attributed to different implementations of these steps. The feature extractor takes the following inputs: index built with Indri, queries file, and qrels file (a column-formatted text file indicating the queries and their relevant documents), and the top N documents returned by a base ranker (as depicted in figure 2.3). The output of our program is a row/column matrix in the LETOR supervised ranking format [97]. Each row is a query-document pair, where the first column is the relevance label of the pair, the second column is the query id, the following columns are the extracted features, and the last

[12]The feature extractor code was written by Claudia Hauff, and then expanded by the author of this thesis to include the four features required by the BM25+extra features baseline mentioned in section 3.2.2 and to specify the stemmer type.

[13]We made use of this tutorial: `http://www.lemurproject.org/lemur/install.php`.

column is the document id. We extract 24 features in total, listed in table 3.4. While features 1-20 are identified in [97], features 21-24 are taken from paper A. Each row in this matrix is accompanied by the corresponding ground truth relevance label; hence, the output is suitable for training our LTR model. To be able to compare the effectiveness results among IR models, we extract features according to the same data splits and cross-validation folds proposed in our selected papers as explained in sections 3.2.2 and 3.2.3. Once the set of features has been extracted, we could train the LTR model with the whole set of features or different subsets, if needed.

Note that different subsets of features as well as how the have been extracted or preprocessed have an impact on the effectiveness results of the LTR algorithms [74]. Thus, feature engineering techniques such as feature selection or dimensionality reduction have been explored to improve the effectiveness of different LTR algorithms [42]. However, in this work we keep the features and the features' set unaltered during our experiments and leave feature engineering as future work.

Feature #	Name	Feature #	Name
1	LM for IR - Jelinek-Mercer	13	Covered query ratio
2	LM for IR - Dirichlet	14	Max query term tf
3	LM for IR - Two-stage	15	Min query term tf
4	tf-idf with BM25 weighting	16	Sum query term tf
5	Okapi BM25	17	Mean query term tf
6	Mean tf-idf	18	Normalized min query term tf
7	Sum tf-idf	19	Normalized max query term tf
8	Min tf-idf	20	Normalized sum query term tf
9	Max tf-idf	21	BM25 z-score
10	Var tf-idf	22	Exact match query percentage
11	Document length	23	Exact match idf weighted query percentage
12	Covered query term tokens	24	Bigram query percentage

Table 3.4: Content-based IR features for LTR approach.

3.3.3 Model training, HPO, and evaluation

With the extracted features split in training, validation, and test sets, the next step in our pipeline is to train LambdaMART with the hyperparameter configuration that yields the highest effectiveness in the validation set. LambdaMART allows direct optimization for specific IR metrics (recall section 2.2.4). We optimize for MAP in paper A because is the main reported IR metric in this study and it also allows us to compare the results obtained in Robust04 with previous research (see section 2.4). Since Paper B's metric is accuracy, which is not an optimization option in LambdaMART, and the objective of the task is to find the unique correct answer, we optimize LambdaMART for *precission at 1* (P@1).

As discussed in section 2.5.2, grid search is the most common HPO method found in IR literature [71, 78, 104], but it is also a time-consuming process. Instead of grid search, we compare two HPO methods: random search (RS), a common HPO baseline [37, 69], and the more sophisticated BOHB (see their

descriptions in section 2.5). To facilitate reproducibility, the hyperparameter configuration search is restricted to a grid that we define. We compare both HPO methods in terms of function evaluations, and set a maximum value of function evaluations that each HPO method should perform before returning the best hyperparameter configuration they found on the validation set. Following the works of [37, 69], we define the resource budget B as the number of training instances for the successive halving component of BOHB.

We evaluate LambdaMART according to the evaluation pipeline displayed in figure 1.2. For each task and dataset from our selected papers we apply LambdaMART with the best hyperparameter configuration found above on the test set. We also evaluate LambdaMART with the default hyperparameter values for comparison purposes. We compute the IR metrics MAP, P@20, and NDGC@20 following Paper A (section 3.2.2), and accuracy for Paper B (section 3.2.3).

At this point we have obtained the effectiveness results for the replicated deep neural IR models, reproduced traditional IR baselines, and proposed LTR baseline; thus we can establish comparisons among the models and check if any difference is significant. We test for statistical significance following the guidelines provided in our selected papers and also relevant literature on this topic [18, 115, 58].

Chapter 4

Experiments

Based on the research approach of the previous chapter and the evaluation pipeline described in section 1.4.2, here we explain the experiments we perform to answer the research question stated in the introduction chapter (section 1.3). In section 4.1 we detail the experimental conditions concerning the lab environment setup 4.1.1, datasets' preprocessing 4.1.2, replication of the deep neural IR models 4.1.3, reproduction of the IR baselines 4.1.4, and reproducibility assessment for papers A and B4.1.5. In section 4.2 we present the experiments we carried out with the LTR baseline with and without hyperparameter optimization4.2. Then, we present the discussion of the effectiveness results for the different models and IR tasks in section 4.3. We close this chapter with a summary of the performed experiments in section 4.4.

4.1 Reproducibility

4.1.1 Preliminaries

All our experiments are run on individual conda[1] environments (one per paper), in our Linux server (see section 3.2.6), to keep track of the software dependencies. Additionally, we installed Oracle Java 8.211[2], independently from the conda environments, to execute the RankLib jar package (see section 3.3.1). Tools that are compiled to be executed from the commandline like `IndriBuildIndex`, `IndriRunQuery`, RankLib, or `trec_eval` are called from Python scripts through the `subprocess` module[3]. Figure 4.1 gives an overview of the main components of our experimental pipeline followed for papers A and B.

4.1.2 Data preprocessing

The data preprocessing we performed for papers A and B includes different steps that depend on the dataset and the IR model. The papers give the preprocessing guidelines and the artifacts required for their deep neural IR models which

[1] https://docs.conda.io/en/latest/miniconda.html
[2] https://www.oracle.com/technetwork/java/javase/downloads/index.html
[3] https://docs.python.org/3/library/subprocess.html

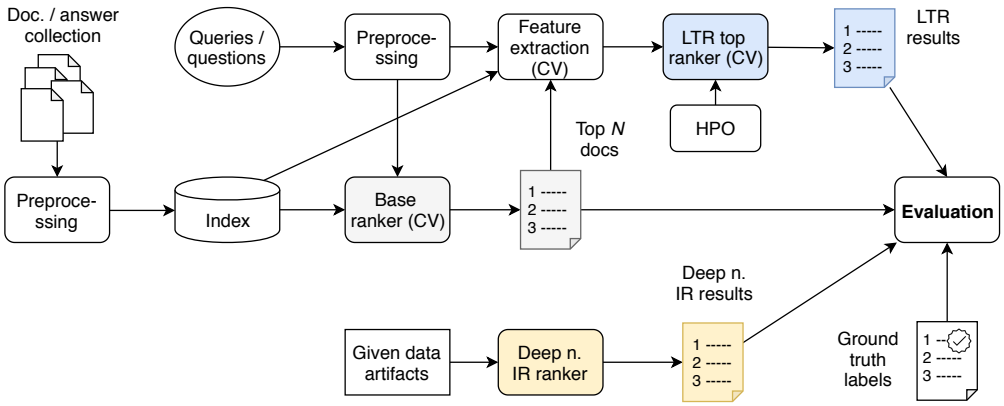


Figure 4.1: Overview of the experimental pipeline for papers A and B.

are also useful for the other IR models we apply. Paper A describes most of its preprocessing steps but does not include automated scripts or code to actually preprocess the document collection in natural language; instead, paper A provides already preprocessed data elements (e.g. IDF values, word2vectors, top N BM25 documents). Paper B does provide the automated scripts for the preprocessing steps besides the pre-trained GloVE embeddings that allow to execute the whole evaluation pipeline of its deep neural IR model. The replication artifacts from papers A and B are accessible through their code repositories (see table 3.3). No other preprocessing steps are assumed to be required to replicate the deep neural IR models from each paper.

On the other hand, the reproduction of the IR baselines and the proposition of the LTR model actually require that we carry out the preprocessing steps we found in the papers' instructions, code, and code documentation. We divided the preprocessing steps in two groups that align with the re-ranking scheme from section 2.2, this is: steps before the application of the base ranker and steps before the application of the LTR top ranker -BM25+extra or LambdaMART- (see figure 4.2). The deployment of the top ranker requires the execution of both groups. The preprocessing steps that are exclusive for the replication of the deep neural IR models (e.g. word embeddings generation) are not included in figure 4.2).

For BioASQ from paper A, we kept only the documents that contain title and abstract after downloading the MEDLINE/PubMed Baseline 2018 document collection[4]. Then, the IR models of paper A require the relevance scores of the top N documents retrieved by BM25 as input. This is expressed in figure 4.2 with the *create index* and *base ranker* preprocessing steps. For BioASQ in particular, we removed the documents that have been published after 2015 from the training set, and after 2016 from the validation and test sets (which are replaced by lower ranked documents up to N=*100*), because these documents do not have relevance annotation [80]. The steps *base ranker* and *generate LTR features* are performed within a cross-validation loop in the case of Robust04, following the same evaluation pipeline as the paper. The base ranker in this

[4]https://www.ncbi.nlm.nih.gov/pubmed/

case is BM25 for which we employed the default hyperparameters (b=0.75, and $k1$=1.2).

For the dataset of paper B, TVQA, we performed an extra preprocessing step: the redefinition of the train, validation, and test splits. This is because the public test set of TVQA does not contain the ground truth labels we need to evaluate the models[5]. We redefined the splits as follows: 80% from the original training set is taken as actual training data, while the remaining 20% is used as actual validation; additionally, we use the original validation set as actual testing. We did not discard the original test set but included its candidate answers in the index we build in section 4.1.4

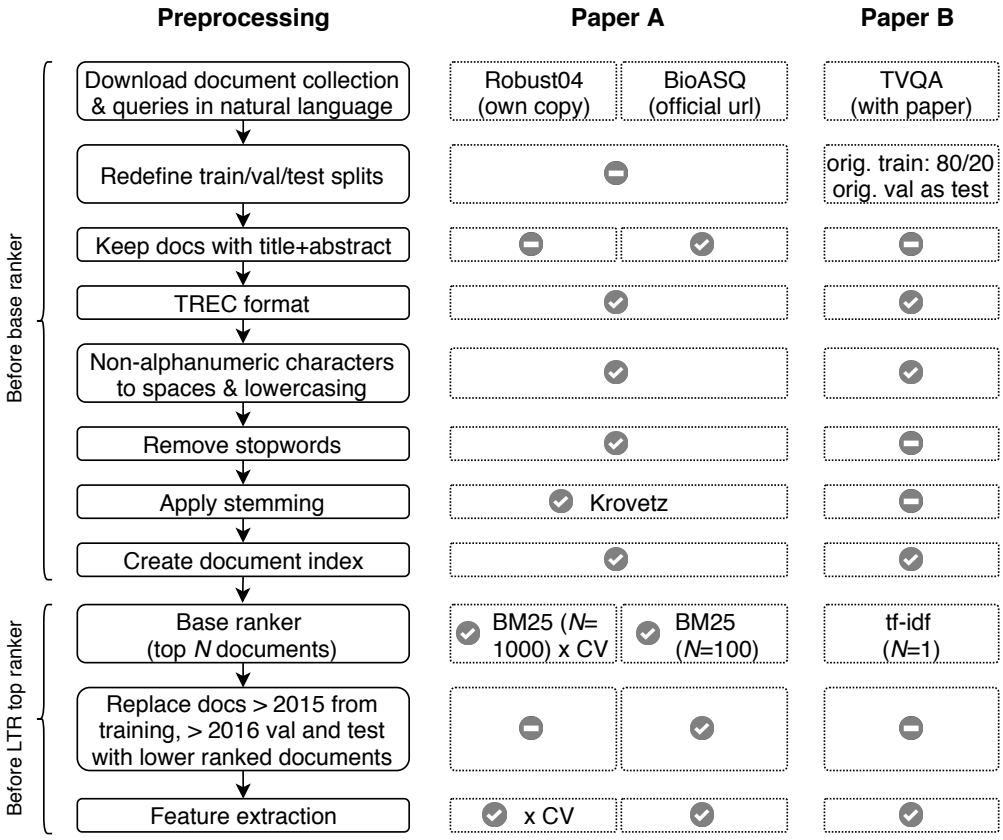


Figure 4.2: Preprocessing steps for the datasets from paper A and B before the application of the IR baselines. Vertically-left-aligned elements indicate the preprocessing steps. Vertically-centered elements and vertically-right-aligned elements indicate the correspondence of the preprocessing steps and the datasets of papers A and B respectively. Each graph element indicates whether the preprocessing step has been applied (✔) or not (⛔) to the dataset and other specific details.

[5]The test set ground truth labels are reserved for the TVQA challenge http://tvqa.cs.unc.edu/leaderboard.html.

4.1.3 Replication of the deep neural IR models

Replication of the deep neural IR model POSIT-DRMM+MV. To replicate POSIT-DRMM+MV, the most effective deep neural IR model proposed in paper A, we reuse the artifacts provided by the original research team, i.e. the model's code, the already preprocessed data elements (index IDF values, word2vec embeddings, and the BM25 top N documents), and the general paper's instructions. Then we proceed to run the code (see figure 4.3).

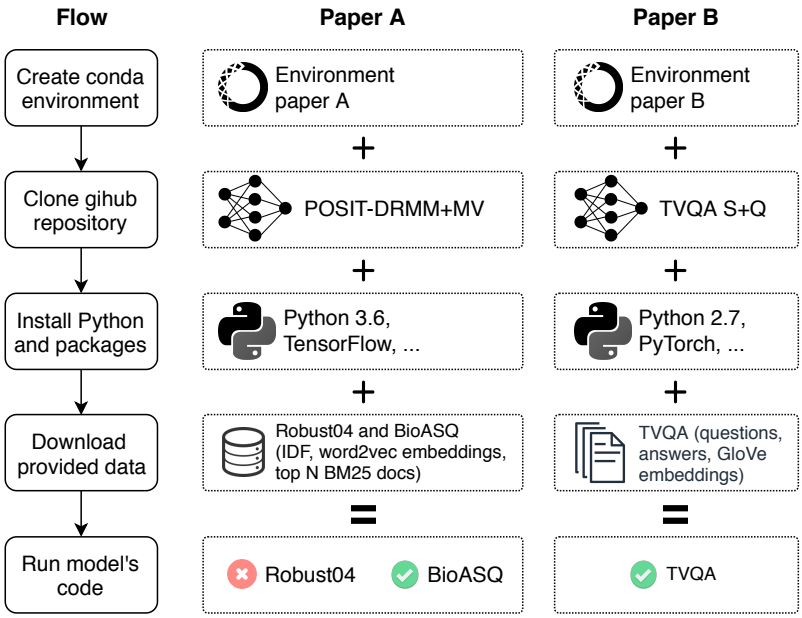


Figure 4.3: Replication flow for the deep neural IR models from paper A and B. Vertically-left aligned elements indicate the replication flow steps. The vertically-centered and vertically-right elements indicate the corresponding replication artifacts for the replication flow for papers A and B respectively. The bottom elements indicate whether the code's run execution is successful (✅) or not (❌).

The code run is divided in three manual steps: (1) train POSIT-DRMM+MV for 40 epochs, (2) check which epoch yields the best model in terms of MAP on the validation set, (3) evaluate the previous model on the test set with `trec_eval` called from a Python script. Due to the random seed of the model this process was repeated five times [80], for which we reported the mean and standard deviation of MAP, P@20, and NDCG@20 for the test set (see appendix B). The hyperparameters of this model, such as the learning rate, batch size, or the number of epochs, are described in the appendix of [80] and were already set in the code.

Remarkably, POSIT-DRMM+MV model's code comes with BioASQ hard-coded as the only dataset option to process. Although we modified and repeatedly debugged the code in an attempt to replicate the results on Robust04, the code fails to run after the first training Epoch. We realized that this issue has

already been reported[6], but it has not been fixed yet by the authors. Consequently, we were able to successfully run POSIT-DRMM+MV model with BioASQ dataset only.

Replication of the deep neural IR model TVQA S+Q. In contrast to paper A, we did not replicate the most effective deep neural IR model from paper B: TVQA S+V+Q (see section 3.2.3), because it depends on video features and we are interested in text features only. Thus, we replicated the **TVQA S+Q** model that exploits the information from the subtitles, questions (and answers) in the form of GloVe embeddings (see figure 3.3).

After preprocessing (see section 4.1.2), TVQA S+Q model was applied through the manual execution of three steps implemented by the authors of paper B in individual Python scripts: (1) vocabulary building and GloVe vectors extraction, (2) model training, where subtitles were specified as the input stream, and (3) inference. The hyperparameters (e.g. learning rate, number of epochs, mini-batch size, etc.) and the PyTorch random seed came set in the code. With the random seed set to a fixed value, we run TVQA S+Q model only once and recorded the accuracy achieved in selecting the right answer for the new test set we defined in section 4.1.2.

4.1.4 Reproduction of the traditional IR baselines

Reproduction of the IR baseline BM25+extra. To reproduce the most effective IR baseline from paper A, BM25+extra, we computed the four features it requires: z-score normalized BM25 scores (based on the BM25 scores obtained in section 4.1.2), percentage of query terms with exact match in the document (regular and IDF weighted), and percentage of query term bi-grams matched in the document. We extracted these features based on the details traced back to [80, 84, 111] since paper A did not provide a code implementation for this task. The features were implemented in C++ and are generated with our feature extractor (see 3.3.2). These features were then combined through a linear regression model (as suggested in paper A) that we implemented in Python to re-rank the top N documents for Robust04 and BioASQ.

For the evaluation we also employed `trec_eval`, but instead of using the same evaluation Python script as before we wrote a different Python script whose input is a TREC runfile. This facilitates the evaluation for this and the LTR baseline. Given that this IR baseline does not involve randomness, we run the experiment once; resulting in a single effectiveness result value per IR metric for BioASQ; however, as Robust04 comes distributed in five folds for cross-validation, we run the experiment for each fold and reported the cross-validation's mean and standard deviation values per IR metric.

Reproduction of the IR baseline tf-idf. Nearest neighbour search tf-idf S[7] is the most effective IR baseline, based on text information only presented in

[6] https://github.com/nlpaueb/deep-relevance-ranking/issues/2

[7] S stands for *subtitle* stream that is used instead of the *question* stream as baseline.

paper B, that we reproduce in this work. Note that the authors did not detail how they implemented tf-idf or which toolkit they employed. Consequently, we were free to try any available implementation, and stayed with Indri. We treated this QA task as an ad-hoc task, i.e. we employed the answers' index built in section 4.1.2, and then applied Indri's tf-idf implementation to rank these answers with respect to the localized subtitle. Note that the task in TVQA is multi-choice answer, where there is only one correct answer [67]; therefore, we restricted the set of answers to be queried and ranked to only the five candidate answers each question has.

The nearest neighbour search in paper B is implemented through cosine similarity to determine the right answer; however, we relied on the scoring scheme from Indri's tf-idf implementation [145] to pick the top one answer as the correct choice (see the *base ranker* for paper B in figure 4.2). Based on this choice, we computed the test accuracy with the same formula specified in the evaluation script from paper B. Default hyperparameter values were preserved for this model (b=0.75, k1=1.2) and the ranking list was stored in one runfile per data split.

4.1.5 Reproducibility assessment

How replicable and reproducible are the IR models from papers A and B? We assess the reproducibility of the models we replicated and reproduced for papers A and B in a similar way to that of [96]. We adapt the assessment criteria to our work as follows: (a) dataset reconstructability/availability, (b) preprocessing, and (c) model approach. The evaluation of these three factors gives an overall reproducibility assessment of *replicability* for the deep neural IR model and *reproducibiilty* for the IR baseline and sets the context to interpret the effectiveness results in section 4.3. Tables 4.1 (paper A) and 4.2 (paper B) summarize our findings with full (●), some (◐), and none (○) specified details for a given reproducibility criterion. We build this information upon the exploratory analysis carried out in section 3.1 and the reproducibility experiments from sections 4.1.3 and 4.1.4. Each reproducibility criterion and the corresponding findings are discussed next.

(a) *Dataset reconstructability.* During the exploratory phase in section 3.1 we purposely selected papers that include the origins for the queries and document collections they employed. However, we found particularly difficult to reconstruct the BioASQ document collection since we could not verify if the documents we downloaded and filtered from the PubMed 2018 baseline website were the actual documents the authors employed in paper A. Our only guide in this regard was the approximate total number of documents from paper A's appendix. On the other hand, Robust04 and TVQA datasets came as well defined collections with a specific list of queries and documents. Likewise, the train, validation, and test data splits, as well as the cross-validation folds, are clearly specified in both papers for all datasets. Therefore, we can rule out that any difference between the original effectiveness results and the ones we reproduce are due to discrepancies in the actual datasets employed, except for BioASQ.

(b) ***Preprocessing.*** Both papers have some preprocessing steps in common, for example tokenization, special character handling, or upper/lower case handling. The main difference we find between the two papers is the provision of automated preprocessing scripts. Although the processing pipeline performed on BioASQ in paper A is the most complex among the analyzed datasets, it does not come accompanied by automated preprocessing scripts but by intermediate preprocessed data elements, e.g. top 100 BM25 ranked documents and IDF scores. We agree that the main goal of paper A is the proposition of its new deep IR model, but since the base ranker's results constitute an important component of the model's re-ranking scheme affected also by other preprocessing like stemming, tokenization, and stopwords handling, we consider that including the automated scripts to get the intermediate results is desirable for reproducibility purposes; even though, we also accept that this task requires a considerable amount of effort that might be out of the interest and scope of a given research team. In contrast, paper B includes the automated script for its simpler preprocessing pipeline's steps. Tokenization is particularly important for BioASQ since it contains many specialized terms whose semantics depends on the correct use of upper letters, e.g. "CNEs, conserved noncoding elements", or special characters, as in chemical compound names such as "Kub5-Hera/RPRD1B".

(c) ***Model approach.*** We split these criteria in two: one for the replicated deep neural IR model and one for the reproduced IR baseline. Code/implementation availability for both types of models is clearly the most important factor in this criterion. Both papers provide an implementation for their proposed deep neural IR models (see section 4.1.3), although paper A did not include the code that allows to replicate the effectiveness results for Robust04, but only for BioASQ. We argue that the reason for this is the same as with the preprocessing steps: a sense of unjustified extra effort. Additionally, not all the software libraries were fully specified for paper A's functioning, which demanded some extra hours of debugging until we fixed the issues. Regarding the IR baselines, both papers identify the employed models, and while paper A gives further details about the name and version of the chosen IR toolkit, paper B omits such information. The availability of the exact same baselines' implementations used in these papers was not a crucial aspect in our reproducibility experiments because we chose a specific IR toolkit to implement the baselines (recall section 3.2.6). Instead, it would have been more advantageous to know the employed baselines' hyperparameters (the hyperparameters for the deep neural IR models were set in the code), but we resorted to the default values due to the lack of this information. Given the amount of information and details both papers present for their deep neural IR models, we could conclude that these papers are self-contained with respect to the deep neural IR models, but only partially with respect to the baselines, whose results' reproduction required that we close the gap with the papers' references (e.g. to implement the extra features for BM25+extra baseline in paper A) and our own domain knowledge. Lastly, the effectiveness results, our main point of reference for our

experiments, are reported in tables in both papers. Furthermore, paper A also specifies the statistical significant effectiveness differences among its models, the type of statistical significance test, and how it was employed (see more details in section 4.3.1). Paper B also mentions that its results are statistical significant, but not with the level of detail we see in paper A.

(d) ***Overall assessment.*** Table 4.1 reveals that neither the deep neural IR model nor the baseline from paper A are fully replicable or reproducibllle respectively. We were able to successfully run the deep neural IR model's code for BioASQ but not for Robust04, even when we had access to all the intermediate preprocessed data elements provided by the authors. Moreover, the three previous assessment criteria show that the implementation of paper A's models is highly complex, what in turn hindered our reproducibility efforts. Such complexity could also explain why the original authors were not able to provide simpler and more complete reproducible artifacts. Note that the IR baseline (BM25) is the cornerstone for the implementation of the LTR model we propose; thus, a successful reproduction of this baseline (in terms of the effectiveness metrics) directly contributes to a successful implementation of the LTR model. Paper B, on the other hand, shows that its deep neural IR model is fully replicable given that all the previous criteria are defined and the tasks automated; however, the tf-idf baseline is reproducible to a certain extent mainly because the paper lacks the details for it's deployment.

4.2 LambdaMART baseline deployment

We apply LambdaMART, our chosen LTR model for strong IR baseline candidate for the ad-hoc and QA tasks from paper A and B, following the re-ranking scheme of section 3.3. The first stage of this scheme is the retrieval of the top N documents with a base ranker, what was already accomplished through the reproduction of the IR baselines in section 4.1.4. This means that we use the IR baselines as our base rankers for the re-ranking scheme, more specifically the BM25[8] ranked list of documents for paper A and the tf-idf ranked list of answers for paper B. Table 4.3 exhibits the hyperparameter values used for LambdaMART base rankers. The remaining two stages are: feature extraction and training with HPO. The application of these two stages is very similar for the ad-hoc and QA tasks; however, there are two main differences between them: the number of documents retrieved by the base banker and the LambdaMART optimization metric, as we will see next.

Feature extraction

The previous implementation of the BM25 and tf-idf base rankers generated the required data inputs for the feature extraction process, these are the relevance of the documents/answers with respect to the queries/questions, Indri's

[8]We use BM25 and not BM25+extra as base ranker to make the results comparable to the deep neural IR model in paper A.

Criteria	**Assessment**	
Dataset reconstructability / availability		
Origin given	●	
Queries and documents available	◐	
Clear train, validation, and test splits	●	
CV folds specified (only Robust04)	●	
Preprocessing specified		
Document filtering (only BioASQ)	○	
Upper/lower case handling	○	
Stemming	●	
Special character handling	○	
Stopwords handling	●	
Tokenization	○	
Base ranker details	◐	
Automated scripts	○	
Model approach	Deep neural IR model	IR baseline
Implementation available with paper	◐	○
Libraries specified	◐	●
Feature engineering	-	◐
Hyperparameter settings	●	○
Paper self-contained	◐	◐
Results reported	●	●
Overall assessment		
Replicability	◐	-
Reproducibility	-	◐

Table 4.1: Reproducibility assessment for deep neural IR model and IR baseline for paper A.

built index, and the base rankers' results (runfiles) corresponding to the data split, and cross-validation fold in the case of Robust04. During the feature extraction process we also followed the same preprocessing that each paper specifies, for instance we applied the krovetz stemmer indicated in paper A or the non-removal of stopwords from paper B (see section 4.1.2). Note that the features are extracted only for the query-document pairs listed in the corresponding data-split/cross-validation-fold base rankers' runfiles; thus, LambdaMART effectively re-ranks only the top N documents/answers and not the entire document/answer collection.

The final outcome of this phase is a text file with query-document pair feature vector with the 24 features specified in table 3.4, one per data split and cross-validation fold, that are the inputs for the subsequent training, validation, and testing phases.

Criteria	**Assessment**	
Dataset reconstructability / availability		
Origin given	●	
Questions and answers available	●	
Clear train, validation, and test splits	●	
Preprocessing specified		
Upper/lower case handling	●	
Stemming	●	
Special character handling	●	
Stopwords handling	●	
Tokenization	●	
Automated scripts	●	
Model approach	Deep neural IR model	IR baseline
Implementation available with paper	●	○
Libraries specified	●	○
Hyperparameter settings	●	○
Paper self-contained	●	◐
Results reported	●	●
Overall assessment		
Replicability	●	-
Reproducibility	-	◐

Table 4.2: Reproducibility assessment for deep neural IR model and IR baseline for paper B.

Paper	**Base ranker**	**Hyperparameter**	**Value**
A	BM25	*b*	0.75
		k1	1.2
		top N docs Robust04	1000
		top N docs BioASQ	100
B	tf-idf	*b*	0.75
		k1	1.2
		k3	0.7
		top N answers	5

Table 4.3: LambdaMART base rankers hyperparameters. The hyperparameters b, k, k1, and k3 are the default values from Indri, while the number of top N docs/answers come defined in the corresponding papers.

Model training with HPO

The training procedure of LambdaMART requires three elements: the training features set (generated before), a target IR optimization metric, and a hyper-

parameter configuration. The IR evaluation metrics are already established in each paper; consequently, we trained LambdaMART with MAP for paper A, but with P@1 for paper B because accuracy is not an optimization target metric option in the Ranklib's implementation of LambdaMART.

Regarding the LambdaMART's hyperparameter configuration composed by the number of leaves K, learning rate ρ, and number of trees T, described in section 2.2.4, we are interested in knowing if there is a hyperparameter configuration that yields better effectiveness results than the default configuration values from RankLib's LambdaMART implementation (see table 4.4). Consequently, we first run an experiment training LambdaMART with the default hyperparameter values and then we compare how efficient random search (RS) and BOHB HPO methods are to find a (possibly) better hyperparameter configuration in the validation set. Other LambdaMART's hyperparameters or configuration options, such as early stopping[9], are set to the default values throughout all our hyperparameter experiments.

For our HPO experiments, we make use of the HpBandSter's Python implementation[10] to compare RS and BOHB. HpBandSter actually minimizes the loss with respect to the chosen metric[11]; thus, we establish the loss to be minimized as: (1 - MAP) for paper A and (1 - P@1) for paper B. Note that P@1 is only employed during the training process but we still measure the effectiveness of LambdaMART in terms of accuracy according to the pipeline of paper B. We further define a hyperparameter configuration search space (shown in table 4.4) with value ranges comparable to those of [137]. The resulting hyperparameter configuration space has 40,000 possible hyperparameter configurations, out of which we determine how efficient RS or BOHB are in finding the best hyperparameter configuration after running a certain number of function evaluations set to 1000 at max. The successive halving mechanism of BOHB also requires the definition of a minimum and maximum training budget B (see section 2.5.4). We express the training budget as the number of training samples (see section 3.3.3), and set the budget limits to B_{min}=10% and B_{max}=100% of the training samples for the minimum and maximum values respectively.

When RS and BOHB have reached the max number of function evaluations or no improvement has been made after several function evaluations, we take note of the hyperparameter configuration that yields the minimum loss for each HPO method. Due to the stochastic nature of the two HPO methods, this experiment is repeated five times. Although we could have exploited paralellization with both RS and BOHB, we evaluated the hyperparameter configurations one at a time to avoid two issues that BOHB faces when using parallelization: late prediction model building and delayed stuck in local optimum[12]. When the HPO process finishes, we take note of the best hyperparameter configuration found by any of the two methods for each dataset and compare it to the default hyperparameter configuration computing the IR metric in test set. The

[9] Early stopping is implemented in RankLib to stop the training of LambdaMART if there is no improvement after a certain number of boosting iterations.

[10] https://automl.github.io/HpBandSter/build/html/index.html

[11] https://automl.github.io/HpBandSter/build/html/quickstart.html#id4

[12] https://automl.github.io/HpBandSter/build/html/best_practices.html

Hyperparameter	Default value	HPO range		
		min	max	step
K	10	5	100	5
ρ	0.1	0.01	0.5	0.01
T	1000	100	2000	50
Early stopping	100	-	-	-
Optimization metric paper A	MAP			
Optimization metric paper B	P@1			

Table 4.4: LambdaMART default hyperparameter values and HPO ranges. Only the hyperparameters K, ρ, and T are subject to HPO in our experiments. MAP is the optimization metric for paper A, while P@1 replaces accuracy for paper B.

test effectiveness results are reported in tables 4.5 and 4.6 for Robust04 and BioASQ respectively.

4.3 IR models' effectiveness results

In this section we compare the effectiveness results we obtained from the replication of the deep neural IR models (section 4.1.3), reproduction of the IR baselines (section 4.1.4), and the application of the LTR model (section 4.2) for the ad-hoc and QA tasks from papers A and B respectively, and doing so, we proceed to answer the research question we posed in section 1.3.

4.3.1 Comparing the effectiveness results on the ad-hoc retrieval task (paper A)

Here we discuss the effectiveness results of the experiments we performed with the three types of IR models on Robust04 (table 4.5) and BioASQ (table 4.6) for paper A. We report the test MAP, P@20, and NDCG@20 scores to enable the comparisons with the results from paper A [80], and also validation MAP to support our observations with the HPO experiments. Note that the effectiveness differences between models are small: less than 0.05 points across all metrics for both datasets; thus, we tested for statistical significance among the models we replicate/reproduce[13] with the paired randomized test specified in paper A, following the instructions from [80, 58, 115], with a significance level of 0.05. Although not specified in the paper, we decided to apply the Bonferroni correction to the p-values which is suitable for multiple hypothesis testing [18]. We run the test for 20,000 iterations. We stopped the test if the p-value does not go over 0.01 or fall under 0.1 after 1,000 iterations. This significance testing method does not make assumptions over the data distribution unlike other common tests such as the Student t-test [58].

[13]To test for significance against the original paper's results we would require the metrics' effectiveness scores for every query, but this information is not available.

IR model	$\mathbf{MAP_{val}}$	$\mathbf{MAP_{test}}$	$\mathbf{P@20_{test}}$	$\mathbf{NDCG@20_{test}}$
BM25 (paper A)	-	0.2380*	0.3540*	0.4250*
BM25 (reproduced)	0.2347±0.02	0.2347±0.02$^{(4,7)}$	0.3464±0.03	0.4166±0.03$^{(4)}$
BM25+extra features (paper A)	-	0.2500*	0.3670*	0.4320*
BM25+extra features (reproduced)	0.2281±0.02	0.2235±0.02	0.3445±0.02	0.3788±0.02
POSIT-DRMM+MV (paper A)	-	**0.2710***	**0.3890***	**0.4640***
POSIT-DRMM+MV (replicated)	-	-	-	-
LambdaMART+default	0.2292±0.03	0.2189±0.05	0.3356±0.04	0.4002±0.05$^{(4)}$
LambdaMART+HPO	**0.2465±0.03**	0.2394±0.04$^{(4,7)}$	0.3564±0.04$^{(7)}$	0.4217±0.04

Table 4.5: Effectiveness results on Robust04. (*) indicates that paper A does not provide the standard deviation for the CV folds. The superscript numbers indicate statistical significance with respect to: BM25 (reproduced)^2, BM25+extra (reproduced)^4, POSIT-DRMM+MV (replicated)^6, and LambdaMART+default^7, with Bonferroni adjusted *p-value* < 0.05. The highest effectiveness value is in bold numbers.

IR model	$\mathbf{MAP_{val}}$	$\mathbf{MAP_{test}}$	$\mathbf{P@20_{test}}$	$\mathbf{NDCG@20_{test}}$
BM25 (paper A)	-	0.4610	0.2550	0.5540
BM25 (reproduced)	0.4320	0.4598	0.2557	0.5518
BM25+extra features (paper A)	-	0.4870	0.2660	0.5810
BM25+extra features (reproduced)	0.4350	0.4641	0.2609	0.5515
POSIT-DRMM+MV (paper A)	-	**0.5100**	**0.2770**	**0.6030**
POSIT-DRMM+MV (replicated)	**0.4628**	0.4763$^{(2)}$	0.2617	0.5701$^{(2)}$
LambdaMART+default	0.4371	0.4703	0.2639	0.5603
LambdaMART+HPO	0.4518	0.4692	0.2605	0.5576

Table 4.6: Effectiveness results on BioASQ. The superscript numbers indicate statistical significance with respect to: BM25 (reproduced)^2, with Bonferroni adjusted *p-value* < 0.05. The highest effectiveness value is in bold numbers.

How effective are the reproduced IR baselines BM25 and BM25+extra?

Rows 1 and 2 in tables 4.5 and 4.6 contain the effectiveness results for the original and reproduced BM25 models. Albeit not the main target IR baseline we reproduce for paper A, we use BM25 results as a sanity check before the application of the other IR models. In fact, we observe that the effectiveness difference between our results (row 2) and the ones reported in paper A (row 1) for both datasets is very small across all IR metrics; thus, we consider that we have successfully reproduced the preprocessing conditions to a large extent in spite of using a different IR toolkit, the lack of details, and the other shortcomings devised in section 4.1.5. In particular, the BM25 results we obtain on Robust04 are in line with [47] that also uses Indri.

We find unexpected outcomes for BM25+extra, the main IR baseline we reproduce from paper A. We observe that our results for this model with Robust04 and BioASQ are 0.02 points or more worse than those reported in paper A (rows 3 and 4 in tables 4.5 and 4.6 respectively), across all reported IR metrics. We also realize that BM25+extra falls under BM25 on Robust04 but

slightly over it on BioASQ. The degraded effectiveness of BM25+extra with respect to BM25 on Robust04 draws our attention since it re-ranks the BM25 results and uses the BM25 normalized scores as part of its features. The contradictory findings about BM25+extra with respect to the original paper leave the open question of how the four *extra* features were exactly generated and combined. This reproducibility issue could have been circumvented with the release of the implementation of BM25+extra alongside paper A.

How effective is the replicated deep neural IR model POSIT-DRMM+MV?

In contrast to Robust04, we actually obtain the effectiveness results for POSIT-DRMM+MV on BioASQ (recall section 4.1.3). Intriguingly, our effectiveness results for the best run out of five trials with this model are 0.0327 MAP points below those reported for the same model (rows 5 and 6 from table 4.6) and the IR baseline in paper A. This is a surprising outcome since we rely on the same artifacts that the authors provided for this model, i.e. the intermediate data elements such as the pretrained word2vec vectors, IDF values, and BM25 top documents and the model's code implementation (recall section 4.1.2). We attribute this discrepancy to the random seeds in POSIT-DRMM+MV model. In fact, the five trials on this model show that the effectiveness can be as low as a MAP=0.0553 depending on the random seed (see appendix B, table B.1 for the five trials' results with mean and standard deviation across the three metrics). Hence, we argue that executing more trials with different random seeds could eventually lead to similar results as reported in paper A or perhaps better. Nonetheless, POSIT-DRMM+MV remains as the most effective model among the results we obtained for BioASQ, although we find that its effectiveness improvement is significant only when compared to BM25.

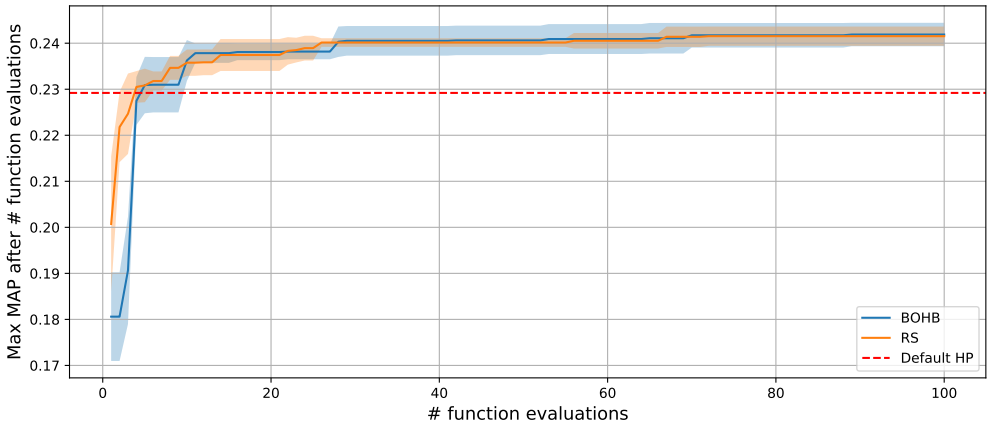
How effective is the proposed LTR baseline LambdaMART with and without HPO?

The second-last row in tables 4.5 and 4.6 display the effectiveness results for the LambdaMART model with default hyperparameter values for Robust04 and BioASQ datasets. LambdaMART+default is located on opposite extremes of the effectiveness scale when compared to the other models depending on the dataset. Its effectiveness is the lowest for Robust04, but it is only surpassed by POSIT-DRMM+MV on BioASQ when considering only the results we replicated/reproduced. Apart from the sensitivity to hyperparameters, the disparity we observe in our results could be caused by the sensitivity each dataset has for the extracted IR features and how they are combined. Likewise BM25+extra, LambdaMART+default is less effective on Robust04 but more effective on BioASQ with respect to the other IR models.

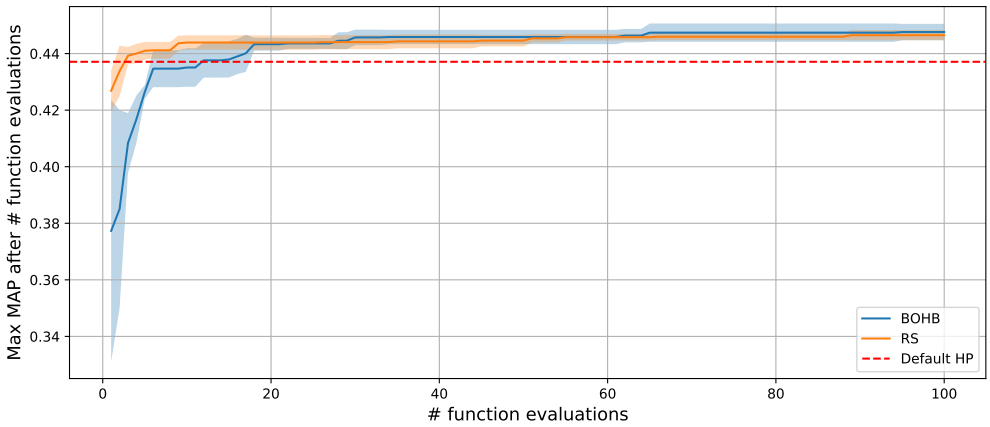
How efficient are the HPO methods RS and BOHB? Now, taking into account the validation MAP scores (second column in tables 4.5 and 4.6), it is evident that the HPO process enhances the effectiveness of LambdaMART with

respect to the default hyperparameter values for the two datasets. We found that depending on the dataset, a different hyperparameter configuration gives the best effectiveness results, confirming the claims of [63]. The HPO learning curves for both Robust04 and BioASQ are illustrated in figures 4.4a and 4.4b respectively. These curves show the max MAP attained after # function evaluations with RS and BOHB HPO methods. Both figures unveil similar convergence patterns for the two datasets, although with different MAP ranges. Both HPO methods start with hyperparameter configurations that perform worst than the default values. RS is the first to overcome this threshold in less than five function evaluations. BOHB follows RS closely in Robust04 but is a bit slower for BioASQ. The curves steadily step up until the 40th function evaluation approximately. They continue growing but at a much slower rate after this point, meaning that it is much harder to find a better hyperparameter configuration later on. We stopped the HPO process after 100 function evaluations and picked the best hyperparameter configuration found among the two HPO methods until then. Interestingly, we found the best hyperparameter configuration for each dataset with BOHB, notwithstanding the two HPO methods have almost identical performance when they become stable. However, we note that BOHB has a larger variance throughout the whole HPO process. The initial slow convergence of BOHB can be explained by the use of low budgets at the beginning of the successive halving mechanism, but this in turns speeds up the overall function evaluation. In average, BOHB completes the 100 function evaluations 8% and 18% faster than RS for Robust04 and BioASQ respectively. More details about the chosen optimized hyperparameters, average wallclock-time, and complementary figures for the HPO process of paper A are presented in appendix C.

How effective is LambdaMART with HPO? The MAP effectiveness scores on the test set of Robust04 show that the LambdaMART+HPO improvement is significant over LambdaMART+default and BM25+extra. However, the MAP gain of the validation BioASQ set is not observed in the test set. Actually, the test effectiveness of LambdaMART+HPO slightly decreases across the three IR metrics when compared to LambdaMART+default. This indicates that the hyperparameter configuration found during the HPO process overfits to the validation set and consequently the model loses generazability power for unseen data. One factor that could be contributing to this overfitting is the uneven 100/400 query validation/test split in BioASQ (recall from section 3.2.2), while in Robust this proportion is 50/50 besides the fact that 5-fold CV is performed with this dataset. Additionally, the small and not significant effectiveness difference between LambdaMART+default and LambdaMART+HPO makes them almost indistinguishable when applied to BioASQ. In short, although we observe a slightly higher effectiveness in LambdaMART+HPO for BioASQ when compared to the IR baselines BM25 and BM25+extra models we reproduce, the improvement is not significant. This is the same conclusion we reach for POSIT-DRMM+MV and LambdaMART+HPO.



(a) Robust04.



(b) BioASQ.

Figure 4.4: HPO processes performed on the validation sets of Robust04 and BioASQ for paper A. The blue and orange solid curves represent the average max MAP over the fives runs of the BOHB and RS HPO methods respectively, with the error shown as the shaded areas around these curves. The dashed horizontal red line indicates the MAP score reached by the LambdaMART+default model.

Does the deep neural IR model keep its advantage before the LTR baseline for the ad-hoc retrieval task? We did not find a significant difference between POSIT-DRMM+MV and any of the two LambdaMART variants on BioASQ, while for Robust04 it was not possible to establish the comparison. However, from the weak IR baseline survey we carried out on Robust04 in section 2.4.2 we see that all the IR models we experiment with in this work would be regarded as weak baselines for this dataset.

4.3.2 Comparing the effectiveness results on the QA task (paper B)

Here we discuss the effectiveness results of the experiments we performed with the three types of IR models on TVQA dataset for paper B. Recall that we redefined the train/val/test splits in section 4.1.2; therefore, we report our validation and test accuracy results in columns 2 and 3 from table table 4.7 and the test results reported in paper B in column 4 from the same table. We observe that the test effectiveness results are still very close between the original and replicated/reproduced models in spite of the data splits redefinition, an effect that can be ascribed to TVQA's large amount of data. In contrast to paper A, the effectiveness differences between the models we try for paper B are larger: around 0.13 accuracy points between the most and least effective models. Since paper B does not specify which statistical significance test it employs, we apply the randomized test explained in section 4.3.1 with the same p-value threshold, Bonferroni correction, and number of iterations.

IR model	**Acc$_{val^*}$**	**Acc$_{test^*}$**	**Acc$_{test^{**}}$**
tf-idf (paper B)	-	-	0.5123
tf-idf (reproduced)	0.5194	0.5242	-
TVQA S+Q (paper B)	-	-	**0.6623**
TVQA S+Q (replicated)	**0.6548**	$0.6586^{(2,5,6)}$	-
LamdaMART+default	0.5560	$0.5629^{(2)}$	-
LamdaMART+HPO	0.5591	$0.5630^{(2)}$	-

Table 4.7: Effectiveness results on TVQA. (*) indicates the accuracy on the redefined validation and test sets. (**) indicates the accuracy on the original test set from paper B. The superscript numbers indicate statistical significance with respect to: tf-idf (reproduced)^2, LambdaMART+default^5, LambdaMART+HPO^6, with Bonferroni adjusted *p-value* < 0.05. The highest accuracy value is in bold numbers.

How effective is the reproduced IR baseline tf-idf?

Rows 1 and 2 of table 4.7 show the effectiveness of the original and reproduced tf-idf models. Although the exact scoring mechanism varies between the two tf-idf implementations (see section 4.1.4) and the data splits in which they are evaluated are not the same, the accuracy difference between them is just above 0.01 accuracy points. Analogous to paper A, we could consider this a sanity check for the evaluation pipeline up to this point. This is not a surprising outcome because the preprocessing was already automated by the original authors and the implementation of the tf-idf baseline did not require specialized steps like the IR baselines of paper A (recall figure 4.2). Likewise BM25 in paper A, tf-idf is the base ranker for LambdaMART in the experiments for paper B; therefore, the successful reproduction of tf-idf underpins the results we later

obtain with LambdaMART. The effectiveness difference of tf-idf with respect to the other models leaves no doubt about it being the least effective model.

How effective is the replicated deep neural IR model TVQA S+Q?

Similar to the reproduction of tf-idf, the replication of the effectiveness results for TVQA S+Q is very close to the numbers reported in paper B as we can appreciate in rows 3 and 4 of table 4.7. This was the expected result since we have all the artifacts for replication available as assessed previously in table 4.2, including the same random seed, in opposition to the deep neural IR model case from paper A. With our experiments we confirm that TVQA S+Q is the most effective IR model (based on textual information only) in paper B, and that the differences it has with the other models are large and significant (>0.13 and >0.9 accuracy points higher than tf-idf and LambdaMART+HPO respectively). The large effectiveness difference of TVQA S+Q with respect to the other two models could be partially explained by the fact that this method uses two text streams: the subtitles and the questions, while the other models only employ the subtitles stream, an implementation choice we kept for reproducibility purposes.

How effective is the proposed LTR baseline LambdaMART with and without HPO?

The last two rows in table 4.7 display the results for LambdaMART+default and LambdaMART+HPO. Based on the validation and test results (of our data splits), the two LambdaMART variants have no significant difference among them; their accuracy in the test set is almost identical. On the other hand, the difference of around 0.04 accuracy points of these two models with respect to tf-idf is significant. However, LambdaMART is still far from the deep neural IR model, regardless the HPO process.

How efficient are the HPO methods RS and BOHB? By checking the validation accuracy we realize that the HPO process has minimal impact for this combination of IR model and dataset, contrasting with the results we got in paper A. The HPO process performed for LambdaMART with the TVQA dataset is shown in figure 4.5. The patterns of the RS and BOHB learning curves are very similar to what we already described for paper A: both HPO methods start with worse hyperparameters than the default values, RS surpasses the default threshold faster than BOHB, but BOHB manages to find the best hyperparameter configuration overall despite having a larger variance. Additionally, BOHB was 42% faster than RS in completing the 100 function evaluations. Nonetheless, finding a better hyperparameter configuration than the default values in the validation set did not translate in significantly better results in the test set, as we saw in the preceding paragraph. The details about the chosen optimized hyperparameters, average wallclock-time, and complementary figures for the HPO process for paper B are presented in appendix C.

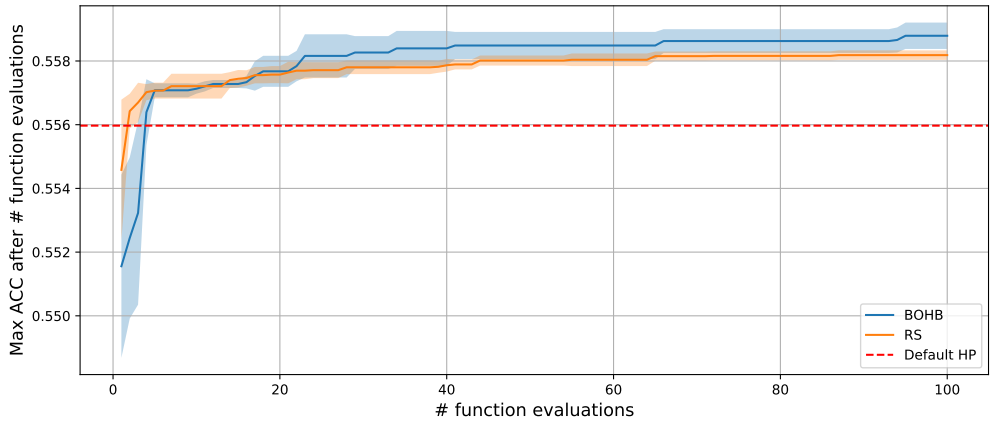


Figure 4.5: HPO processes performed on the validation set of TVQA for paper B. The blue and orange solid curves represent the average max accuracy over the fives runs for the BOHB and RS HPO methods respectively, with the error shown as the shaded areas around these curves. The dashed horizontal red line indicates the accuracy score reached by the LambdaMART+default model.

Does the deep neural IR model keep its advantage before the LTR baseline for the QA task? The TVQA S+Q model proves to be significantly more effective than any of the two variants of LambdaMART we experiment with. Our proposed LTR model improves over the tf-idf baseline, but the gain is not enough to be on par of the deep neural IR model proposed for this QA task.

4.4 Summary

In this chapter we have performed replication and reproduction experiments for the deep neural IR models and IR baselines for the ad-hoc retrieval task of paper A and the QA task of paper B. We also assessed the reproducibility of these models with three criteria: dataset reconstructability, preprocessing, and model approach.

Then, we proposed the application of LambdaMART, a state-of-the-art LTR model with and without HPO, as our candidate for strong baseline for the ad-hoc retrieval and QA tasks. This model was applied as the top ranker in a re-ranking scheme, where the base ranker position was filled with the IR baselines BM25 and tf-idf from papers A and B respectively, depending on the IR task. We extracted 24 content-based IR features to train LambdaMART either with the default hyperparameter values or executing the HPO process. Additionally, we benchmarked two different HPO methods: RS and BOHB, to determine which one is more efficient in finding a better hyperparameter configuration that beats the default configuration values.

Lastly, we performed an effectiveness comparison of the IR models we applied for the particular IR tasks and datasets of papers A and B that enabled us to answer our research question.

Chapter 5

Conclusions and Future Work

In this chapter we present the conclusions regarding the approach we followed and the experiments we performed to fill the gap and answer our research question. Then, we address the limitations of our work and pose directions for future work.

5.1 Conclusions

More than ten years ago, Armstrong et al. [11] showed the issues related to the use of weak IR baselines (non-competitive models that lack the latest advancements in the field). These issues included: the progress stagnation in ad-hoc retrieval, as evidenced with the Robust04 collection [11, 143]; the weak baseline selection bias, where these baselines might be purposely selected because they are easier to beat which favors publication of results; and reproducibility: the comparison between newly introduced IR approaches with the state-of-the-art might not be possible if the models' implementations or other crucial details, e.g. preprocessing steps or model's hyperparameters, are not available. Moreover, past weak IR baseline research has mainly focused on the ad-hoc retrieval task while the reproducibility issues and poor empirical practices when comparing different IR models, persist. It has also been suggested that HPO should be performed when making comparisons to IR baselines, instead of using their *vanilla* versions [71, 115].

However, there is still a gap in the coverage of the weak baseline IR research, as we evidenced with the conducted exploratory analysis of the EMNLP 2018 proceedings. Here we found thirteen cases of week baselines in current deep neural IR research, spread over different IR tasks, datasets, and IR baselines. Most of them have not been part of the weak baseline analysis yet. To fill the gap, we picked two of cases and investigated whether the effectiveness advantage that their recently proposed deep neural IR models show over the (weak) baselines also holds when compared to LambdaMART, the state-of-the-art LTR model we chose to use, plus hyperparameter optimization. The two chosen papers are about two relevant IR tasks: ad-hoc retrieval (paper A [80]) and QA (paper B [67]). Then we proceeded with several reproducibility experiments employing the artifacts provided by these two papers, the deployment of LambdaMART,

and the effectiveness results comparisons among the applied IR models.

5.1.1 Reproducibility experiments

During our reproducibility experiments, we found that the deep neural IR model, POSIT-DRMM+MV, and the IR baseline, BM25+extra, from paper A, are not fully replicable and not fully reproducible respectively on Robust04 and BioASQ datasets. In particular, we can point out that paper A did not provide the scripts to preprocess the document collections in natural language. The fact that the provided code was not meant for Robust04 but only for BioASQ, the unknown random seed for POSIT-DRMM+MV, and the lack of details for the feature extraction process for BM25+extra were the major reproducibility challenges for paper A. On the other hand, the deep neural IR model (TVQA S+Q) and IR baseline (tf-idf) from paper B were fully replicable and reproducible respectively on TVQA dataset, thanks to the completeness of artifacts for TVQA S+Q, and the pipeline simplicity for tf-idf. These details were highlighted in our reproducibility assessment.

Subsequently, we deployed LambdaMART as the top ranker in a re-ranking scheme, where BM25 and tf-idf constituted the base rankers for paper A and B respectively. LambdaMART was trained with 24 IR features and we experimented with the default and optimized hyperparameters of LambdaMART. For the latter we employed the RS and BOHB HPO techniques.

5.1.2 Effectiveness results

The effectiveness results we obtained for POSIT-DRMM+MV and BM25+extra were noticeably lower than reported in paper A, even though our reproduced BM25 model achieved effectiveness results comparable to the original ones. The shortcomings we encounter while recreating the evaluation pipeline for these models might explain the effectiveness degradation. For example, POSIT-DRMM+MV was unstable with respect to the random seeds. On the other hand, the effectiveness results we observed for TVQA S+Q and tf-idf were very close to those reported in paper B; something that is easier to explain based on the completeness of artifacts for TVQA S+Q and the simplicity of implementation of tf-idf, characteristics we also observed in the reproducibility assessment.

HPO efficiency results. The application of LambdaMART with and without HPO revealed that the BOHB HPO method (with the number of training samples as budget) was actually more efficient than RS in finding a better hyperparameter configuration than the default configuration on the validation set across Robust04, BioASQ, and TVQA, although it showed more variance than RS. However, the LambdaMART+HPO's effectiveness was (significantly) higher on Robust04, lower on BioASQ, and almost the same on TVQA, when compared to LambdaMART+default. Thus, the HPO process improved the effectiveness of LambdaMART on Robust04, a relatively small dataset, where five-fold CV was executed. Meanwhile, the HPO process did not (significantly)

change the effectiveness of LambdaMART on BioASQ and TVQA, two relatively large datasets with different train, validation, and test splits. Therefore, the default hyperparameters of LambdaMART already achieve effectiveness close to the optimal values on BioASQ and TVQA for MAP and accuracy respectively.

Considering our research question (section 1.3), we found that *the deep neural IR model from paper A, POSIT-DRMM+MV, did not show a significant effectiveness improvement over the IR baseline, BM25+extra, or over our chosen LTR baseline, LambdaMART (default and HPO variants), for the ad-hoc retrieval task*; which is opposed to the claims of McDonald et al. [80], except for the fact that we found POSIT-DRMM+MV significantly more effective than BM25. Meanwhile, the effectiveness of POSIT-DRMM+MV on Robust04 still requires confirmation. On the other hand, *the effectiveness improvement of the deep neural IR model from paper B, TVQA S+Q, was large and significant over the IR baseline, tf-idf, and over our chosen LTR baseline, LambdaMART (default and HPO variants), for the QA task*; confirming the findings of Lei et al. [67].

Finally, observing the effectiveness results we got, those reported in paper A, and the weak baseline analysis from section 2.4.3, all the IR models we tested on Robust04 can be catalogued as **weak baselines** for this dataset, regardless their condition of traditional, LTR, or deep neural IR models. On the other hand, it is difficult to make such statement about BioASQ since we are to see the effectiveness of other IR models that have not been applied yet to this dataset under the specific conditions described in paper A. Meanwhile, TVQA S+Q appears as a strong (text-only) IR baseline over the two other models for the QA task of paper B.

To conclude, we have unveiled current, non-previously studied cases of weak baselines in deep neural IR research, and we have broaden the coverage of the weak baseline analysis to QA. At the same time we have built research evidence that challenges the findings of McDonald et al. [80] and confirms those of Lei et al. [67] regarding the improvements that new deep neural IR models bring to the field. We expect that our work encourages further research about the issues related to the use of weak IR baselines.

5.2 Future work

In this work we have addressed part of the issues regarding weak IR baselines; however, this research area is wide and pose many questions that are still open and require further research. Next we present the limitations we found during the development of our work and directions for future work.

Reproducibility of state-of-the-art models' results. Previous IR literature showed that the most effective models for ad-hoc retrieval on Robust04 are several years old [31, 33, 132]; however, we do not have public available implementations of these promising models yet in contrast to, e.g. LambdaMART,

the model we proposed in the present work. Therefore, creating public available implementations of these models would be beneficial for the IR community. Another step in the same direction is to release self-contained images of these models that include all the prerequisites to perform end-to-end effectiveness replication, as seen in the OSIRRC [73]. We set out to release the code we developed during this thesis under the same self-contained image format. Additionally, we propose to keep replicating/reproducing the effectiveness of new (deep neural[1]) IR models to build solid evidence that supports or challenges the claims made about them; thus, contributing to construct more scientific knowledge [115]. This also applies to the models found in the leaderboards of the NLP community.

Coverage of the weak baseline analysis. In our exploratory analysis (section 1.4.1) we identified thirteen studies making use of weak IR baselines. We covered only two of these cases due to human and resource limitations. Nonetheless, those studies include IR tasks and datasets that have not been scrutinized yet. Studying these cases is a good opportunity to have a more complete picture of the state of affairs concerning weak baselines. We hope that with the initiatives of banning weak baselines during peer-review [41] the prevalence of these cases decrease.

Effect of features in LTR and feature engineering. One of the open questions that our work leaves, is why BM25+extra and LambdaMART+default performed worse than BM25 on Robust04, given that they re-ranked the top N BM25 docs and the BM25 scores were one of the features for both top rankers. Therefore, we suggest experimenting with different feature engineering techniques such as feature selection [42] and feature normalization [74] to assess the effect of different features on these LTR models. Feature engineering can build more convincing evidence for empirical comparisons, but it is a lengthy process that requires domain expertise and might lead researchers to consider that the required time and effort would be better spent on the hottest research trend.

Different base and top rankers for LTR. During the re-ranking tasks for papers A and B we constraint ourselves to the use of the same base rankers employed in the papers, BM25 and tf-idf, for reproducibility purposes. Nonetheless, different base rankers could have been applied. In fact, it is intriguing what would be the outcome of employing a highly effective base ranker (e.g. [33, 31]) and either POSIT-DRMM+MV, LambdaMART, or other rankers on top, just to name a few examples.

Test BERT. Very recently, BERT [34], a novel deep neural network method for NLP pre-training, has become the state-of-the-art for a wide variety of NLP tasks [118, 15, 70]. The impact of BERT has been so sound, that Google itself

[1] As long as the available resources, e.g. computing time, allow it.

has decided to update its search engine with it[2]. Hence, we propose to assess the effect of BERT on the IR tasks we addressed in this thesis as future work.

Error analysis. In this work we have focused on the reproducibility aspects of the different models we applied. However, we have not closely examined where these models actually fail. Error analysis can be employed to detect the blind spots of each model and, based on this information, improve their effectiveness. Consequently, we propose to perform the error analysis as future work.

HPO. In this thesis we limited the scope of HPO to LambdaMART, and kept the default hyperparameters of the IR baselines, base rankers, and IR features for reproducibility purposes. Thus, it is an open question what is the effect of HPO on the intermediate IR models of the evaluation pipeline we employed in this thesis. Likewise, we propose the experimentation with the training time and the number of boosting iterations as BOHB budget, instead of the number of training samples. Additionally, we could also experiment with parallelization to better exploit the resources of modern computers.

[2] https://searchengineland.com/faq-all-about-the-bert-algorithm-in-google-search-324193

Bibliography

[1] Trec 2004 robust track guidelines. https://trec.nist.gov/data/robust/04.guidelines.html. (Accessed on 10/05/2019).

[2] Emnlp 2018 call for papers. URL https://emnlp2018.org/calls/papers/.

[3] lezzago/lambdamart: Python implementation of lambdamart. https://github.com/lezzago/LambdaMart. (Accessed on 10/16/2019).

[4] pyltr/lambdamart.py at master · jma127/pyltr. https://github.com/jma127/pyltr/blob/master/pyltr/models/lambdamart.py. (Accessed on 10/16/2019).

[5] Acl sigdat. URL https://sigdat.org/.

[6] . A cross-benchmark comparison of 87 learning to rank methods. *Inf. Process. Manage.*, 51(6):757–772, November 2015. ISSN 0306-4573. doi: 10.1016/j.ipm.2015.07.002. URL https://doi.org/10.1016/j.ipm.2015.07.002.

[7] *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1000.

[8] Nasreen Abdul-jaleel, James Allan, W. Bruce Croft, O Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. Umass at trec 2004: Novelty and hard. In *In Proceedings of TREC-13*, 2004.

[9] Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D Lawrence, and David C Gondek. *Learning to Rank for Robust Question Answering*. ISBN 9781450311564. URL http://bit.ly/vViNIG.

[10] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Evaluatir: An online tool for evaluating and comparing ir systems. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09,

pages 833–833, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572153. URL http://doi.acm.org/10.1145/1571941.1572153.

[11] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: Ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 601–610, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1646031. URL http://doi.acm.org/10.1145/1645953.1646031.

[12] Ricardo A Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. 1999. ISBN 020139829X. URL http://people.ischool.berkeley.edu/{~}hearst/irbook/print/chap10.pdf.

[13] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[14] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554, USA, 2011. Curran Associates Inc. ISBN 978-1-61839-599-3. URL http://dl.acm.org/citation.cfm?id=2986459.2986743.

[15] Gabriel Bernier-Colborne, Cyril Goutte, and Serge Léger. Improving cuneiform language identification with BERT. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 17–25, TOBEFILLED-Ann Arbor, Michigan, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-1402. URL https://www.aclweb.org/anthology/W19-1402.

[16] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Claus Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275, 2012.

[17] Carl Boettiger. An introduction to docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79, January 2015. ISSN 0163-5980. doi: 10.1145/2723872.2723882. URL http://doi.acm.org/10.1145/2723872.2723882.

[18] Leonid Boytsov, Anna Belova, and Peter Westfall. Deciding on an adjustment for multiplicity in ir experiments. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 403–412, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2034-4. doi: 10.1145/2484028.2484034. URL http://doi.acm.org/10.1145/2484028.2484034.

[19] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederik Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[20] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102363. URL http://doi.acm.org/10.1145/1102351.1102363.

[21] Christopher Burges, Krysta Svore, Paul Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. In *Proceedings of the learning to rank Challenge*, pages 25–35, 2011.

[22] Christopher J C Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.634{&}rep=rep1{&}type=pdf.

[23] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 193–200, Cambridge, MA, USA, 2006. MIT Press. URL http://dl.acm.org/citation.cfm?id=2976456.2976481.

[24] Stefan Büttcher, Charles L. A. Clarke, and Brad Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. page 621, 2006. doi: 10.1145/1148170.1148285.

[25] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273513. URL http://doi.acm.org/10.1145/1273496.1273513.

[26] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing and Management*, 52(6):1161–1177, 2016. ISSN 03064573. doi: 10.1016/j.ipm.2016.05.004.

[27] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*, pages 1–24, 2011.

[28] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. Keyphrase generation with correlation constraints. In *Proceedings of the*

2018 Conference on Empirical Methods in Natural Language Processing, pages 4057–4066, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1439. URL https://www.aclweb.org/anthology/D18-1439.

[29] Yu-An Chung, Hung-Yi Lee, and James Glass. Supervised and unsupervised transfer learning for question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1585–1594, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1143. URL https://www.aclweb.org/anthology/N18-1143.

[30] Francesco Colace, Massimo De Santo, Luca Greco, and Paolo Napoletano. Improving relevance feedback-based query expansion by the use of a weighted word pairs approach. *J. Assoc. Inf. Sci. Technol.*, 66(11): 2223–2234, November 2015. ISSN 2330-1635. doi: 10.1002/asi.23331. URL https://doi.org/10.1002/asi.23331.

[31] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 758–759, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1572114. URL http://doi.acm.org/10.1145/1571941.1572114.

[32] Nick Craswell, W Bruce Croo, Jiafeng Guo, and Bhaskar Mitra. SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17). 2017. doi: 10.1145/3077136.3084373. URL http://delivery.acm.org/10.1145/3090000/3084373/p1431-craswell.pdf?ip=145.94.4.91{&}id=3084373{&}acc=ACTIVESERVICE{&}key=0C390721DC3021FF.512956D6C5F075DE.4D4702B0C3E38B35.4D4702B0C3E38B35{&}{_}{_}acm{_}{_}=1561121284{_}d69b870dd6465020191f1d2e0198aa81.

[33] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 365–374, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2257-7. doi: 10.1145/2600428.2609628. URL http://doi.acm.org/10.1145/2600428.2609628.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186,

Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[35] Pinar Donmez, Krysta M Svore, and Christopher J C Burges. *On the Local Optimality of LambdaRank*. 2009. ISBN 9781605584836. URL https://www.cs.cmu.edu/{~}pinard/Papers/sigirfp092-donmez.pdf.

[36] Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. A dataset and baselines for sequential open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1077–1083, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1134. URL https://www.aclweb.org/anthology/D18-1134.

[37] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. Technical report, 2018. URL http://arxiv.org/abs/1807.01774.

[38] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. Modeling Diverse Relevance Patterns in Ad-hoc Retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval - SIGIR '18*, pages 375–384, New York, New York, USA, 2018. ACM Press. ISBN 9781450356572. doi: 10.1145/3209978.3209980. URL http://dl.acm.org/citation.cfm?doid=3209978.3209980.

[39] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. Modeling diverse relevance patterns in ad-hoc retrieval. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018*, 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018, pages 375–384. Association for Computing Machinery, Inc, 6 2018. doi: 10.1145/3209978.3209980.

[40] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[41] Yang Gao, Steffen Eger, Ilia Kuznetsov, Iryna Gurevych, and Yusuke Miyao. Does my rebuttal matter? insights from a major NLP conference. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1274–1290, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1129. URL https://www.aclweb.org/anthology/N19-1129.

[42] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In *Proceedings of the 30th annual international ACM SIGIR*

conference on Research and development in information retrieval, pages 407–414. ACM, 2007.

[43] Ana Gonzalez, Isabelle Augenstein, and Anders Søgaard. A strong baseline for question relevancy ranking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4810–4815, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1515. URL https://www.aclweb.org/anthology/D18-1515.

[44] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 55–64, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4073-1. doi: 10.1145/2983323.2983769. URL http://doi.acm.org/10.1145/2983323.2983769.

[45] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A Deep Look into Neural Ranking Models for Information Retrieval. Technical report, 2019. URL http://google.com.

[46] Claudia Hauff. Information retrieval, March 2019. URL: https://github.com/chauff/IN4325/blob/master/slides/coreIR-evaluation.pdf. Last visited on 2019/07/26.

[47] Claudia Hauff. Dockerizing indri for osirrc 2019. 2019.

[48] Claudia Hauff, Matthias Hagen, Anna Beyer, and Benno Stein. *Towards realistic known-item topics for the ClueWeb*. 2012. ISBN 9781450312820. URL http://answers.yahoo.com/.

[49] Hussein Hazimeh and ChengXiang Zhai. Axiomatic analysis of smoothing methods in language models for pseudo-relevance feedback. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 141–150, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3833-2. doi: 10.1145/2808194.2809471. URL http://doi.acm.org/10.1145/2808194.2809471.

[50] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Z Ghahramani, M Welling, C Cortes, N D Lawrence, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences.pdf.

[51] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Z Ghahramani, M Welling, C Cortes, N D Lawrence, and

K Q Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences.pdf.

[52] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2263-8. doi: 10.1145/2505515.2505665. URL http://doi.acm.org/10.1145/2505515.2505665.

[53] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1110. URL https://www.aclweb.org/anthology/D17-1110.

[54] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Automatic machine learning: methods, systems, challenges. *Challenges in Machine Learning*, 2019.

[55] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain, 09–11 May 2016. PMLR. URL http://proceedings.mlr.press/v51/jamieson16.html.

[56] Frederick Jelinek. *Statistical methods for speech recognition*. MIT press, 1997.

[57] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775067. URL http://doi.acm.org/10.1145/775047.775067.

[58] Avinash Kak. Evaluating information retrieval algorithms with significance testing based on randomization and student's paired t-test. *Tutorial Presentation*, 2013.

[59] Jaspreet Kaur. Effective Question Answering Techniques and their Evaluation Metrics. Technical Report 12, 2013. URL http://www.ask.com.

[60] Sadegh Kharazmi, Falk Scholer, David Vallet, and Mark Sanderson. Examining additivity and weak baselines. *ACM Trans. Inf. Syst.*, 34(4): 23:1–23:18, June 2016. ISSN 1046-8188. doi: 10.1145/2882782. URL http://doi.acm.org/10.1145/2882782.

[61] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999. ISSN 0004-5411. doi: 10.1145/324133.324140. URL http://doi.acm.org/10.1145/324133.324140.

[62] Christoph Kofler, Martha Larson, and Alan Hanjalic. User intent in multimedia search: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 49(2):36:1–36:37, August 2016. ISSN 0360-0300. doi: 10.1145/2954930. URL http://doi.acm.org/10.1145/2954930.

[63] Ron Kohavi and George H John. Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995*, pages 304–312. Elsevier, 1995.

[64] Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, dec 2011. doi: 10.1016/j.ins.2011.07.047. URL https://linkinghub.elsevier.com/retrieve/pii/S0020025511003860.

[65] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6. doi: 10.1145/383952.383972. URL http://doi.acm.org/10.1145/383952.383972.

[66] Nayeon Lee, Chien-Sheng Wu, and Pascale Fung. Improving large-scale fact-checking using decomposable attention models and lexical tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1133–1138, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1143. URL https://www.aclweb.org/anthology/D18-1143.

[67] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara Berg. TVQA: Localized, compositional video question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1379, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1167. URL https://www.aclweb.org/anthology/D18-1167.

[68] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4482–4491, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1478. URL https://www.aclweb.org/anthology/D18-1478.

[69] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185): 1–52, 2018. URL http://jmlr.org/papers/v18/16-558.html.

[70] Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. A BERT-based universal model for both within- and cross-sentence clinical temporal relation extraction. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 65–71, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-1908. URL https://www.aclweb.org/anthology/W19-1908.

[71] Jimmy Lin. The Neural Hype and Comparisons Against Weak Baselines. *ACM SIGIR Forum*, 52(1):40–51, 2019. ISSN 01635840. doi: 10.1145/3308774.3308781. URL https://dl.acm.org/citation.cfm?id=3308781.

[72] Jimmy Lin. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 52(2):40–51, January 2019. ISSN 0163-5840. doi: 10.1145/3308774.3308781. URL http://doi.acm.org/10.1145/3308774.3308781.

[73] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig Macdonald, and Sebastiano Vigna. Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval*, pages 408–420, Cham, 2016. Springer International Publishing. ISBN 978-3-319-30671-1.

[74] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[75] Yaqi Liu, Xiaoyu Zhang, Xiaobin Zhu, Qingxiao Guan, and Xianfeng Zhao. ListNet-based object proposals ranking. *Neurocomputing*, 267:182–194, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.06.008. URL http://www.sciencedirect.com/science/article/pii/S0925231217310639.

[76] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1895–1898, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1646259. URL http://doi.acm.org/10.1145/1645953.1646259.

[77] François Mairesse, Paul Raccuglia, and Shiv Vitaladevuni. Search-based evaluation from truth transcripts for voice search applications.

In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 985–988, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4069-4. doi: 10.1145/2911451.2914735. URL http://doi.acm.org/10.1145/2911451.2914735.

[78] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1): 100–103, 2010.

[79] Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*, 2018.

[80] Ryan McDonald, George Brokos, and Ion Androutsopoulos. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1211. URL https://www.aclweb.org/anthology/D18-1211.

[81] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

[82] Bhaskar Mitra and Nick Craswell. Neural Models for Information Retrieval. 2017. URL http://arxiv.org/abs/1705.01509.

[83] Bhaskar Mitra, Nick Craswell, et al. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1): 1–126, 2018.

[84] Sunil Mohan, Nicolas Fiorini, Sun Kim, and Zhiyong Lu. Deep Learning for Biomedical Information Retrieval: Learning Textual Relevance from Click Logs. Technical report, 2017. URL http://ncbi.nlm.nih.gov/pubmed.

[85] Seung-Hoon Na, In-Su Kang, Ye-Ha Lee, and Jong-Hyeok Lee. Completely-arbitrary passage retrieval in language modeling approach. In *Proceedings of the 4th Asia Information Retrieval Conference on Information Retrieval Technology*, AIRS'08, page 22–33, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 3540686339. doi: 10.5555/1786374.1786378. URL https://doi.org/10.5555/1786374.1786378.

[86] Anastasios Nentidis, Anastasia Krithara, Konstantinos Bougiatiotis, Georgios Paliouras, and Ioannis Kakadiaris. Results of the sixth edition of the BioASQ challenge. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*, pages 1–10, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5301. URL https://www.aclweb.org/anthology/W18-5301.

[87] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. 2016.

[88] Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. Multi-level Abstraction Convolutional Model with Weak Supervision for Information Retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, New York, NY, USA, 2018. ACM, ACM. ISBN 978-1-4503-5657-2. doi: 10.1145/3209978.3210123. URL http://doi.acm.org/10.1145/3209978.3210123.

[89] Larry Page, Sergey Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1998.

[90] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 24(4):694–707, April 2016. ISSN 2329-9290. doi: 10.1109/TASLP.2016.2520371. URL https://doi.org/10.1109/TASLP.2016.2520371.

[91] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 257–266, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4918-5. doi: 10.1145/3132847.3132914. URL http://doi.acm.org/10.1145/3132847.3132914.

[92] Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. Mapping natural language commands to web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4970–4976, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1540. URL https://www.aclweb.org/anthology/D18-1540.

[93] Roger D Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

[94] Jay M. Ponte and W. Bruce Croft. 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, pages 275–281, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5. doi: 10.1145/290941.291008. URL http://doi.acm.org/10.1145/290941.291008.

[95] Marco Ponza, Luciano Del Corro, and Gerhard Weikum. Facts that matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1043–1048, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1129. URL https://www.aclweb.org/anthology/D18-1129.

[96] Martin Potthast, Sarah Braun, Tolga Buz, Fabian Duffhauss, Florian Friedrich, Jörg Marvin Gülzow, Jakob Köhler, Winfried Lötzsch, Fabian Müller, Maike Elisa Müller, et al. Who wrote the web? revisiting influential author identification research applicable to information retrieval. In *European Conference on Information Retrieval*, pages 393–407. Springer, 2016.

[97] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. Technical report. URL http://research.microsoft.com/.

[98] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

[99] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*, 2018.

[100] Ridho Reinanda and Dwi H Widyantoro. *Performance Comparison of Learning to Rank Algorithms for Information Retrieval*. 2014. ISBN 9781479968589. URL https://pdfs.semanticscholar.org/cd12/e191d2c2790e5ed60e5186462e6f8027db1f.pdf.

[101] Z Reitermanová. *Data Splitting*. ISBN 9788073781392. URL https://www.mff.cuni.cz/veda/konference/wds/proc/pdf10/WDS10{_}105{_}i1{_}Reitermanova.pdf.

[102] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[103] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1020.

[104] Stephen Robertson. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2010. ISSN 1554-0669. doi: 10.1561/1500000019. URL http://www.staff.city.ac.uk/{~}sb317/papers/foundations{_}bm25{_}review.pdf.

[105] Stephen Robertson, S Walker, S Jones, M M Hancock-Beaulieu, and M Gatford. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST, 1995. URL https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/.

[106] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[107] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL http://doi.acm.org/10.1145/361219.361220.

[108] D. Sculley, Jasper Snoek, Alexander B. Wiltschko, and Ali Rahimi. Winner's curse? on pace, progress, and empirical rigor. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018. URL https://openreview.net/forum?id=rJWF0Fywf.

[109] Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 559–564, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1052. URL https://www.aclweb.org/anthology/D18-1052.

[110] Minjoon Seo, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Phrase-indexed question answering: A new challenge for scalable document comprehension. *arXiv preprint arXiv:1804.07726*, 2018.

[111] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th*

International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pages 373–382, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3621-5. doi: 10.1145/2766462.2767738. URL http://doi.acm.org/10.1145/2766462.2767738.

[112] Darsh J Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. Adversarial domain adaptation for duplicate question detection. *arXiv preprint arXiv:1809.02255*, 2018.

[113] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. Technical report. URL http://www.ibm.com/software/commerce/optimization/cplex-optimizer/.

[114] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[115] Mark D Smucker, James Allan, and Ben Carterette. *A Comparison of Statistical Significance Tests for Information Retrieval Evaluation*. 2007. ISBN 9781595938039. URL https://ciir-publications.cs.umass.edu/getpdf.php?id=744.

[116] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. Technical report. URL https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[117] Victoria Stodden. The scientific method in practice: Reproducibility in the computational sciences. 2010.

[118] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1035. URL https://www.aclweb.org/anthology/N19-1035.

[119] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Multi-granular sequence encoding via dilated compositional units for reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2141–2151, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1238. URL https://www.aclweb.org/anthology/D18-1238.

[120] Simone Teufel. An Overview of Evaluation Methods in TREC Ad Hoc Information Retrieval and TREC Question Answering. In *Evaluation of Text and Speech Systems*, pages 163–186. 2007. doi: 10.1007/978-1-4020-5817-2_6. URL https://ccc.inaoep.mx/{~}villasen/bib/ANOVERVIEWOFEVALUATIONMETHODSINTRECADHOCIRANDTRECQA.pdf.

[121] Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. Frank: A ranking method with fidelity loss. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 383–390, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277808. URL http://doi.acm.org/10.1145/1277741.1277808.

[122] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1), April 2015. doi: 10.1186/s12859-015-0564-6. URL https://doi.org/10.1186/s12859-015-0564-6.

[123] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[124] K. Veningston and R. Shanmugalakshmi. Information Retrieval by Document Re-ranking using Term Association Graph. (2):1–8, 2014. doi: 10.1145/2660859.2660927.

[125] Suzan Verberne, Hans van Halteren, Daphne Theijssen, Stephan Raaijmakers, and Lou Boves. Learning to rank for why-question answering. *Information Retrieval*, 14(2):107–132, Apr 2011. ISSN 1573-7659. doi: 10.1007/s10791-010-9136-6. URL https://doi.org/10.1007/s10791-010-9136-6.

[126] Ellen M Voorhees. The Evaluation of Question Answering Systems: Lessons Learned from the TREC QA Track. Technical report. URL https://pdfs.semanticscholar.org/ba1d/7d0a8934ec960ac6451fb3c48657b3b21743.pdf.

[127] Ellen M. Voorhees. Overview of the trec 2004 robust retrieval track. In *In Proceedings of the Thirteenth Text REtrieval Conference (TREC2004*, page 13, 2004.

[128] Ellen M. Voorhees. The trec 2005 robust track. *SIGIR Forum*, 40(1): 41–48, June 2006. ISSN 0163-5840. doi: 10.1145/1147197.1147205. URL http://doi.acm.org/10.1145/1147197.1147205.

[129] Ellen M Voorhees and Dawn M Tice. Building a Question Answering Test Collection. Technical report. URL www.mS.crosofe.

[130] Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 200–207, New York, NY, USA, 2000. ACM. ISBN 1-58113-226-3. doi: 10.1145/345508.345577. URL http://doi.acm.org/10.1145/345508.345577.

[131] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer, 1999.

[132] Ellen M Voorhees et al. Overview of the trec 2005 robust retrieval track. In *Trec*, 2005.

[133] Jacques Wainer and Gavin Cawley. Nested cross-validation when selecting classifiers is overzealous for most practical applications. Technical report. URL https://arxiv.org/pdf/1809.09446.pdf.

[134] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2835–2841. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3016100.3016298.

[135] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1122. URL https://www.aclweb.org/anthology/P16-1122.

[136] Haocheng Wu, Wei Wu, Ming Zhou, Enhong Chen, Lei Duan, and Heung-Yeung Shum. Improving search relevance for short queries in community question answering. In *Proceedings of the 7th ACM international conference on Web search and data mining - WSDM '14*, pages 43–52, New York, New York, USA, 2014. ACM Press. ISBN 9781450323512. doi: 10.1145/2556195.2556239. URL http://dl.acm.org/citation.cfm?doid=2556195.2556239.

[137] Qiang Wu, Christopher J.C. Burges, Krysta M. Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, jun 2010. ISSN 13864564. doi: 10.1007/s10791-009-9112-1. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.5117{&}rep=rep1{&}type=pdfhttp://link.springer.com/10.1007/s10791-009-9112-1.

[138] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510, 2009.

[139] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[140] Linjun Yang, Bo Geng, Alan Hanjalic, and Xian-Sheng Hua. Contextual image retrieval model. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, pages 406–413, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0117-6. doi: 10.1145/1816041.1816100. URL http://doi.acm.org/10.1145/1816041.1816100.

[141] Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 287–296, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4073-1. doi: 10.1145/2983323.2983818. URL http://doi.acm.org/10.1145/2983323.2983818.

[142] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256. ACM, 2017.

[143] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the" neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. *arXiv preprint arXiv:1904.09171*, 2019.

[144] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271800. URL https://doi.org/10.1145/3269206.3271800.

[145] Chengxiang Zhai. Notes on the lemur tfidf model. *Unpublished report*, 2001.

[146] ChengXiang Zhai. Statistical Language Models for Information Retrieval A Critical Review. *Foundations and Trends® in Information Retrieval*, 2(3):137–213, 2008. ISSN 1554-0669. doi: 10.1561/1500000008. URL http://dx.doi.org/10.1561/1500000008.

[147] Yanzhen Zou, Ting Ye, Yangyang Lu, John Mylopoulos, and Lu Zhang. Learning to rank for question-oriented software text retrieval. In *Proceedings of the 30th IEEE/ACM International Conference on Automated*

Software Engineering, ASE '15, pages 1–11, Piscataway, NJ, USA, 2015. IEEE Press. ISBN 978-1-5090-0024-1. doi: 10.1109/ASE.2015.24. URL https://doi.org/10.1109/ASE.2015.24.

Appendix A

Robust04 cross-validation results

In this appendix we present the test effectiveness results for BM25, BM25+extra, LambdaMART+default, and LambdaMART+HPO with five-fold cross-validation on Robust04.

IR model	Fold	MAP	P@20	NDCG@20
BM25	s1	0.2416	0.3470	0.4188
	s2	0.2114	0.3112	0.3748
	s3	0.2130	0.3280	0.4081
	s4	0.2521	0.3620	0.4337
	s5	0.2552	0.3840	0.4474
	Mean (±std)	0.2347 (±0.02)	0.3464 (±0.03)	0.4166 (±0.03)
BM25+extra	s1	0.2442	0.3530	0.3950
	s2	0.2008	0.3143	0.3402
	s3	0.1993	0.3420	0.3832
	s4	0.2447	0.3640	0.3975
	s5	0.2287	0.3490	0.3783
	Mean (±std)	0.2235 (±0.02)	0.3445 (±0.02)	0.3788 (±0.02)
LambdaMART+ default	s1	0.2371	0.3340	0.4085
	s2	0.1807	0.2918	0.3475
	s3	0.1608	0.2940	0.3482
	s4	0.2740	0.3810	0.4542
	s5	0.2421	0.3770	0.4426
	Mean (±std)	0.2189 (±0.05)	0.3356 (±0.04)	0.4002 (±0.05)
LambdaMART+ HPO	s1	0.2464	0.3470	0.4184
	s2	0.2051	0.3071	0.3681
	s3	0.2016	0.3320	0.3990
	s4	0.2817	0.4010	0.4688
	s5	0.2620	0.3950	0.4542
	Mean (±std)	**0.2394 (±0.04)**	**0.3564 (±0.04)**	**0.4217 (±0.04)**

Table A.1: Test effectiveness results of four IR models with five-fold cross validation on Robust04. The effectiveness results of the most effective model are in bold numbers.

Appendix B

Paper A deep neural IR model trials

In this appendix we present the results of the five trials with the deep neural IR model from paper A, POSIT-DRMM+MV.

Run number	Random seed	MAP	P@20	NDCG@20
1	571150660	0.4743	0.2625	0.5672
2	3666550821	0.0692	0.0303	0.0598
3	2967494122	0.4562	0.2521	0.5506
4	**395698531**	**0.4763**	**0.2617**	**0.5701**
5	606917781	0.0553	0.0115	0.0243
	Mean (±std)	0.3063 (±0.2229)	0.1636 (±0.1305)	0.3544 (±0.2855)

Table B.1: Test effectiveness results of five trials of POSIT-DRMM+MV with different random seeds on BioASQ. Best achieved run in terms of MAP in bold numbers.

Appendix C

HPO process results

In this appendix we present the results for the different HPO methods' runs applied to LambdaMART trained with the datasets of papers A and B. A *function evaluation* starts with the training of LambdaMART with a particular hyperparameter configuration until it is evaluated on the validation set of each dataset (and for each one of the five CV folds in the case of Robust04). We measure the elapsed wallclock-time for the 100 function evaluations of each HPO method run. Note that the wallclock-time variations among HPO runs of the same method for the same dataset is mostly due to the different amount of time it takes to evaluate LambdaMART with different hyperparameters and. BOHB tends to spend less time because of the smaller training budgets that are tried first. The use of smaller training budgets also causes that BOHB starts the HPO process with a smaller effectiveness than RS.

Below we present the individual tables with the values and figures with the learning curves for each HPO method applied to each one of the datasets of papers A and B.

Run	**Initial**				**Optimized$_{100}$**				**wclock$_{100}$**
#	***K***	ρ	***T***	**MAP**	***K***	ρ	***T***	**MAP**	**(min)**
1	41	0.18	209	0.1847	20	0.13	590	0.2403	408.95
2	50	0.43	100	0.1671	20	0.12	350	0.2400	458.48
3	95	0.18	1400	0.1933	80	0.06	1800	0.2399	820.60
4	100	0.24	800	0.1719	25	0.08	1950	0.2428	408.52
5	95	0.10	1450	0.1860	**25**	**0.03**	**450**	**0.2465**	502.33
			Mean	0.1806			Mean	0.2419	519.78
			(±std)	(±0.01)			(±std)	(±0.00)	(±172.64)

Table C.1: HPO results of five BOHB runs applied to LambdaMART trained with Robust04 on validation set with five-fold CV to optimize MAP. Each run consisted of 100 function evaluations. **wclock** indicates the elapsed wallclock-time of each run expressed in minutes. The optimized hyperparameter configuration and the corresponding achieved MAP are in bold numbers.

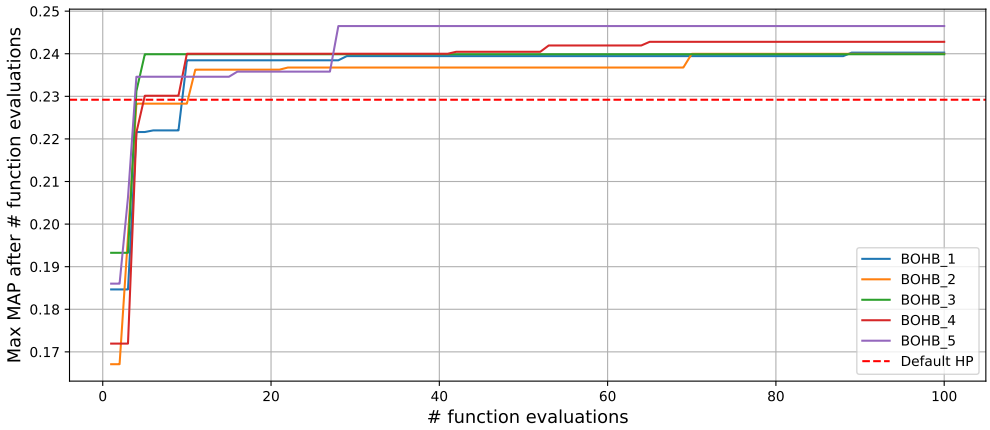


Figure C.1: HPO learning curves BOHB for LambdaMART trained with Robust04 on validation set with five-fold CV to optimize MAP.

Run	**Initial**				**Optimized**$_{100}$				**wclock**$_{100}$
#	**K**	ρ	**T**	**MAP**	**K**	ρ	**T**	**MAP**	**(min)**
1	70	0.45	200	0.1992	**35**	**0.03**	**1500**	**0.2443**	519.08
2	5	0.41	950	0.2095	75	0.03	1400	0.2398	479.75
3	40	0.28	1500	0.2191	35	0.03	150	0.2439	686.04
4	5	0.24	100	0.1751	25	0.09	1600	0.2398	658.44
5	22	0.54	1858	0.2007	70	0.1	1342	0.2398	481.03
			Mean	0.2007			Mean	0.2415	564.87
			(±std)	(±0.02)			(±std)	(±0.00)	(±99.76)

Table C.2: HPO results of five RS runs applied to LambdaMART trained with Robust04 on validation set with five-fold CV to optimize MAP. Each run consisted of 100 function evaluations. **wclock** indicates the elapsed wallclock-time of each run expressed in minutes. The optimized hyperparameter configuration and the corresponding achieved MAP are in bold numbers.

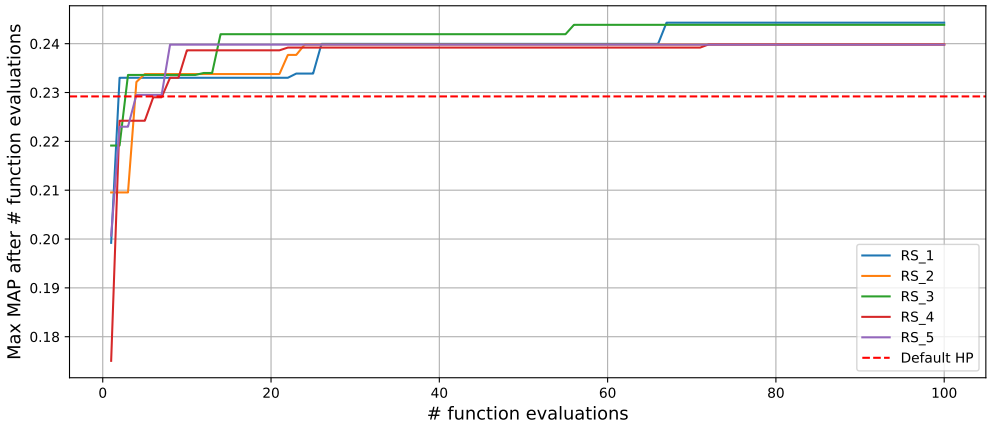


Figure C.2: HPO learning curves RS for LambdaMART trained with Robust04 on validation set with five-fold CV to optimize MAP.

Run	Initial				Optimized$_{100}$				wclock$_{100}$
#	***K***	ρ	***T***	**MAP**	***K***	ρ	***T***	**MAP**	**(min)**
1	25	0.04	1700	0.2966	55	0.16	150	0.4431	76.97
2	191	0.32	2877	0.3591	8	0.55	2093	0.4475	40.3
3	60	0.05	1300	0.4087	90	0.13	1750	0.4492	103.01
4	45	0.31	1950	0.3941	55	0.16	200	0.4465	57.26
5	25	0.24	450	0.4281	**15**	**0.07**	**1400**	**0.4518**	122.29
			Mean	0.3773			Mean	0.4476	79.97
			(±std)	(±0.05)			(±std)	(±0.00)	(±33.24)

Table C.3: HPO results of five BOHB runs applied to LambdaMART trained with BioASQ on validation set to optimize MAP. Each run consisted of 100 function evaluations. **wclock** indicates the elapsed wallclock-time of each run expressed in minutes. The optimized hyperparameter configuration and the corresponding achieved MAP are in bold numbers.

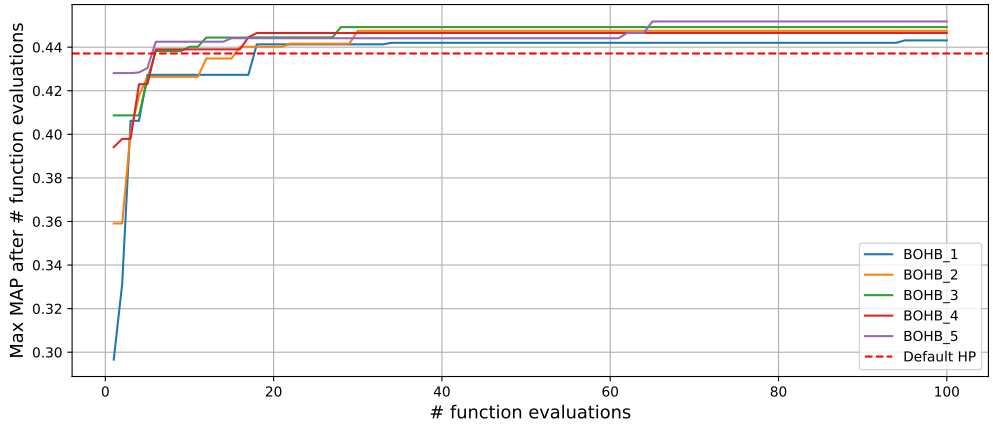


Figure C.3: HPO learning curves BOHB for LambdaMART trained with BioASQ on validation set to optimize MAP.

Run	Initial				Optimized$_{100}$				wclock$_{100}$
#	***K***	ρ	***T***	**MAP**	***K***	ρ	***T***	**MAP**	**(min)**
1	45	0.06	150	0.4271	85	0.07	1450	0.4457	109.7
2	75	0.08	100	0.4176	**90**	**0.13**	**1650**	**0.4492**	111.69
3	5	0.46	400	0.4340	5	0.16	1150	0.4480	115.14
4	40	0.29	750	0.4200	55	0.06	1750	0.4451	80.34
5	65	0.50	1350	0.4128	10	0.45	1050	0.4450	73.39
			Mean	0.4223			Mean	0.4466	98.05
			(±std)	(±0.01)			(±std)	(±0.00)	(±19.59)

Table C.4: HPO results of five RS runs applied to LambdaMART trained with BioASQ on validation set to optimize MAP. Each run consisted of 100 function evaluations. **wclock** indicates the elapsed wallclock-time of each run expressed in minutes. The optimized hyperparameter configuration and the corresponding achieved MAP are in bold numbers.

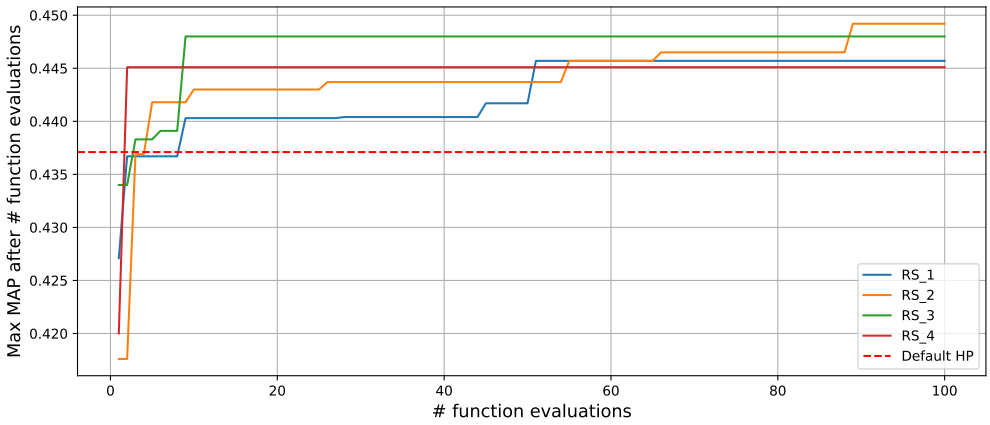


Figure C.4: HPO learning curves RS for LambdaMART trained with BioASQ on validation set to optimize MAP.

Run	**Initial**				**Optimized**$_{100}$				**wclock**$_{100}$
#	***K***	ρ	***T***	**MAP**	***K***	ρ	***T***	**MAP**	**(min)**
1	10	0.20	500	0.5554	5	0.44	1350	0.5591	190.72
2	60	0.08	1950	0.5521	45	0.11	1200	0.5580	308.21
3	90	0.03	1100	0.5488	5	0.44	600	0.5591	252.14
4	10	0.18	600	0.5537	45	0.19	1000	0.5587	210.42
5	85	0.01	1600	0.5478	**5**	**0.44**	**450**	**0.5590**	244.14
			Mean	0.5516			Mean	0.5588	241.13
			(±std)	(±0.00)			(±std)	(±0.00)	(±45.04)

Table C.5: HPO results of five BOHB runs applied to LambdaMART trained with TVQA on validation set to optimize accuracy. Each run consisted of 100 function evaluations. **wclock** indicates the elapsed wallclock-time of each run expressed in minutes. The optimized hyperparameter configuration and the corresponding achieved accuracy are in bold numbers.

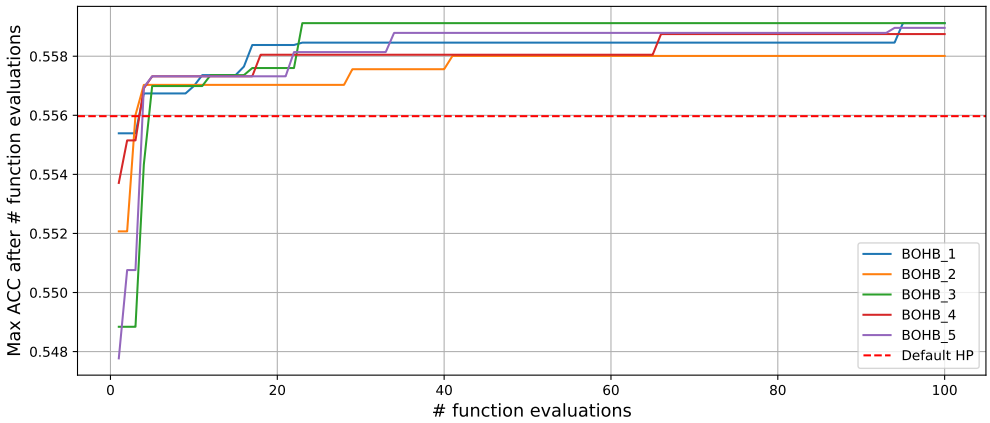


Figure C.5: HPO learning curves BOHB for LambdaMART trained with TVQA on validation set to optimize accuracy.

Run	**Initial**				**Optimized$_{100}$**				**wclock$_{100}$**
#	***K***	ρ	***T***	**MAP**	***K***	ρ	***T***	**MAP**	**(min)**
1	60	0.09	1550	0.5572	**10**	**0.23**	**900**	**0.5584**	410.80
2	55	0.01	1050	0.5511	45	0.14	650	0.5583	373.21
3	35	0.09	1350	0.5566	50	0.06	900	0.5580	457.47
4	100	0.39	1450	0.5535	45	0.11	1150	0.5580	433.58
5	100	0.16	350	0.5544	10	0.47	450	0.5583	417.61
			Mean	0.5546			Mean	0.5582	418.53
			(±std)	(±0.00)			(±std)	(±0.00)	(±31.05)

Table C.6: HPO results of five RS runs applied to LambdaMART trained with TVQA on validation set to optimize accuracy. Each run consisted of 100 function evaluations. **wclock** indicates the elapsed wallclock-time of each run expressed in minutes. The optimized hyperparameter configuration and the corresponding achieved accuracy are in bold numbers.

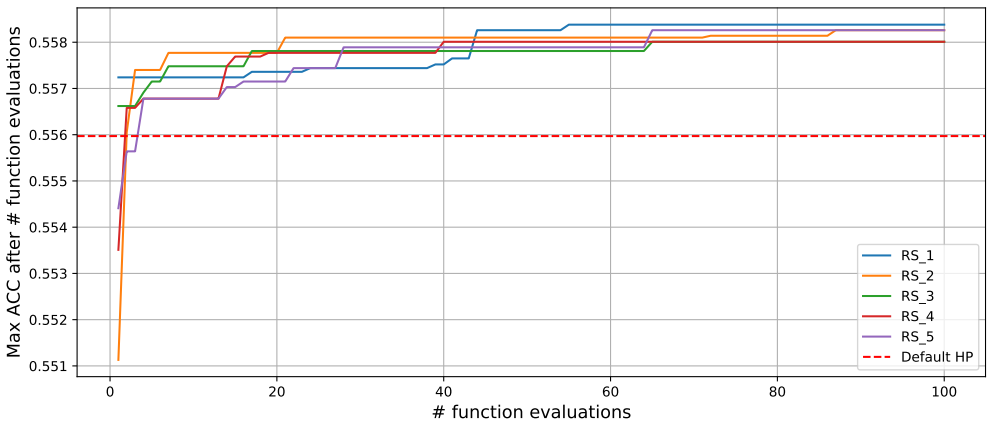


Figure C.6: HPO learning curves RS for LambdaMART trained with TVQA on validation set to optimize accuracy.