Leipzig University
Institute of Computer Science
Degree Programme Computer Science, B.Sc.

Exploiting Pairwise Neural Networks to Improve Early Precision in High-Recall Literature Search

Bachelor's Thesis

Thomas Abel
Born Nov. 30, 1992 in Pfaffenhofen a. d. Ilm

Matriculation Number 3756586

1. Referee: Jun.-Prof. Dr. Martin Potthast

Submission date: March 27, 2023

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Leipzig, March 27, 2023

. . . .
Thomas Abel

Abstract

High recall searches aim to find all documents relevant to some information need which is important to ensure completeness in systematic reviews or related work search. Pairwise retrieval models have been shown to be more effective than pointwise ones in ad-hoc retrieval, though this hasn't been studied in high recall searches because of the quadratic overhead of comparing all the documents. Recent work framed preference judgements as a Dueling Bandits problem and suggested some algorithms that showed high potential in finding the best item while having an acceptable cost. In this thesis we will apply these algorithms on a pointwise ranking approach to investigate if it is possible to obtain a high recall ranking with similar efficiency. We will then evaluate our approaches on the standard Information Retrieval metrics to see how they perform.

Contents

Chapter 1

Introduction

One of the fundamental problems in Information Retrieval and Machine Learning is *Ranking*. Ranking means the ordering of items according to some criterion [Mohri et al., 2012], which is essential for example in search engines, for recommender systems or in document retrieval, where the task is to find an order of documents $d_1,...,d_n$ according to their relevance to a given query q. To obtain such a ranking it is necessary to check if a document is relevant to the query, a very early and easy method to do so is called Boolean Retrieval, where it is only checked if the document contains the query or not [Manning et al., 2008]. For a more refined ranking it is necessary to know to which degree a document is relevant, so some kind of scoring function is needed. To measure this also a better understanding of the underlying document text is needed to determine the relevancy of a query. Therefore computers need a representation of this text to be able to process it. One method to interpret text is to see it as a bag-of-words, which means just seeing it as a set of its words and not paying attention to other properties such as the position of these words relative to each other. With such a representation more properties like term frequency (TF) and inverse document frequency (IDF), which indicate how often a term appears in a document and in how many documents, can be taken into account [Jones, 2004]. An early algorithm for document ranking applying these concepts is called *Okapi BM25* [Robertson et al., 1994]. It is a probabilistic ranking model, which computes a score based on this numerical statistic called TF-IDF. BM25 was also used in the experiments of this thesis and will be further explained later. To represent the document text and process the information contained in it is called Natural Language Processing (NLP) and relies on the use of artificial intelligence and deep learingn [Nadkarni et al., 2011]. The task of constructing ranking models using training data is called *Learning to Rank*. There are different approaches to score documents, which can be generally categorized into three approaches for algorithms, namely *pointwise*, *pairwise* and *listwise* approaches [Liu, 2009]. Pointwise algorithms define the score on each single item, while pairwise algorithms define it on each pair of items and listwise ones on a list of items. Over the years many ideas have been proposed taking up this topic using more and more sophisticated approaches. Modern approaches to NLP also try depict the context of words and their similarity and relations in a text. This can be achieved by representing words as vectors. A technique to accomplish these so called word embeddings is called Word2vec [Mikolov et al., 2013] which uses neural networks to do so. State-of-the-art models use the so called Transformer, a deep learning model architecture which uses mechanisms called attention and self-attention to achieve exceptional language modeling capabilities [Vaswani et al., 2017].

1.1 Goals

Recent pointwise and pairwise LTR approaches apply these transformers in document ranking tasks [Pradeep et al., 2021]. We want to use the models called monoT5 and duoT5 to perform ranking task on our own dataset, the Information Retrieval Anthology. Since the IR Anthology is a corpus consisting of scientific papers and one use case for it is the related work search we want to evaluate the results according to the relevant metrics for this task, namely recall and precision. The IR Anthology does not contain any judged papers yet, so to measure our results we want to judge the papers retrieved from our ranking pipelines along the way and provide the queries and judgments for future work on this dataset. We will especially focus on the duoT5 model. Since it uses a pairwise ranking approach, the number of comparisons it has to make increases quadratically with the number of input documents. We frame this as a Dueling Bandit problem and apply four algorithms proposed in [Yan et al., 2022]. We want to see if we can reduce the number of comparisons that have to be made and how these algorithms perform when compared to the other retrieval pipelines used.

1.2 Research Questions

The research question we therefore derived are:

ResearchQuestion1 *Can we reduce the number of comparisons that have to be made when using the proposed Dueling Bandit algorithms for retrieval compared duoT5?*

ResearchQuestion2 *How do these algorithms perform on the measured metrics compared to duoT5?*

To answer these questions we designed some retrieval pipelines with different settings to get comparable results. The following chapters will describe the process. In Chapter 2 we will give a overview of the field this thesis is set in, in Chapter 3 we will describe the methods and algorithms we used in detail, in Chapter 4 we will describe our experimental setup and how we processed the data we worked with, in Chapter 5 we will present and discuss the results we obtained and in Chapter 6 we will draw our conclusion and describe future work.

Chapter 2

Related Work

In this chapter we will give an overview of the field this thesis is contributing to. We will first give an overview of LTR models, then we will show how the Dueling Bandit approach is used in similar settings before we finally show an overview of other approaches to high recall search.

2.1 Learning-To-Rank with Transformers

Since the introduction of the transformer architecture [Vaswani et al., 2017], it has been applied to many IR tasks including document ranking, where the use of these models has drastically improved the effectiveness compared to traditional models [Lin, 2019]. Especially the release of BERT [Devlin et al., 2019], which is a family of language models based on the transformer architecture, has marked a new era in neural ranking models. Transformers can be used in sparse as well as in dense retrieval methods. In sparse retrieval a vector representation of text, like in a bag-of-words model, has few non-zero elements, whereas in dense retrieval these vectors have more such elements, hense they are more dense [Luan et al., 2021]. A example is DeepCT [Dai and Callan, 2019] which learns to map the text representations of BERT to certain term weights for text. These weights can be stored in a simple inverted index, which can then be used by BM25 for example. Document expansion [Nogueira et al., 2019c] is another example, where transformers are used to predict queries which fit the text and then append them to it, before it gets indexed. The Dense Passage Retriever [Karpukhin et al., 2020] is one of the earlier models for dense retrieval which use large transformer-based models. It uses a dual-encoder based on BERT to learn dense representations of documents and queries to compare them and set a new state-of-the-art in passage retrieval. There are also hybrid models which combine sparse and dense concepts [Luan et al., 2021]. Other approaches to these dual-encoders are so called cross-encoders Reimers and Gurevych [2019]. They do not produce sentence-embeddings to compute similarity between texts. They pass two texts to a transformer simultaneously and produce an output between 0 and 1 indicating the similarity. In [Lin et al., 2020a] an overview of different existing approaches is shown.

A pointwise ranking approach using BERT is for example described in [Nogueira and Cho, 2019], where the authors feed a passage and a query into a simple re-implementation of BERT to compute the probability of it being relevant. The same authors [Nogueira et al., 2019a] later framed this approach as monoBERT and introduced duoBERT in the same paper, a pairwise ranking approach. There are many more approaches using BERT, which we will not describe here in detail [Yang et al., 2019][MacAvaney et al., 2019].

The pairwise approach duoBERT takes in a query and a docuemnt tuple. It then creates three embeddings, one for the query and one for each documents, before computing a probability that the first document is more relevant to the query than the second. A similar pairwise approach was introduced in [Pradeep et al., 2021] which use T5 as a transformer model [Raffel et al., 2019]. Here an algorithm called duoT5 is introduced, together with monoT5, another pointwise approach. These algorithms are used in the experiments of this thesis and will be further explained in a later chapter.

2.2 Improving Efficiency of Transformer-Based Rankers

Transformers are computationally very expensive. But in opposite to increasing their performance, the optimization of their runtime has not been a quite as popular task [Schwartz et al., 2019]. Tackling this by analyzig the runtime of these models has recently gained attention [Hofstätter and Hanbury, 2019]. There are basically two possibilities for improving the effectiveness of transformer-based rankers, one is to actively improve the efficiency of the model itself and the other is to reduce the number of inputs such a model has to process [Gienapp et al., 2022]. For the former approach [MacAvaney et al., 2020] proposed the idea of computing part of the term representation of the document at index time without a query. Another proposed idea for this approach is to use model distillation approaches, which means training a smaller model off of a larger model [Sanh et al., 2019] and improve speed and size while maintaining effectiveness. Decreasing the number of input documents can be done by applying multi-stage ranking architectures as the authors of [Zhang et al., 2021] proposed.

An approach targeting pairwise ranking models is to reduce the number of comparisons by using sparse comparison sets that consider only a subset of all possible document pairs. By using different sampling and aggregation methods, an order of magnitude decrease in comparisons with the best combination could be achieved [Gienapp et al., 2022]. By assuming properties like transitivity best-worst choice models could be applied for pairwise models [Marley and Louviere, 2005]. From a set of $n>2$ documents, the best and the worst item are selected from which one can conclude information about the order of the rest of the items. Using enough of these choices and combining them with statistical models can lead to deriving a ranking for these documents.

2.3 Dueling Bandits

A different approach to improve efficiency is to try to frame the pairwise method as a Dueling Bandits problem [Yan et al., 2022]. Here the authors compare different algorithms to find the best item in a pool of n "arms" through noisy pairwise comparisons between them. We will use this approach and the algorithms provided in [Yan et al., 2022] in this thesis. Dueling Bandits have the advantage that they are simple to implement and do not require training data, which is why they are used in online personalization, online ranker evaluation and search engine optimization [Glowacka, 2019]. They can also be used to optimize information retrieval systems by using implicit feedback from user behaviour [Yue and Joachims, 2009].

Chapter 3

Method

In this chapter we will introduce the different algorithms and tools that we used to perform the experiments. Their functionality will be explained and how they have been adjusted to fit the requirements of this thesis.

3.1 IR Anthology

The Information Retrieval Anthology or IR Anthology is a document corpus consisting of scientific publications on the subject of information retrieval [Potthast et al., 2021]. Published in 2021 it contains 61,097 papers (as of 21.03.2023) from the most important conferences and journals from this field. The papers from this corpus are used to conduct the experiments on. [1]

The raw data from the IR Anthology we used contained 53,673 documents and where retrieved from the Ceph cluster from the Webis Group, which is only accessible from within the Webis network or via VPN. The CephFS path to the file is `/mnt/ceph/storage/data-in-progress/data-research/web-search/SIGIR-21/sigir21-chatonir/es-data-bulk-index.ldjson` and a copy was saved to `/mnt/ceph/storage/data-in-progress/data-teaching/theses/thesis-abel/` which contains the actual version as of 18.01.2023. In 3.1 is an overview of all the data and metadata contained in one document of the dataset. The raw data is in a .ldjson format which stands for line-delimited JSON and means that every line in the file represents a JSON object. To meet the requirements of the framework ir_datasets, the data had to be parsed into a suitable format, since the doc_id field is in a separate line before the other fields and some field names do not meet the conditions of the framework. In this parsing process the three fields title_abstract, title_full_text and ti-

[1]The IR Anthology can be found at `https://ir.webis.de`

tle_text_only were added. These fields are used in the transformer pipelines to compute some scores which will be described in detail in the corresponding section.

3.2 ir_datasets and pyterrier

The python framework ir_dataset [MacAvaney et al., 2021] is a powerful tool which makes it possible to access certain datasets with python code and work with them. Some of these datasets even come with queries and qrels, so they can be easily used for information retrieval experiments. The term qrels was made popular by the Text Retrieval Conference, short TREC, and basically means relevance judgments as to how relevant a document is to a query. ir_datasets also offers integrations into other popular indexing and experimentation toolkits, like pyterrier, which is why it was used in this thesis. As stated above the data from the anthology was parsed into a .jsonl file format to fit the requirements of ir_datasets and could then be loaded with a built-in function to integrate it into this framework. The same process was used to integrate the queries and qrels, but they were provided in the TREC format.[2]

Another important framework used in this thesis is pyterrier [Macdonald and Tonellotto, 2020]. It provides implementations of different information retrieval algorithms and tools for indexing datasets and running and evaluating experiments with these algorithms on them. In this thesis we used the indexing functionality and the implementations of BM25, MonoT5 and DuoT5. The integration from ir_datasets into pyterrier is very simple and can be conducted with one statement. Indexing the dataset is similarly simple with pyterrier and is needed to compute the scores of the BM25 algorithm.

3.3 Algorithms

In this section we will describe how the algorithms used in this thesis work and how we implement them to conduct the experiments. We will also explain which changes where made to the original implementations.

3.3.1 BM25

BM25 [Robertson et al., 1994] uses a bag-of-words model of the underlying document, which means it is represented only as a set of its individual words,

[2]`http://www.cs.cmu.edu/~lemur/LemurGuide.html#data`

dropping certain information like word order. A score is then computed based on the frequency of the query terms in the document and on some relative properties like the length of the document divided by the average document length as shown in the formula in 3.1. [3]

$$BM25(D,Q) = \sum_{i=1}^{n} ln(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1) * \frac{f(q_i, D) * (k_1 + 1)}{f(q_i, D) + k_1 * (1 - b + b * \frac{|D|}{avgdl})} \tag{3.1}$$

As stated above an implementation of this algorithm is contained in pyterrier and was used in this thesis as part of the retrieval pipeline.

3.3.2 MonoT5 and DuoT5

MonoT5 and duoT5 are based on a pretrained sequence-to-sequence transformer called T5 [Pradeep et al., 2021]. T5, which stands for Text-to-Text Transfer Transformer, is an architecture for transformers [Vaswani et al., 2017] used in natural language processing (NLP). The idea behind it is to frame all NLP tasks into a unified unified text-to-text-format, so that the input and the output are text strings and not for example some class labels [Raffel et al., 2019]. To accomplish this, the model is first pre-trained on a large dataset, before it is fine-tuned on a specific downstream task.

In the case of monoT5 a pointwise approach called relevance classificaction is used [Nogueira et al., 2020][Lin et al., 2020b]. Using the prompt shown in 3.2, the models task to predict whether a document is relevant to a given query or not.

$$\text{Query: } q \text{ Document: } d \text{ Relevant:} \tag{3.2}$$

It is therefore fine-tuned to predict the tokens "true" or "false". The score for the document is then computed as the softmax of the logits assigned to the "true" and "false" tokens.

DuoT5 uses a pairwise approach and the idea behind it initially comes from duoBERT [Nogueira et al., 2019b]. It predicts the probability that document d$_i$ is more relevant than document d$_j$ to a given query. Similar to monoT5 the model uses the prompt in 3.3 to predict a token "true" or "false". Then the probabilities of the prediction of the "true" token for every document pair are aggregated and set as score. There are different aggregation methods, the most simple one, which is also used in the implementation of duoT5 used in

[3]`http://terrier.org/docs/v4.0/javadoc/org/terrier/matching/models/BM25.html`

this thesis, is to just sum all of the probabilities. This implementation of token prediction has the effect that the probablities of two corresponding document tuple d_{ij} and d_{ji} do not add up to 1 as one might expect, therefore n*(n-1) probabilities have to be computed.

$$\text{Query: } q \text{ Document0: } d_i \text{ Document1: } d_j \text{ Relevant:} \tag{3.3}$$

Both algorithms are implemented in the pyterrier framework and use a dataset of the MS MARCO collection [Nguyen et al., 2016]. MonoT5 was used as it is published in the corresponding Github repository, duoT5 had to be modified. The first modification made was to add a functionality to save the individual scores of each document pair in a file, so the following Dueling Bandit algorihtms could work with them. Another adjustment that had to be made was to truncate the tokens generated from the document texts, since there was not enough memory capacity available in the Google Colab working environment. [4]

3.3.3 Dueling Bandits

The Multi Armed Bandit problem is a stochastical problem which describes the exploration–exploitation tradeoff dilemma. The name originates in the idea of a gambler at a casino who has to choose to play any of n one-armed bandit machines. To maximize his reward the gambler needs to come up with strategies to decide on which bandit machine to play based on prior information and observation [Weber, 1992]. The Dueling Bandits problem is an extension of MAB [Yue et al., 2009]. Here a gambler (or algorithm) has to choose only two one-armed bandit machines for a comparison (or duel). In our case the machines are an analogy for our documents and the result of such a duel is a binary outcome whether one or the other document wins the duel.

3.3.4 Double Thompson Sampling

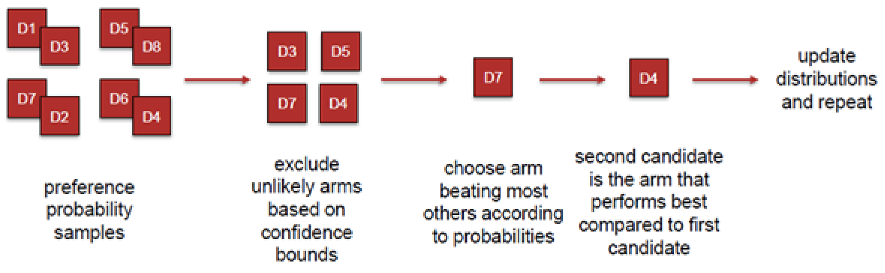
The first of the Dueling Bandit algorithms we want to introduce in this thesis is called Double Thompson Sampling [Wu and Liu, 2016]. As the name hints the two items for a duel are selected by Thompson sampling [Thompson, 1933]. For each pair of items d$_i$ and d$_j$ a beta prior distribution is assumed for the probability of p_{ij}, which is the probability that d$_i$ is preferred over d$_j$. Depending on the outcomes of the duels these distributions are then updated. At each iteration step, the algorithm generates a candidate set by dropping items that are unlikely to be the best item using confidence bounds. Then the

[4] `https://github.com/terrierteam/pyterrier_t5`

first candidate for a duel is selected as the item in this sets which beats most of the other ones according to the sampled probabilities. The second candidate is selected similarly but from the items which perform best compared to the first candidate. In 3.1 the procedure is illustrated.

The algorithm returns only the best item as it is described in the paper. The authors of [Li et al., 2020] provide an implementation of DTS in the Github repository to their paper which was used as a basis in this thesis [5]. We modified the algorithm to remove the returned best document from its input documents, so we could re-run it again on this updated input set. By doing so we could retrieve a ranking from the algorithm. Apart from that the algorithm was not changed in its behaviour and we use the same parameters the authors suggest for the best performance. As suggested in [Yan et al., 2022] we use 1000 as the number of iterations.

Figure 3.1: Double Thompson Sampling


D1
D3
D5
D8
D7
D2
D6
D4
preference probability samples
D3
D5
D7
D4
exclude unlikely arms based on confidence bounds
D7
choose arm beating most others according to probabilities
D4
second candidate is the arm that performs best compared to first candidate
update distributions and repeat

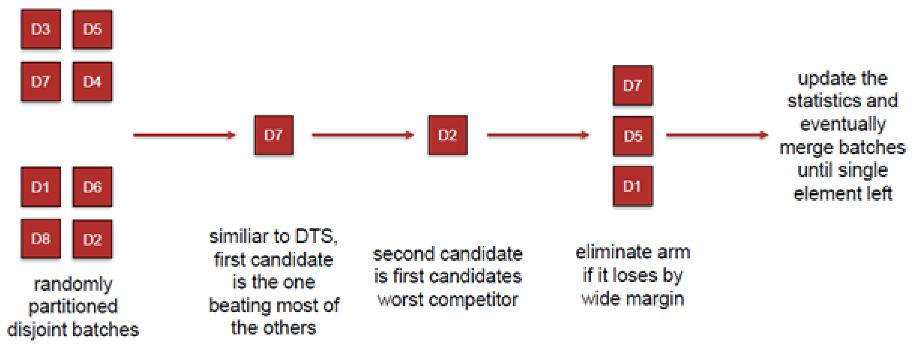
3.3.5 Merge Double Thompson Sampling

The next algorithm, called Merge Double Thompson Sampling, works very similarly to the first one. It uses a divide-and-conquer strategy in addition to Double Thompson Sampling [Li et al., 2020]. In a first step the items are randomly partitioned into disjoint batches. At each iteration a batch is then selected and items that lose to other items within this batch by a wide margin are purged. If the batch then contains a single element it is merged with another batch. By using Thompson Sampling the best candidate from the remaining set is chosen. In contrast to DTS the second candidate of this batch is then selected from the items which perform the worst against the first candidate. This is done to eliminate unlikely winners as fast as possible. The graphic in 3.2 illustrates the process again.

[5]`https://github.com/chang-li/MergeDTS`

This algorithm also only returns the best item. So using the same strategy as in DTS we again excluded the best item to re-run the algorithm on the remaining set and retrieve a ranking. The authors also provided an implementation of MergeDTS in the same repository as for DTS, which was used as basis for our modifications. 1000 was again chosen as the number of iterations and the other parameters according to the suggestions of the authors for best performance.

Figure 3.2: Merge Double Thompson Sampling


D3
D5
D7
D4
D1
D6
D8
D2
randomly partitioned disjoint batches
D7
similiar to DTS, first candidate is the one beating most of the others
D2
second candidate is first candidates worst competitor
D7
D5
D1
eliminate arm if it loses by wide margin
update the statistics and eventually merge batches until single element left

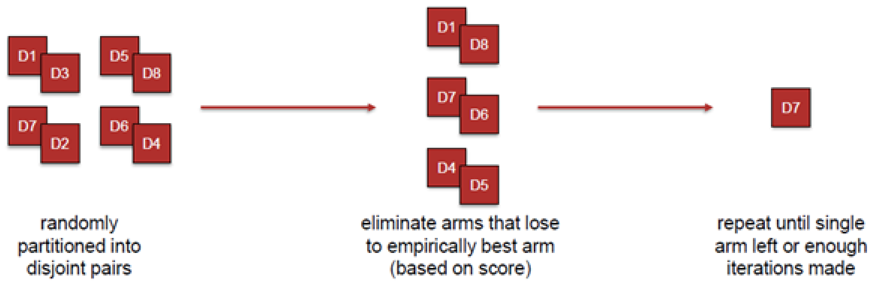
3.3.6 Round Efficient Dueling Bandits

The third algorithm called Round-Efficient Dueling Bandits works in a tournament-like fashion [Lin and Lu, 2018]. First all of the items are randomly partitioned into disjoint pairs. These pairs then have to duel and the individual items update their scores based on the outcome. After this the items, which lose to the item with the best score by a margin based on some confidence interval, are eliminated. Then the remaining pairs are again partitioned into pairs and this process is repeated until a single arm is left or the specified number of iteratios was reached. The process is illustrated in 3.3.

The authors of [Yan et al., 2022] provide an implementation [6] of this algorithm which we use as a basis. It was again modified to return a list of items instead of just the best one. The parameters were also kept the same as in the base implementation and again 1000 iterations were made for each call of the algorithm.

[6]https://github.com/xinyiyan/duelingbandits

Figure 3.3: Round Efficient Dueling Bandits


D1
D3
D5
D8
D7
D2
D6
D4
randomly partitioned into disjoint pairs
D1
D8
D7
D6
D4
D5
eliminate arms that lose to empirically best arm (based on score)
D7
repeat until single arm left or enough iterations made

3.3.7 Clarke et al.

In their original paper the authors present their algorithm in the context of preference judgments and do not frame it as a Dueling Bandit approach [Clarke et al., 2020]. However the authors of [Yan et al., 2022] adapted it and considered it as a solution to the Dueling Bandits problem. The algorithm consists of different phases. In the first phase each item is paired with a certain number of other items (when modeled as a graph with the items being the vertices, each vertex has to have outdegree n). These pairings then perform duels in the next phase and get a score based on the outcome. Then each item with score less than a specified value gets pruned. This process is repeated until a certain threshold for the number of items is reached. In the last phase the remaining items get paired with each other item to form a complete graph, then again perform duels and get a score according to which they get then ranked. The figure 3.4 illustrates this algorithm again.

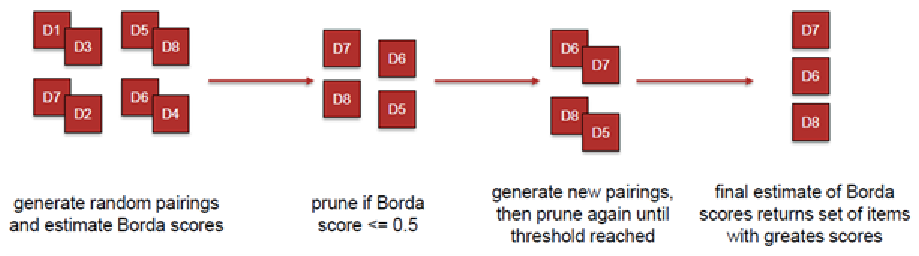
This algorithm actually returns a list of items. With the suggested parameters from the authors this returned list is at most of length 9 so to get the desired top 10 ranking out of it, it was modified so that the algorithm gets called again on the set of the remaining documents, which were not in the returned list, until the desired number was reached. The basis of our implementation is the code provided in the Github repository of the authors. [7]

3.3.8 Pure Random

This algorithm was designed to see how efficient the sampling of the Dueling Bandit algorithms performs against a purely random sampling. Similar to the first stage of the Clarke algorithm it randomly generates pairs of documents for

[7] `https://github.com/claclark/preferences`

Figure 3.4: Clarke Algorithm


D1
D3
D5
D8
D7
D2
D6
D4
generate random pairings and estimate Borda scores
D7
D6
D8
D5
prune if Borda score <= 0.5
D6
D7
D8
D5
generate new pairings, then prune again until threshold reached
D7
D6
D8
final estimate of Borda scores returns set of items with greates scores

a certain number of iterations (1000 iterations as in the other algorithms where chosen). But in contrast to the Clarke algorithm, it does not pay attention to the number of times a single document is paired. Then the pairs have to duel and for each document a score is computed by dividing the number of wins by the number of times it was paired (the score 0 was assigned when the document was not chosen in any pairing). A descending order based on these scores then gives the final ranking.

3.4 Doccano

Doccano is an open-source annotation tool, which was used to judge the documents retrieved from the algorithms.

3.5 Google Colab

All of the algorithms were implement in Python using Jupyter Notebooks and Google Colab as a workspace. In Google Colab a GPU can be selected as a hardware accelerator, which reduces the time needed to run the transformer algorithms by a lot. All of the code and data used and generated in this thesis will be made accessible in the authors repository on `git.webis.de`.

Table 3.1: Metadata Fields from the IR Anthology

Metadata field	Python data type
doc_id	str
personids	List[Dict[str, str]]
xml_checksum	str
series	str
journal	str
volume	int
publisher	str
number	int
pages	str
year	int
url	str
urn	str
doi	str
timestamp	str
biburl	str
bibsource	str
bibkey	str
bibtype	str
booktitle	str
booktitle_html	str
paper_id	int
parent_volume_id	str
title_html	str
dblp	str
authors	List[str]
editors	List[Dict[str, str]]
venue	str
title_lang_en	str
abstract_lang_en	str
full_text_lang_en	str
crossref	str
pdf	str
isbn	str
note	str
title_abstract	str
title_full_text	str
title_text_only	str

Chapter 4

Experimental Setup

In this chapter we will describe the retrieval pipelines of the experiments, how the algorithms were lined up to get results and how the retrieved data was generated and stored.

4.1 Transformer Pipelines

We use a multi-stage ranking architecture similar to the ones used in [Pradeep et al., 2021]. Since we use two different text fields, we get two different representations of one documents, so we have to apply this ranking architecture two times on each text field respectively. The text fields we use are *title_ abstract* and *title_ text_ only*. The field *title_ abstract* was generated by appending the field *abstract_ lang_ en* to the field *title_ lang_ en* so it contains the title and the abstract of the document, the field *title_ text_ only* contains the title and only the text of the document without the abstract. It was generated from the fields *title_ text_ only* and *full_ text_ lang_ en*, but because *full_ text_ lang_ en* also contains the abstract of the document, we had to remove it first before appending the field to the title. See 3.1 for an overview of the metadata fields contained for each document in the IR Anthology.

Using transformers for retrieval improves effectiveness but comes with high computational costs. Multi-stage reranking pipelines have been shown to be an effective way of reducing the number of candidate documents going into these transformer and therefore the computational costs, while still maintainig the quality of these rerankers [Zhang et al., 2021]. We follow the annotation the authors in [Zhang et al., 2021] and [Pradeep et al., 2021] and call the stages in our architecture H_0, H_1 and H_2. We use different multi-stage architectures to obtain results, two two-stage pipelines and one three-stage pipeline. The first two-stage pipeline consists of BM25 and monoT5, the second one of BM25 and duoT5, while the three-stage pipeline consists of a combination of these two, namely BM25, monoT5 and duoT5. We use different numbers for documents to be reranked in each stage, but always retrieve the top-10 of the final stage as our result. In 4.1 we depict an overview.

All our approaches share the same first stage H_0, also called candidate generation. We use pyterrier's built-in indexing function to create a standard inverted index and then use BM25 [Robertson et al., 1994] to get the first top-k documents to feed into the following stages. **k** differs for each of the pipelines we will describe next and we also used this stage as a standalone approach with k equal to 10 to compare performance later. The top-10 documents retrieved were from this approach and from all other approaches were stored using the TREC format.

BM25 » monoT5 For this pipeline the top-1000 documents retrieved from stage H_0 are used to be reranked in stage H_1 with monoT5 [Nogueira et al., 2020].

BM25 » duoT5 Here we use duoT5 in stage H_1 to rerank the top-100 retrieved from stage H_0 and retrieve the top-10 generated from this stage. We use our modified version of duoT5 to generate a python dictionary containing all the scores for the individual pairs which we saved as a jsonl-file. Since duoT5 needs to compute *n*(n-1)* for *n* documents this dictionary contains 9900 entries. It is used as a basis for the Dueling Bandit algorithms to perform their computations. For the text field *title_text_only* we also had to truncate the tokens generated to 512 since the model can not handle longer sequences and the Google Colab environment ran out of memory. For **title_abstract** we did not run into this problem.

BM25 » monoT5 » duoT5 This pipeline uses the same configurations as the pipeline *BM25 » monoT5* and adds a third stage H_2 to rerank the top-100 retrieved from the monoT5 results from stage H_1. We then also retrieved the top-10 documents from the last stage as our final result. Similar to the pipeline *BM25 » duoT5* we used the modified implementation of duoT5 to save the the scores from the individual pairs in a file to use them in the Dueling Bandit algorihtms. We also had to truncate the tokenizer to 512 tokens for the text field *title_text_only* for the same reason as in the other pipeline.

4.2 Dueling Bandits

The idea behind our approach to frame this pairwise ranking approach as a Dueling Bandit problem was to use the individual probabilities for each document pair generated from the duoT5 pipelines as baseline for the noisy comparisons (duels) the algorithms have to work with [Yue et al., 2009]. So in our multi-stage ranking architecture the pipelines for the four Dueling Bandit algorithms look almost the same as the transformer pipelines, with the difference that instead of duoT5 in stage H_1 or H_2 now the respective algorithm takes its place and only uses duoT5 when the score for a certain document pair is needed. For faster performance we used the score files generated from the transformer pipelines in our implementation of these algorithms, so that duoT5 does not have to be called. We modified the duel method from these algorithms, so that it counts the number of individual document pairs the algorithms chose for a duel. Note that the document tuples $(d_i,\ d_j)$ and $(d_j,\ d_i)$ are counted as two different pairs, since the displaying order of the documents affects the score from duoT5. The probabilities do not satisfy the equation

$$p_ij \neq 1 - p_ji \tag{4.1}$$

in general, as stated before.

This led to a problem, because the authors of some of the Dueling Bandit algorithms state to make the assumption that this order does not matter [Wu and Liu, 2016]. We therefore came up with two idea to tackle this. The first approach was to modify the scores from duoT5 and make them consistent, so they sum up to 1. We achieved this by using the formula in 4.2 for every document pair preferance probability and call this the "consistent" approach. We call using the unmodified scores directly from duoT5 the "inconsistent" approach.

$$p'_{ij} = \frac{p_{ij} + (1 - p_{ji})}{2} \tag{4.2}$$

The second idea was to use the lexicographical order of the hashes of the document ids. Before the outcome of the duel for the document pairs is determined we order them according to this, so so it does not matter in which order they are sampled by the algorithm. We call this the "hashed" approach.

We therefore end up with three different methods for each of the four algorithms and the Pure Random algorithm. For a better overview we tried to depict these combinations in the directory structure where the experimental results are saved and came up with this:

```
text field for duoT5: title + abstract OR title + text only
 └─pipeline for scores: bm25 ≫ duoT5 OR bm25 ≫ monoT5 ≫ duoT5
    ├─Double Thompson Sampling
    │  ├─inconsistent
    │  ├─consistent
    │  └─hashed
    ├─Merge Double Thompson Sampling
    │  ├─inconsistent
    │  ├─consistent
    │  └─hashed
    ├─Round Efficient Dueling Bandits
    │  ├─inconsistent
    │  ├─consistent
    │  └─hashed
    ├─Clarke
    │  ├─inconsistent
    │  ├─consistent
    │  └─hashed
    └─Pure Random
       ├─inconsistent
       ├─consistent
       └─hashed
```

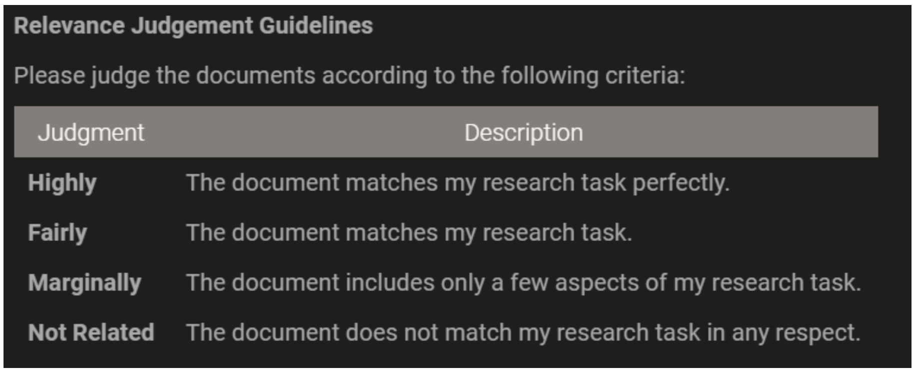
4.3 Reproducibility

To get reproducible results, all of the algorithms including monoT5 and duoT5 use seeds. The used seeds are provided in the code. Apart from the transformer pipelines, we ran each experiment ten times, each time with a different seed. Therefore we get ten result files, each with the top 10 documents retrieved from the corresponding algorithm for each seed. We used 2 different text fields, 2 different duoT5 pipelines to generate the scores on each text field, 5 algorithms and 3 methods for how the algorithms work with the scores. That yields 2*2*5*3=60 individual runs per query, so a total of 600 runs when also taking the seeds into account.

4.4 Queries and Judgments

The supervisors of this thesis and the author provided a total of 8 queries for the document retrieval task. The queries were derived from their research tasks or in case of the author from the research for this thesis and are all thematically related to Information Retrieval, so the IR Anthology is a fitting corpus for them. The table 4.2 shows an overview of the queries. They were used as input for the algorithm pipelines. Since the people who provided them derived them from their research tasks and therefore were experts on this topic they also did the judgments for the results retrieved. This judgment annotation process was done using the framework Doccano. We used a 4-point scale for the relevance judgements derived from [Hagen et al., 2016]. This was done to refine the detail of a binary judgments, but still prevent a neutral judgment from which no trend regarding relevancy could be derived. The retrieved papers of some of the experiments from [Hagen et al., 2016] are also contained in the IR Anthology, so using the same judgment scale makes it possible to merge the queries and results from the paper and this thesis to get a bigger dataset for future work. In 4.1 we show the judgment guidelines as they are displayed in Doccano.

Figure 4.1: Relevance Judgement Guidelines in Doccano



Relevance Judgement Guidelines

Please judge the documents according to the following criteria:

Judgment	Description
Highly	The document matches my research task perfectly.
Fairly	The document matches my research task.
Marginally	The document includes only a few aspects of my research task.
Not Related	The document does not match my research task in any respect.

Table 4.1: Multi-Stage Architecture of the Transformer Pipelines

Ranking Stages		
H_0	H_1	H_2
BM25 (top 10)		
BM25 (top 1000)	monoT5 (top 10)	
BM25 (top 100)	duoT5 (top 10)	
BM25 (top 1000)	monoT5 (top 100)	duoT5 (top 10)

Table 4.2: Queries provided for the Experiments

ID	Query
1	improve pairwise ranking efficiency
2	learning to rank using dueling bandits
3	high recall search approaches
4	autoregressive document retrieval
5	scientific writing text recycling
6	passage ranking transformer pretraining
7	iterative explicit relevance feedback
8	evaluation of explicit relevance feedback

Chapter 5

Results and Discussion

In this chapter we will present the results from the experiments. In the first section we will describe which metrics we used and how to interpret them. Then we will show the results we obtained from the experiments and in the last section we will discuss and evaluate these results.

5.1 Metrics

Precision Precision (P) is the fraction of retrieved documents that are relevant [Manning et al., 2008].

$$Precision = \frac{\#(relevantitemsretrieved)}{\#(retrieveditems)} \tag{5.1}$$

Recall Recall (R) is the fraction of relevant documents that are retrieved [Manning et al., 2008].

$$Recall = \frac{\#(relevantitemsretrieved)}{\#(relevantitems)} \tag{5.2}$$

Mean Average Precision MAP is the mean of the average precision of each query q, here denoted as AveP(q).

$$MAP = \frac{\sum_{i=0}^{|Q|} AveP(Q)}{|Q|} \tag{5.3}$$

The average precision of a query q is the area under the precision-recall curve.

$$AveP(q) = \int_0^1 P(R)dR \tag{5.4}$$

Mean Reciprocal Rank MRR is the mean of the reciprocal rank of each query, where the reciprocal rank is the inverse of the rank of the first relevant document, here denoted as $rank_i$.

$$MRR = \frac{1}{|Q|} \sum_{i=0}^{|Q|} \frac{1}{rank_i} \tag{5.5}$$

Normalized Discounted Cumulative Gain The discounted cumulative gain at a cutoff k, short DCG@k, is a score that penalizes highly relevant documents that appear lower in a retrieved list. Here rel_i is the graded relevance of the result at position i.

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \tag{5.6}$$

The normalized DCG, or nDCG is computed by dividing DCG by the *Ideal DCG*, which is the DCG of the ideal ordering of the retrieved documents. This makes the nDCG of different queries comparable, since it can be that the retrieved lists of results for queries differ in length and therefore queries with more results may have larger DCG but are not necessarily better [Järvelin and Kekäläinen, 2002].

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{5.7}$$

We evaluate all of these metrics on the top 10 retrieved documents from each pipeline. Since precision and recall are binary metrics, we have to translate our 4-point scale into a binary scale. Internally the values 0 to 3 are assigned to the judgments, with 0 being "Not related" and 3 being "Highly". We mark a document as relevant when it has a score of 2 or 3 and not relevant when it has a score of 0 or 1.

5.2 Results from the Transformer Pipelines

We first present the results we obtained from the transformer pipelines, since these will be the benchmark the Dueling Bandit algorithms have to beat. Table 5.1 shows the scores from both text fields and both multi-stage pipelines.

In 5.2 the results of BM25 and the multi-stage pipeline for monoT5 are shown for both text fields.

5.2.1 Results from the Dueling Bandit Algorithms

Here we show the results from Dueling Bandit algorithms. For a better overview we split up the results into two tables for each algorithms, one for each text field.

Table 5.1: Results of the douT5 pipelines on the two Text Fields

Text Fields	**Title + Abstract**		**Title + Text only**	
	duoT5	monoT5 » duoT5	duoT5	monoT5 » duoT5
MAP	0.065363	0.058704	0.046439	0.05127
MRR	0.763889	0.767857	0.583333	0.555556
Precision@10	0.45	0.4375	0.4375	0.4625
Recall@10	0.092519	0.07292	0.06694	0.074774
NDCG@10	0.350925	0.348871	0.318506	0.346429

Table 5.2: Results of BM25 & monoT5 on the two Text Fields

Text Fields	**Title + Abstract**		**Title + Text only**	
	BM25	monoT5	BM25	monoT5
MAP	0.05443	0.051796	0.041257	0.0785
MRR	0.666667	0.682292	0.635417	0.666667
Precision@10	0.4625	0.4375	0.375	0.3875
Recall@10	0.078420	0.082651	0.068703	0.103778
NDCG@10	0.358019	0.345705	0.286536	0.32168

5.3 and 5.8 show the results for Double Thompson Sampling (DTS), 5.4 and 5.9 show the results for Merge Double Thompson Sampling (MergeDTS), 5.5 and 5.10 show the results for Round-Efficient Dueling Bandits (RoundEfficient), 5.6 and 5.11 show the results for the Clarke algorithm and 5.7 and 5.12 show the results for the Pure Random algrithm.

Table 5.3: Results of DTS on the Text Field "Title + Abstract"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.04836	0.050008	0.022498	0.056752	0.051096	0.023607
MRR	0.633889	0.634097	0.427267	0.674320	0.638105	0.440878
Precision@10	0.415	0.415	0.26875	0.4325	0.4225	0.30875
Recall@10	0.069683	0.072320	0.045278	0.073070	0.069851	0.044578
NDCG@10	0.348816	0.343557	0.169281	0.351811	0.348456	0.177793

Table 5.4: Results of MergeDTS on the Text Field "Title + Abstract"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.044832	0.044192	0.022039	0.045687	0.043579	0.020177
MRR	0.636875	0.575779	0.449191	0.538938	0.536786	0.433934
Precision@10	0.40125	0.39375	0.26	0.42	0.41375	0.26375
Recall@10	0.069021	0.068423	0.050978	0.071619	0.068781	0.037346
NDCG@10	0.324085	0.304141	0.162936	0.310229	0.310740	0.162089

Table 5.5: Results of RoundEfficient on the Text Field "Title + Abstract"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.051920	0.049032	0.028238	0.049596	0.048603	0.024444
MRR	0.628661	0.60058	0.451176	0.647827	0.619439	0.439916
Precision@10	0.4325	0.42125	0.275	0.4075	0.42	0.2775
Recall@10	0.077375	0.075246	0.054865	0.065620	0.069019	0.051026
NDCG@10	0.338186	0.336427	0.175221	0.331201	0.336377	0.168908

5.2.2 Results for the Number of Duels

In this section we show the number of individual duels each algorithm performed. The tables 5.13, 5.14, 5.15, 5.16 and 5.17 show the results of the corresponding algorithms. It is no mistake that the numbers for the Pure Random algorithm in 5.17 are the same, but rather a consequence of its implementation and the usage of seeds.

5.3 Results from the Judgments

In this section we will present the results from the relevance judgments. In 5.18 the results can be seen. We received a total of 2108 relevance judgments from all the algorithms combined. Only 171 were marked as relevant, which corresponds to roughly 8.1%.

Table 5.6: Results of Clarke on the Text Field "Title + Abstract"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.046069	0.051170	0.024143	0.056954	0.054472	0.024202
MRR	0.652431	0.67005	0.37812	0.685833	0.667619	0.450084
Precision@10	0.39375	0.4225	0.2612	0.425	0.42375	0.295
Recall@10	0.065266	0.073329	0.045802	0.072102	0.071157	0.044914
NDCG@10	0.337943	0.3523	0.151571	0.353597	0.357026	0.172378

Table 5.7: Results of PureRandom on the Text Field "Title + Abstract"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.047958	0.044033	0.024150	0.049017	0.049094	0.026493
MRR	0.6656	0.624494	0.465625	0.650432	0.642153	0.426195
Precision@10	0.4025	0.405	0.2725	0.4175	0.4275	0.2925
Recall@10	0.066971	0.068577	0.052804	0.070018	0.071349	0.055049
NDCG@10	0.333262	0.320379	0.175422	0.329863	0.338470	0.179847

5.4 Discussion

From the results we can see that the metrics for the text field "Title + Abstract" are in general better than the metrics from the text field "Title + Text". This applies to all pipelines. The reason of this could be that for the text field "Title + Text" the tokens had to be truncated in the implementation of duoT5 and therefore not the whole information contained in the documents could be depicted. A general trend which also can be derived is that all metrics are rather low, which could indicate that the algorithms do not work right or the queries do not fit the corpus. Since even duoT5 has low metrics, this could hint that the queries did not fit the corpus.

We can also see that the "hashed" approach for the Dueling Bandit algorithms performs significantly worse than the other two approaches. It needs less duels but we do not think this makes up for the missing performance. The other two approaches show almost the same results for the metrics and the number of duels needed.

When looking at the metrics we can see that the Dueling Bandit algorithms have a very similar performance, which is not even significantly better than

Table 5.8: Results of DTS on the Text Field "Title + Text"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.039191	0.041597	0.02563	0.040348	0.044882	0.020969
MRR	0.545327	0.56505	0.409524	0.526716	0.562009	0.414563
Precision@10	0.40375	0.4125	0.2575	0.415	0.4325	0.26125
Recall@10	0.05795	0.0608	0.06483	0.0656	0.068663	0.04113
NDCG@10	0.29413	0.302662	0.179202	0.298792	0.31584	0.1719

Table 5.9: Results of MergeDTS on the Text Field "Title + Text"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.040205	0.037528	0.039742	0.037138	0.042331	0.016700
MRR	0.563016	0.511186	0.469459	0.486434	0.58067	0.58067
Precision@10	0.4	0.39	0.25875	0.4075	0.41375	0.24125
Recall@10	0.067027	0.068708	0.087794	0.060894	0.065992	0.038327
NDCG@10	0.283434	0.264592	0.184874	0.287308	0.308625	0.144724

the PureRandom algorithm. They all perform worse compared to monoT5 and duoT5 whereas duoT5 has the best performance according to the metrics. None of the algorithms use reaches the desired high precision or high recall.

When we look at the number of duels, we can order the algorithms according to this number. The best one, needing the least comparisons, is the PureRandom algorithm, then comes the Clarke algorithm, which just needs slightly more. Next is MergeDTS, followed by RoundEfficient which needs an similar amount. The most comparisons needs duoT5 but still only almost half as many as duoT5. The results show that the pipelines or text fields don't really affect the number of comparisons.

Table 5.10: Results of RoundEfficient on the Text Field "Title + Text"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.040667	0.040299	0.018708	0.042409	0.041212	0.022245
MRR	0.55691	0.548294	0.35253	0.542133	0.556592	0.396905
Precision@10	0.4	0.4	0.2175	0.4275	0.4075	0.2725
Recall@10	0.063301	0.067808	0.038302	0.068744	0.064902	0.046495
NDCG@10	0.298868	0.29804	0.125489	0.308765	0.296927	0.155557

Table 5.11: Results of Clarke on the Text Field "Title + Text"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.039734	0.042167	0.021131	0.042114	0.042470	0.02609
MRR	0.529286	0.533452	0.409201	0.560516	0.540253	0.382996
Precision@10	0.41375	0.43125	0.2375	0.42125	0.43375	0.26
Recall@10	0.06449	0.067593	0.050207	0.067725	0.071477	0.046458
NDCG@10	0.30019	0.312728	0.146956	0.313039	0.316639	0.156207

Table 5.12: Results of PureRandom on the Text Field "Title + Text"

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
MAP	0.043045	0.042968	0.018273	0.043233	0.041197	0.029046
MRR	0.593447	0.603710	0.319092	0.596111	0.587307	0.435303
Precision@10	0.40875	0.405	0.21125	0.41375	0.40875	0.28
Recall@10	0.082771	0.073316	0.04054	0.062977	0.063876	0.048125
NDCG@10	0.308206	0.309394	0.131336	0.310186	0.307714	0.181759

Table 5.13: Number of Duels from DTS

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
Title + Abstract	5603.65	5390.475	4211.85	5773.5	5648.1625	4207.9875
Title + Text	5086.0125	5006.6875	4209.0125	5333.2625	5195.65	4219.425

Table 5.14: Number of Duels from MergeDTS

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
Title + Abstract	3749.9375	3588.675	2702.1375	3945.4125	3779.9375	2651.7375
Title + Text	3285.6875	3164.7875	2698.75	3450.2	3403.2625	2659.9875

Table 5.15: Number of Duels from RoundEfficient

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
Title + Abstract	4388.6	4415.95	3365.15	4564.95	4465.275	3297.725
Title + Text	3385.925	3443.4	3307.5125	4150.425	4055.3875	3364.6625

Table 5.16: Number of Duels from Clarke

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
Title + Abstract	1225.3625	1226.8	1116.8875	1235.2625	1231.125	1102.5875
Title + Text	1236.1625	1202.6875	1116.4375	1242.3375	1245.7625	1120.45

Table 5.17: Number of Duels from PureRandom

Pipeline	**BM 25 » duoT5**			**BM 25 » monoT5 » duoT5**		
	incons	cons	hashed	incons	cons	hashed
Title + Abstract	948.8	948.8	900.9	948.8	948.8	900.9
Title + Text	948.8	948.8	900.9	948.8	948.8	900.9

Table 5.18: Relevance Judgments Statistics

Query ID	Highly	Fairly	Marginally	Not Related
1	5	32	22	191
2	2	12	147	84
3	21	40	45	159
4	0	0	3	306
5	1	9	20	261
6	10	17	29	185
7	8	7	3	243
8	3	4	16	223
Total	50	121	285	1652

Chapter 6

Conclusion

Pairwise ranking approaches have been shown to be an effective way of ranking documents. To make them practical to use in a real-time ranking scenario, it is important to improve their efficiency, since the computational cost for them grows quadratically with the input size. We framed this ranking approach as a Dueling Bandit problem to apply certain algorithms on it, which showed promising results in other work. In order to get some comparable results, we used the IR Anthology as our document corpus. We made the IR Anthology accessible for information retrieval toolkits and conducted relevance judgments on it, creating a dataset which can be used for future work. We derived two research questions from our approach, which were:

ResearchQuestion1 *Can we reduce the number of comparisons that have to be made when using the proposed Dueling Bandit algorithms for retrieval compared duoT5?*

Our results showed, that every algorithm used less comparisons than the duoT5 baseline, almost up to an order of magnitude. The second question we wanted to answer was:

ResearchQuestion2 *How do these algorithms perform on the measured metrics compared to duoT5?*

All of the algorithms applied on the IR Anthology performed rather bad when looking at the metrics. We could not really get the high recall we assumed. Since we used only 8 queries, which yielded bad results, this could also be a problem of the data we used and not of the algorithms.

6.1 Future Work

Future work can try to enrich the IR Anthology with more and more queries and judgments, so that it grows to be a valuable dataset for information retrieval experiments. Another interesting task is trying out our algorithms on other datasets or benchmarks, to get some more meaningful results.

Bibliography

Charles L. A. Clarke, Alexandra Vtyurina, and Mark D. Smucker. Assessing top-k preferences. *CoRR*, abs/2007.11682, 2020. URL `https://arxiv.org/abs/2007.11682`.

Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *CoRR*, abs/1910.10687, 2019. URL `http://arxiv.org/abs/1910.10687`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

Lukas Gienapp, Maik Fröbe, Matthias Hagen, and Martin Potthast. Sparse pairwise re-ranking with pre-trained transformers. In *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*, pages 72–80. ACM, 2022. doi: 10.1145/3539813.3545140. URL `https://doi.org/10.1145/3539813.3545140`.

Dorota Glowacka. Bandit algorithms in information retrieval. *Found. Trends Inf. Retr.*, 13(4):299–424, 2019. doi: 10.1561/1500000067. URL `https://doi.org/10.1561/1500000067`.

Matthias Hagen, Anna Beyer, Tim Gollub, Kristof Komlossy, and Benno Stein. Supporting scholarly search with keyqueries. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Ad-*

vances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings, volume 9626 of *Lecture Notes in Computer Science*, pages 507–520. Springer, 2016. doi: 10.1007/978-3-319-30671-1_37. URL `https://doi.org/10.1007/978-3-319-30671-1_37`.

Sebastian Hofstätter and Allan Hanbury. Let's measure run time! extending the IR replicability infrastructure to include performance aspects. In Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu, editors, *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019*, volume 2409 of *CEUR Workshop Proceedings*, pages 12–16. CEUR-WS.org, 2019. URL `https://ceur-ws.org/Vol-2409/position02.pdf`.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL `http://doi.acm.org/10.1145/582415.582418`.

Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502, 2004. doi: 10.1108/00220410410560573. URL `https://doi.org/10.1108/00220410410560573`.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020. URL `https://arxiv.org/abs/2004.04906`.

Chang Li, Ilya Markov, Maarten de Rijke, and Masrour Zoghi. Mergedts: A method for effective large-scale online ranker evaluation. *ACM Trans. Inf. Syst.*, 38(4):40:1–40:28, 2020. doi: 10.1145/3411753. URL `https://doi.org/10.1145/3411753`.

Chuang-Chieh Lin and Chi-Jen Lu. Efficient mechanisms for peer grading and dueling bandits. In *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14-16, 2018*, volume 95 of *Proceedings of Machine Learning Research*, pages 740–755. PMLR, 2018. URL `http://proceedings.mlr.press/v95/lin18a.html`.

Jimmy Lin. The neural hype, justified!: a recantation. *SIGIR Forum*, 53(2): 88–93, 2019. doi: 10.1145/3458553.3458563. URL `https://doi.org/10.1145/3458553.3458563`.

Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. Pretrained transformers for text ranking: BERT and beyond. *CoRR*, abs/2010.06467, 2020a. URL `https://arxiv.org/abs/2010.06467`.

Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. Pretrained transformers for text ranking: BERT and beyond. *CoRR*, abs/2010.06467, 2020b. URL `https://arxiv.org/abs/2010.06467`.

Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009. doi: 10.1561/1500000016. URL `https://doi.org/10.1561/1500000016`.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Trans. Assoc. Comput. Linguistics*, 9:329–345, 2021. doi: 10.1162/tacl_a_00369. URL `https://doi.org/10.1162/tacl_a_00369`.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. CEDR: contextualized embeddings for document ranking. In Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer, editors, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1101–1104. ACM, 2019. doi: 10.1145/3331184.3331317. URL `https://doi.org/10.1145/3331184.3331317`.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. Efficient document re-ranking for transformers by precomputing term representations. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 49–58. ACM, 2020. doi: 10.1145/3397271.3401093. URL `https://doi.org/10.1145/3397271.3401093`.

Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. Simplified data wrangling with ir_datasets. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2429–2436. ACM, 2021. doi: 10.1145/3404835.3463254. URL `https://doi.org/10.1145/3404835.3463254`.

Craig Macdonald and Nicola Tonellotto. Declarative experimentation in information retrieval using pyterrier. In Krisztian Balog, Vinay Setty, Christina Lioma, Yiqun Liu, Min Zhang, and Klaus Berberich, editors, *ICTIR '20: The 2020 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Norway, September 14-17, 2020*, pages 161–168. ACM, 2020. doi: 10.1145/3409256.3409829. URL https://doi.org/10.1145/3409256.3409829.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008. ISBN 978-0-521-86571-5. doi: 10.1017/CBO9780511809071. URL https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf.

Anthony A. J. Marley and Jordan J. Louviere. Some probabilistic models of best, worst, and bestâ€“worst choices. *Journal of Mathematical Psychology*, 49:464–480, 2005.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning.* Adaptive computation and machine learning. MIT Press, 2012. ISBN 978-0-262-01825-8. URL http://mitpress.mit.edu/books/foundations-machine-learning-0.

Prakash M. Nadkarni, Lucila Ohno-Machado, and Wendy Webber Chapman. Natural language processing: an introduction. *J. Am. Medical Informatics Assoc.*, 18(5):544–551, 2011. doi: 10.1136/amiajnl-2011-000464. URL https://doi.org/10.1136/amiajnl-2011-000464.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf.

Rodrigo Frassetto Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019. URL http://arxiv.org/abs/1901.04085.

Rodrigo Frassetto Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *CoRR*, abs/1910.14424, 2019a. URL http://arxiv.org/abs/1910.14424.

Rodrigo Frassetto Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *CoRR*, abs/1910.14424, 2019b. URL http://arxiv.org/abs/1910.14424.

Rodrigo Frassetto Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *CoRR*, abs/1904.08375, 2019c. URL http://arxiv.org/abs/1904.08375.

Rodrigo Frassetto Nogueira, Zhiying Jiang, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. *CoRR*, abs/2003.06713, 2020. URL https://arxiv.org/abs/2003.06713.

Martin Potthast, Sebastian Günther, Janek Bevendorff, Jan Philipp Bittner, Alexander Bondarenko, Maik Fröbe, Christian Kahmann, Andreas Niekler, Michael Völske, Benno Stein, and Matthias Hagen. The Information Retrieval Anthology. In *44th International ACM Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2550–2555. ACM, July 2021. doi: 10.1145/3404835.3462798. URL https://dl.acm.org/doi/10.1145/3404835.3462798.

Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *CoRR*, abs/2101.05667, 2021. URL https://arxiv.org/abs/2101.05667.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. URL http://arxiv.org/abs/1908.10084.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of The*

Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST), 1994. URL `http://trec.nist.gov/pubs/trec3/papers/city.ps.gz`.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL `http://arxiv.org/abs/1910.01108`.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *CoRR*, abs/1907.10597, 2019. URL `http://arxiv.org/abs/1907.10597`.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933. ISSN 00063444. URL `http://www.jstor.org/stable/2332286`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Richard Weber. On the Gittins Index for Multiarmed Bandits. *The Annals of Applied Probability*, 2(4):1024 – 1033, 1992. doi: 10.1214/aoap/1177005588. URL `https://doi.org/10.1214/aoap/1177005588`.

Huasen Wu and Xin Liu. Double thompson sampling for dueling bandits. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 649–657, 2016. URL `https://proceedings.neurips.cc/paper/2016/hash/9de6d14fff9806d4bcd1ef555be766cd-Abstract.html`.

Xinyi Yan, Chengxi Luo, Charles L. A. Clarke, Nick Craswell, Ellen M. Voorhees, and Pablo Castells. Human preferences as dueling bandits. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 567–577. ACM, 2022. doi: 10.1145/3477495.3531991. URL `https://doi.org/10.1145/3477495.3531991`.

Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of BERT for ad hoc document retrieval. *CoRR*, abs/1903.10972, 2019. URL `http://arxiv.org/abs/1903.10972`.

Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 1201–1208. ACM, 2009. doi: 10.1145/1553374.1553527. URL `https://doi.org/10.1145/1553374.1553527`.

Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009. URL `http://www.cs.mcgill.ca/%7Ecolt2009/papers/006.pdf#page=1`.

Yue Zhang, Chengcheng Hu, Yuqi Liu, Hui Fang, and Jimmy Lin. Learning to rank in the age of muppets: Effectiveness-efficiency tradeoffs in multi-stage ranking. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2021, Virtual, November 10, 2021*, pages 64–73. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.sustainlp-1.8. URL `https://doi.org/10.18653/v1/2021.sustainlp-1.8`.