MARTIN-LUTHER-UNIVERSITÄT
HALLE-WITTENBERG

Martin Luther University Halle-Wittenberg
Institute of Computer Science
Degree Programme Informatik

Bias in Learning to Rank Caused by Redundant Web Documents

Bachelor's Thesis

Jan Heinrich Reimer
Born Jan. 25, 1998 in Braunschweig
Matriculation number: 216204166

1st Referee: Prof. Dr. Matthias Hagen
2nd Referee: Maik Fröbe, M.Sc.

Submission date: June 7, 2020

Abstract

Near-duplicate documents are abundant in web corpora. Bernstein and Zobel have shown earlier that this redundancy reduces search effectiveness under the novelty principle, i.e., if subsequent duplicates in rankings are marked irrelevant or removed. We examine the impact of near duplicates on learning to rank, nowadays the standard approach for ranking web search results. Based on the LETOR benchmark dataset and the ClueWeb09 corpus, we build duplicate-aware learning-to-rank datasets and derive worst-case and average-case train/test splits for evaluation. We study nDCG@20 performance impact, ranking bias, and fairness of exposure across domains for common pointwise, pairwise, and listwise learning-to-rank models from the RankLib open source library. Our study shows, that the presence of duplicates in learning-to-rank training data induces severe bias to rankings produced by models trained with redundancy. The implicit overfitting causes performance under Bernstein and Zobel's novelty principle to drop by up to 39 %. We further introduce strategies for deduplicating training features to diminish bias and improve retrieval performance.

Contents

Abstract **i**

1 Introduction **1**

2 Related Work **4**

3 Near-Duplicate Web Documents in Test Collections **7**
3.1 Detecting Content-Equivalent Documents 7
3.2 Representative Document Selection 9

4 Duplicate-Aware Learning-to-Rank Datasets **10**
4.1 Feature Generation 11
4.2 Dataset Partition 12

5 Duplicate-Aware Learning-to-Rank Pipeline **14**
5.1 Deduplication of Feature Vectors 14
5.2 Deduplication of Search Engine Results 17

6 Evaluation **19**
6.1 Performance Impact on Learned Models 20
6.2 Bias on Learned Models 22
6.3 Domain-Based Fairness of Exposure 26

7 Conclusion and Future Work **29**

Bibliography **31**

Postface **41**

List of Figures

1.1 Example of two near-duplicate documents on Wikipedia 2

5.1 Learning-to-rank pipelines for evaluation 15
a Learning-to-rank pipeline 15
b Novelty-aware learning-to-rank pipeline 15

List of Tables

3.1 Most redundant domains in ClueWeb09 / Web Tracks 8
3.2 Most redundant domains in GOV2 / Million Query Tracks 8

4.1 Learning-to-rank features for judged ClueWeb09 documents . . . 11
4.2 Train/test splits for learning-to-rank datasets 12

6.1 nDCG@20 performance for ClueWeb09 splits 20
6.2 nDCG@20 performance for GOV2 splits 21
6.3 First rank of irrelevant duplicates for ClueWeb09 splits 23
6.4 First rank of irrelevant duplicates for GOV2 splits 24
6.5 First rank of irrelevant Wikipedia documents for ClueWeb09 splits 25
6.6 Domain-based fairness of exposure for ClueWeb09 splits 27
6.7 Domain-based fairness of exposure for GOV2 splits 27

Chapter 1

Introduction

Learning to rank is the standard approach for ranking web search results in information retrieval [Liu11, p. 5]. By using machine learning to combine predefined document-based or query-based features, e.g., retrieval scores or click logs, learning-to-rank techniques outmatch traditional ranking functions [CC11].

Web crawls can contain more than 20 % of pages that are content-equivalent to others [Bro+97; FMN03]. Those near duplicates often consist of the same document served at different URLs or for different accessors, like mobiles [Kop+10]. Figure 1.1 shows two identical web documents that—amongst 32 others—form a group of near-duplicate documents.

This redundancy causes a decrease in retrieval performance for traditional ranking functions, as Bernstein and Zobel figure that users generally do not benefit of seeing the same document twice [BZ05], which was later confirmed on many other TREC benchmark datasets [Frö+20]. Their studies call out for better consideration of near-duplicate documents in widely used metrics for ranking search results (Chapter 3). Now that classical ranking has been superseded by learning to rank, which optimizes those metrics, we ask whether the same conclusions can be drawn for the performance of machine-learned rankings.

Imbalanced representation of classes is a common problem in machine learning that can be tackled by a combination of oversampling, i.e., replicating examples in the minority class, and undersampling, i.e., eliminating examples in the majority class [Bar+04]. Though, oversampling prior to partitioning training and test data leads to information leakage from test into training data, weakening ranking reliability [Van+20]. We find that near-duplicate web documents are a form of implicit oversampling that naturally happens before splitting training and test data. Duplicate documents are overrepresented and thus learning-to-rank models are likely to overfit.

The process of creating, crawling, sampling, and parsing a document into features is biased [Bae18]. Sources of bias are diverse, some of which have been discovered in web crawls already [VT04]. We focus on the subsequent step: sampling training data and selecting feature vectors for ranking [Zad04]. With redundant data, partitioning training and test data it is particularly vulnerable as those biases multiply.

The Beatles

From Wikipedia, the free encyclopedia

The Beatles were an English rock band formed in Liverpool in 1960. With a line-up comprising John Lennon, Paul McCartney, George Harrison and Ringo Starr, they are regarded as the most influential band of all time.[1] The group were integral to the development of 1960s counterculture and popular music's recognition as an art form.[2] Rooted in skiffle, beat and 1950s rock and roll, their sound incorporated elements of classical music and traditional pop in innovative ways; the band later explored music styles ranging from ballads and Indian music to psychedelia and hard rock. As pioneers in recording, songwriting

The Beatles

From Wikipedia, the free encyclopedia
(Redirected from Beatles)

The Beatles were an English rock band formed in Liverpool in 1960. With a line-up comprising John Lennon, Paul McCartney, George Harrison and Ringo Starr, they are regarded as the most influential band of all time.[1] The group were integral to the development of 1960s counterculture and popular music's recognition as an art form.[2] Rooted in skiffle, beat and 1950s rock and roll, their sound incorporated elements of classical music and traditional pop in innovative ways; the band later explored music styles ranging from ballads and Indian music to psychedelia and hard rock. As pioneers in recording, songwriting

Figure 1.1: Example of two near-duplicate documents on Wikipedia. The document on the left is returned when requesting *The Beatles*, the right-hand one when requesting *Beatles*. Both documents are identical except for the redirect message.

We conduct a study of the performance deficit caused by overfitting near-duplicate documents, the bias induced on rankings if duplicates are left untouched, and fairness of exposure in machine-learned rankings. We build a new learning-to-rank dataset from the commonly used ClueWeb09 web crawl, using relevance judgements of TREC 2009–2012 Web Tracks (Chapter 4). For our dataset, we compute a set of features similar to those of the LETOR 4.0 benchmark dataset. We compare effects on our deeply judged dataset with the same effects on the more shallowly judged LETOR 4.0 dataset, that is based on the GOV2 corpus [QL13].

We study train/test splits with varying redundancy and compare rankings for different strategies of handling near duplicates in the training data, as well as for modelling in the subsequent evaluation (Chapter 5). For training, we either remove near-duplicate documents and only keep one canonical document of each group of near duplicates, or keep all duplicates but adjust the initial relevance label and features of non-original documents. Before evaluation, we either remove subsequent near duplicates or discount their relevance.

For our train/test splits and strategies of handling near-duplicate documents during training, we compare nDCG@20 performance on the test set when taking novelty into account to a BM25 [RW94] baseline ranking (Chapter 6). On both benchmark datasets based on ClueWeb09 and GOV2, we therefore train a selection of pointwise, pairwise, and listwise models. Beside retrieval performance, we evaluate ranking bias on near-duplicate documents and fairness of exposure per domain. Our experiments show that redundant documents in training cause a severe bias in ranking positions, and not handling them degrades retrieval performance. Even though fairness is not significantly affected on a domain level, we are concerned about the rise of near-duplicate documents in ranking, as the most redundant websites are also very popular.

Our experiments suggest that not handling near-duplicate documents different from original web documents threatens the performance of retrieval systems that use learning to rank and induces additional bias to rankings. This threat should concern particularly because many near duplicates in judged documents from the Web and Million Query tracks stem from already popular websites, that could thus abuse their supremacy. Performance under the more realistic novelty principle decreases significantly, as users are shown near duplicates more frequently. Training with near-duplicate web documents does not only lead to redundant results in ranking, but also produces biased rankings even for testing with deduplicated data. We suggest that in a typical search engine one should handle near-duplicate documents in the test data specially (Chapter 7), in order to fight the vulnerability in ranking stability caused by redundant documents.

Chapter 2

Related Work

Learning to rank still is a new field in information retrieval and machine learning that emerged from heuristic ranking models and combination of predefined features [Liu11, p. 5]. In the past years learning to rank shifted from using pointwise to pairwise [CSS98; Fre+03; Wu+10] to listwise approaches [Cao+07; MC07; XL07], and ranking models can be constructed to directly optimize evaluation measures like MAP or nDCG [Xu+08]. Intuitively, learning to rank outperforms classical ranking [CC11] and therefore has established as the standard approach to ranking web search results [Qin+10]. Niu et al. [Niu+12] introduced a top-k adaption of learning to rank. Specially designed benchmark datasets, like the MS MARCO [Ngu+16] and LETOR [Qin+10; QL13] datasets, were released for comparing effectiveness of new learning-to-rank models. Though, existing datasets only supply shallow human-annotated relevance judgements or are based on proprietary corpora.

In machine learning, algorithms often suffer from imbalanced training data, that causes overfitting [Bar+04; Die95]. The imbalanced training sample problem also occurs in information retrieval, as typically the majority of judged documents is irrelevant to a given query (see for example Table 4.2). Biased selection of training data further threatens robustness of learning-to-rank systems [Zad04]. Fortunately, it has been shown that well-directed oversampling or undersampling can decrease overfitting in most cases [Bar+04; Cha+02; IC14]. Though, training data should not be rebalanced too early, prior to splitting the dataset for training and testing, as the caused label-leakage leads to overly optimistic thus misleading results [Van+20]. We figure that the implicit undirected oversampling caused by redundant documents may pose a risk to machine-learned ranking, as it can worsen the imbalance in training data.

In 1997, Broder et al. [Bro+97] first studied clusters of near-duplicate documents on the Web. By comparing k-grams of words within documents, they discovered that 18 % of all documents were near duplicates, i.e., documents with very high syntactic similarity. Later, Fetterly, Manasse, and Najork [FMN03] confirmed that those groups of duplicate documents are stable in time and that 22 % of documents are near duplicates. These early studies were mainly concerned about improving crawler efficiency and quality [MJS07]. Hashing [Mey+03] and fingerprinting documents [BZ04; MJS07] established as an efficient way of clustering near-duplicate documents. Alternative approaches like Ioannou et al.'s semantic-aware duplicate detection [Ioa+10] or normalizing URL patterns [Kop+10] are limited in domain or prone to false positives, unlike for example fingerprinting. With all above methods, it is ambiguous which representative document from each group of near duplicates should be shown to users. While Dulitz et al. [Dul+11] suggest the most popular document should be chosen, no standard approach has been agreed on in literature. On the Web, canonical link relations [OK12] and HTTP redirects [BFN96; FR14] indicate a preference from each website's authors.

Bernstein and Zobel [BZ05] found, that of relevant documents from submitted runs of the TREC 2004 Terabyte Track 16 % are near duplicates. In information retrieval, normalized discounted cumulative gain nDCG [JK02] and mean average precision (MAP)—despite its discouragement [Fuh17]—are common ranking evaluation measures. Those evaluation measures are not aware of near-duplicate documents [BZ05; Frö+20]. Bernstein and Zobel suggest cleansing retrieval evaluation by introducing the novelty principle: A document with a relevant judgement is considered irrelevant if an equivalent near duplicate appears beforehand in the ranking. Applying their novelty principle causes a decrease in MAP performance by 20 % on average. Fröbe et al. [Frö+20] confirmed Bernstein and Zobel's analysis on runs from all TREC Terabyte, Web, and 2017–2018 Common Core Tracks, promoting an improved implementation of the novelty principle. In runs submitted to the TREC tasks, nDCG@20 performance under the novelty principle dropped by as much as 17 % [Frö+20]. Adaptions of nDCG and MAP that reward novel and diverse content have since then been introduced [BZ05; Cla+08]. Though, no research has yet been made on the effect of near duplicates on learning to rank for information retrieval.

A current trend in machine learning in general and learning to rank in particular is the concept of fair ranking [Cas18; CDS18]. Instead of technical quality measures like retrieval performance, Biega et al. [Bie+20] suggest to focus on fair and diverse representation and ranking of search results. Especially the inherent bias induced in every step of a search engines pipeline needs to be tackled or regularized [Bae18]. Many learning-to-rank recommendation algorithms are biased towards popular content [KSL20]. In learning to rank for information retrieval, user feedback is skewed as well [Ai+18; JSS17]. This causes training data to be imbalanced, i.e., documents are being judged irrelevant much more frequently than relevant [IC14]. While a small set of biases have been studied in learning to rank [Ova+20; Wan+18], Chapelle, Chang, and Liu [CCL11] still see bias as an ongoing research topic. Fair ranking frameworks consider exposure weighted against relevance [Bie+20; GS20; SJ18; SJ19; Zeh+17]. We question that practice, as near-duplicate documents being judged relevant in isolation is an unfair assumption in the first place [BZ05].

Previous research either focuses on diversity irrespective of near duplicates or on retrieval performance impact when penalizing duplicates only. Yet, we do not know how machine-learned rankings are biased by redundant web documents. The selection bias caused by near-duplicate documents currently is unmitigated in evaluation of learning-to-rank models. The risk of abuse and unfairness is mostly undiminished as well, exposing many state-of-the-art learning-to-rank algorithms. We motivate further research to prevent unfair ranking due to near-duplicate documents in the first place.

Chapter 3

Near-Duplicate Web Documents in Test Collections

For building a duplicate-aware dataset (Chapter 4), we first need to denote groups of near duplicates, i.e., content-equivalent documents, from our source web corpora, i.e., ClueWeb09 and GOV2. Second, in order to be able to deduplicate learning to rank feature vectors, we select representatives from each group of near-duplicate documents. We identify prominent contributors of near duplicates and compare their relevance.

3.1 Detecting Content-Equivalent Documents

Several techniques are used for finding near duplicates in web corpora, but comparing fingerprints/hashes of documents is used most often [BZ04; Ioa+10; Kop+10; Mey+03; MJS07]. To identify content-equivalent, near-duplicate document pairs, Bernstein and Zobel propose the lossless S_3 fingerprint similarity [BZ04] that counts common words normalized to both documents' sizes. For each pair of documents from the corpus, an S_3 score of 0 indicates no overlap while an S_3 score of 1 means that both documents are exactly equivalent. Near-duplicate pairs transitively form groups of content-equivalent documents. For our experiments and dataset, we use document groups provided by Fröbe et al., who calculated similarity using word 8-grams for various corpora, including ClueWeb09 and GOV2 [Frö+20]. They identify a S_3 score threshold of 0.84 for ClueWeb09 documents and 0.68 for GOV2 documents. Documents with S_3 scores above this threshold are content-equivalent with a precision of 0.95 [Frö+20]. Fröbe et al. confirmed their thresholds by manually reviewing 100 samples. Both corpora, GOV2 and ClueWeb09, contain large proportions of near duplicates: of the Million Query Tracks 20 % are content-equivalent and 25 % of judged documents from the Web Tracks.

In Tables 3.1 and 3.2, we list domains with the highest amounts of redundant documents from the TREC 2009–2012 Web Tracks and TREC 2007–2008 Million Query

Table 3.1: Number of near-duplicate documents from the most redundant domains in judged documents from ClueWeb09 / Web Tracks and proportions of relevant documents (Rel.) for redundant (Red.) or all judged documents. Domains without Alexa rank are not found within top 1 Million Alexa ranks.

Domain	**Tag**	**Alexa**	**Red. Doc.**		**All Doc.**	
			Count	% Rel.	Count	% Rel.
`wikipedia.org`	Research	7	7225	32 %	10694	26 %
`memoryx.net`	Technology	71949	166	0 %	175	0 %
`supercrawler.com`	Technology	—	98	29 %	130	25 %
`meetup.com`	Social	514	82	0 %	247	4 %
`acclinet.com`	Technology	—	58	0 %	63	0 %
`yahoo.net`	Search	4*	57	0 %	248	56 %
`opm.gov`	Government	11294	57	33 %	101	21 %
`newyork-hotels.tv`	—	—	55	0 %	84	2 %
`state.tn.us`	Government	23553	53	4 %	65	3 %
`nih.gov`	Government	479	52	50 %	132	43 %

*The `yahoo.net` domain is listed by Alexa as `yahoo.com` with a rank of 4.

Table 3.2: Number of near-duplicate documents from the most redundant domains in judged documents from GOV2 / Million Query Tracks and proportions of relevant documents (Rel.) for redundant (Red.) or all judged documents.

Domain	**Tag**	**Alexa**	**Red. Docs.**		**All Docs.**	
			Count	Rel.	Count	Rel.
`nih.gov`	Government	479	2557	32 %	6952	28 %
`state.gov`	Government	1593	670	51 %	1666	38 %
`noaa.gov`	Government	1023	625	29 %	3233	23 %
`nasa.gov`	Government	787	607	32 %	4279	19 %
`usda.gov`	Government	4080	531	34 %	2836	24 %
`usgs.gov`	Government	1715	483	34 %	2279	28 %
`tempe.gov`	Government	135152	392	6 %	1170	4 %
`ca.gov`	Government	729	385	28 %	2688	19 %
`michigan.gov`	Government	4560	378	29 %	547	26 %
`nara.gov`	Government	227048	320	43 %	559	33 %

Tracks respectively. We denote Alexa ranks[1] derived from a list of the 1 Million domains with the most average daily visitors and page views. We use Alexa top ranks from 23 Jun 2010 as that is the closest snapshot to the GOV2 and ClueWeb09 crawls on the Internet Archive.[2] Additionally, we report domain tags which we retrieved from OpenDNS.[3]

The domains that contain the most near-duplicate documents also have comparatively high Alexa ranks, indicating high popularity. For example, the Wikipedia encyclopedia is the 7th most popular domain shortly after the crawl, and makes up for 42 % of all near-duplicate documents from the Web Tracks. In case of `nih.gov`, the most popular domain of the corpus (i.e., the first GOV domain in the Alexa list) is also the most redundant domain. Furthermore, near-duplicate documents are relevant with a higher probability. Of the Web Tracks' judged documents 22 % are relevant but 29 % of near duplicates within. Similarly, 25 % of documents from the Million Query Tracks are relevant but 34 % of near-duplicate documents. The same effect can be seen for most individual domains, e.g., near-duplicate documents from `wikipedia.org` contain 32 % relevant documents, 6 percentage points more than all Wikipedia documents.

3.2 Representative Document Selection

For evaluating different strategies of handling near-duplicate content in a learning-to-rank pipeline (Chapter 5), we parse canonical link relations [OK12].[4] Canonical links are a good way for web document authors to hint a representative document that should be shown to users when there are alternate forms of that same original document available. Compared to choosing the most popular document like Dulitz et al. [Dul+11], canonical links resemble the web page author's intent. We use thus denoted representative documents as ground truth for later deduplication. From the largest contributor of near duplicate documents in ClueWeb09, the `wikipedia.org` domain, most near-duplicate documents (60 %) have an associated canonical document. For example the *The Beatles* document in Figure 1.1 on page 2 contains a canonical link to the *Beatles* article. If in a group of near-duplicate documents different canonical documents are linked, we choose the one that is linked the most frequently as the most likely candidate. We review a sample of 100 ambiguous cases. Most ambiguities are caused by missing canonical links that were added on Wikipedia only halfway through the ClueWeb09 crawl.

[1] `https://alexa.com/topsites/`

[2] `https://web.archive.org/web/20100623204449/http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`

[3] `https://domain.opendns.com`

[4] `https://en.wikipedia.org/wiki/Canonical_link_tag`

Chapter 4

Duplicate-Aware Learning-to-Rank Datasets

For evaluation of learning-to-rank algorithms, there exists already a variety of benchmark datasets:

- The first LETOR benchmark dataset from 2007,[1]
- the Internet Mathematics 2009 dataset by Yandex,[2]
- LETOR 3.0 datasets [Qin+10], an update on the initial LETOR datasets, based on the OHSUMED and GOV corpora,
- WCL2R [Alc+10], crawled from a Chilean search engine,
- the Yahoo! Learning to Rank Challenge benchmark dataset [CC11],
- LETOR 4.0 [QL13], based on the GOV2 corpus, and
- MS MARCO [Ngu+16], crawled from query logs of the Bing search engine.

For most of these, we are unable to detect near-duplicate documents; duplicate detection is only possible for the LETOR datasets and MS MARCO, as raw documents from the Yandex, WCL2R, and Yahoo! datasets are proprietary. We choose the LETOR 4.0 dataset using TREC 2007–2008 Million Query Track judgements, as that dataset is relatively large (2500 queries, 84834 documents) and because it features a large amount of near duplicates. The judged documents from the GOV2 corpus, on which it is based, contain 23 % content-equivalent documents [Frö+20]. The LETOR 4.0 dataset contains feature vectors for four different ranking settings: supervised ranking, semi-supervised ranking, rank aggregation, and listwise ranking [QL13]. We use supervised learning feature vectors normalized on a query level, because that dataset can directly be used for training learning-to-rank models.

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

[2] https://web.archive.org/web/20100410222404/http://imat2009.yandex.ru/en/datasets

Table 4.1: Learning-to-rank features generated for judged documents from the ClueWeb09 corpus.

Query-dependent		**Query-independent**	
Description	Count	Description	Count
Term frequency	4	URL length	1
TF · IDF	4	Number of slashes in URL	1
BM25 score	4	PageRank	1
F2 exp score	4	SpamRank	1
F2 log score	4	Number of inlinks	1
QL score	4	Number of outlinks	1
QLJM score	4		
PL2 score	4		
SPL score	4		
Σ Total			42

To contrast the shallowly judged LETOR dataset (i.e., queries in Million Query Tracks only contain few judged documents), we create a new learning-to-rank dataset from the ClueWeb09 corpus with deep relevance judgements from the TREC 2009–2012 Web Tracks. Though the ClueWeb09 corpus has been used in previous learning-to-rank research [MSO12], no dataset has been published that we could re-use for our experiments. Our new dataset aims to provide similar features to the LETOR Million Query datasets to allow for a detailed comparison of learning-to-rank performance on both, the GOV2 and ClueWeb09 corpus.

4.1 Feature Generation

Table 4.1 shows query-dependent and query-independent features that we computed for judged documents from TREC 2009–2012 Web Tracks. All query-dependent features are based on raw documents of the ClueWeb09 corpus that were indexed with Anserini.[3] Each query-dependent feature was computed for document title, main content, body, and anchor texts. The raw, non-normalized, query-dependent features were kindly provided by the author's supervisors. We supplement query-dependent features with query-independent features that are based on each document's URL, as well as link analysis of the ClueWeb09 web graph:[4] First, we count a document's URL length and the number of slashes in the URL. Second, we include

[3] `http://anserini.io/`

[4] `https://lemurproject.org/clueweb09/webGraph.php`

Table 4.2: Labeled documents in train/test splits, proportion of relevant documents (Rel.), number of near-duplicate documents (Red.), and canonical documents amongst near-duplicates (Can.).

		Train/test split	Training Labels				Test Labels			
			Count	Rel.	Red.	Can.	Count	Rel.	Red.	Can.
ClueWeb09	Web 2009–2012	Worst-case scenario	16675	25 %	4996	1285	9994	8 %	2615	804
		5-fold cross-validation 1	21337	19 %	6127	1666	5309	16 %	1441	414
		5-fold cross-validation 2	21062	18 %	6054	1669	5584	20 %	1514	411
		5-fold cross-validation 3	21960	18 %	6293	1702	4686	17 %	1275	378
		5-fold cross-validation 4	21642	19 %	6084	1661	5004	16 %	1484	419
		5-fold cross-validation 5	20583	18 %	5714	1622	6063	20 %	1854	458
GOV2 / LETOR	MQ 2007	5-fold cross-validation 1	42147	25 %	10010	3971	13652	28 %	3020	1241
		5-fold cross-validation 2	41947	25 %	10171	3991	14013	27 %	3119	1263
		5-fold cross-validation 3	41309	26 %	9572	3828	14290	24 %	3619	1404
		5-fold cross-validation 4	41478	27 %	9419	3787	13844	25 %	3272	1304
		5-fold cross-validation 5	41955	26 %	9758	3908	13813	26 %	3280	1283
	MQ 2008	5-fold cross-validation 1	9630	19 %	826	339	2874	19 %	246	97
		5-fold cross-validation 2	9404	19 %	759	309	2933	21 %	272	112
		5-fold cross-validation 3	8643	20 %	710	280	3635	15 %	295	126
		5-fold cross-validation 4	8514	20 %	723	291	3062	21 %	259	101
		5-fold cross-validation 5	9442	18 %	813	335	2707	21 %	205	82

PageRank scores by the ClueWeb09 creators,[5] as well as Cormack, Smucker, and Clarke's SpamRank score [CSC11]. Third, we calculate inlink counts and outlink counts for documents in our dataset using Spark.[6] We do not include any query features [MSO12] which are also absent from LETOR dataset [QL13]. In comparison to LETOR, we do not compute query-dependent features on document URLs, as we don't know how Qin and Liu [QL13] tokenized the URL. Also, some features that exist in LETOR 4.0 are missing from our generated features and we add similar features for replacement. We normalize features in our generated dataset on a query level for better performance with simple learning-to-rank models.

4.2 Dataset Partition

From the feature vectors for documents from the GOV2 and ClueWeb09 corpora, we derive 16 train/test splits that we use for evaluating the bias caused by near-duplicate

[5] `https://lemurproject.org/clueweb09/pageRank.php`

[6] `https://spark.apache.org/`

documents in learning to rank. Table 4.2 shows the six splits for the ClueWeb09 corpus and ten splits for the GOV2 corpus that we use for evaluation. The LETOR 4.0 datasets include predefined partitions for 5-fold cross-validation [QL13], five splits for TREC 2007 Million Query Track and five for the TREC 2008 Million Query Track. We keep Qin and Liu's train/test splits on the GOV2 corpus and ignore their validation splits, as we do not tune any model's hyperparameters.

For experiments on the ClueWeb09 corpus, we define six train/test splits by selecting the most redundant 60 queries from the TREC 2009–2012 Web Tracks, and compute learning to rank features for all documents within.

First, we construct a worst-case train/test split. We select the 40 most redundant topics for training and use the 20 less redundant topics for testing. This train/test split should allow for more extreme effects in evaluation, even for simple learning-to-rank models. Thus, the worst-case split serves as an empirical upper bound for the impact of near-duplicate documents on learning to rank. In the worst case, 25 % of the documents used for training are relevant, but only 8 % of the documents used for testing.

Second, we derive five average-case train/test splits using 5-fold cross-validation. Training and test data in this split contain equal proportions of near-duplicate documents: 28.4 % of documents in the test sets and 28.3 % of documents in the train sets are near duplicates. Additionally, we observe that cross-validation train/test splits have balanced amounts of relevant documents.

Chapter 5

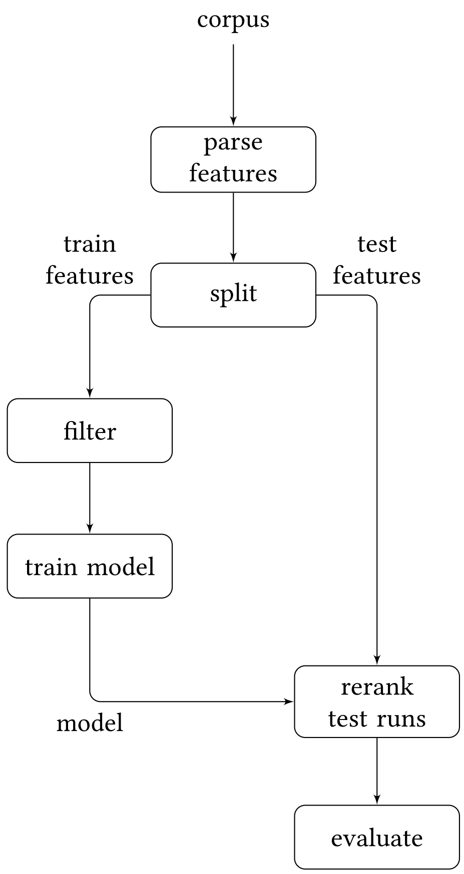
Duplicate-Aware Learning-to-Rank Pipeline

In a typical information retrieval system, learning to rank is used to rerank a top-k subset of all documents that has been retrieved with respect to a baseline ranking like BM25 [Niu+12; Qin+10]. For evaluation purposes, we rank all test documents regardless of whether they were selected by a baseline ranking or not. Parsed feature vectors from documents of the Web Tracks and Million Query Tracks are split into sets for training and testing (Chapter 4). In the learning-to-rank pipeline for evaluation [Liu11, pp. 18 sq.], as shown in Figure 5.1a, we then train a learning-to-rank model with feature vectors and ground truth labels from the training split. Afterwards all documents from the test set are ranked using the trained model. Finally, the test ranking is being evaluated for performance, fairness, or bias.

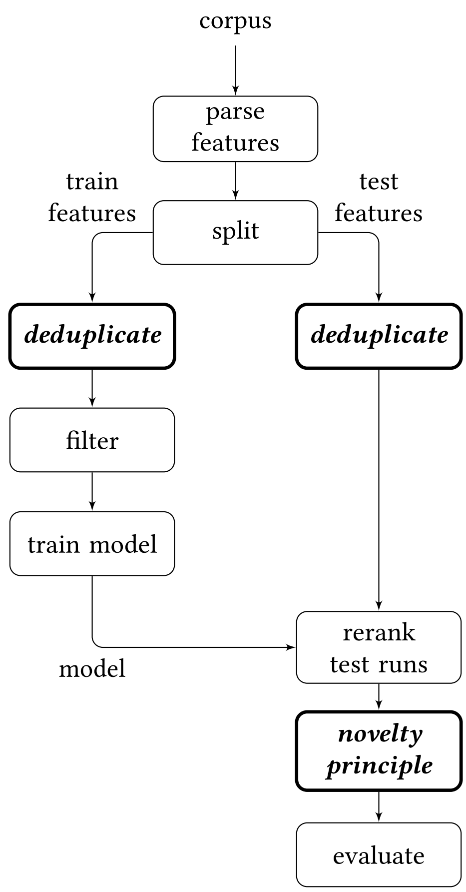
We hook into the pipeline at two places to handle near duplicates, both for training and for evaluation. In Figure 5.1b, we introduce a learning-to-rank pipeline supplemented with steps for deduplication. First, after splitting feature vectors, we use different forms of deduplication on the training and test splits. Second, we apply Bernstein and Zobel's novelty principle for evaluation [BZ05]. While deduplicating feature vectors has an active impact on training of learning-to-rank models, the novelty principle is a postprocessing step that models user behavior, but does not influence a learning-to-rank model's decisions.

5.1 Deduplication of Feature Vectors

Using features parsed from imbalanced sources like the TREC Tracks directly is prone to overfitting [Bar+04; Die95]. To counteract overfitting, it is common practice to oversample underrepresented or to undersample overrepresented feature vectors [Bar+04; Cha+02; IC14]. Vandewiele et al. [Van+20] warn that sampling should occur after splitting feature vectors, as otherwise test labels may leak into the training set. We therefore apply deduplication on training and test vector sets individually. We introduce two distinct ways of handling near-duplicate documents in learning to rank and contrast both methods with full redundancy.



(a) Learning-to-rank pipeline for supervised learning [Liu11, pp. 18 sq.].



(b) Novelty-aware learning-to-rank pipeline. We highlight steps where novelty can be addressed.

Figure 5.1: Learning-to-rank pipelines for evaluation.

Full Redundancy (100 %) We use the original feature vectors for ranking and do not remove any document from the training or test set. This scenario corresponds to classic learning to rank, but does not take redundancy into account. A learning-to-rank model thus has no specific information to classify whether a document is redundant. The full redundancy strategy serves as a baseline to removing or penalizing near duplicates.

No Redundancy (0 %) We remove all near-duplicate documents except for the representative document of each group of content-equivalent documents in the training set. To choose duplicates that should be removed, we parse each document's canonical link element [OK12]. A document is considered more canonical if more documents link it as their canonical document. Per group of content-equivalent documents, we choose only the document that is linked as canonical document most often, and remove all other documents from that group. If a group contains two documents with equally many canonical inlinks, we choose one document at random to be the group's representative document. This especially applies to LETOR feature vectors, because no canonical links are defined in documents from the GOV2 corpus. As an example of deduplication, the feature vector for the *Beatles* article of Wikipedia (see Figure 1.1, page 2) would be removed from the training set while the *The Beatles* vector would be retained. If a document has no near duplicates it is considered canonical on its own.

In the 0 % scenario, learning-to-rank models only know that every document they see is canonical, i.e., only canonical documents are contained in training/test feature vectors. Though, with this strategy learning-to-rank models have no knowledge about non-canonical documents, as they were filtered out before training. It is thus difficult for ranking algorithms to identify canonical documents from the test set, to rank them higher than their near duplicates. Additionally, in undersampling near-duplicate documents, we remove up to 22 % of our initial training data for learning (Worst-case scenario, Table 4.2). With much less training data available, we also expect learning-to-rank algorithms to be less effective.

Novelty-Aware Penalization of Duplicates (*NOV*) To not remove training data but still give learning-to-rank algorithms the opportunity to learn about novelty of documents, we propose a second deduplication strategy: We penalize relevance judgements of non-canonical feature vectors and add a boolean feature indicating canonical documents. Our goal is to achieve a total order of the following three groups: relevant canonical documents first, relevant near-duplicated documents second, and irrelevant documents last. We argue, that non-canonical near duplicates are considered less relevant than canonical documents, but still more relevant than irrelevant documents. This assumption has been suggested for evaluation by Bernstein and Zobel [BZ05], but has not yet been modelled for ground truth labels that are a key input to learning-to-rank algorithms. Therefore, we penalize relevant non-canonical near-duplicate documents by discounting their relevance judgements by 90 %. We suppose, that the order of irrelevant documents regarding their novelty does not matter, and thus leave irrelevant labels unmodified.

We observe, that many learning-to-rank features, e.g., BM25, are similar or equal for all documents in a group of near duplicates. The added boolean feature, indicating whether a vector's represented document is most canonical (Section 3.2), could be used by learning-to-rank models to discount scores according to a document's originality.

5.2 Deduplication of Search Engine Results

Apart from thoughtful sampling of feature vectors that affects the learned models, we need to model redundancy in evaluation. For modelling deduplication from a user's perspective, after ranking we adjust ranked runs according to the novelty principle and strategies introduced by Bernstein and Zobel [BZ05]. They suggest that a document, regardless of its original judgement, should be considered irrelevant if a near-duplicate document is ranked higher. We employ both, Bernstein and Zobel's original adjustment of relevance labels and removing near-duplicate documents from ranked runs. Both strategies account for novelty in search result representation. While for comparing performance (Section 6.1) that is indispensable, we also adjust relevance labels for reporting bias (Section 6.2) and fairness (Section 6.3) for more realistic estimations of both experiments.

Duplicates Unmodified As a baseline, we keep near-duplicate documents in ranked runs unmodified, duplicates are included in search results. This would model users to consider relevant near duplicates still relevant, even though they have seen a content-equivalent document before. Evaluation of retrieval performance with this baseline strategy could significantly overestimate user experience [BZ05].

Duplicates Irrelevant Second, in each ranking we mark documents appearing after another document from the same group of near-duplicate documents irrelevant, regardless of their original relevance label [BZ05]. With this adjustment of relevance labels, users would still be shown near-duplicate documents. Though, in evaluation redundant documents are treated like irrelevant documents, accounting for users' impression of not receiving new information from those documents, thus finding them irrelevant.

Duplicates Removed Often however, an information retrieval system would hide duplicate results and only show one canonical document for each group of near-duplicate documents [BZ05; CCL10; Dul+11]. We model that behavior in a third strategy: if a content-equivalent document has been seen earlier in the ranking, we remove subsequent near duplicates. With that strategy, users do not see redundant content. We expect this model to be the most realistic scenario, and evaluating runs with it should yield better results than marking near-duplicate documents irrelevant.

Chapter 6

Evaluation

We use our learning-to-rank datasets for ClueWeb09 and GOV2 (Chapter 4) to train common pointwise, pairwise, and listwise learning-to-rank models with features from each train set. For training and ranking, we use the RankLib open source library.[1] We discuss results for AdaRank [XL07], Coordinate Ascent [MC07], LambdaMART [Wu+10], ListNET [Cao+07], RankBoost [Fre+03], and linear regression. These models cover all three learning-to-rank approaches: linear regression is a pointwise learning-to-rank model, predicting the ground truth label for each single documents [Liu11, p. 20]; LambdaMART and RankBoost are pairwise models, minimizing inconsistencies in pairwise preferences [Liu11, pp. 20 sq.]; AdaRank, ListNET, and Coordinate Ascent are listwise ranking algorithms, that optimize a loss function on a ranked list [Liu11, pp. 21 sq.].

From our learning-to-rank dataset, we first filter training feature vectors according to each train/test split. We deduplicate feature vectors, train the learning-to-rank model, rerank documents from the test set, and evaluate reranked runs.

We only train with and rerank judged documents, in order to be able to use feature vectors directly for learning. To prevent the selection bias in LETOR discovered by Minka and Robertson [MR08], we prune training vectors with zero BM25@body. We deduplicate only training vectors; novelty-aware penalization of near duplicates is done on both the training and test set individually (Section 5.1). We do not tune any model's hyperparameters, but instead keep them at RankLib's default values. Also, we do not regularize our data to prevent overfitting.

We compute baseline runs ranked by descending BM25@body, because the BM25 model [RW94] is independent of the presence of near duplicates. It is thus a good comparison for the bias induced by near-duplicate documents to learning-to-rank models. Also, we should be able to see similar effects like Fröbe et al. for adjustment of relevance labels (Section 5.2), as they studied deterministically ranked runs [Frö+20]. We expect all learning-to-rank algorithms to outperform the BM25@body baseline.

Each experiment is run five times, to account for non-deterministic behaviour of most of the trained models. In Sections 6.1, 6.2, and 6.3, we report averages of all

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

Table 6.1: nDCG@20 performance on test splits for the ClueWeb09 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	nDCG@20 Performance								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
Worst-Case Scenario	BM25	0.063	—	—	0.056	—	—	0.072	—	—
	AdaRank	0.128	$0.139^{\uparrow 0.1}$	$0.164^{\uparrow 0.3}$	0.125	$0.150^{\uparrow 0.2}$	$\mathbf{0.161^{\uparrow 0.3}}$	0.156	$0.164^{\uparrow 0.1}$	$0.164^{\uparrow 0.1}$
	Coor. Ascent	0.153	$0.152^{\downarrow 0.0}$	$0.163^{\uparrow 0.1}$	0.129	$0.149^{\uparrow 0.1}$	$\mathbf{0.177^{\uparrow 0.3}}$	0.169	$0.174^{\uparrow 0.0}$	$0.177^{\uparrow 0.0}$
	LambdaMART	0.113	$\mathbf{0.151^{\uparrow 0.3}}$	$0.145^{\uparrow 0.2}$	0.110	$\mathbf{0.154^{\uparrow 0.3}}$	$\mathbf{0.159^{\uparrow 0.4}}$	0.142	$\mathbf{0.182^{\uparrow 0.3}}$	$0.159^{\uparrow 0.1}$
	ListNET	0.124	$0.132^{\uparrow 0.1}$	$0.125^{\uparrow 0.0}$	0.120	$0.135^{\uparrow 0.1}$	$0.131^{\uparrow 0.1}$	0.141	$0.150^{\uparrow 0.1}$	$0.142^{\uparrow 0.0}$
	RankBoost	0.155	$0.171^{\uparrow 0.1}$	$0.183^{\uparrow 0.2}$	0.144	$0.161^{\uparrow 0.1}$	$0.181^{\uparrow 0.3}$	0.178	$0.176^{\downarrow 0.0}$	$0.195^{\uparrow 0.1}$
	Regression	0.109	$0.142^{\uparrow 0.2}$	$0.117^{\uparrow 0.1}$	0.098	$0.139^{\uparrow 0.3}$	$0.128^{\uparrow 0.2}$	0.127	$0.159^{\uparrow 0.2}$	$0.130^{\uparrow 0.0}$
5-Fold Cross-Validation	BM25	0.143	—	—	0.112	—	—	0.143	—	—
	AdaRank	0.217	$0.210^{\downarrow 0.0}$	$0.213^{\downarrow 0.0}$	0.199	$0.196^{\downarrow 0.0}$	$0.198^{\downarrow 0.0}$	0.223	$0.211^{\downarrow 0.1}$	$0.212^{\downarrow 0.1}$
	Coor. Ascent	0.259	$0.249^{\downarrow 0.0}$	$0.231^{\downarrow 0.1}$	0.158	$\mathbf{0.197^{\uparrow 0.3}}$	$\mathbf{0.247^{\uparrow 0.6}}$	0.226	$0.240^{\uparrow 0.1}$	$0.247^{\uparrow 0.1}$
	LambdaMART	0.267	$0.222^{\downarrow 0.2}$	$0.204^{\downarrow 0.3}$	0.190	$0.181^{\downarrow 0.1}$	$\mathbf{0.215^{\uparrow 0.2}}$	0.237	$0.218^{\downarrow 0.1}$	$0.216^{\downarrow 0.1}$
	ListNET	0.185	$0.189^{\uparrow 0.0}$	$0.172^{\downarrow 0.1}$	0.156	$0.160^{\uparrow 0.0}$	$0.166^{\uparrow 0.1}$	0.186	$0.188^{\uparrow 0.0}$	$0.176^{\downarrow 0.1}$
	RankBoost	0.278	$0.254^{\downarrow 0.1}$	$0.260^{\downarrow 0.1}$	0.198	$0.204^{\uparrow 0.0}$	$0.220^{\uparrow 0.1}$	0.255	$0.256^{\uparrow 0.0}$	$0.261^{\uparrow 0.0}$
	Regression	0.233	$0.215^{\downarrow 0.1}$	$0.184^{\downarrow 0.3}$	0.145	$\mathbf{0.189^{\uparrow 0.3}}$	$\mathbf{0.183^{\uparrow 0.3}}$	0.204	$0.218^{\uparrow 0.1}$	$0.195^{\downarrow 0.1}$

five runs. We also aggregate results of both sets of Million Query cross-validation splits by reporting averages of the 2007 and 2008 tracks.

6.1 Performance Impact on Learned Models

For all three relevance adjustments (Section 5.2), we evaluate nDCG@20 [JK02] per topic of each reranked test set, and report the average performance for all topics per experimental configuration. We compute each run's performance using RankLib's nDCG implementation. For each relevance adjustment, we report performance for the three deduplication strategies described in Section 5.1. We compare the 0 % and *NOV* strategy's effect in relation to full redundancy by reporting Cohen's d [Coh88, p. 20], and highlight significant changes for the t test with $p \leqslant 0.05$.

ClueWeb09 Table 6.1 shows nDCG@20 retrieval performance on the test set for ClueWeb09 train/test splits. The upper half contains results for our worst-case train/test split and the lower half contains averaged results for the average-case 5-fold cross-validation splits. In our worst-case scenario, most learning-to-rank algorithms perform slightly better or equally good when either near-duplicate documents are removed from the training set or a duplicate non-canonical documents'

Table 6.2: nDCG@20 performance on test splits for the GOV2 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	nDCG@20 Performance								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
5-Fold Cross-Validation	BM25	0.384	—	—	0.378	—	—	0.402	—	—
	AdaRank	0.451	0.451$^{=0.0}$	0.453$^{\uparrow 0.0}$	0.432	0.432$^{=0.0}$	**0.481$^{\uparrow 0.2}$**	0.467	0.467$^{=0.0}$	**0.483$^{\uparrow 0.0}$**
	Coor. Ascent	0.500	0.501$^{\uparrow 0.0}$	0.478$^{\downarrow 0.1}$	0.477	0.478$^{\uparrow 0.0}$	**0.511$^{\uparrow 0.1}$**	0.514	0.515$^{\uparrow 0.0}$	0.511$^{\downarrow 0.0}$
	LambdaMART	0.467	0.467$^{=0.0}$	0.451$^{\downarrow 0.0}$	0.452	0.452$^{=0.0}$	**0.481$^{\uparrow 0.1}$**	0.483	0.483$^{=0.0}$	0.481$^{\downarrow 0.0}$
	ListNET	0.490	0.490$^{\downarrow 0.0}$	0.483$^{\downarrow 0.0}$	0.469	0.469$^{\downarrow 0.0}$	**0.492$^{\uparrow 0.1}$**	0.505	0.505$^{\uparrow 0.0}$	0.504$^{\downarrow 0.0}$
	RankBoost	0.503	0.503$^{=0.0}$	0.494$^{\downarrow 0.0}$	0.480	0.480$^{=0.0}$	**0.507$^{\uparrow 0.1}$**	0.517	0.517$^{=0.0}$	0.517$^{\downarrow 0.0}$
	Regression	0.496	0.496$^{=0.0}$	0.496$^{\uparrow 0.0}$	0.476	0.476$^{=0.0}$	0.472$^{\downarrow 0.0}$	0.510	0.510$^{=0.0}$	0.511$^{\uparrow 0.0}$

relevance is discounted. Only the LambdaMART algorithm improves significantly with the 0 % strategy. This effect is probably caused by LambdaMART overfitting the training data more than any other modelaw when training with full redundancy, and thus cannot generalize to the test set. Without redundancy, it does not overfit as much, resulting in a performance improvement relative to training with redundancy.

All models trained on the cross-validation train/test split perform better than models trained on the worst-case split. Though, deduplication of training vectors no longer has a significant effect on nDCG@20 performance. The rankers are not overfitting redundant documents as much as in the worst-case training set, that includes much more duplicates. Expectedly, most learning-to-rank models perform slightly worse when removing training data, like we do in the 0 % redundancy strategy. However, adding new information in the *NOV* strategy seems to confuse all algorithms except Coordinate Ascent. We see a minimal decrease in performance when not adjusting relevance labels after ranking.

We also confirm Bernstein et al.'s and Fröbe et al.'s findings, that under novelty-aware rejudgement not handling near-duplicate documents other than original content causes decreased performance throughout all algorithms [BZ05; Frö+20]. In the worst-case scenario, this effect is worse than in the average case, because the majority of relevant documents are used for training. The wost-case train set contains 39 % relevant documents, whereas the test set only contains 7 %. All learning-to-rank models perform better than the BM25@body baseline ranking and better than linear regression, for both train/test splits.

GOV2 Table 6.2 shows nDCG@20 performance on reranked test runs for the GOV2 corpus. We present averaged results for the 5-fold cross-validation splits included in the LETOR dataset [QL13] for the TREC 2007–2008 Million Query Tracks.

On this corpus we do not see any significant effect when removing near duplicates from training vectors, and only minimal effect when discounting duplicate vector's relevance. Instead, rankings are equally good for all deduplication strategies. We expectedly observe similar effects to the ClueWeb09 cross-validation train/test splits, and further confirm that nDCG@20 performance decreases under application of the novelty principle [BZ05; Frö+20]. Additionally, performance is overall higher than for the ClueWeb09 corpus. This can be caused by two reasons: First, near duplicates from the GOV2 corpus have a higher probability of being relevant than near duplicates from ClueWeb09 (Chapter 3). Irrelevant documents are thus less likely to appear on top ranks, which might be another reason for the nDCG@20 performance to be uninfluenced by deduplication. Second, LETOR train/test splits contain much more training vectors and are more shallowly judged, of which learning-to-rank algorithms benefit [YR09].

Though we cannot measure an improvement in search effectiveness when removing near duplicates during training in the average case on both corpora, we see deduplication as an opportunity for optimizing a search engine's efficiency. Duplicate documents can safely be removed from the train and test sets without degrading effectiveness, which should speed up learning to rank because we now need to calculate features for fewer documents.

6.2 Bias on Learned Models

We contrast our evaluations on general retrieval performance with studying the bias in rankings that redundant documents in training data pose to learning-to-rank models. On both corpora, we evaluate ranks of irrelevant near-duplicate documents. We add a more detailed study of the `wikipedia.org` domain as an exemplary, very popular domain. As domains of GOV2 documents are much less popular, and because no documents from `wikipedia.org` are included, we limit the latter study to ClueWeb09 splits

For both studies, we report the first rank of irrelevant documents per topic, and report averages over all test topics. For a perfect ranker, we expect irrelevant documents to be ranked lower, at the end of the ranking. Relevant documents should be ranked higher, at the start of the list. We reason, that a system which ranks irrelevant near-duplicate documents higher is biased towards redundant documents, especially when irrelevant duplicates appear on very high ranks. We report absolute ranks, not reciprocal ranks, as that would disallow us to report averages and significance [Fuh17].

Similar to the previous evaluation, we report first ranks for all learning-to-rank models and deduplication strategies. We compare the effect size in relation to full redundancy by reporting Cohen's d, and highlight significant changes for $p \leqslant 0.05$.

Table 6.3: First rank of irrelevant near-duplicate documents on test splits for the ClueWeb09 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	**First Rank of Irrelevant Duplicate Documents**								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
Worst-Case Scenario	BM25	14	—	—	13	—	—	14	—	—
	AdaRank	8	$\mathbf{13^{\uparrow 0.5}}$	$\mathbf{16^{\uparrow 0.7}}$	7	$\mathbf{12^{\uparrow 0.5}}$	$\mathbf{14^{\uparrow 0.6}}$	9	$\mathbf{16^{\uparrow 0.4}}$	$\mathbf{18^{\uparrow 0.6}}$
	Coor. Ascent	5	$\mathbf{8^{\uparrow 0.5}}$	$\mathbf{9^{\uparrow 0.6}}$	3	$\mathbf{7^{\uparrow 0.7}}$	$\mathbf{9^{\uparrow 0.8}}$	4	$\mathbf{9^{\uparrow 0.5}}$	$\mathbf{9^{\uparrow 0.6}}$
	LambdaMART	4	$5^{\uparrow 0.2}$	$\mathbf{6^{\uparrow 0.3}}$	4	$5^{\uparrow 0.2}$	$\mathbf{6^{\uparrow 0.4}}$	4	$5^{\uparrow 0.2}$	$\mathbf{6^{\uparrow 0.4}}$
	ListNET	11	$13^{\uparrow 0.1}$	$8^{\downarrow 0.2}$	9	$11^{\uparrow 0.1}$	$8^{\downarrow 0.1}$	11	$14^{\uparrow 0.2}$	$9^{\downarrow 0.2}$
	RankBoost	6	$7^{\uparrow 0.2}$	$7^{\uparrow 0.2}$	4	$5^{\uparrow 0.2}$	$\mathbf{6^{\uparrow 0.4}}$	7	$8^{\uparrow 0.1}$	$8^{\uparrow 0.2}$
	Regression	5	$5^{\downarrow 0.0}$	$\mathbf{11^{\uparrow 0.6}}$	5	$5^{\downarrow 0.1}$	$\mathbf{10^{\uparrow 0.7}}$	6	$6^{\uparrow 0.0}$	$\mathbf{11^{\uparrow 0.6}}$
5-Fold Cross-Validation	BM25	19	—	—	13	—	—	19	—	—
	AdaRank	24	$24^{\uparrow 0.0}$	$24^{\uparrow 0.0}$	15	$15^{\downarrow 0.0}$	$15^{\downarrow 0.0}$	24	$25^{\uparrow 0.0}$	$24^{\uparrow 0.0}$
	Coor. Ascent	12	$14^{\uparrow 0.1}$	$\mathbf{18^{\uparrow 0.3}}$	5	$\mathbf{7^{\uparrow 0.3}}$	$\mathbf{18^{\uparrow 0.8}}$	10	$\mathbf{13^{\uparrow 0.2}}$	$\mathbf{18^{\uparrow 0.4}}$
	LambdaMART	13	$13^{\downarrow 0.0}$	$\mathbf{16^{\uparrow 0.2}}$	7	$7^{\uparrow 0.1}$	$\mathbf{16^{\uparrow 0.6}}$	11	$12^{\uparrow 0.1}$	$\mathbf{16^{\uparrow 0.3}}$
	ListNET	17	$16^{\downarrow 0.0}$	$18^{\uparrow 0.1}$	11	$11^{\uparrow 0.0}$	$\mathbf{15^{\uparrow 0.2}}$	16	$16^{\downarrow 0.0}$	$18^{\uparrow 0.1}$
	RankBoost	15	$15^{\downarrow 0.0}$	$15^{\downarrow 0.0}$	7	$7^{\uparrow 0.0}$	$8^{\uparrow 0.1}$	14	$16^{\uparrow 0.1}$	$15^{\uparrow 0.0}$
	Regression	12	$12^{\uparrow 0.0}$	$\mathbf{17^{\uparrow 0.3}}$	6	$\mathbf{9^{\uparrow 0.3}}$	$\mathbf{15^{\uparrow 0.7}}$	10	$\mathbf{13^{\uparrow 0.2}}$	$\mathbf{17^{\uparrow 0.4}}$

Ranks of Near-Duplicate Documents Tables 6.3 and 6.4 show the first rank of irrelevant near-duplicate documents for models trained on the ClueWeb09 and GOV2 corpus respectively. For both corpora, we see that most models rank irrelevant near-duplicate documents at high ranks if feature vectors are not being deduplicated. It concerns, that for learning with LETOR feature vectors and for learning with the worst-case ClueWeb09 split, we often see top-10 ranks, highlighting that any ranking algorithm is prone to misuse by redundant documents. On the average case, all learning-to-rank models rank near duplicates higher than the BM25@body baseline.

Of our deduplication strategies, we see the *NOV* strategy to be most efficient in ranking irrelevant near-duplicate documents lower for cross-validation splits. With that strategy, Coordinate Ascent pushes irrelevant near duplicates significantly down by 6 positions on ClueWeb09 and 1 position on GOV2, when duplicates are left unmodified after ranking. If subsequent near duplicates are marked irrelevant after ranking, we see Coordinate Ascent ranks increasing 13 positions on ClueWeb09 and 4 positions on GOV2; if duplicates are removed ranks increase by 41 positions on ClueWeb09 and by 1 position on GOV2.

While we see all learning-to-rank algorithms to rank irrelevant near duplicates lower with both deduplication strategies, we do not see proportional improvement in nDCG performance (Section 6.1). We assume, that in ranked lists, as near-

Table 6.4: First rank of irrelevant near-duplicate documents on test splits for the GOV2 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	**First Rank of Irrelevant Duplicate Documents**								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
5-Fold Cross-Validation	BM25	7	—	—	7	—	—	7	—	—
	AdaRank	7	$7^{=0.0}$	$\mathbf{8}^{\uparrow\mathbf{0.1}}$	6	$6^{=0.0}$	$\mathbf{10}^{\uparrow\mathbf{0.3}}$	6	$6^{=0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.1}}$
	Coor. Ascent	7	$7^{\uparrow 0.0}$	$\mathbf{8}^{\uparrow\mathbf{0.1}}$	6	$6^{\downarrow 0.0}$	$\mathbf{10}^{\uparrow\mathbf{0.4}}$	6	$6^{\downarrow 0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.1}}$
	LambdaMART	6	$6^{=0.0}$	$\mathbf{8}^{\uparrow\mathbf{0.1}}$	6	$6^{=0.0}$	$\mathbf{10}^{\uparrow\mathbf{0.4}}$	6	$6^{=0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.1}}$
	ListNET	7	$7^{\downarrow 0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.0}}$	6	$6^{\downarrow 0.0}$	$\mathbf{8}^{\uparrow\mathbf{0.2}}$	6	$6^{\downarrow 0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.0}}$
	RankBoost	7	$7^{=0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.1}}$	6	$6^{=0.0}$	$\mathbf{8}^{\uparrow\mathbf{0.3}}$	6	$6^{=0.0}$	$\mathbf{7}^{\uparrow\mathbf{0.0}}$
	Regression	7	$7^{=0.0}$	$5^{\downarrow 0.3}$	6	$6^{=0.0}$	$4^{\downarrow 0.2}$	7	$7^{=0.0}$	$4^{\downarrow 0.3}$

duplicate documents are pushed down, non-duplicate irrelevant documents could move further up the ranking. This seems plausible, because most domains that contribute redundancy also contain large numbers of relevant documents.

Comparing the two deduplication strategies, we also see the *NOV* strategy to outmatch the 0 % strategy for the average case. This confirms again that removing information from which a ranking algorithm could learn decreases its effectiveness, even though we remove information, that would otherwise tend to confuse learning-to-rank models.

As another effect—independent from deduplication of the training set—we observe a decrease in ranks of irrelevant near-duplicate documents under the novelty principle, especially when subsequent near duplicates are marked irrelevant. We see this effect to be much stronger on the ClueWeb09 corpus than on the GOV2 corpus, and in general irrelevant near-duplicate documents are ranked higher by models trained on GOV2. Irrelevant near-duplicate documents being ranked much lower if consecutive duplicates are removed before evaluation makes a strong argument to support the findings of Bernstein et al. and Fröbe et al. [BZ05; Frö+20]. The difference in top ranks between the Million Query Tracks and Web Tracks may stem from different topic selection or crawling strategies.

Ranks of Documents from Wikipedia In addition to measuring ranks of near-duplicate documents, we add a supporting study of the `wikipedia.org` domain on rankings of ClueWeb09 documents. Wikipedia contributes by far the most near-duplicate documents and furthermore is a very popular domain (Alexa Rank: 7). As domains contributing redundancy in the GOV2 corpus are much more leveled, but also because government websites are generally less popular (Alexa rank of `nih.gov`: 479), we limit this study of ranks for documents from popular domains

Table 6.5: First rank of irrelevant Wikipedia documents on test splits for the ClueWeb09 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	**First Rank of Irrelevant Wikipedia Documents**								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
Worst-Case Scenario	BM25	83	—	—	79	—	—	74	—	—
	AdaRank	37	$\mathbf{54}^{\uparrow\mathbf{0.4}}$	$\mathbf{115}^{\uparrow\mathbf{1.3}}$	37	$\mathbf{54}^{\uparrow\mathbf{0.4}}$	$\mathbf{114}^{\uparrow\mathbf{1.3}}$	33	$\mathbf{52}^{\uparrow\mathbf{0.5}}$	$\mathbf{107}^{\uparrow\mathbf{1.3}}$
	Coor. Ascent	12	$\mathbf{46}^{\uparrow\mathbf{1.0}}$	$\mathbf{40}^{\uparrow\mathbf{1.0}}$	11	$\mathbf{45}^{\uparrow\mathbf{1.0}}$	$\mathbf{40}^{\uparrow\mathbf{1.0}}$	10	$\mathbf{40}^{\uparrow\mathbf{1.0}}$	$\mathbf{40}^{\uparrow\mathbf{1.1}}$
	LambdaMART	17	$15^{\downarrow 0.1}$	$19^{\uparrow 0.1}$	13	$12^{\downarrow 0.1}$	$\mathbf{19}^{\uparrow\mathbf{0.3}}$	14	$13^{\downarrow 0.1}$	$19^{\uparrow 0.3}$
	ListNET	64	$66^{\uparrow 0.0}$	$44^{\downarrow 0.4}$	62	$64^{\uparrow 0.0}$	$44^{\downarrow 0.4}$	57	$60^{\uparrow 0.0}$	$42^{\downarrow 0.3}$
	RankBoost	28	$\mathbf{41}^{\uparrow\mathbf{0.3}}$	$31^{\uparrow 0.1}$	24	$\mathbf{38}^{\uparrow\mathbf{0.4}}$	$29^{\uparrow 0.2}$	23	$\mathbf{34}^{\uparrow\mathbf{0.3}}$	$27^{\uparrow 0.1}$
	Regression	14	$16^{\uparrow 0.1}$	$\mathbf{25}^{\uparrow\mathbf{0.5}}$	13	$16^{\uparrow 0.1}$	$\mathbf{25}^{\uparrow\mathbf{0.5}}$	11	$14^{\uparrow 0.2}$	$\mathbf{25}^{\uparrow\mathbf{0.7}}$
5-Fold Cross-Validation	BM25	83	—	—	72	—	—	69	—	—
	AdaRank	72	$68^{\downarrow 0.1}$	$\mathbf{83}^{\uparrow\mathbf{0.2}}$	65	$63^{\downarrow 0.1}$	$\mathbf{76}^{\uparrow\mathbf{0.2}}$	65	$63^{\downarrow 0.0}$	$\mathbf{77}^{\uparrow\mathbf{0.2}}$
	Coor. Ascent	26	$\mathbf{40}^{\uparrow\mathbf{0.4}}$	$\mathbf{37}^{\uparrow\mathbf{0.3}}$	18	$\mathbf{30}^{\uparrow\mathbf{0.4}}$	$\mathbf{37}^{\uparrow\mathbf{0.6}}$	20	$\mathbf{34}^{\uparrow\mathbf{0.5}}$	$\mathbf{37}^{\uparrow\mathbf{0.6}}$
	LambdaMART	27	$28^{\uparrow 0.0}$	$29^{\uparrow 0.1}$	19	$20^{\uparrow 0.0}$	$\mathbf{28}^{\uparrow\mathbf{0.3}}$	24	$24^{\downarrow 0.0}$	$29^{\uparrow 0.2}$
	ListNET	55	$54^{\downarrow 0.0}$	$57^{\uparrow 0.0}$	47	$47^{\uparrow 0.0}$	$53^{\uparrow 0.1}$	47	$47^{\downarrow 0.0}$	$53^{\uparrow 0.1}$
	RankBoost	42	$48^{\uparrow 0.2}$	$45^{\uparrow 0.1}$	31	$\mathbf{42}^{\uparrow\mathbf{0.3}}$	$37^{\uparrow 0.1}$	34	$\mathbf{43}^{\uparrow\mathbf{0.3}}$	$39^{\uparrow 0.1}$
	Regression	26	$\mathbf{38}^{\uparrow\mathbf{0.3}}$	$\mathbf{44}^{\uparrow\mathbf{0.5}}$	18	$\mathbf{32}^{\uparrow\mathbf{0.5}}$	$\mathbf{42}^{\uparrow\mathbf{0.7}}$	19	$\mathbf{32}^{\uparrow\mathbf{0.5}}$	$\mathbf{42}^{\uparrow\mathbf{0.7}}$

to ClueWeb09 documents.

In Table 6.5, we report average first ranks of irrelevant documents from the `wikipedia.org` domain for models trained on the ClueWeb09 across topics. If all irrelevant documents from Wikipedia were near duplicates, we would expect very similar results to Table 6.3, especially if near duplicates are removed after ranking. Although, even with that assumption being false, we see similarities to first ranks of irrelevant near-duplicate documents (Section 6.2). By all learning-to-rank models, Wikipedia documents are ranked at much higher positions than in the BM25@body baseline ranking. Without deduplication of search engine results, documents from `wikipedia.org` are ranked 42 positions higher on average than the baseline ranking. This is cause for alarm, as all rankers are clearly biased towards ranking near-duplicate documents higher. Additionally, we observe that `wikipedia.org`, besides other domains that contribute high amounts of near-duplicate documents (e.g., `yahoo.com`, `meetup.com`, and `nih.gov`), is very popular (Tables 3.1, 3.2, page 8). Redundant websites such as Wikipedia are often popular, thus resulting in a bias towards documents from popular domains.

Deduplication of feature vectors for learning to rank helps to reduce that bias. We see irrelevant Wikipedia documents to be ranked at lower positions by all ranking models for at least one of the 0 % or *NOV* deduplication strategies. Interestingly, some learning-to-rank algorithms rank fairer with the 0 % strategy

(RankBoost), and some with the *NOV* strategy (AdaRank, LambdaMART). Though, in general the *NOV* strategy is better able to reduce the ranking bias. Training the AdaRank model with *NOV* deduplication fully compensates the bias on the average case and irrelevant Wikipedia documents are ranked on similar positions as with BM25. For our worst-case scenario AdaRank even ranks irrelevant documents from `wikipedia.org` lower, i.e., fairer, than the BM25@body baseline ranking.

Our results constitute additional motivation to handle near-duplicate documents specially when learning to rank. Both evaluations of irrelevant near duplicate ranks and irrelevant Wikipedia ranks highlight a severe bias in representation of redundant documents. The ranking bias for Wikipedia documents concerns particularly, as in that case a very popular domain is overrepresented by ranking algorithms. Models vulnerable to this kind of bias are prone to SEO abuse and can be tricked especially by popular domains.

6.3 Domain-Based Fairness of Exposure

As a third means of evaluating bias induced in learning-to-rank models by near-duplicate documents, we study fairness of exposure, as defined by the recent TREC 2019 Fair Ranking Track [Bie+20]. Similar to the Fair Ranking Track's organizers, we want to examine, whether some content providers are preferred by a learning-to-rank model, and if deduplication of feature vectors could help to improve fairness.

For our evaluation, instead of arbitrary groups of authors, we compute fairness based on each document's normalized domain name. We extract the hostname from each document's URL, and strip subdomains until only one non-top-level subdomain is left. We use the Mozilla Foundation's Public Suffix List[2] to determine a top-level domain, but fall back to the last domain part if no matching suffix is found in the Public Suffix List. For instance the domain `m.en.wikipedia.org` is reduced to `wikipedia.org`, but `data.gov.uk` stays unmodified.

We compute domain-based fairness per topic, and exclude topics without any relevant document in the ranking. All parameters of Biega et al.'s fairness measure are kept to TREC defaults [Bie+20]. In Tables 6.6, and 6.7, we report average fairness for all topics, effect size (Cohen's d) compared to 100 % redundancy, and highlight significant changes for $p \leqslant 0.05$.

For both corpora, ClueWeb09 and GOV2, we see little changes with different deduplication strategies. However, in the average-case scenario with full redundancy and no adjustment of relevance labels after ranking, most rankers generate slightly less fair rankings, compared to the BM25@body baseline. Conversely, when the most redundant topics are used for learning in the worst-case scenario, fairness

[2] `https://publicsuffix.org/`

Table 6.6: Fairness of exposure across domains on test splits for the ClueWeb09 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	Fairness of Exposure Across Domains								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
Worst-Case Scenario	BM25	0.560	—	—	0.558	—	—	0.554	—	—
	AdaRank	0.565	0.570$^{\uparrow 0.0}$	0.579$^{\uparrow 0.0}$	0.559	0.563$^{\uparrow 0.0}$	0.572$^{\uparrow 0.0}$	0.545	0.561$^{\uparrow 0.0}$	0.572$^{\uparrow 0.1}$
	Coor. Ascent	0.606	0.572$^{\downarrow 0.1}$	0.579$^{\downarrow 0.1}$	0.632	0.570$^{\downarrow 0.2}$	0.580$^{\downarrow 0.1}$	0.575	0.563$^{\downarrow 0.0}$	0.580$^{\uparrow 0.0}$
	LambdaMART	0.570	0.576$^{\uparrow 0.0}$	0.559$^{\downarrow 0.0}$	0.579	0.592$^{\uparrow 0.0}$	0.562$^{\downarrow 0.0}$	0.557	0.556$^{\downarrow 0.0}$	0.562$^{\uparrow 0.0}$
	ListNET	0.586	0.579$^{\downarrow 0.0}$	0.580$^{\downarrow 0.0}$	0.585	0.579$^{\downarrow 0.0}$	0.583$^{\downarrow 0.0}$	0.566	0.571$^{\uparrow 0.0}$	0.564$^{\downarrow 0.0}$
	RankBoost	0.599	0.582$^{\downarrow 0.0}$	0.600$^{\uparrow 0.0}$	0.600	0.578$^{\downarrow 0.1}$	0.601$^{\uparrow 0.0}$	0.587	0.564$^{\downarrow 0.1}$	0.597$^{\uparrow 0.0}$
	Regression	0.575	0.555$^{\downarrow 0.1}$	0.539$^{\downarrow 0.1}$	0.597	0.570$^{\downarrow 0.1}$	0.550$^{\downarrow 0.1}$	0.533	0.563$^{\uparrow 0.1}$	0.550$^{\uparrow 0.0}$
5-Fold Cross-Validation	BM25	0.689	—	—	0.628	—	—	0.619	—	—
	AdaRank	0.688	0.695$^{\uparrow 0.0}$	0.688$^{\downarrow 0.0}$	0.628	0.634$^{\uparrow 0.0}$	0.626$^{\downarrow 0.0}$	0.627	0.630$^{\uparrow 0.0}$	0.622$^{\downarrow 0.0}$
	Coor. Ascent	0.642	0.676$^{\uparrow 0.1}$	**0.700$^{\uparrow 0.2}$**	0.665	0.641$^{\downarrow 0.1}$	0.653$^{\downarrow 0.0}$	0.625	0.631$^{\uparrow 0.0}$	0.653$^{\uparrow 0.1}$
	LambdaMART	0.650	0.646$^{\downarrow 0.0}$	0.661$^{\uparrow 0.0}$	0.640	0.640$^{\uparrow 0.0}$	0.630$^{\downarrow 0.0}$	0.626	0.627$^{\uparrow 0.0}$	0.631$^{\uparrow 0.0}$
	ListNET	0.675	0.687$^{\uparrow 0.0}$	0.670$^{\downarrow 0.0}$	0.622	0.638$^{\uparrow 0.1}$	0.622$^{\downarrow 0.0}$	0.615	0.625$^{\uparrow 0.0}$	0.620$^{\uparrow 0.0}$
	RankBoost	0.692	0.719$^{\uparrow 0.1}$	0.711$^{\uparrow 0.1}$	0.659	0.659$^{\downarrow 0.0}$	0.658$^{\downarrow 0.0}$	0.649	0.652$^{\uparrow 0.0}$	0.655$^{\uparrow 0.0}$
	Regression	0.644	0.681$^{\uparrow 0.1}$	0.674$^{\uparrow 0.1}$	0.639	0.635$^{\downarrow 0.0}$	0.623$^{\downarrow 0.1}$	0.606	0.628$^{\uparrow 0.1}$	0.617$^{\uparrow 0.0}$

Table 6.7: Fairness of exposure across domains on test splits for the GOV2 corpus. Superscripts indicate effect size, significant changes are highlighted bold.

	Algorithm	Fairness of Exposure Across Domains								
		Duplicates Unmodified			Duplicates Irrelevant			Duplicates Removed		
		100 %	0 %	*NOV*	100 %	0 %	*NOV*	100 %	0 %	*NOV*
5-Fold Cross-Validation	BM25	0.352	—	—	0.347	—	—	0.344	—	—
	AdaRank	0.314	0.314$^{=0.0}$	0.309$^{\downarrow 0.0}$	0.309	0.309$^{=0.0}$	0.304$^{\downarrow 0.0}$	0.305	0.305$^{=0.0}$	0.304$^{\downarrow 0.0}$
	Coor. Ascent	0.296	0.295$^{\downarrow 0.0}$	0.291$^{\downarrow 0.0}$	0.290	0.289$^{\downarrow 0.0}$	0.287$^{\downarrow 0.0}$	0.285	0.284$^{\downarrow 0.0}$	0.287$^{\uparrow 0.0}$
	LambdaMART	0.296	0.296$^{=0.0}$	0.295$^{\downarrow 0.0}$	0.290	0.290$^{=0.0}$	0.290$^{\downarrow 0.0}$	0.286	0.286$^{=0.0}$	0.290$^{\uparrow 0.0}$
	ListNET	0.294	0.295$^{\uparrow 0.0}$	0.289$^{\downarrow 0.0}$	0.289	0.289$^{\uparrow 0.0}$	0.284$^{\downarrow 0.0}$	0.286	0.285$^{\downarrow 0.0}$	0.283$^{\downarrow 0.0}$
	RankBoost	0.294	0.294$^{=0.0}$	0.289$^{\downarrow 0.0}$	0.288	0.288$^{=0.0}$	0.284$^{\downarrow 0.0}$	0.283	0.283$^{=0.0}$	0.284$^{\uparrow 0.0}$
	Regression	0.301	0.301$^{=0.0}$	0.295$^{\downarrow 0.0}$	0.293	0.293$^{=0.0}$	0.293$^{\downarrow 0.0}$	0.291	0.291$^{=0.0}$	0.288$^{\downarrow 0.0}$

improves upon the baseline ranking, even though all rankings in the average-case cross-validation train/test splits are fairer. Either marking consecutive near duplicates irrelevant or removing them worsens fairness of exposure in all train/test splits for all ranking algorithms. In both cases, learning-to-rank models perform better than the BM25@body baseline ranking on the deeply judged Web Track documents, but worse on shallowly judged Million Query Track topics. The decreased fairness of novelty-aware rejudged rankings concerns and questions its use, as currently the focus in information retrieval evaluation drifts from pure relevance-related measures to novelty, diversity, and fair representation. Unfortunately, we are unable to show significant changes regarding different deduplication strategies, except for a small improvement of the *NOV* strategy on the Coordinate Ascent model with no adjustment of relevance labels.

A deeper analysis of different measures of fairness, including tuning Biega et al.'s parameters for trading off fairness versus relevance [Bie+20], is required to show significantly, how redundancy in training labels affects the fairness of search result representation in learning-to-rank models.

Chapter 7

Conclusion and Future Work

Near-duplicate documents in web crawls affect retrieval performance [BZ05] and are present in learning-to-rank datasets. Our experiments, using the LETOR benchmark dataset and the ClueWeb09 corpus, approach three different effects of near duplicates on learning to rank: nDCG@20 performance, ranking bias, and fairness of exposure across domains. We find that popular learning-to-rank models are affected by near-duplicate web documents.

Average performance of ranked runs decreases by as much as 39 % (Coordinate Ascent, ClueWeb09) when novelty is modeled by marking already seen near duplicates irrelevant, compared to keeping near duplicates' labels unmodified. We evaluate two strategies to counteract this performance impact. First, we mutate training feature vectors to introduce novelty based on canoncial links to learning-to-rank models. Second, subsequent near duplicates are removed after ranking, as was proposed by Bernstein and Zobel [BZ05]. Both strategies help to reduce the performance deficit from marking near-duplicate documents irrelevant. All studied learning-to-rank models benefit from novelty-aware feature mutation, often significantly and with medium effect.

Additionally, we measure bias in ranks of near-duplicate documents. Irrelevant near-duplicate documents are ranked on top ranks by nearly all learning-to-rank algorithms, indicating a clear bias towards duplicate content. The observed bias is particularly strong for documents from the popular `wikipedia.org` domain. We find that nearly all learning-to-rank models in our study profit from novelty-aware feature mutation. Feature mutation counteracts ranking bias by helping learning to rank models to rank irrelevant near duplicates significantly lower in almost all considered learning-to-rank models.

Conversely, fairness of exposure per domain does not change with any deduplication strategy we studied. Though, we observe a minimal decrease in fairness if relevance labels are adjusted according to Bernstein and Zobel's novelty strategy [BZ05]. We figure that Biega et al.'s fairness measure might penalize the removal of relevant duplicates.

We conclude, that introducing novelty information to learning-to-rank training features is a good strategy for improving a retrieval system's performance and robustness against redundancy. Not handling near duplicates especially poses a risk on search engines, as performance decreases and rankings are biased. We therefore encourage further research in the field and call out for systematic evaluation of biases caused by redundancy in learning to rank.

Future Work We encountered several research directions during our studies that we see as important questions to be considered for future research. Some learning-to-rank models can directly optimize performance metrics [Xu+08; Zeh+17]. Training those models with metrics that are novelty-aware could counteract bias towards redundant documents. Existing research on learning diversity may be adjusted to take novelty in terms of duplication into account as well [SJ18; SJ19; Zeh+17; Zhu+14]. A popular approach for detecting bias and measuring fairness in information retrieval is scoring exposure in relation to relevance [Bie+20]. Our experiments show, that this approach does not accurately model fairness in terms of redundancy. For other pointwise and pairwise learning-to-rank models, the effect of near-duplicate documents on their loss functions for learning to rank should be considered. Similarly, each feature's importance in trained models is an interesting future research direction. We advocate studying empirically and theoretically the direct bias induced to a learning-to-rank model, without observing their effects indirectly like we did. If particularly vulnerable features are found, those could be discounted or removed from training data. Alternatively, similar to query features [MSO12], some learning-to-rank algorithms could be improved to generate better rankings within a group of near-duplicate documents, to ensure the best document within a group is always ranked first.

Source code We release the source code for reproduction of our results under a free license, to incubate future research in the field.[1] The open source repository includes instructions for downloading and using the free datasets we derived from ClueWeb09 documents and the LETOR 4.0 dataset.

[1] https://github.com/webis-de/sigir20-sampling-bias-due-to-near-duplicates-in-learning-to-rank

Bibliography

[Ai+18] Qingyao Ai, Jiaxin Mao, Yiqun Liu, and W. Bruce Croft. "Unbiased Learning to Rank: Theory and Practice." In: 27th ACM International Conference on Information and Knowledge Management. CIKM '18 (Torino, Italy, Oct. 22–26, 2018). Ed. by Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang. Association for Computing Machinery, 2018, pp. 2305–2306. ISBN: 978-1-4503-6014-2. DOI: 10.1145/3269206.3274274 (cit. on p. 5).

[Alc+10] Otávio D. A. Alcântara, Álvaro R. Pereira Jr., Humberto Mossri de Almeida, Marcos André Gonçalves, Christian Middleton, and Ricardo Baeza-Yates. "WCL2R: A Benchmark Collection for Learning to Rank Research with Clickthrough Data." In: *Journal of Information and Data Management* 1.3 (2010), pp. 551–566. ISSN: 2178-7107. URL: https://periodicos.ufmg.br/index.php/jidm/article/view/71 (cit. on p. 10).

[Bae18] Ricardo Baeza-Yates. "Bias on the web." In: *Communications of the ACM* 61.6 (2018), pp. 54–61. ISSN: 0001-0782. DOI: 10.1145/3209581 (cit. on pp. 1, 5).

[Bar+04] Ricardo Barandela, Rosa Maria Valdovinos, José Salvador Sánchez, and Francesc J. Ferri. "The Imbalanced Training Sample Problem: Under or over Sampling?" In: *Structural, Syntactic, and Statistical Pattern Recognition.* Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition. SSPR 2004 and SPR 2004 (Lisbon, Portugal, Aug. 18–20, 2004). Ed. by Ana L. N. Fred, Terry Caelli, Robert P. W. Duin, Aurélio C. Campilho, and Dick de Ridder. Vol. 3138. Lecture Notes in Computer Science. Springer, 2004, p. 806. ISBN: 978-3-540-22570-6. DOI: 10.1007/978-3-540-27868-9_88 (cit. on pp. 1, 4, 14).

[BFN96] Tim Berners-Lee, Roy T. Fielding, and Henrik Frystyk Nielsen. *Hypertext Transfer Protocol – HTTP/1.0.* RFC 1945. RFC Editor, May 1996. DOI: 10.17487/RFC1945 (cit. on p. 5).

[Bie+20] Asia J. Biega, Fernando Diaz, Michael D. Ekstrand, and Sebastian Kohlmeier. “Overview of the TREC 2019 Fair Ranking Track.” In: *Computing Research Repository. CoRR* abs/2003.11650 (2020). ISSN: 2331-8422. arXiv: `2003.11650` (cit. on pp. 5, 26, 28–30).

[Bro+97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. “Syntactic Clustering of the Web.” In: *Computer Networks and ISDN Systems* 29.8–13 (1997), pp. 1157–1166. ISSN: 0169-7552. DOI: `10.1016/S0169-7552(97)00031-7` (cit. on pp. 1, 4).

[BZ04] Yaniv Bernstein and Justin Zobel. “A Scalable System for Identifying Coderivative Documents.” In: *String Processing and Information Retrieval.* 11th International Symposium on String Processing and Information Retrieval. SPIRE 2004 (Padova, Italy, Oct. 5–8, 2004). Ed. by Alberto Apostolico and Massimo Melucci. Vol. 3246. Lecture Notes in Computer Science. Springer, 2004, pp. 55–67. ISBN: 978-3-540-23210-0. DOI: `10.1007/978-3-540-30213-1_6` (cit. on pp. 5, 7).

[BZ05] Yaniv Bernstein and Justin Zobel. “Redundant documents and search effectiveness.” In: 14th ACM International Conference on Information and Knowledge Management. CIKM ’05 (Bremen, Germany, Oct. 31–Nov. 5, 2005). Ed. by Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken. Association for Computing Machinery, 2005, pp. 736–743. ISBN: 978-1-59593-140-5. DOI: `10.1145/1099554.1099733` (cit. on pp. i, 1, 5, 14, 16–18, 21, 22, 24, 29).

[Cao+07] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. “Learning to rank: from pairwise approach to listwise approach.” In: Twenty-Fourth International Conference on Machine Learning. ICML ’07 (Corvallis, Oregon, United States, June 20–24, 2007). Ed. by Zoubin Ghahramani. Vol. 227. International Conference Proceeding Series. Association for Computing Machinery, 2007, pp. 129–136. ISBN: 978-1-59593-793-3. DOI: `10.1145/1273496.1273513` (cit. on pp. 4, 19).

[Cas18] Carlos Castillo. “Fairness and Transparency in Ranking.” In: *ACM SIGIR Forum* 52.2 (2018), pp. 64–71. ISSN: 0163-5840. DOI: `10.1145/3308774.3308783` (cit. on p. 5).

[CC11] Olivier Chapelle and Yi Chang. “Yahoo! Learning to Rank Challenge Overview.” In: Yahoo! Learning to Rank Challenge. Held at ICML ’10 (Haifa, Israel, June 25, 2010). Ed. by Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Vol. 14. Proceedings of Machine Learning Research. 2011, pp. 1–24. URL: `http://proceedings.mlr.press/v14/chapelle11a` (cit. on pp. 1, 4, 10).

[CCL10] Jack G. Conrad, Joanne R.S. Claussen, and Jie Lin. "Information retrieval systems with duplicate document detection and presentation functions." US 7,809,695 B2 (United States). Thomson Reuters Global Resources. 2010. URL: https://pimg-fpiw.uspto.gov/fdd/95/096/078/0.pdf (cit. on p. 18).

[CCL11] Olivier Chapelle, Yi Chang, and Tie-Yan Liu. "Future directions in learning to rank." In: Yahoo! Learning to Rank Challenge. Held at ICML '10 (Haifa, Israel, June 25, 2010). Ed. by Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Vol. 14. Proceedings of Machine Learning Research. 2011, pp. 91–100. URL: http://proceedings.mlr.press/v14/chapelle11b (cit. on p. 5).

[CDS18] J. Shane Culpepper, Fernando Diaz, and Mark D. Smucker. "Research Frontiers in Information Retrieval: Report from the Third Strategic Workshop on Information Retrieval in Lorne." SWIRL 2018. In: *ACM SIGIR Forum* 52.1 (2018), pp. 34–90. ISSN: 0163-5840. DOI: 10.1145/3274784.3274788 (cit. on p. 5).

[Cha+02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique." In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953 (cit. on pp. 4, 14).

[Cla+08] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. "Novelty and diversity in information retrieval evaluation." In: 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '08 (Singapore, July 20–24, 2008). Ed. by Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong. Association for Computing Machinery, 2008, pp. 659–666. ISBN: 978-1-60558-164-4. DOI: 10.1145/1390334.1390446 (cit. on p. 5).

[Coh88] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. Erlbaum, 1988. ISBN: 978-0-8058-0283-2 (cit. on pp. 20, 22, 26).

[CSC11] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. "Efficient and effective spam filtering and re-ranking for large web datasets." In: *Information Retrieval Journal* 14.5 (2011), pp. 441–465. ISSN: 1386-4564. DOI: 10.1007/s10791-011-9162-z (cit. on p. 12).

[CSS98] William W. Cohen, Robert E. Schapire, and Yoram Singer. “Learning to Order Things.” In: *Advances in Neural Information Processing Systems 10*. Neural Information Processing Systems 1997. NIPS 1997 (Denver, Colorado, United States, Dec. 1–6, 1997). Ed. by Michael I. Jordan, Michael J. Kearns, and Sara A. Solla. MIT Press, 1998, pp. 451–457. ISBN: 978-0-262-10076-2. URL: `https://papers.nips.cc/paper/1431-learning-to-order-things` (cit. on p. 4).

[Die95] Thomas G. Dietterich. “Overfitting and Undercomputing in Machine Learning.” In: *ACM Computing Surveys* 27.3 (1995), pp. 326–327. ISSN: 0360-0300. DOI: `10.1145/212094.212114` (cit. on pp. 4, 14).

[Dul+11] Daniel Dulitz, Alexandre A. Verstak, Sanjay Ghemawat, and Jeffrey A. Dean. “Representative Document Selection for Sets of Duplicate Documents in a Web Crawler System.” US 7,984,054 B2 (United States). Google Inc. 2011. URL: `https://pimg-fpiw.uspto.gov/fdd/54/840/079/0.pdf` (cit. on pp. 5, 9, 18).

[FMN03] Dennis Fetterly, Mark S. Manasse, and Marc Najork. “On the Evolution of Clusters of Near-Duplicate Web Pages.” In: *Empowering Our Web*. 1st Latin American Web Congress. LA-WEB 2003 (Sanitago, Chile, Nov. 10–12, 2003). IEEE Computer Society, 2003, pp. 37–45. ISBN: 978-0-7695-2058-2. DOI: `10.1109/LAWEB.2003.1250280` (cit. on pp. 1, 4).

[FR14] Roy T. Fielding and Julian F. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. RFC Editor, June 2014. DOI: `10.17487/RFC7231` (cit. on p. 5).

[Fre+03] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. “An Efficient Boosting Algorithm for Combining Preferences.” In: *Journal of Machine Learning Research* 4 (2003), pp. 933–969. ISSN: 1533-7928. URL: `http://jmlr.org/papers/v4/freund03a.html` (cit. on pp. 4, 19).

[Frö+20] Maik Fröbe, Jan Philipp Bittner, Martin Potthast, and Matthias Hagen. “The Effect of Content-Equivalent Near-Duplicates on the Evaluation of Search Engines.” In: *Advances in Information Retrieval*. 42nd European Conference on IR Research. ECIR 2020 (Lisbon, Portugal, Apr. 14–17, 2020). Ed. by Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins. Vol. 12036.2. Lecture Notes in Computer Science. Springer, 2020, pp. 12–19. ISBN: 978-3-030-45441-8. DOI: `10.1007/978-3-030-45442-5_2` (cit. on pp. 1, 5, 7, 10, 19, 21, 22, 24).

[Fuh17] Norbert Fuhr. "Some Common Mistakes In IR Evaluation, And How They Can Be Avoided." In: *ACM SIGIR Forum* 51.3 (2017), pp. 32–41. ISSN: 0163-5840. DOI: 10.1145/3190580.3190586 (cit. on pp. 5, 22).

[GS20] Ruoyuan Gao and Chirag Shah. "Toward creating a fairer ranking in search engine results." In: *Information Processing and Management* 57.1 (2020). ISSN: 0306-4573. DOI: 10.1016/j.ipm.2019.102138 (cit. on p. 5).

[IC14] Muhammad Ibrahim and Mark James Carman. "Undersampling Techniques to Re-balance Training Data for Large Scale Learning-to-Rank." In: *Information Retrieval Technology*. 10th Asia Information Retrieval Societies Conference. AIRS 2014 (Kuching, Malaysia, Dec. 3–5, 2014). Ed. by Azizah Jaafar, Nazlena Mohamad Ali, Shahrul Azman Mohd. Noah, Alan F. Smeaton, Peter Bruza, Zainab Abu Bakar, Nursuriati Jamil, and Tengku Mohd Tengku Sembok. Vol. 8870. Lecture Notes in Computer Science. Springer, 2014, pp. 444–457. ISBN: 978-3-319-12843-6. DOI: 10.1007/978-3-319-12844-3_38 (cit. on pp. 4, 5, 14).

[Ioa+10] Ekaterini Ioannou, Odysseas Papapetrou, Dimitrios Skoutas, and Wolfgang Nejdl. "Efficient Semantic-Aware Detection of Near Duplicate Resources." In: *The Semantic Web: Research and Applications*. 7th Extended Semantic Web Conference. ESWC 2010 (Heraklion, Crete, Greece, May 30–June 3, 2010). Ed. by Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache. Vol. 6089.2. Lecture Notes in Computer Science. Springer, 2010, pp. 136–150. ISBN: 978-3-642-13488-3. DOI: 10.1007/978-3-642-13489-0_10 (cit. on pp. 5, 7).

[JK02] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." In: *ACM Transactions on Information Systems* 20.4 (2002), pp. 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418 (cit. on pp. 5, 20).

[JSS17] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. "Unbiased Learning-to-Rank with Biased Feedback." In: Tenth ACM International Conference on Web Search and Data Mining. WSDM '17 (Cambridge, United Kingdom, Feb. 6–10, 2017). Ed. by Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang. Association for Computing Machinery, 2017, pp. 781–789. ISBN: 978-1-4503-4675-7. DOI: 10.1145/3018661.3018699 (cit. on p. 5).

[Kop+10] Hema Swetha Koppula, Krishna P. Leela, Amit Agarwal, Krishna Prasad Chitrapura, Sachin Garg, and Amit Sasturkar. "Learning URL patterns for webpage de-duplication." In: Third ACM International Conference on Web Search and Web Data Mining. WSDM '10 (New York, New York, United States, Feb. 4–6, 2010). Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu. Association for Computing Machinery, 2010, pp. 381–390. ISBN: 978-1-60558-889-6. DOI: `10.1145/1718487.1718535` (cit. on pp. 1, 5, 7).

[KSL20] Dominik Kowald, Markus Schedl, and Elisabeth Lex. "The Unfairness of Popularity Bias in Music Recommendation: A Reproducibility Study." In: *Advances in Information Retrieval.* 42nd European Conference on IR Research. ECIR 2020 (Lisbon, Portugal, Apr. 14–17, 2020). Ed. by Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins. Vol. 12036.2. Lecture Notes in Computer Science. Springer, 2020, pp. 35–42. ISBN: 978-3-030-45441-8. DOI: `10.1007/978-3-030-45442-5_5` (cit. on p. 5).

[Liu11] Tie-Yan Liu. *Learning to Rank for Information Retrieval.* 1st ed. Springer, 2011. ISBN: 978-3-642-14266-6. DOI: `10.1007/978-3-642-14267-3` (cit. on pp. 1, 4, 14, 15, 19).

[MC07] Donald Metzler and W. Bruce Croft. "Linear feature-based models for information retrieval." In: *Information Retrieval Journal* 10.3 (2007), pp. 257–274. ISSN: 1386-4564. DOI: `10.1007/s10791-006-9019-z` (cit. on pp. 4, 19).

[Mey+03] Dmitriy Meyerzon, Srikanth Shoroff, F. Soner Terek, and Scott Norin. "Method and System for Detecting Duplicate Documents in Web Crawls." US 6,547,829 B1 (United States). Microsoft Corporation. 2003. URL: `https://pimg-fpiw.uspto.gov/fdd/29/478/065/0.pdf` (cit. on pp. 4, 7).

[MJS07] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. "Detecting near-duplicates for web crawling." In: 16th International Conference on World Wide Web. WWW '07 (Banff, Alberta, Canada, May 8–12, 2007). Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. Association for Computing Machinery, 2007, pp. 141–150. ISBN: 978-1-59593-654-7. DOI: `10.1145/1242572.1242592` (cit. on pp. 4, 5, 7).

[MR08] Tom Minka and Stephen E. Robertson. "Selection bias in the LETOR datasets." In: Workshop on Learning to Rank for Information Retrieval. LR4IR 2008. Held in conjunction with SIGIR '08 (Singapore, July 24, 2008). Ed. by Hang Li, Tie-Yan Liu, and Chengxiang Zhai. 2008, pp. 48–

51. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.7837 (cit. on p. 19).

[MSO12] Craig Macdonald, Rodrygo L. T. Santos, and Iadh Ounis. "On the usefulness of query features for learning to rank." In: 21st ACM International Conference on Information and Knowledge Management. CIKM '12 (Maui, Hawaii, United States, Oct. 29–Nov. 2, 2012). Ed. by Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki. Association for Computing Machinery, 2012, pp. 2559–2562. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2398691 (cit. on pp. 11, 12, 30).

[Ngu+16] 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: Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016. CoCo 2016. Co-located with NIPS 2016 (Barcelona, Spain, Dec. 9, 2016). Ed. by Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne. Vol. 1773. CEUR Workshop Proceedings. Sun SITE Central Europe, 2016. URL: http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf (cit. on pp. 4, 10).

[Niu+12] Shuzi Niu, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. "Top-k learning to rank: labeling, ranking and evaluation." In: 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '12 (Portland, Oregon, United States, Aug. 12–16, 2012). Ed. by William R. Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson. Association for Computing Machinery, 2012, pp. 751–760. ISBN: 978-1-4503-1472-5. DOI: 10.1145/2348283.2348384 (cit. on pp. 4, 14).

[OK12] Maile Ohye and Joachim Kupke. *The Canonical Link Relation.* RFC 6596. RFC Editor, Apr. 2012. DOI: 10.17487/RFC6596 (cit. on pp. 5, 9, 16).

[Ova+20] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. "Correcting for Selection Bias in Learning-to-rank Systems." In: The Web Conference 2020. WWW '20 (Taipei, Taiwan, Apr. 20–24, 2020). Ed. by Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen. Association for Computing Machinery, 2020, pp. 1863–1873. ISBN: 978-1-4503-7023-3. DOI: 10.1145/3366423.3380255 (cit. on p. 5).

[Qin+10] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. "LETOR: A benchmark collection for research on learning to rank for information retrieval." In:

Information Retrieval Journal 13.4 (2010), pp. 346–374. ISSN: 1386-4564. DOI: `10.1007/s10791-009-9123-y` (cit. on pp. 4, 10, 14).

[QL13] Tao Qin and Tie-Yan Liu. "Introducing LETOR 4.0 Datasets." In: *Computing Research Repository. CoRR* abs/1306.2597 (2013). ISSN: 2331-8422. arXiv: `1306.2597` (cit. on pp. 2, 4, 10, 12, 13, 21).

[RW94] Stephen E. Robertson and Steve Walker. "Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval." In: 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. SIGIR '94 (Dublin, Ireland, July 3–6, 1994). Ed. by W. Bruce Croft and C. J. van Rijsbergen. Springer, 1994, pp. 232–241. ISBN: 978-3-540-19889-5. DOI: `10.1007/978-1-4471-2099-5_24` (cit. on pp. 2, 19).

[SJ18] Ashudeep Singh and Thorsten Joachims. "Fairness of Exposure in Rankings." In: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '18 (London, United Kingdom, Aug. 19–23, 2018). Ed. by Yike Guo and Faisal Farooq. Association for Computing Machinery, 2018, pp. 2219–2228. ISBN: 978-1-4503-5552-0. DOI: `10.1145/3219819.3220088` (cit. on pp. 5, 30).

[SJ19] Ashudeep Singh and Thorsten Joachims. "Policy Learning for Fairness in Ranking." In: *Advances in Neural Information Processing Systems 32*. Annual Conference on Neural Information Processing Systems 2019. NeurIPS 2019 (Vancouver, British Columbia, Canada, Dec. 8–14, 2019). Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 5427–5437. URL: `https://papers.nips.cc/paper/8782-policy-learning-for-fairness-in-ranking` (cit. on pp. 5, 30).

[Van+20] Gilles Vandewiele, Isabelle Dehaene, György Kovács, Lucas Sterckx, Olivier Janssens, Femke Ongenae, Femke De Backere, Filip De Turck, Kristien Roelens, Johan Decruyenaere, Sofie Van Hoecke, and Thomas Demeester. "Overly Optimistic Prediction Results on Imbalanced Data: Flaws and Benefits of Applying Over-sampling." In: *Computing Research Repository. CoRR* abs/2001.06296 (2020). ISSN: 2331-8422. arXiv: `2001.06296` (cit. on pp. 1, 4, 14).

[VT04] Liwen Vaughan and Mike Thelwall. "Search engine coverage bias: evidence and possible causes." In: *Information Processing and Management* 40.4 (2004), pp. 693–707. ISSN: 0306-4573. DOI: `10.1016/S0306-4573(03)00063-3` (cit. on p. 1).

[Wan+18] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. "Position Bias Estimation for Unbiased Learning to Rank in Personal Search." In: Eleventh ACM International Conference on Web Search and Data Mining. WSDM '18 (Marina Del Rey, California, United States, Feb. 5–9, 2018). Ed. by Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek. Association for Computing Machinery, 2018, pp. 610–618. ISBN: 978-1-4503-5581-0. DOI: 10.1145/3159652.3159732 (cit. on p. 5).

[Wu+10] Qiang Wu, Christopher J. C. Burges, Krysta M. Svore, and Jianfeng Gao. "Adapting boosting for information retrieval measures." In: *Information Retrieval Journal* 13.3 (2010), pp. 254–270. ISSN: 1386-4564. DOI: 10.1007/s10791-009-9112-1 (cit. on pp. 4, 19).

[XL07] Jun Xu and Hang Li. "AdaRank: a boosting algorithm for information retrieval." In: 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '07 (Amsterdam, The Netherlands, July 23–27, 2007). Ed. by Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando. Association for Computing Machinery, 2007, pp. 391–398. ISBN: 978-1-59593-597-7. DOI: 10.1145/1277741.1277809 (cit. on pp. 4, 19).

[Xu+08] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. "Directly optimizing evaluation measures in learning to rank." In: 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '08 (Singapore, July 20–24, 2008). Ed. by Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong. Association for Computing Machinery, 2008, pp. 107–114. ISBN: 978-1-60558-164-4. DOI: 10.1145/1390334.1390355 (cit. on pp. 4, 30).

[YR09] Emine Yilmaz and Stephen Robertson. "Deep versus shallow judgments in learning to rank." In: 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '09 (Boston, Massachusetts, United States, July 19–23, 2009). Ed. by James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel. Association for Computing Machinery, 2009, pp. 662–663. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1572066 (cit. on p. 22).

[Zad04] Bianca Zadrozny. "Learning and evaluating classifiers under sample selection bias." In: Twenty-first International Conference on Machine Learning. ICML '04 (Banff, Alberta, Canada, July 4–8, 2004). Ed. by Carla E. Brodley. Vol. 69. International Conference Proceeding Series. Association for Computing Machinery, 2004. ISBN: 978-1-58113-838-2. DOI: 10.1145/1015330.1015425 (cit. on pp. 1, 4).

[Zeh+17] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. "FA*IR: A Fair Top-k Ranking Algorithm." In: 2017 ACM on Conference on Information and Knowledge Management. CIKM '17 (Singapore, Nov. 6–10, 2017). Ed. by Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li. Association for Computing Machinery, 2017, pp. 1569–1578. ISBN: 978-1-4503-4918-5. DOI: 10.1145/3132847.3132938 (cit. on pp. 5, 30).

[Zhu+14] Yadong Zhu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Shuzi Niu. "Learning for search result diversification." In: 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '14 (Gold Coast, Queensland, Australia, July 6–11, 2014). Ed. by Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin. Association for Computing Machinery, 2014, pp. 293–302. ISBN: 978-1-4503-2257-7. DOI: 10.1145/2600428.2609634 (cit. on p. 30).

Postface

This version of the thesis has been revised and edited after its defence for online publication on `webis.de`, and may thus slightly differ from from the submitted version in wording and grammar.