How Train–Test Leakage Affects Zero-shot Retrieval

Maik Fröbe,[1] Christopher Akiki,[2] Martin Potthast,[2] Matthias Hagen[1]

[1] Martin-Luther-Universität Halle-Wittenberg
[2] Leipzig University

Abstract Neural retrieval models are often trained on (subsets of) the millions of queries of the MS MARCO / ORCAS datasets and then tested on the 250 Robust04 queries or other TREC benchmarks with often only 50 queries. In such setups, many of the few test queries can be very similar to queries from the huge training data—in fact, 69% of the Robust04 queries have near-duplicates in MS MARCO / ORCAS. We investigate the impact of this unintended train–test leakage by training neural retrieval models on combinations of a fixed number of MS MARCO / ORCAS queries, which are very similar to actual test queries, and an increasing number of other queries. We find that leakage can improve effectiveness and even change the ranking of systems. However, these effects diminish the smaller, and thus more realistic, the extent of leakage is in all training instances.

Keywords: Neural information retrieval; Train–test leakage; BERT; T5

1 Introduction

Training transformer-based retrieval models requires large amounts of data unavailable in many traditional retrieval benchmarks [34]. Data-hungry training regimes became possible with the 2019 release of MS MARCO [10] and its 367,013 queries that were subsequently enriched by the ORCAS click log [8] with 10 million queries. Fine-tuning models trained on MS MARCO to other benchmarks or using them without fine-tuning in zero-shot scenarios is often very effective [34, 36, 47]. For example, monoT5 [36] which was trained on MS MARCO data only is the most effective model for the Robust04 document ranking task at the time of writing.[3] Furthermore, the reference implementations of monoT5 and monoBERT [37] in retrieval frameworks such as PyTerrier [32] or Pyserini / PyGaggle [26] all use models trained only on MS MARCO by default. However, when MS MARCO was officially split into train and test data, cross-benchmark use was not anticipated, so that MS MARCO's training queries may overlap with the test queries of other datasets (e.g., Robust04). We investigate the impact of such a train–test leakage by training neural models on MS MARCO document ranking data with different proportions of controlled leakage to Robust04 and the TREC 2017 and 2018 Common Core tracks as test datasets.

[3] https://paperswithcode.com/sota/ad-hoc-information-retrieval-on-trec-robust04

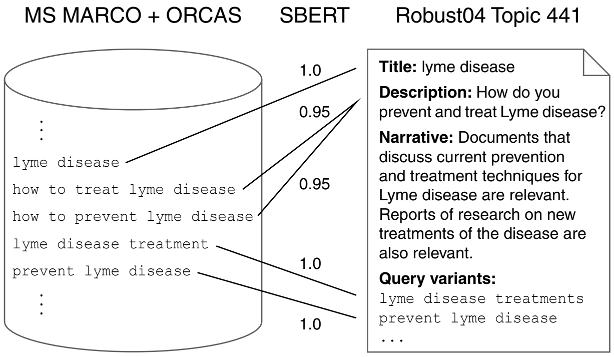
MS MARCO + ORCAS
SBERT
Robust04 Topic 441
⋮
lyme disease
how to treat lyme disease
how to prevent lyme disease
lyme disease treatment
prevent lyme disease
⋮
1.0
0.95
0.95
1.0
1.0
Title: lyme disease
Description: How do you prevent and treat Lyme disease?
Narrative: Documents that discuss current prevention and treatment techniques for Lyme disease are relevant. Reports of research on new treatments of the disease are also relevant.
Query variants:
lyme disease treatments
prevent lyme disease
...

Figure 1. MS MARCO / ORCAS queries with high Sentence-BERT (SBERT) similarity to Robust04 Topic 441.

To identify probably leaking queries, we run a semantic nearest-neighbor search using Sentence-BERT [38] and compare each MS MARCO / ORCAS query to the title, description, and manual query variants [3, 4] of the topics in Robust04 and the TREC 2017 and 2018 Common Core tracks. Figure 1 illustrates this procedure for Topic 441 (`lyme disease`) from Robust04. Our manual review of the leakage candidates shows that 69% to 76% of the topics have near-duplicates in MS MARCO / ORCAS. To analyze the effect of this potential train–test leakage on neural retrieval models, we create three types of training datasets per test corpus, in variants with 1,000 to 128,000 training instances (query + (non-)relevant document): (1) a fixed number of instances derived from test queries from the test corpora (1000 for Robust04 and 200 for each of the two Common Core tracks), augmented by other random non-leaking MS MARCO / ORCAS instances to simulate an upper bound on train–test leakage effects, (2) a fixed number of leaking MS MARCO / ORCAS instances (1000 for Robust04 and 200 each for the two Common Core tracks) supplemented by other random non-leaking instances, and (3) random MS MARCO / ORCAS instances, ensuring that no train–test leakage candidates are included.

In our experiments, we observe leakage-induced improvements in effectiveness for Robust04 and the two Common Core tracks, which can even change the ranking of systems. However, the average improvements in overall effectiveness are often not significant and decrease as the proportion of leakage in the training data becomes smaller and more representative of realistic training scenarios. However, given the swaps in system rankings as well as leakage effects on search results that we observed, we do advise caution: In any case, a rigorous experimental setup demands for maximizing its reliability, so that train–test leaks should still be avoided.[4]

[4] All code and data is publicly available at https://github.com/webis-de/SPIRE-22.

2 Background and Related Work

Disjoint training, validation, and test datasets are essential to properly evaluate the effectiveness of machine learning models [7]. Duplication between training and test data can lead to incorrectly high "effectiveness" by memorizing instances rather than learning the target concept. In practice, however, the training and test data often still contain redundancies. For text data, paraphrases, synonyms, etc., can be especially problematic, resulting in train–test leaks [19, 24, 29]. For instance, the training and test sets of the ELI5 dataset [13] for question answering were created using TF-IDF as a heuristic to eliminate redundancies between them. This proved insufficient as 81% of the test questions turned out to be paraphrases of training questions, which clearly favored models that memorized the training data [24]. Recently, Zhan et al. [46] found that 79% of the TREC 2019 Deep Learning track topics have similar or duplicated queries in the training data and proposed new data splits to evaluate the interpolation and extrapolation effectiveness of models. However, not all types of train–test leaks are unintentional. The TREC 2017 and 2018 Common Core tracks [1] intentionally reused topics from Robust04 to allow participants to use the relevance judgments for training. Indeed, approaches trained on the Robust04 judgments were more effective than others [1]. In this paper, we study whether a similar effect can be observed for unintentional leakage from the large MS MARCO and ORCAS datasets.

Training retrieval models on MS MARCO and applying them to another corpus is a form of transfer learning [34]. Transfer learning is susceptible to train–test leakage since the train and test data are often generated independently without precautions to prevent leaks [6]. Research on leakage in transfer learning focuses on membership inference [35, 41] (predicting if a model has seen an instance during training) and property inference [2, 17] (predicting properties of the training data). Both inferences rely on the observation that neural models may memorize some training instances to generalize through interpolation [5, 7] and to similar test instances [15, 16]. It is unclear whether and how neural retrieval models in a transfer learning scenario are affected by leakage. Memorized relevant instances might reduce effectiveness for different test queries while improving it for similar queries, like the examples in Figure 1. We take the first steps to investigate the effects of such a train–test leakage.

When the target corpus contains only few training instances, transferred retrieval models are often more effective without fine-tuning, in a zero-shot setting [47]; for instance, when training on MS MARCO and testing on TREC datasets [34, 36, 47]. A frequently used target TREC dataset is Robust04 [42] with 250 topics and a collection of 528,155 documents published between 1989 and 1996 by the Financial Times, the Federal Register, the Foreign Broadcast Information Service, and the LA Times.[5] Later, the TREC Common Core track 2017 [1] reused 50 of the 250 Robust04 topics on the New York Times Anno-

[5] https://trec.nist.gov/data/cd45/index.html

tated Corpus [39][6] (1,864,661 documents published between 1987 and 2007) and the Common Core track 2018 reused another 25 Robust04 topics (and introduced 25 new topics) on the Washington Post Corpus[7] (595,037 documents published between 2012 and 2017). At the Robust04 track, 311,410 relevance judgments were collected, 30,030 at the TREC 2017 Common Core track, and 26,233 at the TREC 2018 Common Core track. Interestingly, every Robust04 topic and every topic from the Common Core tracks 2017 and 2018 was augmented with at least eight query variants compiled by expert searchers, and made available as an additional resource [3, 4].

Research on paraphrase detection [12, 43] and semantic question matching [40] is of great relevance to the identification of potentially leaking queries between training and test data. Reimers and Gurevych [38] and Lin et al. [28] showed that pooling or averaging the output of contextual word embeddings of pre-trained transformer encoders like BERT [11] is not suited for paraphrase detection—both, with respect to efficiency and accuracy. Sentence-BERT [38] solves this issue by adopting a BERT-based triplet network structure and a contrastive loss that attempts to learn a global and a local structure suited for detecting semantically related sentences. We therefore use Sentence-BERT in a version specifically trained for paraphrase detection to identify leaking queries.

3 Identifying Leaking Queries

To examine the impact of possible leaks from MS MARCO / ORCAS to the TREC datasets Robust04 and Common Core 2017 and 2018, we compare the former's queries (367,013 plus 10 million) to the 275 topics of the latter three. Since lexical similarity may not be sufficient, as indicated by the ELI5 issue [24] mentioned above, we compute semantic similarity scores using Sentence-BERT [38].[8] We store the Sentence-BERT embeddings of all MS MARCO and ORCAS queries in two Faiss indexes [23] and query them for the 100 nearest neighbors (exact; cosine similarity) of each topic's title, description, and query variants.

To determine the threshold for the Sentence-BERT similarity score beyond which we consider a query a source of leakage for a topic, one human annotator assessed whether a query is leaking for a TREC topic (title, description, query variants) for a stratified sample of 100 pairs of queries and topics with a similarity above 0.8. Against these manual judgments, a similarity threshold of 0.91 is the lowest that yields a 0.9 precision for deciding that a query is leaking for a topic. Table 1 shows the number of topics for which queries above this threshold can be found. From MS MARCO and ORCAS combined, 3,960 queries are leakage candidates for one of 181 Robust04 topics (72% of the 250 topics). From the two Common Core tracks, 37 and 38 topics have leakage candidates (76% of the 50 topics, respectively)—high similarities mostly against the query variants.

[6] https://catalog.ldc.upenn.edu/LDC2008T19

[7] https://trec.nist.gov/data/wapost/

[8] Of the available pre-trained Sentence-BERT models, we use the paraphrase detection model: https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2

Table 1. Number of topics (T) in Robust04 and the TREC 2017 and 2018 Common Core tracks for which similar queries (number as Q) in MS MARCO (MSM) and the union of MSM and ORCAS (+ORC) exist in terms of the query having a Sentence-BERT score > 0.91 against the topic's title, description, or a query variant.

Candidates	**Robust04**				**Core 2017**				**Core 2018**			
	MSM		+ ORC		MSM		+ ORC		MSM		+ ORC	
	T	Q	T	Q	T	Q	T	Q	T	Q	T	Q
Title	33	83	140	1,775	2	12	23	176	2	2	21	110
Description	2	3	8	50	0	0	0	0	0	0	1	2
Variants	45	116	167	3,356	6	16	38	602	9	26	38	973
Union	53	151	181	3,960	7	18	38	645	9	26	38	973

Some of these leakage candidates still are false positives (threshold precision of 0.9). To only use actual leaking queries in our train–test leakage experiments, two annotators manually reviewed the 5 most similar candidates per topic above the 0.91 threshold (a total of 827 candidates; not all topics had 5 candidates). Given the title, description, and narrative of a topic, the annotators labeled the similarity of a query to the topic title according to Jansen et al.'s reformulation types [22]: a query can be *identical* to the topic title (differences only in inflection or word order), be a *generalization* (subset of words), a *specialization* (superset of words), a *reformulation* (some synonymous terms), or it can be on a *different topic*. An initial kappa test on 103 random of the 827 candidates showed moderate agreement (Cohen's kappa of 0.59; 103 queries: we aimed for 100 but included all queries for a topic when one was selected). After discussing the 103 cases with the annotators, they agreed on all cases and we had them each independently label half of the remaining 724 candidates. Table 2 shows the annotation results: 172 topics of Robust04 (i.e., 69%) have manually verified leaking queries (648 total), as well as 37 topics of Common Core 2017 (74%) and 38 of Common Core 2018 (76%). A large portion of the true-positive leaking queries are identical to or specializations of a topic's title (57.5% of 721). In our below train–test leakage experiments, we only use manually verified true-positive leaking queries as the source of leakage from MS MARCO / ORCAS.

4 Experimental Analysis

Focusing on zero-shot settings, we train neural retrieval models on specifically designed datasets to assess the effect of train–test leakage from MS MARCO / ORCAS to Robust04 and TREC 2017 and 2018 Common Core. We analyze the models' effectiveness in five-fold cross-validation experiments, report detailed results for varying training set sizes for monoT5 (which has the highest effectiveness in our experiments), and study the effects of leaked instances on the retrieval scores and the resulting rankings.

Table 2. Statistics of the 827 manually annotated leakage candidate queries. (a) Number of true and false candidates. (b) Annotated query reformulation types.

(a) Manually annotated candidates.

Corpus	Candidates	Queries	Topics
Robust04	true	648	172
	false	93	53
Core 2017	true	138	37
	false	21	11
Core 2018	true	157	38
	false	19	7

(b) Reformulation types.

Type	Queries
Identical	187
Generalization	124
Specialization	228
Reformulation	182
Different Topic	106

Training Datasets. For each of the three test scenarios (Robust04 and the two Common Core scenarios), we construct three types of training datasets: (1) 'No Leakage' with random non-leaking queries (balanced between MS MARCO and ORCAS as in previous experiments [8]), (2) 'MSM Leakage' with a fixed number of random manually verified leaking queries from MS MARCO / ORCAS (500 queries for Robust04, 100 queries for Common Core) supplemented by no-leakage queries until a desired size is reached, and (3) 'Test Leakage' with a fixed number of queries from the actual test data (500 for Robust04, 100 for Common Core; oversampling: each topic twice (but different documents) to match 'MSM Leakage') supplemented by no-leakage queries until a desired size is reached. 'Test Leakage' is meant as an "upper bound" for any train–test leakage effect.

For each type, we construct datasets with 1,000 to 128,000 instances (500 to 64,000 queries; each with one relevant and one non-relevant document). Since MS MARCO / ORCAS queries only have annotated relevant documents, we follow Nogueira et al. [36] and sample "non-relevant" instances from the top-100 BM25 results for such queries. For the 'Test Leakage' scenario, we use the actual TREC judgments to sample the non-/relevant instances. In our 72 training datasets (3 test scenarios, 3 types, 8 sizes), the number of leaked instances is held constant to analyze the effect of a decreasing (and thus more realistic) ratio of leakage. Larger training data would result in more costly training, but our chosen sizes already suffice to observe a diminishing effect of train–test leakage.

Trained Models. For our experimental analyses, we use models based on monoBERT [37] and monoT5 [36] as implemented in PyGaggle [26], and models based on Duet [33], KNRM [44], and PACRR [21] as implemented in Capreolus [45] (default configurations each). In pilot experiments with 32,000 'No Leakage' instances, these models had higher nDCG@10 scores on Robust04 than CEDR [30], HINT [14], PARADE [25], and TK [20]. Following Nogueira et al. [36], we use the base versions of BERT and T5 to spend the computational budget on training many models instead of a single large one. Since the training is not deterministic, each model is trained on each of the 72 training sets five times for one epoch with varying seeds (used to shuffle the training queries; configured in PyTorch). We use `ir_datasets` [31] for data wrangling and, following previously suggested

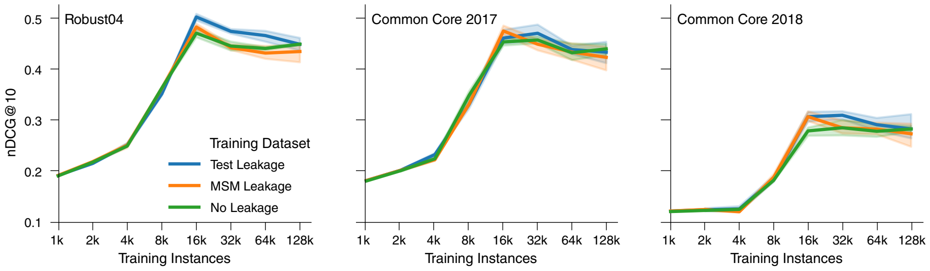

Figure 2. Effectiveness of monoT5 measured as nDCG@10 on the topics with leakage (172 topics for Robust04, 37 and 38 for the 2017 and 2018 editions of the Common Core track). Models trained on datasets of varying size with no leakage (No), leakage from MS MARCO / ORCAS (MSM), or leakage from the test data (Test).

training regimes [36, 37, 45], pass the relevant and the non-relevant document of a query consecutively to a model in the same batch during training. During inference, all models re-rank the top-100 BM25 results (Capreolus, default configuration) and we break potential score ties in rankings via alphanumeric ordering by document ID (with random IDs, this leads to a random distribution for other document properties such as text length [27]).

Leakage-Induced nDCG Improvements for MonoT5. Figure 2 shows the average nDCG@10 of monoT5, the most effective model in our experiments, for different training set sizes, tested only on topics with leaked queries. For small training sets, monoT5 achieves rather low nDCG@10 values and cannot exploit the leakage. The nDCG@10 increases with more training data on all benchmarks, peaking at 16,000 or 32,000 instances. At the peaks, monoT5 trained with leakage is more effective than without, and training on test leakage leads to a slightly higher nDCG@10 than leakage from MS MARCO / ORCAS (MSM). However, the difference between test and MSM leakage is larger for Robust04 (with some documents published as early as 1989) compared to the newer Common Core tracks (with documents published closer to the publication date of MS MARCO). On the Common Core data, MSM leakage is almost as effective as test leakage.

Leakage-Induced Effectiveness Improvements for Other Models. We employ a five-fold cross-validation setup for Duet, KNRM, monoBERT, monoT5, and PACRR to study whether leakage-induced effectiveness improvements can also be observed for other models when a grid search in the cross-validation setup can choose the training set size with the highest leakage effect for each model. We report the effectiveness of the models as nDCG@10, Precision@1, and the mean first rank of a relevant document (MFR) [18].[9] While effectiveness scores measured via nDCG@10 and Precision@1 have the property that higher values are better (a score of 1 indicates "best" effectiveness), for MFR, lower scores

[9] We use MFR instead of the mean reciprocal rank (MRR) as suggested by Fuhr [18]. His criticism of MRR was recently supported by further empirical evidence [48].

Table 3. Effectiveness on Robust04 (R04) as nDCG@10, mean first rank of a relevant document (MFR), and Precision@1 (Prec@1) in a five-fold cross-validation setup on all test topics. Models are trained with no leakage (None), leakage from MS MARCO / ORCAS (MSM), or leakage from the test data (Test). Highest scores in bold; † marks Bonferroni-corrected significant differences to the no-leakage baseline (Student's t-test, $p = 0.05$). Model order swaps induced by MSM leakage in red.

Model	**nDCG@10 on R04**			**MFR on R04**			**Prec@1 on R04**		
	None	MSM	Test	None	MSM	Test	None	MSM	Test
Duet [33]	0.201	0.198	**0.224**†	2.420	2.682	**2.340**	0.297	0.261	**0.304**
KNRM [44]	0.194	0.214†	**0.309**†	2.348	2.309	**1.976**†	0.293	0.313	**0.329**
monoBERT [37]	0.394	0.373†	**0.396**	1.688	1.725	**1.639**	0.434	**0.454**	0.414
monoT5 [36]	0.461	0.457	**0.478**†	1.443	**1.416**	1.417	0.562	0.578	**0.590**
PACRR [21]	0.382	0.364†	**0.391**	1.663	1.604	**1.579**†	0.458	0.462	**0.502**

are better—but still a score of 1 is the best case indicating that the document on rank 1 always is relevant. In all effectiveness evaluations, we conduct significance tests via Student's t-test ($p = 0.05$) with Bonferroni correction for multiple testing as implemented in PyTerrier [32].

Table 3 shows the five-fold cross-validated effectiveness on Robust04 for the five models when optimizing each fold for nDCG@10, MFR, or Precision@1 in a grid search. Models trained on test leakage almost always are more effective than their no-leakage counterparts (exception: Precision@1 of monoBERT) and actual test leakage usually helps more than leakage from MS MARCO / ORCAS (MSM; exceptions: MFR of monoT5 and Precision@1 of monoBERT). Overall, on Robust04, models trained with MSM leakage are often less effective than their no-leakage counterparts (e.g., the nDCG@10 of monoBERT). A possible explanation might be the large time gap between the Robust04 document publication dates (documents published between 1987 and 2007) and the MS MARCO data (crawled in 2018). A similar observation was made during the TREC 2021 Deep Learning track [9]. The transition from Version 1 of MS MARCO to Version 2 (crawled three years later) caused some models to prefer older documents since they saw old documents as positive instances and newer ones as negative instances during training. Still, MSM leakage can lead to swaps in model ranking on Robust04. For instance, KNRM trained with MSM leakage achieves a higher nDCG@10 and Precision@1 than Duet without leakage, while KNRM trained without leakage is less effective than Duet.

Table 4 shows the five-fold cross-validated effectiveness on the TREC 2017 Common Core track for the five models when optimizing each fold for nDCG@10, MFR, or Precision@1 in a grid search. In contrast to Robust04, more models improve their effectiveness when trained with MSM leakage as the time gap between the New York Times Annotated Corpus and MS MARCO is smaller than for Robust04. MonoT5 with actual test leakage is the most effective model for all three measures, and monoT5 trained on MSM leakage is more effective

Table 4. Effectiveness on Common Core 2017 (CC17) as nDCG@10, mean first rank of a relevant document (MFR), and Precision@1 (Prec@1) in a five-fold cross-validation setup on all test topics. Models are trained with no leakage (None), leakage from MS MARCO / ORCAS (MSM), or leakage from the test data (Test). Highest scores in bold; † marks Bonferroni-corrected significant differences to the no-leakage baseline (Student's t-test, $p = 0.05$). Model order swaps induced by MSM leakage in red.

Model	**nDCG@10 on R04**			**MFR on R04**			**Prec@1 on R04**		
	None	MSM	Test	None	MSM	Test	None	MSM	Test
Duet [33]	0.374	0.373	**0.376**	1.620	1.512	**1.485**	0.500	0.480	**0.540**
KNRM [44]	0.316	**0.343**†	0.330	1.587	**1.512**	1.568	0.480	**0.520**	0.480
monoBERT [37]	0.402	0.407	**0.419**	1.625	**1.605**	1.634	**0.480**	0.460	0.460
monoT5 [36]	0.445	0.464	**0.490**†	1.363	1.384	**1.359**	0.660	0.620	**0.680**
PACRR [21]	0.406	0.403	**0.413**	**1.390**	1.515	1.546	0.540	0.520	**0.580**

Table 5. Effectiveness on Common Core 2018 (CC18) as nDCG@10, mean first rank of a relevant document (MFR), and Precision@1 (Prec@1) in a five-fold cross-validation setup on all test topics. Models are trained with no leakage (None), leakage from MS MARCO / ORCAS (MSM), or leakage from the test data (Test). Highest scores in bold; † marks Bonferroni-corrected significant differences to the no-leakage baseline (Student's t-test, $p = 0.05$). Model order swaps induced by MSM leakage in red.

Model	**nDCG@10 on R04**			**MFR on R04**			**Prec@1 on R04**		
	None	MSM	Test	None	MSM	Test	None	MSM	Test
Duet [33]	0.285	**0.301**	0.295	1.993	**1.812**	2.231	0.320	**0.380**	0.260
KNRM [44]	0.201	**0.256**†	0.238†	3.099	**2.768**	3.125	0.100	**0.160**	0.140
monoBERT [37]	0.364	**0.380**	0.366	1.810	**1.683**	1.719	0.460	**0.560**	0.460
monoT5 [36]	0.417	**0.448**	0.445	1.577	**1.503**	1.512	0.480	**0.540**	**0.540**
PACRR [21]	0.376	**0.406**	0.393	1.649	**1.383**†	1.485	0.520	**0.560**	0.540

than the no-leakage counterpart in nDCG@10 and MFR. MSM leakage also may cause model order swaps at higher positions in the systems' nDCG@10 ordering: monoBERT with MSM leakage would slightly overtake PACRR. Still, most of the effectiveness improvements on this dataset caused by MSM leakage or test leakage are not significant (exception: the nDCG@10 differences for monoT5 with test leakage and KNRM with MSM leakage to the no-leakage counterparts).

Table 5 shows the five-fold cross-validated effectiveness on the TREC 2018 Common Core track for the five models when optimizing each fold for nDCG@10, MFR, or Precision@1 in a grid search. In contrast to Robust04 and the 2017 edition of the Common Core track, training with MSM leakage improves the effectiveness in all cases for all three measures. While most of the leakage-induced effectiveness improvements are not statistically significant, the model order even changes on the top MFR position, where PACRR with MSM leakage would overtake monoT5 without leakage.

Table 6. Mean rank of the (leaked) relevant training documents (± standard deviation) for models trained with and without leakage from MS MARCO / ORCAS (MSM leakage) or from the test data (test leakage). Ranks macro-averaged over all topics for test leakage and over all topics with leaking queries for MSM leakage.

	Model	Robust04		Common Core 17		Common Core 18	
		With	Without	With	Without	With	Without
MSM leak.	Duet	41.70 ±45.88	46.79 ±46.51	34.52 ±32.67	35.98 ±32.93	43.39 ±33.52	45.67 ±32.99
	KNRM	82.36 ±31.88	84.74 ±30.15	43.24 ±31.74	43.68 ±31.50	53.12 ±32.14	53.45 ±32.14
	monoBERT	23.08 ±28.71	23.58 ±28.22	46.97 ±34.95	47.11 ±35.49	41.79 ±36.16	42.48 ±36.39
	monoT5	20.13 ±26.77	20.15 ±26.64	35.68 ±31.69	36.46 ±31.88	29.86 ±28.24	30.31 ±28.27
	PACRR	42.41 ±44.86	42.43 ±44.67	35.79 ±33.71	36.28 ±33.83	34.76 ±36.45	35.70 ±36.87
Test leak.	Duet	90.04 ±26.98	90.65 ±26.41	45.78 ±30.03	46.55 ±30.34	46.31 ±29.85	46.35 ±30.13
	KNRM	89.95 ±26.43	91.20 ±25.24	47.37 ±32.80	47.49 ±32.81	50.53 ±32.40	50.13 ±32.26
	monoBERT	47.01 ±31.84	47.39 ±31.80	46.64 ±31.51	47.12 ±31.51	43.19 ±31.51	44.04 ±31.66
	monoT5	45.28 ±32.09	45.37 ±31.96	46.35 ±31.47	47.45 ±31.83	40.16 ±31.18	40.95 ±31.24
	PACRR	80.89 ±34.05	82.60 ±33.07	53.59 ±31.49	52.91 ±31.26	52.25 ±32.69	52.28 ±32.32

Discussion. The results in Tables 3–5 show that leakage from MS MARCO / ORCAS (MSM) can have an impact on the retrieval effectiveness, even when only a small number of instances are leaked, as in our experiments. While the changes on Robust04 are rather negligible, the impact is larger for the Common Core tracks with document publication dates closer to the ones from MS MARCO. Interestingly, MSM leakage-induced nDCG@10 improvements sometimes can lead to swaps in model ordering despite the improvements not being significant in most cases. This exemplifies that experimental effectiveness comparisons might be invalid when some models had access to leaked instances during training.

Memorization of Leaked Instances. To analyze whether the models memorize leaked instances, we compare the retrieval scores and resulting ranks of leaked documents in the test rankings when training includes or does not include leakage. For leaked documents not returned in the top-100 BM25 results—the models only re-rank these—, we determine a hypothetical rank by calculating the score of this document for the query and inserting the document at the corresponding rank in the to-be-re-ranked 100 documents (including breaking score-ties by document ID). Each leaked document thus has a maximal rank of 101.

Table 6 shows the mean rank of relevant documents when they were included during training (with leakage) or not (without leakage). Models perfectly memorizing their positive training instances (i.e., relevant documents for test queries) would rank these documents at substantially higher positions than models that did not see the same instance during training. However, while the mean rank of leaked relevant documents improves for most cases, the improvement is mostly negligible. For instance, the mean rank of leaked relevant documents for the very effective monoT5 and monoBERT models improves only slightly compared to their no-leakage counterparts on all three corpora. But the difference increases

Table 7. Mean retrieval score of the (leaked) relevant training documents (± standard deviation; higher scores = "more relevant") for models trained with / without leakage from MS MARCO / ORCAS (MSM) or the test data (Test). Scores macro-averaged over all topics for test leakage and over all topics with leaking queries for MSM leakage.

	Model	**Robust04**		**Common Core 17**		**Common Core 18**	
		With	Without	With	Without	With	Without
MSM leak.	Duet	0.89 ±1.22	0.78 ±1.18	0.52 ±1.18	0.47 ±1.17	0.16 ±0.79	0.09 ±0.75
	KNRM	-2.06 ±3.64	-2.58 ±3.43	-2.53 ±3.75	-3.08 ±3.52	-2.32 ±3.24	-2.78 ±3.01
	monoBERT	-0.75 ±0.44	-0.72 ±0.41	-0.88 ±0.49	-0.85 ±0.47	-0.92 ±0.50	-0.89 ±0.48
	monoT5	-1.05 ±1.14	-1.19 ±1.20	-1.32 ±1.26	-1.48 ±1.31	-1.51 ±1.34	-1.65 ±1.38
	PACRR	2.59 ±3.25	2.29 ±3.11	2.78 ±3.35	2.46 ±3.20	2.25 ±3.08	1.95 ±2.96
Test leak.	Duet	0.07 ±0.61	-0.11 ±0.56	0.22 ±0.68	-0.01 ±0.66	0.30 ±0.69	0.09 ±0.68
	KNRM	-2.71 ±3.79	-3.41 ±3.71	-2.78 ±3.65	-3.28 ±3.63	-3.08 ±4.14	-3.59 ±4.14
	monoBERT	-0.91 ±0.45	-1.04 ±0.53	-0.85 ±0.44	-0.90 ±0.48	-0.85 ±0.46	-0.92 ±0.49
	monoT5	-1.70 ±1.24	-2.37 ±1.53	-1.47 ±1.09	-1.98 ±1.37	-1.52 ±1.24	-2.01 ±1.50
	PACRR	2.31 ±4.27	1.92 ±3.09	1.83 ±4.40	1.98 ±3.13	2.66 ±3.31	2.26 ±3.23

(still rather negligibly, though) on the corpora on which leakage was more effective. In combination with the high standard deviations, one can hardly see memorization effects for the positions of leaked relevant documents in the final rankings. We thus also inspect the retrieval scores of the leaked documents.

Table 7 shows the mean retrieval score of the relevant documents when they were included during training (with leakage) or not (without leakage). Models that memorize the leaked relevant training documents should increase their score, and we indeed observe that the retrieval score of leaked relevant documents increases in most cases compared to their no-leakage counterpart (exception: monoBERT for MSM leakage and PACRR for test leakage from Common Core 2017). The difference between the score differences of leakage models and non-leakage models is larger for leakage from the test data than for MSM leakage in 13 of the 15 cases (with a maximum difference for monoT5 from a test leakage difference of $0.67 = 2.37 - 1.70$ to an MSM leakage difference of $0.14 = 1.19 - 1.05$). To investigate the "full picture" with respect to also negative leaked instances (i.e., non-relevant documents), we next also study the rank offsets between the positive and the negative leaked instances.

Table 8 shows the macro-averaged increase in the rank difference of the leaked relevant and non-relevant documents between models trained with and without leakage. The leakage increases the rank offset for all five analyzed models (e.g., 6.4 ranks for Duet on Robust04 with MSM leakage). Interestingly, an in-depth inspection showed that most of the increased differences are caused by lowered ranks of the leaked non-relevant documents (e.g., 2 ranks lower for monoT5) while the leaked relevant documents improve their ranks only slightly (e.g., 0.3 ranks higher for monoT5).

Table 8. Macro-averaged increase of the rank-offset between the leaked relevant and non-relevant documents (± standard deviation) for models trained on MSM leakage (*Δ* on MSM) or on test leakage (*Δ* on Test) over the no-leakage variants.

Model	***Δ* on MSM**			***Δ* on Test**		
	R04	C17	C18	R04	C17	C18
Duet	6.378 ±32.15	3.119 ±19.17	2.647 ±19.23	0.809 ±17.69	1.430 ±19.33	1.023 ±20.10
KNRM	0.640 ±19.22	0.979 ±15.23	0.398 ±14.55	1.335 ±11.75	0.012 ±14.92	0.140 ±15.18
monoBERT	0.692 ±17.97	0.076 ±17.19	0.369 ±20.04	3.886 ±20.39	0.980 ±17.44	3.497 ±25.98
monoT5	0.443 ±8.60	0.390 ±9.28	0.789 ±9.91	3.443 ±19.96	2.242 ±9.84	1.819 ±10.98
PACRR	0.043 ±19.30	0.764 ±10.93	0.452 ±12.38	1.952 ±17.71	0.271 ±16.96	0.753 ±14.16

Discussion. Overall, our results in Tables 6–8 indicate that memorization happens but has little impact. Larger memorization effects might be desirable in practical scenarios where a retrieval system that memorizes good results can simply present them when the same query is submitted again. However, for empirical evaluations in scientific publications or at shared tasks, (unintended) leakage memorization at a larger scale might still lead to unwanted outcomes.

5 Conclusion

Our study of train–test leakage effects for neural retrieval models was inspired by the observation that 69% of the Robust04 topics, a dataset often used to test neural models, have very similar queries in the MS MARCO / ORCAS datasets, that are often used to train neural models. At first glance, this overlap might seem alarming since train–test leakage is known to invalidate experimental evaluations. We thus analyzed train–test leakage effects for five neural models (Duet, KNRM, monoBERT, monoT5, and PACRR) in scenarios with different amounts of leakage. While our experiments show leakage-induced effectiveness improvements that may even lead to swaps in model ranking, our overall results are reassuring: the effects on nDCG@10 are rather small and not significant in most cases. They also become smaller the smaller (and more realistic) the amount of leakage among all training instances is. Still, even if only a few nDCG@10 differences were significant, we noticed a memorization effect: the rank offset between leaked relevant and non-relevant documents increased on all scenarios.

Train–test leakage should thus still be avoided in academic experiments but the practical consequences for real search engines might be different. The observed increased rank offset might be a highly desirable effect when presuming that queries already seen during training are submitted again after a model has been deployed to production. An interesting direction for future research is to enlarge our experiments to investigate more of the few cases where train–test leakage slightly reduced the effectiveness.

Bibliography

[1] Allan, J., Harman, D., Kanoulas, E., Li, D., Gysel, C., Voorhees, E.: TREC 2017 Common Core track overview. In: Proc. of TREC 2017, vol. 500-324, NIST (2017)

[2] Ateniese, G., Mancini, L., Spognardi, A., Villani, A., Vitali, D., Felici, G.: Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. Int. J. Secur. Networks **10**(3), 137–150 (2015)

[3] Benham, R., Gallagher, L., Mackenzie, J., Damessie, T., Chen, R., Scholer, F., Moffat, A., Culpepper, J.: RMIT at the 2017 TREC CORE track. In: Proc. of TREC 2017, NIST Special Publication, vol. 500-324, NIST (2017)

[4] Benham, R., Gallagher, L., Mackenzie, J., Liu, B., Lu, X., Scholer, F., Culpepper, J., Moffat, A.: RMIT at the 2018 TREC CORE track. In: Proc. of TREC 2018, NIST Special Publication, vol. 500-331, NIST (2018)

[5] Berthelot, D., Raffel, C., Roy, A., Goodfellow, I.: Understanding and improving interpolation in autoencoders via an adversarial regularizer. In: Proc. of ICLR 2019, OpenReview.net (2019)

[6] Chen, C., Wu, B., Qiu, M., Wang, L., Zhou, J.: A comprehensive analysis of information leakage in deep transfer learning. CoRR **abs/2009.01989** (2020)

[7] Chollet, F.: Deep Learning with Python. Simon and Schuster (2021)

[8] Craswell, N., Campos, D., Mitra, B., Yilmaz, E., Billerbeck, B.: ORCAS: 20 million clicked query-document pairs for analyzing search. In: Proc. of CIKM 2020, pp. 2983–2989, ACM (2020)

[9] Craswell, N., Mitra, B., Yilmaz, E., Campos, D.: Overview of the TREC 2021 Deep Learning Track. In: Voorhees, E.M., Ellis, A. (eds.) Notebook, NIST (2021)

[10] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.: Overview of the TREC 2019 Deep Learning Track. In: Proc. of TREC 2019, NIST Special Publication, NIST (2019)

[11] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of NAACL 2019, pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota (2019)

[12] Dolan, W.B., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. In: Proc. of the Third International Workshop on Paraphrasing (IWP 2005) (2005)

[13] Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., Auli, M.: ELI5: Long form question answering. In: Proc. of ACL 2019, pp. 3558–3567, ACL (2019)

[14] Fan, Y., Guo, J., Lan, Y., Xu, J., Zhai, C., Cheng, X.: Modeling diverse relevance patterns in ad-hoc retrieval. In: Proc. of SIGIR 2018, pp. 375–384, ACM (2018)

[15] Feldman, V.: Does learning require memorization? A short tale about a long tail. In: Proc. of STOC 2020, pp. 954–959, ACM (2020)

[16] Feldman, V., Zhang, C.: What neural networks memorize and why: Discovering the long tail via influence estimation. In: Proc. of NeurIPS 2020 (2020)

[17] Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proc. of CCS 2015, pp. 1322–1333, ACM (2015)

[18] Fuhr, N.: Some common mistakes in IR evaluation, and how they can be avoided. SIGIR Forum **51**(3), 32–41 (2017)

[19] He, H., Garcia, E.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)

[20] Hofstätter, S., Zlabinger, M., Hanbury, A.: Interpretable & time-budget-constrained contextualization for re-ranking. In: Proc. of ECAI 2020, Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 513–520, IOS Press (2020)

[21] Hui, K., Yates, A., Berberich, K., Melo, G.: PACRR: A position-aware neural IR model for relevance matching. In: Proc. of EMNLP 2017, pp. 1049–1058, ACL (2017)

[22] Jansen, B., Booth, D., Spink, A.: Patterns of query reformulation during web searching. J. Assoc. Inf. Sci. Technol. **60**(7), 1358–1371 (2009)

[23] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2021)

[24] Krishna, K., Roy, A., Iyyer, M.: Hurdles to progress in long-form question answering. In: Proc. of NAACL 2021, pp. 4940–4957, ACL (2021)

[25] Li, C., Yates, A., MacAvaney, S., He, B., Sun, Y.: PARADE: Passage representation aggregation for document reranking. CoRR **abs/2008.09093** (2020)

[26] Lin, J., Ma, X., Lin, S., Yang, J., Pradeep, R., Nogueira, R.: Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In: Proc. of SIGIR 2021, pp. 2356–2362, ACM (2021)

[27] Lin, J., Yang, P.: The impact of score ties on repeatability in document ranking. In: Proc. of SIGIR 2019, pp. 1125–1128, ACM (2019)

[28] Lin, S., Yang, J., Lin, J.: Distilling dense representations for ranking using tightly-coupled teachers. CoRR **abs/2010.11386** (2020)

[29] Linjordet, T., Balog, K.: Sanitizing synthetic training data generation for question answering over knowledge graphs. In: Proc. of ICTIR 2020, pp. 121–128, ACM (2020)

[30] MacAvaney, S., Yates, A., Cohan, A., Goharian, N.: CEDR: Contextualized embeddings for document ranking. In: Proc. of SIGIR 2019, pp. 1101–1104, ACM (2019)

[31] MacAvaney, S., Yates, A., Feldman, S., Downey, D., Cohan, A., Goharian, N.: Simplified data wrangling with `ir_datasets`. In: Proc. of SIGIR 2021, pp. 2429–2436, ACM (2021)

[32] Macdonald, C., Tonellotto, N., MacAvaney, S., Ounis, I.: PyTerrier: Declarative experimentation in Python from BM25 to dense retrieval. In: Proc. of CIKM 2021, pp. 4526–4533, ACM (2021)

[33] Mitra, B., Diaz, F., Craswell, N.: Learning to match using local and distributed representations of text for web search. In: Proc. of WWW 2017, pp. 1291–1299, ACM (2017)

[34] Mokrii, I., Boytsov, L., Braslavski, P.: A systematic evaluation of transfer learning and pseudo-labeling with BERT-based ranking models. In: Proc. of SIGIR 2021, pp. 2081–2085, ACM (2021)

[35] Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: Proc. of SP 2019, pp. 739–753, IEEE (2019)

[36] Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Findings of EMNLP 2020, vol. EMNLP 2020, pp. 708–718, ACL (2020)

[37] Nogueira, R., Yang, W., Cho, K., Lin, J.: Multi-stage document ranking with BERT. CoRR **abs/1910.14424** (2019)

[38] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proc. of EMNLP 2019, pp. 3980–3990, ACL (2019)

[39] Sandhaus, E.: The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia **6**(12), e26752 (2008)

[40] Sharma, L., Graesser, L., Nangia, N., Evci, U.: Natural language understanding with the Quora question pairs dataset. CoRR **abs/1907.01041** (2019)

[41] Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: Proc. of SP 2017, pp. 3–18, IEEE (2017)

[42] Voorhees, E.: The TREC Robust Retrieval track. SIGIR Forum **39**(1), 11–20 (2005)

[43] Wahle, J.P., Ruas, T., Meuschke, N., Gipp, B.: Are neural language models good plagiarists? A benchmark for neural paraphrase detection. In: Proc. of JCDL 2021, pp. 226–229 (2021)

[44] Xiong, C., Dai, Z., Callan, J., Liu, Z., Power, R.: End-to-end neural adhoc ranking with kernel pooling. In: Proc. of SIGIR 2017, pp. 55–64, ACM (2017)

[45] Yates, A., Arora, S., Zhang, X., Yang, W., Jose, K., Lin, J.: Capreolus: A toolkit for end-to-end neural ad hoc retrieval. In: Proc. of WSDM 2020, pp. 861–864, ACM (2020)

[46] Zhan, J., Xie, X., Mao, J., Liu, Y., Zhang, M., Ma, S.: Evaluating extrapolation performance of dense retrieval. CoRR **abs/2204.11447** (2022)

[47] Zhang, X., Yates, A., Lin, J.: A little bit is worse than none: Ranking with limited training data. In: Proc. of SustaiNLP 2020, pp. 107–112, Association for Computational Linguistics (2020)

[48] Zobel, J., Rashidi, L.: Corpus bootstrapping for assessment of the properties of effectiveness measures. In: Proc. of CIKM 2020, pp. 1933–1952, ACM (2020)