

# Are Neural Ranking Models Robust?

CHEN WU, RUQING ZHANG, JIAFENG GUO\*, YIXING FAN, and XUEQI CHENG\*, CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; University of Chinese Academy of Sciences, Beijing, China

Recently, we have witnessed the bloom of neural ranking models in the information retrieval (IR) field. So far, much effort has been devoted to developing effective neural ranking models that can generalize well on new data. There has been less attention paid to the robustness perspective. Unlike the effectiveness which is about the average performance of a system under normal purpose, robustness cares more about the system performance in the worst case or under malicious operations instead. When a new technique enters into the real-world application, it is critical to know not only how it works in average, but also how would it behave in abnormal situations. So we raise the question in this work: Are neural ranking models robust? To answer this question, firstly, we need to clarify what we refer to when we talk about the robustness of ranking models in IR. We show that robustness is actually a multi-dimensional concept and there are three ways to define it in IR: 1) The *performance variance* under the independent and identically distributed (I.I.D.) setting; 2) The *out-of-distribution (OOD) generalizability*; and 3) The *defensive ability* against adversarial operations. The latter two definitions can be further specified into two different perspectives respectively, leading to 5 robustness tasks in total. Based on this taxonomy, we build corresponding benchmark datasets, design empirical experiments, and systematically analyze the robustness of several representative neural ranking models against traditional probabilistic ranking models and learning-to-rank (LTR) models. The empirical results show that there is no simple answer to our question. While neural ranking models are less robust against other IR models in most cases, some of them can still win 2 out of 5 tasks. This is the first comprehensive study on the robustness of neural ranking models. We believe the way we study the robustness as well as our findings would be beneficial to the IR community. We will also release all the data and codes to facilitate the future research in this direction.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**.

Additional Key Words and Phrases: Robustness, Ranking Models, Systematic Analysis

## ACM Reference Format:

Chen Wu, Ruqing Zhang, Jiafeng Guo\*, Yixing Fan, and Xueqi Cheng\*. \*. Are Neural Ranking Models Robust?. *ACM Transactions on Information Systems* \*, \*, Article \* (February \*), 37 pages. [https://doi.org/\\*](https://doi.org/*)

## 1 INTRODUCTION

Relevance ranking is a core problem of information retrieval (IR). Given a query and a set of candidate documents, a scoring function is usually learned to determine the relevance degree of a document with respect to the query. With the advance of deep learning technology, we have

\* Jiafeng Guo and Xueqi Cheng are corresponding authors.

---

Author's address: Chen Wu, Ruqing Zhang, Jiafeng Guo\*, Yixing Fan, and Xueqi Cheng\*, {wuchen17z,zhangruqing, guojiafeng,fanyixing,cxq}@ict.ac.cn, CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; University of Chinese Academy of Sciences, Beijing, NO. 6 Kexueyuan South Road, Haidian District, Beijing, China, 100190.

---

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© \* Copyright held by the owner/author(s).

1046-8188/\*2-ART\*

[https://doi.org/\\*](https://doi.org/*)

witnessed substantial growth of neural ranking models [17, 18, 29, 60, 91], achieving promising results in learning query-document relevance patterns. Yet, due to the hype surrounding neural ranking models, some researchers have doubted whether they are actually effective for the classic ad-hoc retrieval problem without vast quantities of training data [53]. Recently, pre-trained language representation models such as BERT [45] have brought breakthrough to various downstream natural language processing (NLP) tasks [1, 28, 44, 94]. The success of pre-trained models in NLP has also attracted a lot of attention in the IR community. Researchers have applied the popular models, e.g., BERT and ELMo [69], for ad-hoc document ranking, and shown that they can achieve remarkable success on a broad range of ranking problems [28, 43, 56].

However, up to now, there has been little attention paid to the robustness, an important issue beyond the effectiveness, of neural ranking models. Unlike the effectiveness which is about the average performance of a system, robustness cares more about the worst-case performance instead. When a new technique enters into the real-world retrieval scenarios, it is critical to know not only how it works in average, but also how would it behave in abnormal situations. Therefore, in this work, we raise the following question: Are neural ranking models robust?

To answer this question, we need to first clarify what we refer to when we talk about the robustness of ranking models in IR. Although the robustness of neural ranking models has not been analyzed so far, there have been some related studies on the robustness of traditional probabilistic ranking models. For example, Goren et al. [27] analyzed the robustness of LTR models under adversarial document manipulation, and found that increased regularization of linear ranking functions results in increased ranking robustness. Voorhees [86] proposed to measure the robustness of traditional probabilistic ranking models by focusing on poorly performing queries. As we can see, the existing studies on the robustness of ranking models took quite different perspectives. In fact, the robustness is not a simple concept, but with multi-dimensional definition as shown in the research from the machine learning (ML) community [80, 98]. Therefore, if one only picks up one perspective, he/she is very likely to obtain misleading conclusions with respect to robustness of neural ranking models. Unfortunately, there has been no comprehensive study on this question yet.

Therefore, in this paper, based on the previous work in IR and also inspired by the study of robustness in ML, we propose to define the robustness of ranking models via a comprehensive taxonomy which contains three major perspectives as shown in Figure 1.

Before introducing the taxonomy of the robustness of ranking models, we first describe the key notations and the formulation of the effectiveness in this work. Suppose that  $Y = \{r_1, r_2, \dots, r_l\}$  is the set of ranks, where  $l$  denotes the number of ranks. There exists a total order between the ranks  $r_l > r_{l-1} > \dots > r_1$ , where  $>$  denotes the order. Specifically,  $r_1$  usually takes 0 as value, representing being irrelevant. Suppose that  $Q = \{q_1, q_2, \dots, q_m\}$  is the set of queries in training. Each query  $q_i$  is associated with a list of documents  $\mathbf{d}_i = \{d_{i1}, d_{i2}, \dots, d_{i,n(q_i)}\}$  and a list of corresponding labels  $\mathbf{y}_i = \{y_{i1}, y_{i2}, \dots, y_{i,n(q_i)}\}$ , where  $n(q_i)$  denotes the size of list  $\mathbf{d}_i$ ,  $d_{ij} \in \mathbf{D}$  denotes the  $j^{\text{th}}$  document in  $\mathbf{d}_i$ , and  $y_{ij} \in Y$  denotes the label of document  $d_{ij}$ .

We consider the ranking model  $f$  learned on the examples  $\{q_i, \mathbf{d}_i, \mathbf{y}_i\}_{i=1}^m$ , which are drawn from the training distribution  $\mathcal{G}$ . In this way, for each query  $q_i$ , the elements in its corresponding document list  $\mathbf{d}_i$  can be assigned relevance scores using the model and then be ranked according to the scores. Let  $\mathbf{d}_i$  be identified by the list of integers  $\{1, 2, \dots, n(q_i)\}$ , then permutation  $\pi(q_i, \mathbf{d}_i, f)$  is defined as a bijection from  $\{1, 2, \dots, n(q_i)\}$  to itself. Ranking is nothing but to produce a permutation  $\pi(q_i, \mathbf{d}_i, f)$  for the given query  $q_i$  and the associated list of documents  $\mathbf{d}_i$  using the ranking model. Given an effectiveness evaluation metric  $M$ , the ranking models are usually evaluated by the average performance over the test queries under the I.I.D. setting, i.e.,

$$\mathbb{E}_{(q_t, \mathbf{d}_t, \mathbf{y}_t) \sim \mathcal{G}} M(\pi(q_t, \mathbf{d}_t, f), \mathbf{y}_t), \quad (1)$$

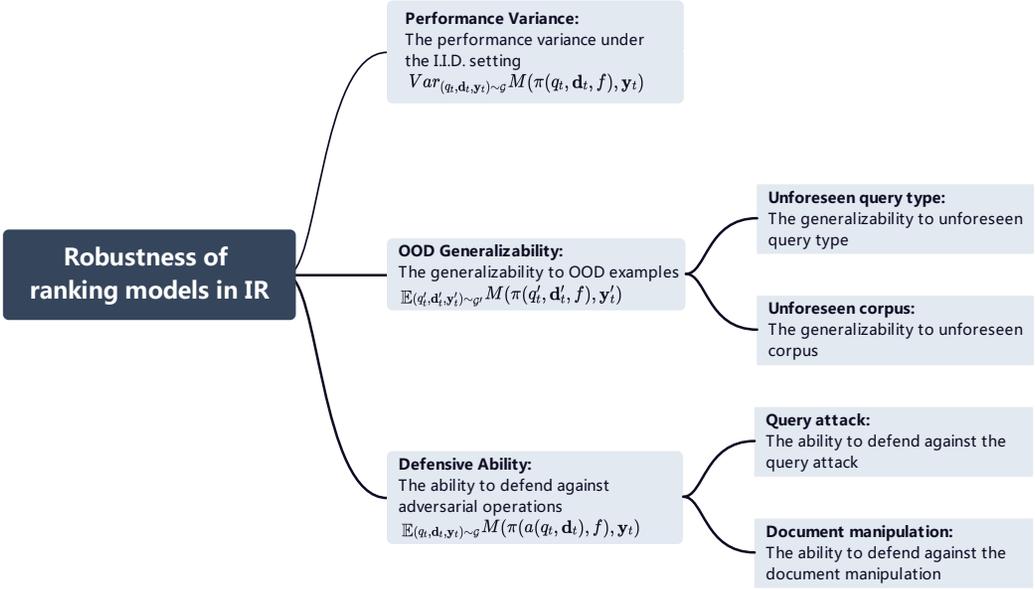


Fig. 1. The taxonomy of the robustness of ranking models in IR.

where  $q_t$ ,  $\mathbf{d}_t$  and  $y_t$  denotes the query, the document list and the label in the test set, respectively. Specifically, the test samples are supposed to be drawn from the same distribution as the training distribution  $\mathcal{G}$ .

We define the robustness of ranking models from three major perspectives as follows:

- **Performance Variance:** The performance variance aims to analyze the robustness of ranking models by emphasizing the worst-case performance across different individual queries under the I.I.D. setting. Formally, the performance variance is defined as

$$Var_{(q_t, \mathbf{d}_t, y_t) \sim \mathcal{G}} M(\pi(q_t, \mathbf{d}_t, f), y_t), \quad (2)$$

where  $Var(\cdot)$  denotes the variance of effectiveness over all the test queries. Besides, a special case of performance variance, i.e., the poorly-performing queries, is emphasized to analyze the ranking robustness in the worst case.

- **OOD Generalizability:** The OOD generalizability aims to analyze the robustness of ranking models according to the transfer effectiveness on OOD examples. Formally, suppose that OOD examples  $q'_t$ ,  $\mathbf{d}'_t$  and  $y'_t$  are drawn from a new distribution  $\mathcal{G}'$ . The OOD generalizability is defined as

$$\mathbb{E}_{(q'_t, \mathbf{d}'_t, y'_t) \sim \mathcal{G}'} M(\pi(q'_t, \mathbf{d}'_t, f), y'_t), \quad (3)$$

Specifically, the OOD generalizability can be further defined in two ways, i.e., OOD generalizability on unforeseen query types and OOD generalizability on unforeseen corpus.

- **Defensive Ability:** The defensive ability aims to analyze the robustness of ranking models according to their ability to defend against adversarial operations. Given an adversarial attack function  $a$  for the query and document, the defensive ability is formalized as

$$\mathbb{E}_{(q_t, \mathbf{d}_t, y_t) \sim \mathcal{G}} M(\pi(a(q_t, \mathbf{d}_t), f), y_t), \quad (4)$$

Specifically, the defensive ability can be measured with respect to two types of adversarial operations, i.e., query attack and document manipulation.

Based on this taxonomy, we design the corresponding experiments, and conduct empirical studies to analyze the robustness of several representative neural ranking models against traditional probabilistic ranking models and LTR models. All the experimental datasets and codes<sup>1</sup> used in our study would be publicly available for the research community. Specifically, the design of our empirical experiments for each task are as follows.

- To facilitate the study of the performance variance, we conduct experiments on three ad-hoc retrieval datasets, i.e., MQ2007 [71], Robust04, and MS MARCO, and follow the previous works [86, 99] to measure the robustness of different ranking models.
- To facilitate the study of the OOD generalizability, we first build two benchmark datasets for unforeseen query type and unforeseen corpus, respectively. Specifically, we construct the benchmark dataset for unforeseen query type by splitting the MS MARCO dataset into 5 sub-datasets, with respect to the official query type. Besides, we construct the benchmark dataset for unforeseen corpus based on three ad-hoc retrieval corpora, i.e., MQ2007, Robust04, and MS MARCO. Furthermore, we propose the drop rate metric to measure the OOD generalizability of different ranking models.
- To facilitate the study of the defensive ability, we first build a benchmark dataset for query attack. Specifically, we introduce four types of character-level edits [42, 70] and three types of word-level edits to attack the query in the MS MARCO dataset. Then, we propose the drop rate metric to measure the defensive ability against query attack. For document manipulation, we conduct experiments on the ASRC dataset [74] and follow the previous work [27] to measure the robustness of different ranking models.

The empirical results demonstrate that: 1) Neural ranking models are in general not robust as compared with other IR models. The finding is consistent with the previous study [84] in ML which demonstrates that the complexity in the model could lead to robustness issues. 2) Pre-trained ranking models exhibit the best robustness against all the other models from the perspective of the performance variance. 3) Some neural ranking models show the superiority of the defensive ability. For example, DSSM, Duet and Conv-KNRM are robust to the query attack. 4) Based on the empirical evaluation results, there remains clearly room for future robustness improvements.

We organize this work as follows. We first introduce the ranking models which are evaluated through all the robustness metrics, including traditional probabilistic ranking models, LTR models and neural ranking models in Section 2. We then analyze the robustness of ranking models in terms of the performance variance, the OOD generalizability, and the defensive ability, in Section 3, Section 4 and Section 5 respectively. Finally, we briefly review the related work in Section 6 and conclude this work in Section 7.

## 2 RANKING MODELS

In this paper, we aim to investigate whether the neural ranking models are robust. Specifically, we adopt three types of representative neural ranking models for analysis, i.e., representation-focused deep matching models, interaction-focused deep matching models, hybrid deep matching models and advanced pre-trained models.

- **Representative-focused Deep Matching Models** include,

<sup>1</sup>The experimental datasets and codes will be available at URL.

- **DSSM**: DSSM [36] is a representative-focused deep matching model designed for Web search, which contains a letter n-gram based word hashing layer, two non-linear hidden layers and an output layer.
- **Interaction-focused Deep Matching Models** include,
  - **DRMM**: DRMM [29] is an interaction-focused deep matching model designed for ad-hoc retrieval. It consists of a matching histogram mapping, a feed forward matching network and a term gating network. In this paper, we use LogCount-based Histogram (LCH) as the matching histogram mapping and Term Vector (TV) as the term gating function.
  - **Conv-KNRM**: Conv-KNRM [18] is another popular interaction-focused deep matching model, which models n-gram soft matches for ad-hoc retrieval based on convolutional neural networks (CNN) and kernel-pooling.
- **Hybrid Deep Matching Models** include,
  - **Duet**: Duet [60] is a hybrid deep matching model which combines both the representation-focused architecture and the interaction-focused architecture. Specifically, it contains two separate deep neural networks, one that matches using a local representation of text, and another that learns a distributed representation before matching.
- **Pre-trained Models** include,
  - **BERT**: The key technical innovation of BERT [45] is applying the multi-layer bidirectional Transformer encoder architecture for language modeling. BERT uses two different types of pre-training objectives including Masked Language Model (MLM) and Next Sentence Prediction (NSP). In this paper, the query and the document are concatenated as the input to BERT with special tokens delimiting them, i.e., [CLS] and [SEP]. To obtain the relevance score of the document to a given query, we apply a sigmoid function over the representation of [CLS] following previous studies [17, 63].
  - **ColBERT**: ColBERT [46] employs contextualized late interaction over deep language models (in particular, BERT) for efficient retrieval. It independently encodes queries and documents into fine-grained representations that interact via cheap and pruning-friendly computations. Here, we analyze ColBERT’s ability to re-rank documents following the original paper.

Specifically, we use the implementations of DRMM, Conv-KNRM, and Duet from MatchZoo [30]. We adopt the parameters of BERT<sub>base</sub> released by Google<sup>2</sup> to initialize BERT. For ColBERT, we use the original code<sup>3</sup> released by the authors for implementation.

In order to better analyze the robustness of neural ranking models, we adopt three types of LTR models for comparison, including pointwise LTR models, pairwise LTR models and listwise LTR models.

- **Pointwise LTR Models** include,
  - **Prank**: Prank [14] is a famous pointwise LTR model on ordinal regression.
- **Pairwise LTR Models** include,
  - **RankSVM**: RankSVM [40] is a representative pairwise LTR model based on Structural Support Vector Machine (SVM).
- **Listwise LTR Models** include,
  - **LambdaMART**: LambdaMART [7] is a state-of-the-art listwise LTR algorithm which uses gradient boosting to produce an ensemble of retrieval models.

<sup>2</sup><https://github.com/google-research/bert>

<sup>3</sup><https://github.com/stanford-futuredata/ColBERT/>

For Prank, we implement it according to the original paper since there is no publicly available code. For RankSVM, we directly use the implementation in  $SVM^{rank}$  [41]. LambdaMART is implemented using RankLib<sup>4</sup>, which is a widely used LTR tool.

Besides, we compare the above neural models with two representative traditional probabilistic ranking models, including,

- **QL**: Query likelihood model based on Dirichlet smoothing [97] is one of the best performing language models.
- **BM25**: The BM25 formula [76] is another highly effective retrieval model that represents the classical probabilistic retrieval model.

Specifically, we use the implementations of QL and BM25 from Anserini [93] toolkit<sup>5</sup>.

### 3 PERFORMANCE VARIANCE UNDER THE I.I.D. SETTING

Most ranking models are designed under the simple assumption that the observations are from I.I.D. random variables, and focus on improving the average effectiveness of retrieval results. Recently, it has been recognized that, when we attempt to improve the mean retrieval effectiveness over all queries, the stability of performance across different individual queries could be hurt [99]. Therefore, in this section, we analyze the robustness of ranking models by emphasizing the worst-case performance across different individual queries, i.e., the performance variance under the I.I.D. settings. In the following, we first introduce the definition and metric of the performance variance, and then conduct experiments to analyze the robustness of ranking models.

#### 3.1 Definition of Performance Variance

The performance variance of ranking models refers to the variance of effectiveness across different individual queries, which has been formulated in Eq. (2). When a ranking model achieves improvements in mean retrieval effectiveness (e.g., mean average precision (MAP) [79]), the performance of some individual queries could be hurt [99]. Although failures on a small number of queries may not have a significant effect on the average performance, users who are interested in such queries are unlikely to be tolerant of this kind of deficiency. Accordingly, an ideal ranking model is expected to balance the trade-off between effectiveness and robustness by achieving high average effectiveness and low variance of effectiveness [100].

#### 3.2 Metric of Performance Variance

Here, we propose the variance of normalized average precision (VNAP) to measure the performance variance, which is defined as,

$$VNAP = \mathbb{E}(NAP(q_t) - \mathbb{E}(NAP(q_t)))^2, \quad (5)$$

where the expectation  $\mathbb{E}(\cdot)$  is over a set of queries that are assumed to be uniformly distributed.  $NAP(q_t)$  denotes the normalized average precision with respect to the query  $q_t$ , which is defined as,

$$NAP(q_t) = \frac{AP(q_t)}{\mathbb{E}(AP(q_t))}, \quad (6)$$

where  $AP(q_t)$  denotes the average precision with respect to the query  $q_t$ , which is defined as,

$$AP(q_t) = \frac{1}{R_{q_t}} \sum_{k=1}^{R_{q_t}} \frac{1}{o_k} \sum_{n=1}^{o_k} \delta(y_{tn} > 0), \quad (7)$$

<sup>4</sup><https://sourceforge.net/p/lemur/wiki/RankLib/>

<sup>5</sup><http://anserini.io/>

Table 1. Statistics of datasets used for evaluating the performance variance.

|                         | MQ2007 | Robust04 | MS MARCO |
|-------------------------|--------|----------|----------|
| #Queries                | 1692   | 250      | 0.37M    |
| #Documents              | 65,323 | 528k     | 3.21M    |
| Avg document length     | 3587   | 471      | 1129     |
| Avg #Relevant Documents | 10     | 70       | 1        |

where  $R_{q_t}$  denotes the number of relevant documents to the query  $q_t$ .  $o_k$  is the rank of the relevant document  $k$  predicted by the ranking model, which ranges from 1 to the size of document list.  $y_{tn}$  denotes the ground-truth label of document  $d_{tn}$ .  $\delta$  is the indicator function, which aims to count the number of relevant documents.

Note that VNAP is similar to VAP [99]. The difference between them is that we have normalized the average precision to eliminate the influence of the mean performance. Since different models are likely to have different mean performance, it is necessary to eliminate the influence of the mean performance to better measure the variance of ranking models. The ranking model would be more robust with a lower VNAP value.

### 3.3 Experimental Settings

In this section, we introduce our experimental settings, including datasets and implementations details.

**3.3.1 Datasets.** To evaluate the performance variance of different ranking models, we conduct experiments on several representative ad-hoc retrieval datasets, i.e.,

- **MQ2007.** Million Query Track 2007 (MQ2007) is a LETOR [71] benchmark dataset based on the GOV2 collection which includes 25 million documents in 426 gigabytes. Queries for MQ2007 are divided into 5 folds as described in LETOR4.0.
- **Robust04.** Robust04 contains 250 queries and 528k news articles, whose queries are collected from TREC 2004 Robust Track<sup>6</sup>. There are about 70 relevant documents (news articles) for each query.
- **MS MARCO.** MicroSoft MAchine Reading COmprehension Document Ranking Dataset (MS MARCO)<sup>7</sup> is a large-scale benchmark dataset for web document retrieval. MS MARCO contains 4 million documents and 0.37 million training queries, where each query has one relevant document. Evaluation was on the dev set, which contains 5193 queries<sup>8</sup>.

The detailed statistics of datasets are shown in Table 1. We choose these three datasets according to three criteria: 1) The datasets are public and the original document contents are available, 2) The query numbers are diverse (e.g., 250 queries in Robust04 and 0.37 million queries in MS MARCO), and 3) The literary style of the documents ranges from newswire articles to Web document.

**3.3.2 Implementation Details.** In preprocessing, all the words in the documents and queries are white-space tokenized, lower-cased, and stemmed using the Krovetz stemmer [49]. For the MQ2007

<sup>6</sup><https://trec.nist.gov/data/robust.html>

<sup>7</sup><https://github.com/microsoft/MSMARCO-Document-Ranking>

<sup>8</sup>The relevance assessments for the official test set were not public at the time the paper was written.

dataset, we use the data partition and top 40 ranked documents released by the official LETOR4.0 [71]. For the Robust04 dataset, an initial retrieval is performed using the Anserini toolkit with the QL model to obtain the top 100 ranked documents. For the MS MARCO dataset, we use the official top 100 ranked documents retrieved by the QL model.

For traditional probabilistic ranking models, we adopt the static parameters suggested by [37] on the Robust04 dataset and tune the parameters on the corresponding validation set on the MQ2007 and MS MARCO dataset<sup>9</sup>. For LTR models, we use standard features released by LETOR 4.0 for MQ2007. In LETOR, there are 46 hand-crafted features in total, among which 42 features are constructed based on the textual elements (e.g., term frequencies, BM25 and language model scores based on title, body, anchor texts, and URL), and 4 features are based on link analysis (e.g., PageRank, inlink number, outlink number, and number of child page). Due to the lack of officially released LTR features, we construct the LTR features for Robust04 and MS-MARCO respectively, by computing the TF, IDF, TF-IDF, Document Length, BM25 score, QL-DIR score and QL-JM scores on the title, url, body and document respectively as the final 28-dimensional LTR features following [71]. For RankSVM, we use the linear kernel with the hyper-parameter  $C$  selected from the range  $[0.0001, 10]$  on the validation set for three datasets following [18]. For LambdaMART, we select the number of trees (in 100-500, by 10), leafs (in 2-20, by 2) and shrinkage coefficient (in  $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ ) on the validation set for three datasets.

For neural ranking models, the model-specific hyper-parameters are as follows. For DSSM, we use a three-layer DNN and set the node number of each layer as 100, 100 and 50 to avoid overfitting on Robust04 and MQ2007 datasets following [65]. We set the node number of each layer as 300, 300, 128 on the MS MARCO dataset. For DRMM, we use a four-layer architecture and set the node number of each layer as 30 (histogram bins), 5, 1 and 1 on Robust04 and MQ2007 datasets following [29, 65]. We set the node number of each layer as 30 (histogram bins), 10, 1 and 1 on MS MARCO dataset. For Conv-KNRM, we set the  $n$ -gram size as 3 and the number of CNN filters as 128. For kernel-pooling, we set the number of kernels to 11 with the mean values  $\mu$  of the Gaussian kernels varying from  $-1$  to  $+1$ , and standard deviation  $\sigma$  of 0.1 for all kernels (the first kernel's  $\sigma$  is set as 0.001 for exact matching ) on three datasets following [18, 35]. For Duet, we used 10 filters with both the local and distributed model, with hidden dimensions set to 30 and 699, respectively on Robust04 dataset following [95]. We set the filter size as 10 in both local model and distributed model, and the hidden size as 20 in the fully-connected layer on MQ2007 dataset following [21]. We set the filter size as 32 in both local and distributed model and the hidden size as 32 in the fully-connected layer on MS MARCO dataset. The learning rates of above models are tuned on the corresponding validation set in the range of  $[1e-5, 5e-3]$ . For BERT, we choose the fine-tuning learning rate from  $\{5e-5, 3e-5, 2e-5\}$  as recommended in [45] on three datasets. For ColBERT, we set the fine-tuning learning rate as  $3e-6$  on three datasets following [46]. For all neural ranking models, we use the Adam [47] optimizer. We use LTR and neural ranking models to re-rank the top candidate documents.

### 3.4 Empirical Results on Performance Variance

In this section, we analyze the empirical results on the performance variance under the I.I.D. setting. Specifically, we first analyze the average ranking effectiveness and the variance of effectiveness of different ranking models, respectively. Furthermore, we analyze the relationship between the average ranking effectiveness and the variance of effectiveness.

<sup>9</sup> $\mu$  in 1-2000 (by 10) for QL and  $k_1$  in 0.1-5 (by 0.1),  $b$  in 0.1-1 (by 0.1) for BM25.

Table 2. The average ranking effectiveness over all the queries on the Robust04, MQ2007 and MS MARCO dataset in terms of MAP, NDCG@10, NDCG@20, P@10, P@20, R@10, R@20, MRR@10 and MRR@100.

| Model      | Robust04      |               |               |               | MQ2007        |               |               |               | MS MARCO      |               |               |
|------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
|            | MAP           | NDCG@20       | P@20          | R@20          | MAP           | NDCG@10       | P@10          | R@10          | MRR@10        | MRR@100       | R@10          |
| QL         | 0.2109        | 0.4113        | 0.3516        | 0.2094        | 0.3928        | 0.3891        | 0.3348        | 0.3192        | 0.2144        | 0.2271        | 0.4489        |
| BM25       | 0.2162        | 0.4200        | 0.3594        | 0.2105        | 0.3991        | 0.4000        | 0.3435        | 0.3293        | 0.2184        | 0.2301        | 0.4439        |
| Prank      | 0.1190        | 0.2181        | 0.2157        | 0.1168        | 0.3650        | 0.3162        | 0.3008        | 0.2665        | 0.2184        | 0.2311        | 0.4645        |
| RankSVM    | <b>0.2179</b> | 0.4234        | 0.3610        | 0.2120        | 0.4637        | <b>0.4428</b> | 0.3811        | 0.3657        | 0.2398        | 0.2522        | 0.4818        |
| LambdaMart | 0.2171        | 0.4218        | 0.3584        | 0.2116        | <b>0.4646</b> | 0.4401        | 0.3741        | 0.3633        | 0.2602        | 0.2710        | 0.4986        |
| DSSM       | 0.1216        | 0.2306        | 0.2283        | 0.1198        | 0.3998        | 0.3606        | 0.3421        | 0.3070        | 0.1051        | 0.1220        | 0.2070        |
| DRMM       | 0.1649        | 0.3096        | 0.2748        | 0.1585        | 0.4317        | 0.4009        | 0.3662        | 0.3338        | 0.1168        | 0.1336        | 0.3046        |
| Conv-KNRM  | 0.1332        | 0.2526        | 0.2426        | 0.1220        | 0.3957        | 0.3553        | 0.3436        | 0.3047        | 0.2075        | 0.2209        | 0.4435        |
| Duet       | 0.1308        | 0.2503        | 0.2421        | 0.1299        | 0.4089        | 0.3727        | 0.3561        | 0.3223        | 0.1233        | 0.1402        | 0.3054        |
| BERT       | 0.2108        | <b>0.4330</b> | <b>0.3808</b> | <b>0.2127</b> | 0.4620        | 0.4420        | <b>0.3878</b> | <b>0.3708</b> | 0.3093        | 0.3181        | 0.5748        |
| ColBERT    | 0.1989        | 0.4010        | 0.3594        | 0.2043        | 0.4420        | 0.4153        | 0.3752        | 0.3529        | <b>0.3469</b> | <b>0.3541</b> | <b>0.6212</b> |

### 3.4.1 How does different ranking models perform in terms of the average effectiveness?

To better understand the performance variance, we first analyze the average ranking effectiveness over the queries. Following [21, 29], we take the topic “title” as queries and conduct 5-fold cross-validation on Robust04 and MQ2007 datasets to minimize over-fitting without reducing the number of learning instances. For Robust04 dataset, we use the mean average precision (MAP) [79], normalized discounted cumulative gain at rank 20 (NDCG@20) [6], and precision at position 20 (P@20) [15] as the evaluation metrics following [29]. For MQ2007 dataset, we report the P@10 and NDCG@10 following [21]. Moreover, we also use the recall at position 20 (R@20) [64] and recall at position 10 (R@10) as the recall oriented evaluation metrics for Robust04 and MQ2007, respectively. For MS MARCO dataset, we report the Mean Reciprocal Rank at 100 (MRR@100 [73]), which is suggested in the official instructions. We also report the MRR@10 and R@10 for MS MARCO dataset.

The main results are shown in Table 2. From the results, we can find that: (1) For the two traditional probabilistic ranking models, we can see that *BM25* is a strong baseline which performs better than *QL* in most cases. (2) For LTR models, RankSVM and LambdaMart perform better than the traditional probabilistic ranking models. It is not surprising since LTR models combine various features including the two traditional probabilistic ranking models (i.e., *BM25* and *QL* scores), which are capable of characterizing the relevance between a document and a query. However, Prank performs worse than the traditional probabilistic ranking models on the Robust04 and

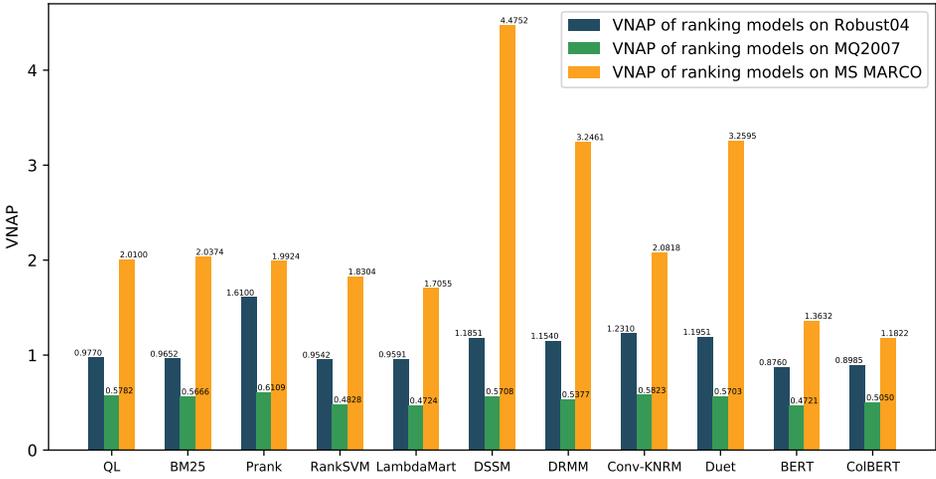


Fig. 2. The performance variance of different ranking models on the Robust04, MQ2007 and MS MARCO dataset in terms of VNAP.

MQ2007 dataset. The reason might be that pointwise LTR ignores the fact that some documents are associated with the same query and some others are not [54]. (3) Most neural ranking models perform worse than traditional probabilistic ranking models and LTR models. The reason might be that it is difficult for a deep neural model to train from scratch with such a few supervised pairs [72]. (4) Pre-trained models (i.e., *BERT* and *ColBERT*) perform the best on all the three datasets. These results indicate that the rich language information from text captured by the pre-trained model is useful for relevance modeling.

**3.4.2 How does different ranking models perform in terms of the variance of normalized average precision?** The performance variance comparisons among different ranking models in terms of VNAP are shown in Figure 2. Recall that high values of VNAP attest to decreased ranking robustness.

From the results, we can observe that: (1) For the two traditional probabilistic ranking models, BM25 achieves lower VNAP values than QL on the Robust04 and MQ2007 datasets, indicating that BM25 is more robust than QL on these datasets. (2) For the LTR models, RankSVM and LambdaMart achieve lower VNAP than traditional probabilistic ranking models on the three datasets, indicating that LTR models are more robust than traditional probabilistic ranking models. (3) For neural ranking models, DSSM achieves the highest VNAP on the MS MARCO dataset. The reason might be that DSSM achieves poor mean performance on the MS MARCO dataset and the corresponding VNAP is relatively higher. (4) When we look at the pre-trained models (i.e., *BERT* and *ColBERT*), we find they achieve the lowest VNAP among all the ranking models on the three datasets. It is interesting to find that pre-trained models exhibit the strongest robustness in terms of VNAP. As shown in Table 2, the pre-trained models have shown great success on the average effectiveness. The result indicates that pre-trained models present a great potential since it performs well in terms of the mean performance and the robustness (e.g, in terms of the variance).

**3.4.3 What is the relationship between the average effectiveness and the variance of effectiveness?** Here, we analyze the relationship between the average effectiveness and the robustness. Specifically, we consider the variance of effectiveness in Figure 2 and the average effectiveness in Table 2 simultaneously. Overall, the robustness of different models generally increases (i.e., the VNAP decreases) with the increase of the average ranking effectiveness (i.e., the MAP increases). In addition, we have computed the correlation between the effectiveness and the variance. For example, the Pearson correlation coefficient between the MAP and VNAP is  $-0.9869$ . The result indicates that there is a strong correlation between the average effectiveness and the variance of effectiveness.

### 3.5 Analysis on Poorly-performing Queries

The previous experiments have analyzed the performance variance over all the queries. A natural question is how the ranking models perform in the worse case, which can be well evaluated on the poorly-performing queries. Therefore, in this section, we emphasize on the poorly-performing queries to analyze the robustness of the ranking models, which refer to the queries that have the poor ranking performance among all the test queries. We first introduce the evaluation metric of the poorly-performing queries, and then analyze the experimental results.

**3.5.1 Metric of Poorly-performing Queries.** To evaluate the robustness of ranking models in terms of the poorly-performing queries, we use two metrics following the previous works [86].

- **%no** denotes the percentage of queries with no relevant documents in the top 10 retrieved, i.e.,

$$\%no = \frac{1}{|Q|} \sum_{t=1}^{|Q|} \prod_{n=1}^{10} \delta(y_{tn} = 0), \quad (8)$$

where  $y_{tn} \in Y$  is the relevance label (i.e., the larger the relevance label, the more relevant the query-document pair) of the top  $n \in [1, 10]$  document with respect to the query  $q_t$ .  $\delta$  is the indicator function and  $|Q|$  is the total number of evaluated queries. The ranking model would be more robust with a lower %no value.

- **gMAP** denotes the geometric mean average precision, which is defined as,

$$gMAP = \exp\left(\frac{1}{|Q|} \sum_{t=1}^Q \log(AP(q_t) + \epsilon)\right) - \epsilon, \quad (9)$$

where  $AP(q_t)$  is formulated as in Eq. (7).  $\epsilon$  is a minimal positive since the average precision score for a single query may approximate to zero [87]. The ranking model would be more robust with a higher gMAP value.

**3.5.2 Empirical Results on Poorly-performing Queries.** Here, we first measure the performance of the poorly-performing queries in terms of %no and gMAP. Then, we analyze the relationship between the performance of all the queries and the performance of the poorly-performing queries.

- **How does different ranking models perform over poorly-performing queries?** We first focus on the poorly-performing queries to analyze the robustness of different ranking models in terms of %no and gMAP. The main results are shown in Table 3. Recall that low values of %no and high values of gMAP attest to increased ranking robustness. From the results, we can observe that: (1) For traditional probabilistic ranking models, *QL* achieves a lower %no value than BM25 on the MQ2007 and MS MARCO, demonstrating that *QL* is more robust than BM25 over the poorly-performing queries. (2) For LTR models, RankSVM and LambdaMart achieve lower %no values and higher gMAP values as compared with

Table 3. The performance of poorly-performing queries on the Robust04, MQ2007, and MS MARCO dataset in terms of %no and gMAP.

| Model      | Robust04    |               | MQ2007       |               | MS MARCO     |               |
|------------|-------------|---------------|--------------|---------------|--------------|---------------|
|            | %no         | gMAP          | %no          | gMAP          | %no          | gMAP          |
| QL         | 11.24       | 0.0933        | 26.25        | 0.0802        | 55.11        | 0.0144        |
| BM25       | <b>8.02</b> | 0.0952        | 26.84        | 0.0805        | 55.61        | 0.0108        |
| Prank      | 9.22        | 0.0947        | 28.43        | 0.0760        | 53.55        | 0.0141        |
| RankSVM    | 9.63        | 0.0965        | 22.39        | 0.0972        | 51.82        | 0.0154        |
| LambdaMart | 8.42        | 0.0961        | 22.28        | 0.0978        | 50.14        | 0.0160        |
| DSSM       | 22.11       | 0.0631        | 25.71        | 0.0821        | 79.30        | 0.0065        |
| DRMM       | 20.51       | 0.0711        | 24.58        | 0.0889        | 69.54        | 0.0086        |
| Conv-KNRM  | 23.29       | 0.0609        | 25.41        | 0.0814        | 55.65        | 0.0135        |
| Duet       | 26.92       | 0.0585        | 24.35        | 0.0840        | 69.46        | 0.0087        |
| BERT       | 9.23        | <b>0.1067</b> | <b>20.98</b> | <b>0.0979</b> | 42.52        | 0.0205        |
| ColBERT    | 14.86       | 0.0949        | 22.70        | 0.0919        | <b>37.88</b> | <b>0.0237</b> |

traditional probabilistic ranking models on the MQ2007 and MS MARCO. The results indicate that combining different kinds of human knowledge (i.e., relevance features) could achieve good improvements on the ranking robustness over the poorly-performing queries. (3) When we look at the pre-trained models (i.e., *BERT* and *ColBERT*), we find that *BERT* achieves the lowest %no value and the highest gMAP value among all the models on the MQ2007 dataset, while *ColBERT* achieves the lowest %no value and the highest gMAP value on the MS MARCO dataset. It is interesting that pre-trained models exhibit the strongest robustness over the poorly-performing queries under the I.I.D. setting.

- **What is the relationship between the average effectiveness and the robustness over the poorly-performing queries?** Specifically, we visualize the experimental results on the MQ2007 dataset from Table 2 and 3 into Figure 3, where the horizontal axis represents the average effectiveness metric (i.e., MAP) and the vertical axis represents the robustness metric (i.e., gMAP and %no). We have the following observations: (1) The robustness over the poorly-performing queries generally increases (i.e., the gMAP increases), with the increase of the average effectiveness (i.e., the MAP increases). The reason might be that the good performance over poorly-performing queries, in turn, would improve the performance over all the queries. This is consistent with the finding in previous studies where the degree of ranking robustness is positively correlated with retrieval performance [102]. (2) The pre-trained models are the most robust model over the poorly-performing queries and the most effective model over all the queries under the I.I.D. setting. Future work could propose new self-supervised objectives tailored for IR that enhance model robustness.

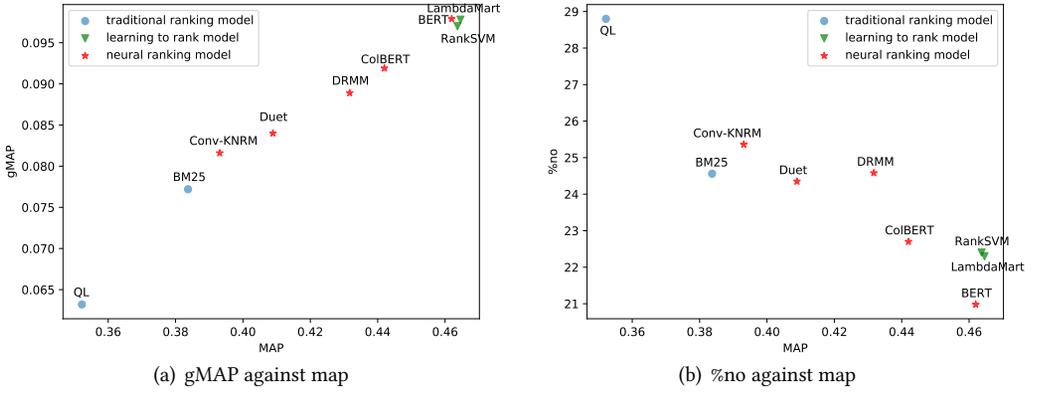


Fig. 3. The relationship between the effectiveness (e.g., MAP) and the robustness over the poorly-performing queries (e.g., gMAP and %no) on the MQ2007 dataset.

## 4 OOD GENERALIZABILITY

As described above, most evaluations in IR assume that the train and test examples are I.I.D.. Some advanced ranking models (e.g., pre-trained model) can achieve promising performance on some IR tasks [28, 43, 56]. However, in the real-world scenarios, the train and test distributions are often not identically distributed caused by the data bias [83]. Without any special supervision, the ranking models may come at the cost of poor generalization and performance on examples not observed during training [10].

Therefore, in this section, we analyze the robustness of ranking models according to the transfer effectiveness on OOD examples, i.e., the OOD generalizability which has been formulated in Eq. (3). Specifically, we introduce two ways to define the OOD generalizability, i.e., 1) The OOD generalizability on unforeseen query type, and 2) The OOD generalizability on unforeseen corpus.

### 4.1 OOD Generalizability on Unforeseen Query Type

In this section, we introduce the definition and metric of the OOD generalizability on unforeseen query type.

**4.1.1 Definition of OOD Generalizability on Unforeseen Query Type.** The OOD generalizability on unforeseen query type refers to the transfer effectiveness on unseen query types. Nowadays, users formulate their queries in the form of various types [5] that can describe their information needs properly to find the right results quickly. When submitting a new query type to a search engine, it tends to fail due to the large gap between normal in-distribution and undesired OOD query types. To improve users' satisfaction, a ranking model should be robust to such new query types.

Formally, given a ranking model  $f$  learned on  $\{q_i, \mathbf{d}_i, \mathbf{y}_i\}_{i=1}^m$  which are drawn from the training distribution  $\mathcal{G}_q$ , we aim to evaluate its performance on the test examples with unforeseen query types, which are drawn from a new distribution  $\mathcal{G}'_q$ . Specifically, the OOD generalizability on unforeseen query type is defined as

$$\mathbb{E}_{(q'_t, \mathbf{d}_t, \mathbf{y}'_t) \sim \mathcal{G}'_q} M(\pi(q'_t, \mathbf{d}_t, f), \mathbf{y}'_t), \quad (10)$$

where  $q'_t$ ,  $\mathbf{d}_t$  and  $\mathbf{y}'_t$  denotes the query, the document list and the label with new query types. Note that the same document collection is used for both the training and test sets.

**4.1.2 Metric of OOD Generalizability on Unforeseen Query Type.** To measure the OOD generalizability of the ranking models on unforeseen query type, we propose one automatic metric, namely,

- $\mathbf{DR}_{OOD}$  evaluates the *drop rate* between the ranking performance  $P_{OOD}$  on the OOD test set (i.e.,  $(q'_t, \mathbf{d}_t, y'_t) \sim \mathcal{G}'_q$ ) and the ranking performance  $P_{I.I.D.}$  on the I.I.D. test set (i.e.,  $(q_t, \mathbf{d}_t, y_t) \sim \mathcal{G}_q$ ), which is defined as

$$DR_{OOD} = \frac{P_{OOD} - P_{I.I.D.}}{P_{I.I.D.}}, \quad (11)$$

where  $P_{I.I.D.}$  and  $P_{OOD}$  are defined as

$$P_{I.I.D.} = \mathbb{E}_{(q_t, \mathbf{d}_t, y_t) \sim \mathcal{G}_q} M(\pi(q_t, \mathbf{d}_t, f), y_t), \quad (12)$$

$$P_{OOD} = \mathbb{E}_{(q'_t, \mathbf{d}_t, y'_t) \sim \mathcal{G}'_q} M(\pi(q'_t, \mathbf{d}_t, f), y'_t), \quad (13)$$

where the effectiveness evaluation metric  $M$  can be defined in different ways, such as mean reciprocal rank (MRR) [73], mean average precision (MAP) [79], normalized discounted cumulative gain (NDCG) [6] and Precision (P) [15], with respect to the specific experimental dataset. The ranking model would be more robust with a higher  $DR_{OOD}$ .

## 4.2 OOD Generalizability on Unforeseen Corpus

In this section, we introduce the definition and metric of the OOD generalizability on unforeseen corpus.

**4.2.1 Definition of OOD Generalizability on Unforeseen Corpus.** The OOD generalizability on unforeseen corpus refers to the transfer effectiveness on unseen corpus. In practice, the training corpus usually has a limited volume and hardly characterizes the entire distribution. Chasing an evolving data distribution is costly, and even if the training corpus does not become stale, models will still encounter unexpected situations at the test time [34]. Accordingly, training a ranking model on the given corpus that can well generalize to another new corpus is necessary.

Formally, given a ranking model  $f$  learned on  $\{q_i, \mathbf{d}_i, y_i\}_{i=1}^m$  which are drawn from the training distribution  $\mathcal{G}_d$ , we aim to evaluate its performance on the test examples from unseen corpus, which are drawn from a new distribution  $\mathcal{G}'_d$ . Specifically, the OOD generalizability on unforeseen corpus is defined as

$$\mathbb{E}_{(q'_t, \mathbf{d}'_t, y'_t) \sim \mathcal{G}'_d} M(\pi(q'_t, \mathbf{d}'_t, f), y'_t), \quad (14)$$

where  $q'_t$ ,  $\mathbf{d}'_t$  and  $y'_t$  denotes the query, the document list and the label from unseen corpus.

**4.2.2 Metric of OOD Generalizability on Unforeseen Corpus.** To measure the OOD generalizability on unforeseen corpus, we also employ the  $\mathbf{DR}_{OOD}$  metric based on the ranking performance  $P_{OOD}$  on the OOD test set (i.e.,  $(q'_t, \mathbf{d}'_t, y'_t) \sim \mathcal{G}'_d$ ) and the ranking performance  $P_{I.I.D.}$  on the I.I.D. test set (i.e.,  $(q_t, \mathbf{d}_t, y_t) \sim \mathcal{G}_d$ ).

## 4.3 Experimental Settings

In this section, we introduce our experimental settings, including data construction, and implementation details.

**4.3.1 Data Construction.** For evaluation purposes, we build two benchmark dataset based on several existing IR collections. Specifically, we use **Robust04**, **MQ2007**, and **MS MARCO**, which have been described in Section 3.3.1. In these IR collections, queries are associated with the meta-data information, which helps differentiate the examples with respect to query type and corpus,

Table 4. Statistics of our constructed dataset for unforeseen query type

| type                    | Location | Numeric | Person | Description | Entity |
|-------------------------|----------|---------|--------|-------------|--------|
| #Training Queries       | 15,000   | 15,000  | 15,000 | 15,000      | 15,000 |
| #Test Queries           | 300      | 300     | 300    | 300         | 300    |
| Avg #Relevant Documents | 1        | 1       | 1      | 1           | 1      |
| Avg query words         | 5.68     | 6.77    | 5.96   | 5.56        | 6.27   |

respectively. We now describe the detail of the two datasets for evaluating the OOD generalizability on unforeseen query type and corpus as follows.

- **Dataset for Unforeseen Query Type.** Here, we take the MS MARCO dataset as our source data. The reason is that MS MARCO contains various types of questions, and the amount of questions are much larger than other datasets. Firstly, we leverage the official query types<sup>10</sup>, including *Location*, *Numeric*, *Person*, *Description* and *Entity*, to identify the query type for each query. Then, for each query type, we construct the new training/test set, by randomly sampling 15,000/300 queries from the original training/development set respectively. In this way, we obtain a benchmark dataset with 5 different query types. The detailed statistics are shown in Table 4.
- **Dataset for Unforeseen Corpus.** We take Robust04 from news articles, MQ2007 from Gov2 documents, and MS MARCO from web documents, as the whole benchmark dataset to mimic the different data distributions in different corpora. As we can see, they represent different sizes and genres of heterogeneous text collections. For Robust04 and MQ2007, we randomly divide them into a training set (80%) and a test set (20%). For MS MARCO, to obtain a similar volume of relevant query-document pairs for fair comparison, we build the new training set by sampling a quarter of queries from the original training set and directly leverage the original development set as the new test set following [10]. For three corpora, we also randomly sample 20% queries from the training set for validation, respectively. Thus, we can obtain a benchmark dataset with 3 different corpora.

**4.3.2 Implementation Details.** The implementation details, including the pre-processing and model implementation, are similar to that in Section 3.3.2. The only expectation is that the parameters of traditional probabilistic ranking models are tuned on the corresponding development set in the two constructed benchmark dataset.

#### 4.4 Analysis of OOD Generalizability on Unforeseen Query Type

In this section, we analyze the empirical results on the OOD generalizability on unforeseen query type. Specifically, we adopt the widely used metric for MS MARCO, i.e., **MRR@100**, as the implementation of the effectiveness evaluation metric  $M$  in Eq. (12) and (13). Table 5 shows the MRR@100 performance of different ranking models under both the OOD (i.e., the training and test query types are different.) and I.I.D. (i.e., the training and test query types are the same.) settings. For the OOD setting, we train the model on one query type and test it on a different query type. Since the model may be too specific to the single query type it was trained on, we also train the model on

<sup>10</sup><https://github.com/microsoft/MSMARCO-Question-Answering>

Table 5. The MRR@100 performance of different ranking models with respect to unforeseen query type. The five query types in MS MARCO, i.e., Location, Numeric, Person, Description and Entity, are denoted as LOC, NUM, PER, DES and ENT, respectively. The column “train” and “test” denote the training and test set for the ranking models, and the MRR@100 performance is reported on the test set. ALL\* denotes the other four query types where the test query type has been removed from the entire query set. Significant performance degradation with respect to the corresponding I.I.D. setting is denoted as ‘-’ ( $p\text{-value} \leq 0.05$ ).

| Train | Test | QL    | BM25         | Prank | RankSVM | LambdaMART         | DSSM  | DRMM               | Conv-KNRM          | Duet               | BERT               | ColBERT                  |
|-------|------|-------|--------------|-------|---------|--------------------|-------|--------------------|--------------------|--------------------|--------------------|--------------------------|
| LOC   | LOC  | 0.277 | 0.312        | 0.250 | 0.299   | 0.319              | 0.146 | 0.278              | 0.168              | 0.221              | 0.379              | <b>0.432</b>             |
| NUM   | LOC  | 0.276 | 0.296        | 0.243 | 0.276   | 0.300              | 0.120 | 0.216 <sup>-</sup> | 0.105 <sup>-</sup> | 0.120 <sup>-</sup> | 0.295 <sup>-</sup> | <b>0.330<sup>-</sup></b> |
| PER   | LOC  | 0.267 | 0.306        | 0.272 | 0.293   | 0.310              | 0.202 | 0.232 <sup>-</sup> | 0.089 <sup>-</sup> | 0.120 <sup>-</sup> | 0.314 <sup>-</sup> | <b>0.456</b>             |
| DES   | LOC  | 0.274 | 0.299        | 0.242 | 0.275   | 0.289              | 0.176 | 0.202 <sup>-</sup> | 0.097 <sup>-</sup> | 0.132 <sup>-</sup> | 0.292 <sup>-</sup> | <b>0.326<sup>-</sup></b> |
| ENT   | LOC  | 0.276 | 0.305        | 0.265 | 0.278   | 0.306              | 0.091 | 0.203 <sup>-</sup> | 0.110 <sup>-</sup> | 0.143 <sup>-</sup> | 0.300 <sup>-</sup> | <b>0.322<sup>-</sup></b> |
| ALL*  | LOC  | 0.275 | 0.305        | 0.254 | 0.289   | 0.318              | 0.153 | 0.247              | 0.200              | 0.133              | 0.360              | <b>0.366</b>             |
| NUM   | NUM  | 0.249 | 0.270        | 0.223 | 0.230   | 0.278              | 0.137 | 0.190              | 0.115              | 0.130              | 0.250              | <b>0.325</b>             |
| LOC   | NUM  | 0.247 | <b>0.253</b> | 0.224 | 0.237   | 0.238 <sup>-</sup> | 0.065 | 0.186              | 0.091              | 0.091 <sup>-</sup> | 0.190 <sup>-</sup> | 0.227 <sup>-</sup>       |
| PER   | NUM  | 0.242 | <b>0.265</b> | 0.232 | 0.241   | 0.251              | 0.124 | 0.196              | 0.084              | 0.091 <sup>-</sup> | 0.181 <sup>-</sup> | 0.251 <sup>-</sup>       |
| DES   | NUM  | 0.240 | <b>0.267</b> | 0.235 | 0.232   | 0.253              | 0.165 | 0.198              | 0.077 <sup>-</sup> | 0.107              | 0.207              | 0.253 <sup>-</sup>       |
| ENT   | NUM  | 0.245 | <b>0.262</b> | 0.239 | 0.226   | <b>0.262</b>       | 0.078 | 0.180              | 0.083              | 0.120              | 0.203              | 0.260 <sup>-</sup>       |
| ALL*  | NUM  | 0.245 | 0.265        | 0.235 | 0.239   | 0.267              | 0.096 | 0.209              | 0.137              | 0.097              | 0.247              | <b>0.285</b>             |
| PER   | PER  | 0.283 | 0.300        | 0.261 | 0.264   | 0.313              | 0.194 | 0.242              | 0.119              | 0.134              | 0.277              | <b>0.398</b>             |
| LOC   | PER  | 0.271 | 0.290        | 0.258 | 0.260   | 0.284              | 0.060 | 0.206 <sup>-</sup> | 0.102              | 0.110              | 0.271              | <b>0.339</b>             |
| NUM   | PER  | 0.274 | 0.285        | 0.230 | 0.241   | 0.306              | 0.064 | 0.198 <sup>-</sup> | 0.095              | 0.101              | 0.259              | <b>0.320<sup>-</sup></b> |
| DES   | PER  | 0.273 | 0.294        | 0.254 | 0.246   | 0.289              | 0.138 | 0.200 <sup>-</sup> | 0.083 <sup>-</sup> | 0.089 <sup>-</sup> | 0.276              | <b>0.349</b>             |
| ENT   | PER  | 0.274 | 0.292        | 0.242 | 0.244   | 0.291              | 0.078 | 0.203 <sup>-</sup> | 0.086 <sup>-</sup> | 0.119              | 0.249              | <b>0.336<sup>-</sup></b> |
| ALL*  | PER  | 0.271 | 0.296        | 0.249 | 0.248   | 0.295              | 0.074 | 0.206              | 0.132              | 0.107              | 0.308              | <b>0.367</b>             |
| DES   | DES  | 0.236 | 0.251        | 0.211 | 0.235   | 0.283              | 0.195 | 0.233              | 0.104              | 0.151              | 0.250              | <b>0.332</b>             |
| LOC   | DES  | 0.232 | 0.238        | 0.215 | 0.247   | 0.268              | 0.083 | 0.190 <sup>-</sup> | 0.117              | 0.107 <sup>-</sup> | 0.213              | <b>0.291</b>             |
| NUM   | DES  | 0.232 | 0.238        | 0.211 | 0.234   | 0.279              | 0.098 | 0.178 <sup>-</sup> | 0.082              | 0.104 <sup>-</sup> | 0.220              | <b>0.284</b>             |
| PER   | DES  | 0.223 | 0.249        | 0.237 | 0.252   | 0.273              | 0.161 | 0.192 <sup>-</sup> | 0.084              | 0.116              | 0.206              | <b>0.320</b>             |
| ENT   | DES  | 0.232 | 0.239        | 0.212 | 0.244   | 0.278              | 0.112 | 0.200 <sup>-</sup> | 0.099              | 0.133              | 0.214              | <b>0.310</b>             |
| ALL*  | DES  | 0.231 | 0.249        | 0.209 | 0.246   | 0.287              | 0.141 | 0.201              | 0.123              | 0.113              | 0.258              | <b>0.310</b>             |
| ENT   | ENT  | 0.226 | 0.270        | 0.211 | 0.235   | 0.255              | 0.104 | 0.196              | 0.114              | 0.148              | 0.232              | <b>0.293</b>             |
| LOC   | ENT  | 0.223 | <b>0.260</b> | 0.181 | 0.237   | 0.241              | 0.080 | 0.203              | 0.111              | 0.103 <sup>-</sup> | 0.209              | <b>0.260</b>             |
| NUM   | ENT  | 0.224 | 0.251        | 0.174 | 0.231   | 0.256              | 0.073 | 0.201              | 0.089              | 0.097 <sup>-</sup> | 0.229              | <b>0.257</b>             |
| PER   | ENT  | 0.222 | 0.264        | 0.220 | 0.254   | 0.249              | 0.122 | 0.193              | 0.106              | 0.129              | 0.212              | <b>0.293</b>             |
| DES   | ENT  | 0.222 | 0.258        | 0.180 | 0.240   | 0.231              | 0.130 | 0.207              | 0.109              | 0.121              | 0.217              | <b>0.275</b>             |
| ALL*  | ENT  | 0.222 | 0.263        | 0.222 | 0.249   | 0.256              | 0.117 | 0.218              | 0.146              | 0.125              | 0.249              | <b>0.311</b>             |

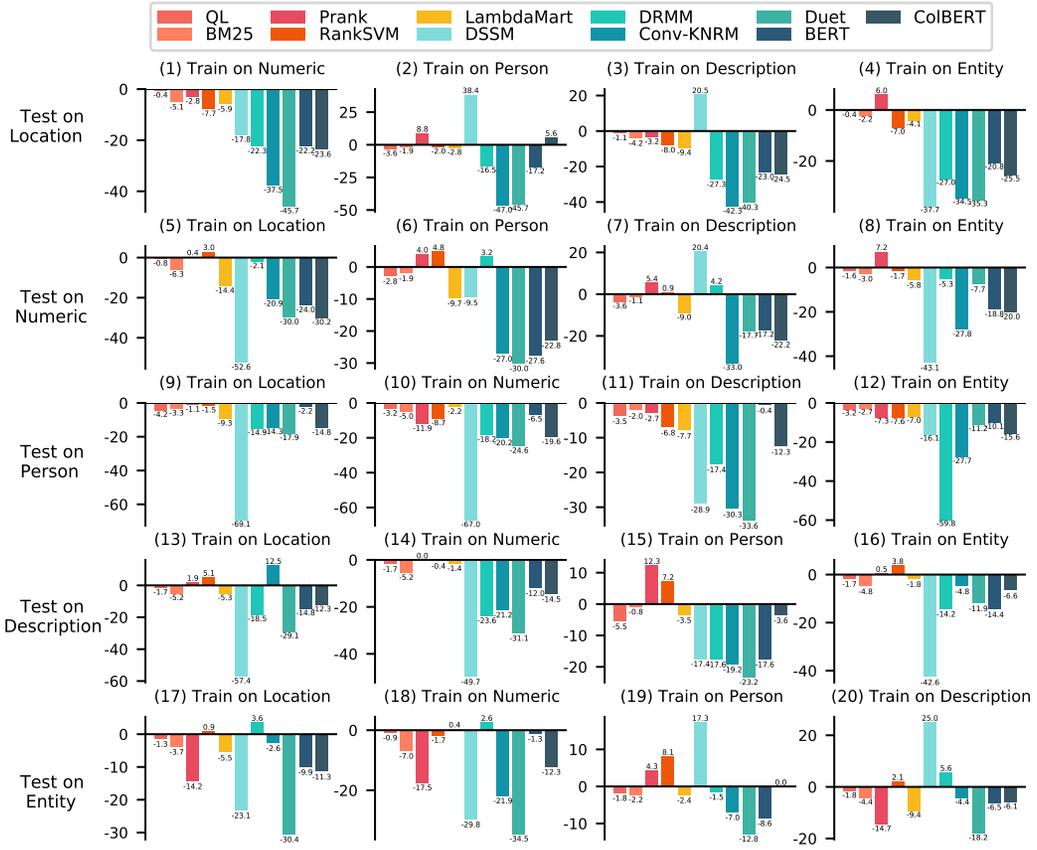


Fig. 4. The OOD generalizability of different ranking models on unforeseen query type in terms of DR<sub>OOD</sub>(%). The test query type is described on the left and the training query type is described above each subfigure.

four query types and test it on the one remaining type. Figure 4 shows the DR<sub>OOD</sub> performance with respect to unforeseen query type.

In the following, we first give an overall analysis of different ranking models. Then we analyze the OOD generalizability of traditional probabilistic ranking models, LTR models and neural ranking models, respectively. Recall that high values of DR<sub>OOD</sub> attest to increased ranking robustness.

**4.4.1 Overall analysis on all the ranking models.** Firstly, we give an overall performance analysis on all the ranking models. We can observe that: (1) Under the I.I.D. setting (e.g., the training and test query type is Person), *ColBERT* generally performs the best (0.398 on Person) followed by *LambdaMART* (0.313 on Person), *BERT* (0.277 on Person) and then *RankSVM* (0.264 on Person), in terms of MRR@100. (2) As show in Figure 4, most ranking models are not able to well generalize to OOD query types. Take the most effective model *ColBERT* as an example, the DR<sub>OOD</sub> value is -30.2% when the training query type is Location and the test query type is Numeric. It indicates that higher effectiveness does not reliably improve OOD generalizability. (3) When being trained on four query types and tested on the one remaining type, most ranking models perform worse than the I.I.D. setting, while perform better than being trained on the single query type. It indicates

that the training data for a single query type may be inadequate. When the training query type increases, the OOD generalizability improves.

**4.4.2 Analysis on traditional probabilistic ranking models and LTR models.** When we look at the traditional probabilistic ranking models and LTR models, we find that: (1) In general, traditional probabilistic ranking models achieve the highest  $DR_{OOD}$  values, indicating that traditional probabilistic ranking models are the most robust models when facing unforeseen query types. A possible reason would be that as unsupervised methods, traditional probabilistic ranking models avoid the problem of overfitting the training data and thus have a better OOD generalizability on unforeseen query types. (2) The  $DR_{OOD}$  values of LTR models are higher than that of neural ranking models, indicating that LTR models are more robust than neural ranking models. The reason might be that hand-crafted features in LTR models could better generalize to unforeseen query type than automatically learned features by neural networks.

**4.4.3 Analysis on neural ranking models.** When we look at the neural ranking models, we find that: (1) In general, neural ranking models achieve the lowest  $DR_{OOD}$  values among all the ranking models. The reason might be that neural ranking models with a deeper network architecture fit the normal in-distribution query types better, at the cost of further loss in performance on the held out OOD query types. It is consistent with the finding in [10]. (2) Among these five neural ranking models, pre-trained models (i.e., *BERT* and *ColBERT*) have the best OOD generalizability on unforeseen query type. For example, *ColBERT* trained (fine-tuned) on Person and tested on Location, the  $MRR@100$  value even improves 5.6% compared with *ColBERT* trained and tested on Location. A possible explanation would be that pre-training on the huge text corpus can improve OOD generalization. It is consistent with the finding in [34], which indicates that pre-trained models are more robust to OOD examples on several NLP tasks. (3) Pre-trained models have shown great effectiveness under both the OOD and I.I.D. settings, and good robustness to OOD query types. Future works could design novel pre-training objectives tailored for IR that enhance ranking robustness.

## 4.5 Analysis of OOD Generalizability on Unforeseen Corpus

In this section, we analyze the empirical results on the OOD generalizability on unforeseen corpus. Similar to the used effectiveness evaluation metric in Section 3.4.1, we also adopt the **MAP**, **NDCG@10**, **NDCG@20**, **P@10**, **P@20**, **MRR@10** and **MRR@100** as the implementation of  $M$  in Eq. (12) and (13). Table 6 shows the MAP, NDCG@10, NDCG@20, P@10, P@20, MRR@10 and MRR@100 performance of all the ranking models under both the OOD (i.e., the training and test corpora are different) and I.I.D. (i.e., the training and test corpora are the same) settings. Figure 5 shows the  $DR_{OOD}$  performance based on the MAP, NDCG, P and MRR, respectively.

In the following, we first give an overall analysis of different ranking models. Then we analyze the OOD generalizability of traditional probabilistic ranking models, LTR models and neural ranking models, respectively. Recall that high values of  $DR_{OOD}$  attest to increased ranking robustness.

**4.5.1 Overall analysis on all the ranking models.** Firstly, we give an overall analysis on all the ranking models. We can observe that: (1) Under the I.I.D. setting (e.g., the training and test corpus is MQ2007), *ColBERT* generally performs the best (0.4759) followed by *BERT* (0.4656) and *RankSVM* (0.4601), in terms of MAP. (2) As show in Figure 5, almost all  $DR_{OOD}$  values are negative. It indicates that most ranking models are not able to well generalize to OOD examples. (3) Besides, the  $DR_{OOD}$  value with respect to MRR for unforeseen MS MARCO is much lower than that for unforeseen query types sampled from MS MARCO. A possible reason is that compared with different query types sampled from the same corpus, different corpora have greater differences among samples.

Table 6. The performance of different ranking models with respect to unforeseen corpus. The “train” and “test” denote the training and test corpus for the ranking models, and the MAP, NDCG@10, NDCG@20, P@10, P@20, MRR@10 and MRR@100 performance are reported on the test set. Significant performance degradation with respect to the corresponding I.I.D. setting is denoted as ‘-’ ( $p$ -value  $\leq 0.05$ ).

| Model      | test     | Robust04            |                     |                     | MQ2007              |                     |                     | MS MARCO            |                     |
|------------|----------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
|            | train    | MAP                 | NDCG@20             | P@20                | MAP                 | NDCG@10             | P@10                | MRR@10              | MRR@100             |
| QL         | Robust04 | <b>0.2553</b>       | 0.4575              | 0.3860              | 0.3646              | 0.3662              | 0.3349              | 0.2301              | 0.2426              |
|            | MQ2007   | 0.2492              | 0.4561              | 0.3960              | 0.3736              | 0.3675              | 0.3293              | 0.2368              | 0.2488              |
|            | MS MARCO | 0.2490              | 0.4525              | 0.3920              | 0.3714              | 0.3622              | 0.3240              | 0.2374              | 0.2495              |
| BM25       | Robust04 | 0.2522              | 0.4514              | 0.3760              | 0.3132 <sup>-</sup> | 0.3197 <sup>-</sup> | 0.2843 <sup>-</sup> | 0.2051 <sup>-</sup> | 0.2174 <sup>-</sup> |
|            | MQ2007   | 0.2227              | 0.4280              | 0.3600              | 0.3863              | 0.3936              | 0.3438              | 0.2618              | 0.2732              |
|            | MS MARCO | 0.1971              | 0.3995              | 0.3370              | 0.3837              | 0.3831              | 0.3391              | 0.2660              | 0.2782              |
| Prank      | Robust04 | 0.2186              | 0.4011              | 0.3620              | 0.3410 <sup>-</sup> | 0.2847 <sup>-</sup> | 0.2719              | 0.2038 <sup>-</sup> | 0.2171 <sup>-</sup> |
|            | MQ2007   | 0.2395              | 0.4357              | 0.3830              | 0.3834              | 0.3382              | 0.3056              | 0.1531 <sup>-</sup> | 0.1676 <sup>-</sup> |
|            | MS MARCO | 0.2162              | 0.3848              | 0.3440              | 0.3583              | 0.3076              | 0.3030              | 0.2373              | 0.2490              |
| RankSVM    | Robust04 | 0.2501              | 0.4510              | 0.3760              | 0.3884 <sup>-</sup> | 0.3502 <sup>-</sup> | 0.3145 <sup>-</sup> | 0.2350              | 0.2468              |
|            | MQ2007   | 0.2340              | 0.4190              | 0.3680              | 0.4601              | 0.4376              | 0.3820              | 0.2021 <sup>-</sup> | 0.2160 <sup>-</sup> |
|            | MS MARCO | 0.2523              | 0.4529              | 0.3860              | 0.3948 <sup>-</sup> | 0.3652 <sup>-</sup> | 0.3343 <sup>-</sup> | 0.2410              | 0.2535              |
| LambdaMart | Robust04 | 0.2397              | 0.4357              | 0.3650              | 0.3562 <sup>-</sup> | 0.3148 <sup>-</sup> | 0.2970 <sup>-</sup> | 0.2172 <sup>-</sup> | 0.2301 <sup>-</sup> |
|            | MQ2007   | 0.2057              | 0.3780              | 0.3300              | 0.4578              | 0.4381              | 0.3787              | 0.2068 <sup>-</sup> | 0.2192 <sup>-</sup> |
|            | MS MARCO | 0.2174              | 0.4026              | 0.3160              | 0.3365 <sup>-</sup> | 0.2829 <sup>-</sup> | 0.2675 <sup>-</sup> | 0.2602              | 0.2710              |
| DSSM       | Robust04 | 0.1392              | 0.2455              | 0.2270              | 0.3158 <sup>-</sup> | 0.2407 <sup>-</sup> | 0.2441 <sup>-</sup> | 0.1231              | 0.1350              |
|            | MQ2007   | 0.1229              | 0.2295              | 0.2260              | 0.3988              | 0.3530              | 0.3438              | 0.0159 <sup>-</sup> | 0.0336 <sup>-</sup> |
|            | MS MARCO | 0.1262              | 0.2360              | 0.2160              | 0.2785 <sup>-</sup> | 0.1761 <sup>-</sup> | 0.1902 <sup>-</sup> | 0.1051              | 0.1220              |
| DRMM       | Robust04 | 0.2541              | <b>0.4744</b>       | <b>0.4060</b>       | 0.4231              | 0.3859              | 0.3524              | 0.1864              | 0.2002              |
|            | MQ2007   | 0.1829              | 0.3427 <sup>-</sup> | 0.2930 <sup>-</sup> | 0.4390              | 0.4038              | 0.3719              | 0.0552 <sup>-</sup> | 0.0711 <sup>-</sup> |
|            | MS MARCO | 0.1357 <sup>-</sup> | 0.2445 <sup>-</sup> | 0.2380 <sup>-</sup> | 0.3024 <sup>-</sup> | 0.2101 <sup>-</sup> | 0.2257 <sup>-</sup> | 0.1165              | 0.1334              |
| Conv-KNRM  | Robust04 | 0.1735              | 0.3084              | 0.2710              | 0.3089 <sup>-</sup> | 0.2332 <sup>-</sup> | 0.2479 <sup>-</sup> | 0.0301 <sup>-</sup> | 0.0483 <sup>-</sup> |
|            | MQ2007   | 0.1372              | 0.2439              | 0.2210              | 0.4115              | 0.3670              | 0.3568              | 0.0123 <sup>-</sup> | 0.0278 <sup>-</sup> |
|            | MS MARCO | 0.1389              | 0.2654              | 0.2480              | 0.2665 <sup>-</sup> | 0.1515 <sup>-</sup> | 0.1787 <sup>-</sup> | 0.1559              | 0.1716              |
| Duet       | Robust04 | 0.1512              | 0.2592              | 0.2380              | 0.3156 <sup>-</sup> | 0.2470 <sup>-</sup> | 0.2509 <sup>-</sup> | 0.0161 <sup>-</sup> | 0.0337 <sup>-</sup> |
|            | MQ2007   | 0.1319              | 0.2332              | 0.2030              | 0.4126              | 0.3629              | 0.3527              | 0.0114 <sup>-</sup> | 0.0270 <sup>-</sup> |
|            | MS MARCO | 0.1338              | 0.2469              | 0.2270              | 0.2616 <sup>-</sup> | 0.1442 <sup>-</sup> | 0.1749 <sup>-</sup> | 0.1219              | 0.1392              |
| BERT       | Robust04 | 0.2386              | 0.4407              | 0.3830              | 0.3772 <sup>-</sup> | 0.3178 <sup>-</sup> | 0.2988 <sup>-</sup> | 0.1509 <sup>-</sup> | 0.1667 <sup>-</sup> |
|            | MQ2007   | 0.2178              | 0.4262              | 0.3740              | 0.4656              | 0.4435              | 0.3929              | 0.0910 <sup>-</sup> | 0.1091 <sup>-</sup> |
|            | MS MARCO | 0.2091              | 0.4048              | 0.3680              | 0.3215 <sup>-</sup> | 0.2488 <sup>-</sup> | 0.2509 <sup>-</sup> | 0.2691              | 0.2798              |
| ColBERT    | Robust04 | 0.2136              | 0.4126              | 0.3600              | 0.3741 <sup>-</sup> | 0.3231 <sup>-</sup> | 0.3133 <sup>-</sup> | 0.1048 <sup>-</sup> | 0.1226 <sup>-</sup> |
|            | MQ2007   | 0.1967              | 0.3785              | 0.3460              | <b>0.4759</b>       | <b>0.4500</b>       | <b>0.4059</b>       | 0.0937 <sup>-</sup> | 0.1116 <sup>-</sup> |
|            | MS MARCO | 0.2077              | 0.4149              | 0.3580              | 0.3087 <sup>-</sup> | 0.2263 <sup>-</sup> | 0.2328 <sup>-</sup> | <b>0.3279</b>       | <b>0.3360</b>       |

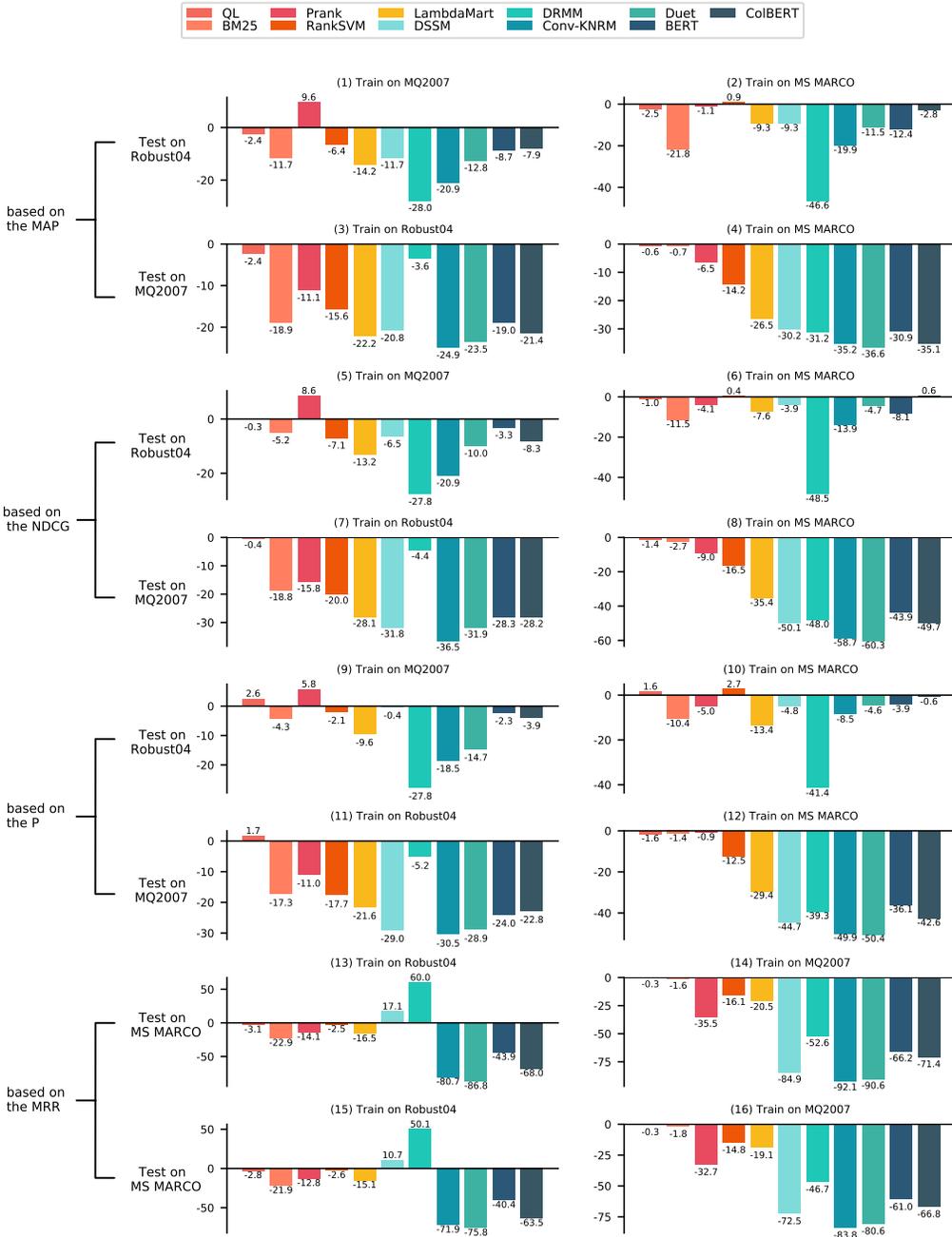


Fig. 5. The OOD generalizability on unforeseen corpus in terms of DR<sub>OOD</sub>(%) based on the MAP, NDCG, P, and MRR respectively. For each subfigure, the test corpus is described on the left and the training corpus is described above.

**4.5.2 Analysis on traditional probabilistic ranking models and LTR models.** When we look at the traditional probabilistic ranking models and LTR models, we can find that: (1) Under the I.I.D. setting, *BM25* performs better than *QL* on most corpus, and *BM25* is a strong baseline which performs better than LTR models on Robust04. Under the OOD setting, *QL* is more robust to the unforeseen corpus than *BM25*. (2) LTR models show a good transfer effectiveness from MS MARCO to Robust04. For instance, the  $DR_{OOD}$  value based on MAP is close to 0 (i.e., -1.1%) when we train *Prank* on MS MARCO and then test it on Robust04. Besides, it is surprising that when we train *RankSVM* on MS MARCO and test it on Robust04, the  $DR_{OOD}$  value based on P@20 is positive (i.e., 2.7%). The reason might be that the distribution of hand-crafted features for Robust04 is similar to that for MS MARCO. (3) *Prank* shows a better OOD generalizability on unforeseen corpus than *RankSVM* and *LambdaMART* if we test them on Robust04 and MQ2007. For example, when we train *Prank* on MQ2007 and test it on Robust04, the  $DR_{OOD}$  value based on NDCG@20 is even positive (i.e., 8.6%), while *LambdaMART* achieves -13.2%  $DR_{OOD}$  under the same setting. One possible explanation is that *Prank* has simpler model architecture and less parameters compared with *RankSVM* and *LambdaMART*, which alleviates the problem of overfitting and increases the OOD generalizability.

**4.5.3 Analysis on neural ranking models.** When we look at the neural ranking models, we can see that: (1) In general, neural ranking models show the worst OOD generalizability on unforeseen corpus. The finding is consistent with that on unforeseen query type. For example, when we train *Conv-KNRM* on MQ2007 and test it on MS MARCO, the  $DR_{OOD}$  value based on MRR@100 is even -83.8%. (2) *DRMM* shows a good transfer effectiveness from Robust04 to both the MS MARCO and MQ2007. As shown in previous studies [29], *DRMM* have performed quite well on Robust04. In this way, *DRMM* trained on Robust04 may lean to generalize to other corpora.

## 5 DEFENSIVE ABILITY AGAINST ADVERSARIAL OPERATIONS

Deep neural networks have been found vulnerable to adversarial attack, where an imperceptible perturbation can trigger dramatic changes in the final result [3, 101]. However, the vulnerability of neural ranking models remains under-explored. This poses a serious security risk on the practical search engines, where an imperceptible adversarial perturbation to the query/document can be sufficient to intentionally fool the neural ranking model to make wrong decisions, i.e., raise or lower the ranks of selected candidate documents.

Therefore, in this section, we analyze the robustness of ranking models according to their ability to defend against adversarial operations, i.e., the defensive ability which has been formulated in Eq. (4). Specifically, we introduce two types of adversarial operations to measure the defensive ability, i.e., 1) The ability to defend against the query attack, and 2) The ability to defend against the document manipulation.

### 5.1 Defensive Ability against Query Attack

In this section, we introduce the definition and metric of the defensive ability against query attack.

**5.1.1 Definition of Defensive Ability against Query Attack.** The defensive ability against query attack refers to that the ranking performance on the attacked queries. In a real-world scenario, the users' queries may be attacked unconsciously. For example, users may occasionally misspell or mistype a query keyword when performing a search. Such query typo is a longstanding real-world problem for IR, which has been extensively studied [31, 67, 103]. From the perspective of a user, if the search engine cannot tolerate the query typo, the user's satisfaction will remarkably drop. A reliable system that always produces acceptable retrieval performance, is more preferred than

another system that fails on the occasional query typo. Specifically, we consider the effects of misspelled or mistyped queries by exploring several types of character-level and word-level edits.

Formally, given a ranking model  $f$  learned on  $\{q_i, \mathbf{d}_i, \mathbf{y}_i\}_{i=1}^m$  which are drawn from the training distribution  $\mathcal{G}$ , we aim to evaluate its performance on the I.I.D. test queries  $q_t$  attacked by a query attack function  $a_q$ , i.e.,  $a_q(q_t)$ . Specifically, the defensive ability against query attack is defined as

$$\mathbb{E}_{(q_t, \mathbf{d}_t, \mathbf{y}_t) \sim \mathcal{G}} M(\pi(a_q(q_t), \mathbf{d}_t, f), \mathbf{y}_t), \quad (15)$$

where  $q_t$ ,  $\mathbf{d}_t$  and  $\mathbf{y}_t$  denotes the query, the document list and the label from the I.I.D. test set, respectively.

**5.1.2 Metric of Defensive Ability against Query Attack.** As for evaluation measure, we employ one automatic metric, namely,

- **DR<sub>query</sub>** evaluates the *drop rate* between the model performance  $P_o$  on the original queries (i.e.,  $q_t$ ) and the ranking performance  $P_t$  on the attacked queries (i.e.,  $a_q(q_t)$ ), i.e.,

$$DR_{query} = \frac{P_t - P_o}{P_o}, \quad (16)$$

where  $P_o$  and  $P_t$  are defined as

$$P_o = \mathbb{E}_{(q_t, \mathbf{d}_t, \mathbf{y}_t) \sim \mathcal{G}} M(\pi(q_t, \mathbf{d}_t, f), \mathbf{y}_t), \quad (17)$$

$$P_t = \mathbb{E}_{(q_t, \mathbf{d}_t, \mathbf{y}_t) \sim \mathcal{G}} M(\pi(a_q(q_t), \mathbf{d}_t, f), \mathbf{y}_t), \quad (18)$$

where the effectiveness evaluation metric  $M$  can be defined in different ways, such as MRR@100, with respect to the specific experimental dataset. The ranking model would be more robust with a higher DR<sub>query</sub>.

## 5.2 Defensive Ability against Document Manipulation

In this section, we introduce the definition and metric of the ability to defend against document manipulation.

**5.2.1 Definition of Defensive Ability against Document Manipulation.** The defensive ability against document manipulation refers to that the ranking performance on the adversarial documents. In the Web search setting, many document authors are “ranking-incentivized” [26]. That is, they may introduce modifications to their documents, hoping to rank them higher for some queries by search engines. In this way, these modifications could be almost indiscernible by users and users will lose faith in the search system if they observe these rapid ranking changes. Thus, a ranking model should be robust to such manipulations for maintaining high-quality search results.

Formally, given a ranking model  $f$  learned on  $\{q_i, \mathbf{d}_i, \mathbf{y}_i\}_{i=1}^m$  which are drawn from the training distribution  $\mathcal{G}$ , we aim to evaluate its performance on the I.I.D. test queries  $q_t$  with the corresponding document list  $\mathbf{d}_t$  attacked based on a document attack function  $a_d$ , i.e.,  $a_d(\mathbf{d}_t)$ . Specifically, the defensive ability against document manipulation is defined as

$$\mathbb{E}_{(q_t, \mathbf{d}_t, \mathbf{y}_t) \sim \mathcal{G}} M(\pi(q_t, a_d(\mathbf{d}_t), f), \mathbf{y}_t), \quad (19)$$

where  $q_t$ ,  $\mathbf{d}_t$  and  $\mathbf{y}_t$  denotes the query, the document list and the label from the I.I.D. test set, respectively.

**5.2.2 Metric of Defensive Ability against Document Manipulation.** As for evaluation measure, we employ Top Change (TC) and Kendall’s- $\tau$  distance (KT) following the previous work [27].

Suppose that  $L^{(q_t)}$  is a ranked document list with respect to a given query  $q_t$  achieved by a ranking model. After the document manipulation on the document list  $\mathbf{d}_t$ , a new ranked list  $L^{(q_t)'}$  is obtained by the ranking model. The key idea to quantify robustness is to measure the distance

between  $L^{(q_t)}$  and  $L^{(q_t)'}$ . Specifically, the lower the TC and KT value, the lower the “distance” between the two lists, i.e., the more robust the ranking model is.

- **TC** indicates the similarity between the two ranked lists based on the highest ranked document, i.e.,

$$TC = \mathbb{E}_{(q_t, d_t, y_t) \sim \mathcal{G}} \delta(L_1^{(q_t)} \neq L_1^{(q_t)'}), \quad (20)$$

where  $L_1^{(q_t)}$  and  $L_1^{(q_t)'}$  denotes the highest ranked document in the ranked list  $L^{(q_t)}$  and  $L^{(q_t)'}$ , respectively.  $\delta$  is the indicator function, where the value is 1 if the highest ranked document in  $L^{(q_t)}$  and  $L^{(q_t)'}$  is different, and 0 otherwise.

- **KT** is defined as the number of discordant pairs between two paired lists normalized with respect to the number of pairs of items in a list, i.e.,

$$KT = \mathbb{E}_{(q_t, d_t, y_t) \sim \mathcal{G}} \frac{\sum_{i,j \in P} \bar{K}_{ij}(L^{(q_t)}, L^{(q_t)'})}{n_{q_t}(n_{q_t} - 1)/2}, \quad (21)$$

where  $P$  is the set of unordered pairs of distinct documents in  $L^{(q_t)}$  and  $L^{(q_t)'}$ .  $n_{q_t}$  is the size of the ranked list.  $\bar{K}_{ij}(L^{(q_t)}, L^{(q_t)'})$  denotes whether the document pair  $(d_{ti}, d_{tj})$  is a discordant pair. A discordant pair is two documents whose relative ranking in one list is different than that in the other list. Specifically,  $\bar{K}_{ij}(L^{(q_t)}, L^{(q_t)'})$  is defined as,

$$\bar{K}_{ij}(L^{(q_t)}, L^{(q_t)'}) = \delta((L^{(q_t)}(d_{ti}) - L^{(q_t)}(d_{tj}))(L^{(q_t)'}(d_{ti}) - L^{(q_t)'}(d_{tj})) < 0), \quad (22)$$

where  $L^{(q_t)}(d_{ti})$  and  $L^{(q_t)'}(d_{ti})$  are the rankings of the document  $d_{ti}$  in  $L^{(q)}$  and  $L^{(q)'}$ , respectively.

### 5.3 Experimental Settings

We first introduce our experimental settings, including datasets and implementation details.

**5.3.1 Datasets.** The datasets used for evaluating the defensive ability are as follows.

To evaluate the defensive ability against query attack, we conduct experiments on the **MS MARCO** dataset, which has been described in Section 4. We construct the novel training set by randomly sampling a quarter of queries from the original training set. We randomly sample 20% queries from the novel training set for validation. Besides, we directly use the original development set as the novel test set.

To evaluate the defensive ability against document manipulation, we follow the previous work [27] to conduct experiments on the **ASRC** and **ClueWeb09-B** dataset. The detailed statistics of these datasets are shown in Table 7.

- **ASRC.** Adversarial Search Collection (ASRC)<sup>11</sup> is an adversarial ranking dataset, which aims to analyze content-based ranking competitions so as to shed light on the strategic behavior of publishers [74]. The competition included 31 different repeated matches, each of which was with respect to a different TREC’s ClueWeb09 query. The competition was run for eight rounds. Students are incentivized by course-grade rewards to manipulate their documents, in order to have them ranked higher in the next round.
- **ClueWeb09-B.** ClueWeb09-B is a large Web collection with 150 queries and over 50M English documents, whose queries are accumulated from TREC Web Track 2009, 2010, and 2011.

<sup>11</sup><https://github.com/asrcdataset/>

Table 7. Statistics of datasets used for evaluating the ability to defend against document manipulation

|                         | ASRC  | ClueWeb09-B |
|-------------------------|-------|-------------|
| #Queries                | 31    | 200         |
| #Documents              | 1,279 | 50M         |
| Avg #Relevant Documents | 36    | 55          |

5.3.2 *Implementation Details.* The model implementation details are similar to that in Section 4.3.2. In the following, we will introduce our methods to simulate the adversarial query attack and document manipulation.

- *Implementation of Query Attack.* The MS MARCO dataset is preprocessed in the same way as described in Section 4.3.2. We use the official top 100 ranked documents retrieved by QL. Then, to implement the query attack, we simulate the query attack at the character-level and word-level.

For character-level query attack, we follow the previous works [42, 70], which are inspired by psycholinguistic studies [19, 75]. The psycholinguistic studies demonstrate that humans can comprehend text altered by jumbling internal characters, provided that the first and last characters of each word remain unperturbed.

Specifically, we explore to perturb queries with four types of character-level edits: (1) Add, inserting a new lower-case character internally in a word; (2) Remove, deleting an internal character of a word; (3) Substitute: substituting an internal character for any letter; (4) Swap, swapping two adjacent internal characters of a word. For each query in the test set, we firstly randomly choose a word and then attack it using one randomly-chosen character-level edit. We denote such attacked queries as *1-char* attacked queries and there are 25.32% Add, 24.96% Remove, 23.34% Substitute and 24.42% Swap in the final obtained *1-char* attacked queries.

Furthermore, we also randomly select one word from the *1-char* attacked queries and then attack it using one randomly-chosen character-level edit [70]. We denote such attacked queries as *2-char* attacked queries. We evaluate the performance of different ranking models on the original test queries, *1-char* and *2-char* attacked test queries respectively.

For word-level query attack, we explore to perturb queries with three types of word-level edits: (1) Add, inserting a new lower-case word in a query; (2) Remove, deleting a word of a query; (3) Substitute: substituting a word with a random word. For each query in the test set, we firstly randomly choose a word and then attack it using one randomly-chosen word-level edit.

- *Implementation of Document Manipulation.* We use the public dataset ASRC that was created as a result of an on-going ranking competition [74], to measure the document manipulation [27]. Specifically, the ranking competition involved 31 repeated matches that lasted for 8 rounds, where each match was with respect to a different query. Students in an IR course served as documents' authors. In the first round, in addition to the query itself, students were provided with an example relevant document, and were incentivized by bonus points to the course's grade to modify their documents so as to have them ranked as high as possible in the next round. Starting from the second round, students were presented with the ranking as well as the content of all documents submitted in the previous round in the match. To assure the fairness of the ranking competition, students had no prior knowledge of the ranking

function and all data was anonymized. In this way, the ASRC dataset could simulate the document manipulation in the ranking competition.

Since the ASRC dataset is too small to effectively train a ranking model, we firstly train the ranking models on the ClueWeb09-B, and then evaluate their defensive abilities on the ASRC [27]. Specifically, we randomly sample 75% queries from the ClueWeb09-B for training and leverage the remaining 25% queries for validation. For the ClueWeb09-B dataset, we use the QL model to retrieve the top 100 ranked documents to build an initial document list.

## 5.4 Ranking Robustness to Query Attack

In this section, we analyze the empirical results on the defensive ability against query attack. Specifically, we adopt the widely used metric for MS MARCO, i.e., **MRR@100**, as the implementation of the effectiveness evaluation metric  $M$  in Eq. (17) and (18). Table 8 and Table 9 show the MRR@100 and  $DR_{query}$  performance of ranking models against the character-level and the word-level query attack, respectively. We first analyze the defensive ability of different ranking models against the character-level query attack. Then, we analyze the defensive ability of different ranking models against the word-level query attack. Recall that high values of  $DR_{query}$  attest to increased ranking robustness.

**5.4.1 Analysis on the defensive ability against the character-level query attack.** We first analyze the defensive ability against the character-level query attack. In the following, we first give an overall analysis of different ranking models. Then, we analyze traditional probabilistic ranking models, LTR models and neural ranking models, respectively.

- Overall analysis on all the ranking models.** Firstly, we give an overall analysis on all the ranking models against the query attack. We can observe that: (1) In the absence of any query attack, *ColBERT* performs the best (0.3360) followed by *BERT* (0.2798), *LambdaMART* (0.2710) and then *BM25* (0.2612), in terms of MRR@100. However, even single-character attacks can be catastrophic, resulting in a significantly degraded performance of 26.1%, 27.6%, 13.2% and 23.3% in terms of  $DR_{query}$ , for *ColBERT*, *BERT*, *LambdaMART*, and *BM25*, respectively. (2) For all the ranking models, the MRR@100 performances on both *1-char* and *2-char* attacked queries are worse than that on the original queries. These results indicate that all the ranking models are not able to generalize to attacked queries quite well. (3) The absolute drop rate value between *2-char* attacked queries and original queries are much higher than that between *1-char* attacked queries and original queries (e.g., 46.4% v.s. 20.8% for *RankSVM*). By conducting further analysis, we find that the absolute drop rate value could be larger with more characters in a query attacked. Therefore, it is necessary to improve the robustness of ranking models against the query attack. (4) Overall, in terms of the  $DR_{query}$  between *1-char* attacked and original queries, the relative robustness order of defending against query attack is  $DRMM < QL < BERT < ColBERT < BM25 < RankSVM < Prank < LambdaMART < Conv-KNRM < Duet < DSSM$ .
- Analysis on traditional probabilistic ranking models and LTR models.** When we look at the traditional probabilistic ranking models and LTR models, we find that: (1) Traditional probabilistic ranking models are less robust than LTR models under *1-char* attack (e.g., -30.7% of *QL* v.s. -13.2% of *LambdaMART*). Specifically, the adversarial edits might flip words either to a different word in the vocabulary or, more often, to the out-of-vocabulary token UNK. Consequently, adversarial edits can degrade ranking models by transforming the informative words in a query to UNK. For traditional probabilistic ranking models, they emphasize too much on exact matching signals between the query and the document, and treat each unique character combination differently, resulting in the vulnerability to query attack. (2) For the LTR models, *LambdaMART* is more robust than *RankSVM* and *Prank*, especially against the *2-char* attack. One

Table 8. The performance of different ranking models against the character-level query attack on the MS MARCO dataset. ‘-’ indicates statistically significant degradation with respect to the original query performance ( $p$ -value  $\leq 0.05$ ).

|            | Original      | 1-char                    |                     | 2-char                    |                     |
|------------|---------------|---------------------------|---------------------|---------------------------|---------------------|
|            | MRR@100       | MRR@100                   | DR <sub>query</sub> | MRR@100                   | DR <sub>query</sub> |
| QL         | 0.2221        | 0.1540 <sup>-</sup>       | -30.7%              | 0.1213 <sup>-</sup>       | -45.4%              |
| BM25       | 0.2612        | 0.2004 <sup>-</sup>       | -23.3%              | 0.1453 <sup>-</sup>       | -44.4%              |
| Prank      | 0.2311        | 0.1839 <sup>-</sup>       | -20.4%              | 0.1218 <sup>-</sup>       | -47.3%              |
| RankSVM    | 0.2535        | 0.2008 <sup>-</sup>       | -20.8%              | 0.1360 <sup>-</sup>       | -46.4%              |
| LambdaMART | 0.2710        | 0.2353 <sup>-</sup>       | -13.2%              | <b>0.2100<sup>-</sup></b> | -22.5%              |
| DSSM       | 0.1220        | 0.1210                    | <b>-0.8%</b>        | 0.1209                    | <b>-0.9%</b>        |
| DRMM       | 0.1335        | 0.0875 <sup>-</sup>       | -34.5%              | 0.0828 <sup>-</sup>       | -38.0%              |
| Conv-KNRM  | 0.1716        | 0.1526 <sup>-</sup>       | -11.1%              | 0.1359 <sup>-</sup>       | -20.8%              |
| Duet       | 0.1392        | 0.1315                    | -5.5%               | 0.1310                    | -5.9%               |
| BERT       | 0.2798        | 0.2026 <sup>-</sup>       | -27.6%              | 0.1376 <sup>-</sup>       | -50.8%              |
| ColBERT    | <b>0.3360</b> | <b>0.2605<sup>-</sup></b> | -26.1%              | 0.1640 <sup>-</sup>       | -51.2%              |

possible explanation is that as a listwise LTR method, *LambdaMART* can make use of the whole ranked document list, which can better reduce the effect of character-level adversarial attacks with more context information than the pairwise LTR method *RankSVM* and the pointwise LTR method *Prank*.

- **Analysis on neural ranking models.** When we look at the neural ranking models, we find that: (1) *DSSM* is the most robust model against both the *1-char* and *2-char* query attack, but performs poorly on the original query set. The reason might be that *DSSM* utilizes a character-level n-gram based word hashing, which is more robust to misspelling problems than just treating each word as the basic semantic units. (2) *Duet* is the second most robust model against both the *1-char* and *2-char* query attack, while also performs poorly on the original query set. Specifically, *Duet* leverages both the local and distributed representations for text matching. In the distributed representation, an activation pattern that has some errors or other differences from past data can still be mapped to the query, using a similarity function. In this way, *Duet* is robust to noise and has the ability to generalize [60]. (3) Since *DSSM* and *Duet* have shown strong robustness, a promising direction to design a robust neural ranking model against the misspelling lies in applying the character-level operations. For instance, we could combine pre-training objectives with the n-gram word hashing layer to simultaneously achieve good ranking effectiveness and robustness. (4) *DRMM* is the least robust model against the *1-char* attacked queries. The result indicates that the adversarial edits significantly degrade the *DRMM* model which directly uses GloVe [68] word embeddings, by transforming the informative words to UNK. Besides, for *DRMM*, the improvement of the absolute drop rate value with respect to *2-char* queries over *1-char* queries is only 3.5%. It is an interesting finding that while *DRMM* is susceptible to adversarial typos, it shows some robustness against the further attacks. (5) When we take a look at the pre-trained models (i.e., *BERT* and *ColBERT*), the phenomenon is absolutely different with *DRMM*. For example, for *ColBERT*, the improvement of absolute drop rate value with respect to *2-char*

Table 9. The performance of different ranking models against the word-level query attack on the MS MARCO dataset. ‘-’ indicates statistically significant degradation with respect to the original query performance ( $p\text{-value} \leq 0.05$ ).

|            | Original      | word-level query attack   |                     |
|------------|---------------|---------------------------|---------------------|
|            | MRR@100       | MRR@100                   | DR <sub>query</sub> |
| QL         | 0.2221        | 0.1433 <sup>-</sup>       | -35.5%              |
| BM25       | 0.2612        | 0.1915 <sup>-</sup>       | -24.3%              |
| Prank      | 0.2311        | 0.1976 <sup>-</sup>       | -14.5%              |
| RankSVM    | 0.2535        | 0.2184 <sup>-</sup>       | -13.8%              |
| LambdaMART | 0.2710        | <b>0.2483<sup>-</sup></b> | -8.4%               |
| DSSM       | 0.1220        | 0.1208                    | <b>-1.0%</b>        |
| DRMM       | 0.1335        | 0.0898 <sup>-</sup>       | -32.7%              |
| Conv-KNRM  | 0.1716        | 0.1528 <sup>-</sup>       | -11.0%              |
| Duet       | 0.1392        | 0.1325                    | -4.8%               |
| BERT       | 0.2798        | 0.1931 <sup>-</sup>       | -31.0%              |
| ColBERT    | <b>0.3360</b> | 0.2221 <sup>-</sup>       | -33.9%              |

queries over *1-char* queries is 25.1%, which is the highest among all the ranking models. These results indicate that the pre-trained models are less robust than other ranking models against the query attack. One possible explanation could be that such pre-trained models apply WordPiece [90] tokenization, where a word is broken down into more than one sub-words. In this way, an attacked word in a query may be decomposed into more than one attacked sub-words, which have a more significant impact on the performance.

**5.4.2 Analysis on the defensive ability against the word-level query attack.** We analyze the defensive ability against the word-level query attack. From the results, we can observe that: (1) Overall, for all the ranking models, the MRR@100 performance on the word-level query attack is worse than that on the original queries. The results indicate that all the ranking models are not able to generalize to word-level attacked queries quite well. (2) In terms of the DR<sub>query</sub> between *word-level* attacked and original queries, the relative robustness order of defending against word-level query attack is  $QL < ColBERT < DRMM < BERT < BM25 < Prank < RankSVM < Conv-KNRM < LambdaMart < Duet < DSSM$ . (3) Traditional probabilistic ranking models are less robust than LTR models under *word-level* attack (e.g., -35.5% of *QL* v.s. -8.4% of *LambdaMART*). Meanwhile, for the LTR models, *LambdaMART* is more robust than *RankSVM* and *Prank*. The reason is similar to what we have analyzed in Section 5.4.1. (4) It is surprising to find that *DSSM* and *Duet* are the most robust ranking models against the *word-level* query attack. In addition, the DR<sub>query</sub> of *Conv-KNRM* is also relatively high (e.g., -11.0%). The three ranking models all apply the character-level n-gram operations. The results indicates that ranking models which contains a character-level operation will be more robust than those ranking models which treat each word as the basic semantic units. (5) When we take a look at pre-trained models (i.e., BERT and ColBERT), we could find they are not robust compared with other neural ranking models. For example, the DR<sub>query</sub> of *ColBERT* is -33.9%, which is the highest among all the neural ranking models. One possible explanation could

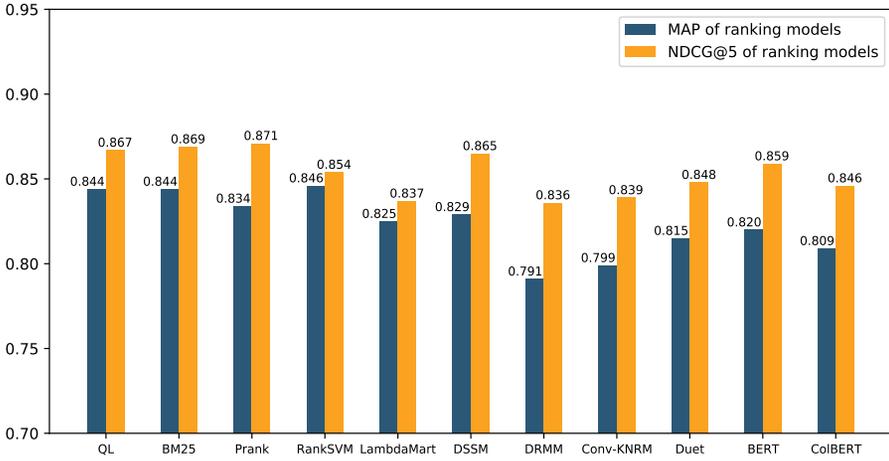


Fig. 6. The ranking effectiveness of ranking models on the ASRC dataset in terms of MAP and NDCG@5.

be that WordPiece [90] tokenization decompose an attacked word into more attacked sub-words, which have a more significant impact on the performance.

## 5.5 Ranking Robustness under Document Manipulation

In this section, we analyze the empirical results on the defensive ability against document manipulation. We first measure the average effectiveness of different ranking models and then show the robustness evaluation results of different models in terms of TC and KT.

**5.5.1 Analysis on the effectiveness of ranking models.** We first analyze the average effectiveness of ranking models on the ASRC dataset. Specifically, we use the mean average precision (MAP) [79] and normalized discounted cumulative gain at rank 5 (NDCG@5) [6] to evaluate ranking effectiveness [27]. The results are shown in Figure 6.

From the results we observe that: (1) All the ranking models can generally achieve good effectiveness on the ASRC dataset. The high MAP and NDCG values can be attributed to the fact that most documents generated by the students were judged to be relevant (e.g., 1113 out of 1279 documents are relevant) [27]. (2) Overall, traditional probabilistic ranking models perform better than neural ranking models and LTR models. (3) Among all the neural ranking models, *DSSM* performs the best. (4) *BERT* performs second only to *DSSM* in neural ranking models. Specifically, we train the ranking models on the ClueWeb09-B dataset and then evaluate them on the ASRC dataset, i.e., under the OOD setting. As noted in Section 4, we have found that *BERT* shows greater robustness than other neural ranking models in terms of the OOD generalizability, which is quite consistent with the result on the ASRC. (5) Overall, the relative average effectiveness order in terms of MAP is  $DRMM < Conv-KNRM < ColBERT < Duet < BERT < LambdaMART < DSSM < Prank < QL = BM25 < RankSVM$ .

**5.5.2 Analysis on the robustness of ranking models.** Here, we analyze the robustness of ranking models against document manipulation on the ASRC dataset. The results are shown in

Figure 7. Recall that high values of KT and TC attest to decreased ranking robustness, as these are measures of inter-list distance.

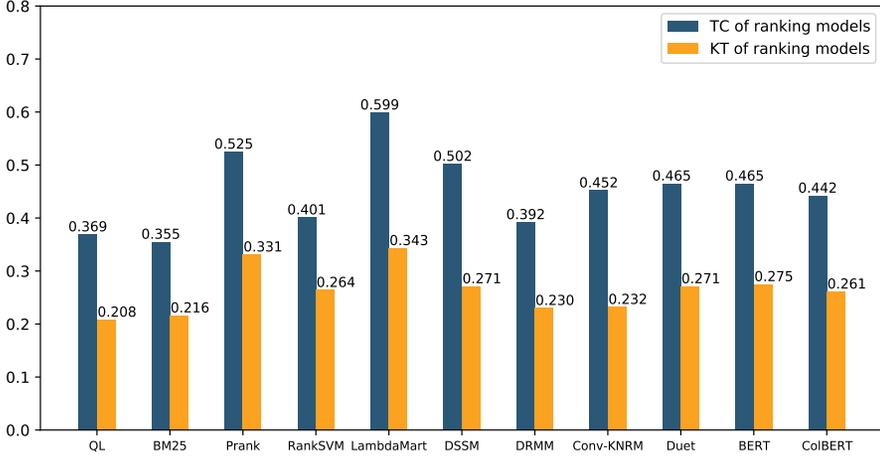


Fig. 7. The ranking robustness of ranking models on the ASRC dataset against document manipulation in terms of TC and KT.

From the results we observe that: (1) For the traditional probabilistic ranking models, *QL* and *BM25* achieve the lowest TC and KT performance among all the ranking models, indicating that traditional probabilistic ranking models are the most robust to defend against document manipulations. In the generation process of the ASRC dataset, students were asked to modify relevant documents so as to have them ranked as high as possible in the next round. This suggests that traditional probabilistic ranking models have the lowest sensitivity to the difference between relevant documents. (2) For the LTR models, the ranked lists induced by *RankSVM* and *Prank* are more robust than those induced by *LambdaMART*. Since *RankSVM* and *Prank* is linear and *LambdaMART* is not, the variance of *RankSVM* and *Prank* is in general lower, and we saw that its ranking robustness is higher. Meanwhile, the ranked lists induced by *Prank* is less robust than those induced by *RankSVM*, indicating that pointwise learning objective may suffer more from the document manipulation compared with pairwise learning objective. (3) For the neural ranking models, they achieve comparable results with LTR models against document manipulation. The reason might be that, flipping original words in a document to new words for higher rank, both the statistical features and word embeddings change significantly, resulting in the performance drop. (4) Overall, the relative robustness order of defending against document manipulation (e.g., in terms of TC) is  $LambdaMART < Prank < DSSM < BERT = Duet < Conv-KNRM < ColBERT < RankSVM < DRMM < QL < BM25$ .

## 6 RELATED WORK

In this section, we briefly review the works related to our study, including the performance variance over different queries, the OOD generalizability and the adversarial operation.

## 6.1 Performance Variance

The performance variance over different queries has been studied as an early exploration of robustness in IR [11, 85–87, 99, 100, 104]. The key idea is to pay more attention to the difference on ranking performance, rather than the average effectiveness performance which is considered by most ranking models.

Early works [85–87] mainly focused on the poorly performing queries of ranking systems. They evaluated the ranking robustness by proposing new metrics which emphasized the poorly performing queries. Beyond the widely-used average performance metrics such as the mean average precision (MAP) and the average of precision at rank 10 (P@10), Voorhees [85] proposed the percentage of queries with no relevant in the top 10 retrieved (%no) and the area underneath the MAP(X) vs. X curve to test the robustness of a ranking model. Besides, Voorhees [86] proposed the gMAP, which is a variant of the traditional MAP measure that used a geometric mean rather than an arithmetic mean to average individual query results, to evaluate the ranking robustness. The proposed gMAP showed promise of giving appropriate emphasis to poorly performing queries while being more stable at equal query set sizes. In addition, Voorhees [87] used the document from AQUAINT Corpus of English News Text to test poorly performing queries and summarized the results of the three-year run of the Robust track.

Besides, some works [11, 104] proposed the evaluation metrics to analyze the the performance variance over queries. For example, Collins-Thompson [11] proposed the *R-Loss* and *Robustness Index* to measure the robustness on the query expansion. *R-Loss* computed the averaged net loss of relevant documents in the retrieved document, due to the failure on query expansion. And *Robustness Index* was defined as the difference between the number of improved queries and the number of hurt queries. The improved/hurt queries refer to the queries whose performance is improved/hurt over the original model. Besides, Zighelnic and Kurland [104] proposed *< Init* to calculate the percentage of queries for which the retrieval performance of a query model is worse than that of the original query model.

However, such robustness metrics were often defined separately from the effectiveness metric (e.g., MAP). There was a lack of a unified formulation to allow them to be evaluated in an integrated manner. Later, a series of methods were proposed based from the perspective of bias-variance decomposition in IR to solve the problem. Zhang et al. [99, 100] analyzed the IR robustness from the perspective of bias-variance decomposition of IR evaluation. They proposed the variance of AP to analyze the robustness of ranking models in a unified framework with ranking effectiveness. Cormack and Grossman [13] quantified bias and variance of system rankings by treating rankings as embeddings in a Euclidean space. They then showed that shallow pooling has substantially higher bias and insubstantially lower variance than probability-proportional-to-size sampling.

Recently, Shivaswamy and Chandrashekar [81] defined notions of bias and variance directly on pairwise ordering of items. They showed that ranking disagreements between true orderings and a ranking function could be decomposed into bias and variance components. This decomposition is similar to the squared loss and other losses that have been previously studied.

## 6.2 OOD Generalizability

The OOD generalizability refers to the model's ability to generalize to various new test sets. We first briefly review the works which study the OOD generalizability in Computer Vision (CV) and Natural language processing (NLP). Then we review the recent works in the IR field.

Extensive prior works [32–34, 48] have studied the OOD generalizability on CV and NLP. These works aimed to fairly measure the neural models' OOD generalizability in benchmark dataset of image/text classification. Hendrycks et al. [33] systematically studied seven robustness hypotheses

on image classification and proposed three new robustness benchmarks to analyze them. The authors showed that using large models and synthetic data augmentation could improve robustness on real-world distribution shifts. Hendrycks et al. [34] systematically measured OOD generalization for seven NLP datasets by constructing a new robustness benchmark with realistic distribution shifts. By measuring OOD generalization and OOD detection of different neural models, they showed that pre-trained transformers are more effective at detecting OOD examples. Koh et al. [48] focused on the real world distribution shifts and summarized them into two types: domain generalization and subpopulation shift. The authors then proposed eight benchmark datasets in the wild to evaluate these distribution shifts.

Recently, since deep learning methods became more prevalent to solve the ranking problems, many works began to study the problem of OOD generalizability in IR. Such works mainly focused on the cross domain adaption for neural ranking models. Cohen et al. [10] proposed cross domain regularization on the ranking models to improve their performance on unforeseen domains. They adopted the adversarial learning by using an adversarial discriminator and the gradient reversal layers [22] to train their neural ranking model on a small set of domains. The effectiveness of their proposed architecture was evaluated on cross domain question answering data. Long et al. [55] also leveraged the adversarial learning framework by devising a domain discriminator to solve the problem of domain adaptation. They evaluated the effectiveness of their proposed method on three domain transfer tasks, including cross-domain digits retrieval, image to image and image to videos transfers, on several benchmarks. Mao et al. [58] utilized the multi-layer joint kernelized mean distance to measure the distance between the target data and the source data. They measured the distance based on deep features extracted from the deep networks. The target data which was found by their method was then iteratively added to the training data. In ad-hoc retrieval, Yilmaz et al. [96] aggregated sentence-level evidence to fine-tune BERT models. The methods were proposed to solve the challenges of exceeding document length and document-level relevance judgments for BERT. By this way, the fine-tuned BERT models could capture cross-domain notions of relevance and can be directly used for ranking documents from other domains.

### 6.3 Adversarial Operation

Deep neural networks have been found vulnerable to adversarial operations [25, 82]. Specifically, the key idea of adversarial operation is to find a minimal perturbation in the data that can maximize the risk of making wrong predictions.

Early works have extensively studied the adversarial operation in CV [25, 57, 61, 66]. Adversarial operation can be classified as white-box adversarial attacks and black-box adversarial attacks, which means that adversaries have full access to target models (e.g., model's architecture and parameters) or no knowledge about target models, respectively. Fast gradient sign method (FGSM) [25] and its variants (e.g., FGM [61] and PGD [57]) were classical white-box gradient-based attack methods. These methods added the noise to the whole image based on the gradient of loss. On the contrary, another popular approach Jacobian-based saliency map (JSMA) [12, 66, 89] only perturbed one pixel at a time. It chose the pixel with the highest saliency score, which was defined as the gradient of the target class multiplied by the sum of the gradients of other classes. As a result, JSMA produced adversarial examples which increased the probability of the target class while decreased others. While the above works focused on the gradient information of models, Naseer et al. [62] utilized the transferability of adversarial examples to train substitute models, in order to conduct a black-box attack. Another representative black-box attack was score-based method [9, 51], which utilized the output score of model to conduct attack.

Adversarial operations have also been conducted to the NLP field, including text classification [20, 23, 52, 78], sentiment analysis [4, 50, 52, 78] and natural language inference [4, 59]. Different

from CV where the image is represented as continuous data, the main problem of generating adversarial text in NLP is the discrete input space. To address the problem, Gong et al. [24] proposed to attack in the embedding space. They used FGSM to produce perturbations in word embedding and used nearest neighbor search to find the closest words. However, this approach treated all tokens as equally vulnerable and replace all tokens with their nearest neighbors, which led to non-sensical, word-salad outputs [92]. A number of works [38, 39, 52, 77] utilized the ideas of JSMA (e.g., find important pixel) to solve this problem. For example, Jin et al. [39] first found the most important words by sorting all the original words in terms of their importance score. The importance score was calculated as the prediction change before and after deleting the word. Then the word was replaced by synonym which has the most similar semantic meaning with the original word. The candidate synonyms were selected to force the target model to make wrong predictions. This method was more efficient since it attacked the most importance words. Meanwhile, it was classified as a black-box method since it only required the prediction score of the model.

In IR, early works on adversarial information retrieval mainly focused on identifying and addressing different types of spam [8]. Recently, Raifer et al. [74] studied the ranking competition on the web retrieval, which means that the document authors manipulate their documents intentionally to have them highly ranked. They constructed the ASRC dataset to simulate the ranking competition and used game theory to analyze the strategies employed by document authors. Following the above work, Goren et al. [27] defined the ranking robustness and analyzed the robustness of ranking functions based on the ASRC dataset. For the work related to the query attack, query spelling correction has been a lively research topic since the mid 2000's, especially in the NLP community [2, 16]. The Microsoft Speller Challenge held in the year 2011 [88] also attracted much attention for the problem of query spelling correction. Hagen [31] presented a new large-scale collection of 54,772 queries with manually annotated spelling corrections. A recent work [103] proposed to train a more robust Dense Retriever and BERT re-ranker which are robust to typos in queries. Overall, the research area on the adversarial operation in IR is largely unexplored up till now, and more efforts are expected in this direction in the short future.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we systematically analyzed the robustness of several representative neural ranking models against traditional probabilistic ranking models and LTR models. Specifically, we proposed three ways to define the robustness, i.e., the performance variance under I.I.D. settings, the OOD generalizability and the defensive ability against adversarial operations. The latter two definitions were further specialized into two different perspectives respectively. Overall, the results showed that neural ranking models are in general not robust as compared with other IR models in terms of 3 out of 5 robustness tasks. The exception is that the pre-trained ranking models achieve the best robustness from the perspective of the performance variance, while DSSM, Duet and Conv-KNRM model show robustness in terms of the defensive ability against query attacks. More research efforts are needed to develop robust neural ranking models for IR. Future research efforts could explore novel pre-training objectives and model architectures with character-level operations that enhance the robustness of ranking models. The analysis of the ranking robustness is a foundation for designing ideal ranking models in real world applications. We believe the way we study the robustness (definition and metric) as well as our findings would be beneficial to the IR community.

In the future work, we would like to apply the findings to improve the robustness of existing ranking models. We would also try to design new robust neural ranking models based on the findings in this work. Moreover, it is valuable to define a unified formulation based on the different perspectives of the robustness to analyze the ranking models comprehensively.

## REFERENCES

- [1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398* (2019).
- [2] Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 955–962.
- [3] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access* 6 (2018), 14410–14430.
- [4] Moustafa Alzantot, Yash Sharma Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating Natural Language Adversarial Examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
- [5] Andrei Broder. 2002. A taxonomy of web search. In *ACM Sigir forum*, Vol. 36. ACM New York, NY, USA, 3–10.
- [6] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.
- [7] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.
- [8] Carlos Castillo and Brian D Davison. 2011. Adversarial Web Search. *Foundations and Trends in Information Retrieval* 4, 5 (2011), 377–486.
- [9] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 15–26.
- [10] Daniel Cohen, Bhaskar Mitra, Katja Hofmann, and W Bruce Croft. 2018. Cross domain regularization for neural ranking models using adversarial learning. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1025–1028.
- [11] Kevyn Collins-Thompson. 2009. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 837–846.
- [12] Théo Combey, António Loison, Maxime Faucher, and Hatem Hajri. 2020. Probabilistic Jacobian-based Saliency Maps Attacks. *Machine Learning and Knowledge Extraction* 2, 4 (2020), 558–578.
- [13] Gordon V Cormack and Maura R Grossman. 2019. Quantifying Bias and Variance of System Rankings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1089–1092.
- [14] Koby Crammer, Yoram Singer, et al. 2001. Pranking with ranking. In *Nips*, Vol. 1. 641–647.
- [15] W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*. Vol. 520. Addison-Wesley Reading.
- [16] Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. 293–300.
- [17] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 985–988.
- [18] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 126–134.
- [19] Matt Davis. 2003. Psycholinguistic evidence on scrambled letters in reading.
- [20] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 31–36.
- [21] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 375–384.
- [22] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.
- [23] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 50–56.
- [24] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. 2018. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175* (2018).
- [25] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

- [26] Gregory Goren. 2019. Ranking Robustness In Adversarial Retrieval Settings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1446–1446.
- [27] Gregory Goren, Oren Kurland, Moshe Tenenholz, and Fiana Raiber. 2018. Ranking robustness under adversarial document manipulations. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 395–404.
- [28] Jia-Chen Gu, Tianda Li, Quan Liu, Zhen-Hua Ling, Zhiming Su, Si Wei, and Xiaodan Zhu. 2020. Speaker-aware bert for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2041–2044.
- [29] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 55–64.
- [30] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. MatchZoo: A Learning, Practicing, and Developing System for Neural Text Matching. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. ACM, New York, NY, USA, 1297–1300. <https://doi.org/10.1145/3331184.3331403>
- [31] Matthias Hagen, Martin Potthast, Marcel Gohsen, Anja Rathgeber, and Benno Stein. 2017. A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1261–1264.
- [32] Dan Hendrycks, Steven Basart, Mantas Mazeika, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. 2019. Scaling Out-of-Distribution Detection for Real-World Settings. *arXiv preprint arXiv:1911.11132* (2019).
- [33] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. 2021. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 8340–8349.
- [34] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained Transformers Improve Out-of-Distribution Robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2744–2751.
- [35] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. In *ECAI 2020*. IOS Press, 513–520.
- [36] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.
- [37] Samuel Huston and W Bruce Croft. 2014. Parameters learned in the comparison of retrieval models using term dependencies. *Ir, University of Massachusetts* (2014).
- [38] Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2021–2031.
- [39] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 8018–8025.
- [40] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 133–142.
- [41] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 217–226.
- [42] Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust Encodings: A Framework for Combating Adversarial Typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2752–2765.
- [43] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics* 8 (2020), 64–77.
- [44] Masahiro Kaneko and Mamoru Komachi. 2019. Multi-head multi-layer attention to deep language representations for grammatical error detection. *Computación y Sistemas* 23, 3 (2019), 883–891.
- [45] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.
- [46] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48.
- [47] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [48] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution

- shifts. In *International Conference on Machine Learning*. PMLR, 5637–5664.
- [49] Robert Krovetz. 2000. Viewing morphology as an inference process. *Artificial intelligence* 118, 1-2 (2000), 277–294.
- [50] J Li, S Ji, T Du, B Li, and T Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *26th Annual Network and Distributed System Security Symposium*.
- [51] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. 2019. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning*. PMLR, 3866–3876.
- [52] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 4208–4215.
- [53] Jimmy Lin. 2019. The neural hype and comparisons against weak baselines. In *ACM SIGIR Forum*, Vol. 52. ACM New York, NY, USA, 40–51.
- [54] Tie-Yan Liu. 2011. Learning to rank for information retrieval. (2011).
- [55] Fuchen Long, Ting Yao, Qi Dai, Xinmei Tian, Jiebo Luo, and Tao Mei. 2018. Deep domain adaptation hashing with adversarial learning. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 725–734.
- [56] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. *B-PROP: Bootstrapped Pre-Training with Representative Words Prediction for Ad-Hoc Retrieval*. Association for Computing Machinery, New York, NY, USA, 1513–1522. <https://doi.org/10.1145/3404835.3462869>
- [57] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.
- [58] Sitong Mao, Xiao Shen, and Fu-lai Chung. 2018. Deep domain adaptation based on multi-layer joint kernelized distance. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1049–1052.
- [59] Pasquale Minervini and Sebastian Riedel. 2018. Adversarially Regularising Neural NLI Models to Integrate Logical Background Knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. 65–74.
- [60] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*. 1291–1299.
- [61] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* (2016).
- [62] Muzammal Naseer, Salman H Khan, Shafin Rahman, and Fatih Porikli. 2018. Task-generalizable Adversarial Attack based on Perceptual Metric. *arXiv preprint arXiv:1811.09020* (2018).
- [63] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
- [64] David L Olson and Dursun Delen. 2008. *Advanced data mining techniques*. Springer Science & Business Media.
- [65] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 257–266.
- [66] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.
- [67] Gustavo Penha, Arthur Câmara, and Claudia Hauff. 2021. Evaluating the Robustness of Retrieval Pipelines with Query Variation Generators. *arXiv preprint arXiv:2111.13057* (2021).
- [68] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
- [69] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. <https://doi.org/10.18653/v1/N18-1202>
- [70] Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating Adversarial Misspellings with Robust Word Recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5582–5591.
- [71] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).
- [72] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Are Neural Rankers still Outperformed by Gradient Boosted Decision Trees?. In *International Conference on Learning Representations*.
- [73] Dragomir R Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating Web-based Question Answering Systems.. In *LREC*. Citeseer.
- [74] Nimrod Raifer, Fiana Raiber, Moshe Tennenholtz, and Oren Kurland. 2017. Information retrieval meets game theory: The ranking competition between documents’ authors. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 465–474.

- [75] Graham Rawlinson. 2007. The significance of letter position in word recognition. *IEEE Aerospace and Electronic Systems Magazine* 22, 1 (2007), 26–27.
- [76] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*. Springer, 232–241.
- [77] Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812* (2017).
- [78] Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Interpretable Adversarial Perturbation in Input Embedding Space for Text. In *IJCAI*.
- [79] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.
- [80] Muhammad Shafique, Mahum Naseer, Theocharis Theocharides, Christos Kyrkou, Onur Mutlu, Lois Orosa, and Jungwook Choi. 2020. Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead. *IEEE Design & Test* 37, 2 (2020), 30–57.
- [81] Pannaga Shivaswamy and Ashok Chandrashekar. 2021. Bias-Variance Decomposition for Ranking. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 472–480.
- [82] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna Estrach, Dumitru Erhan, Ian Goodfellow, and Robert Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*.
- [83] Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.
- [84] Shailesh Tripathi, David Muhr, Manuel Brunner, Herbert Jodlbauer, Matthias Dehmer, and Frank Emmert-Streib. 2021. Ensuring the Robustness and Reliability of Data-Driven Knowledge Discovery Models in Production and Manufacturing. *Frontiers in Artificial Intelligence* 4 (2021), 22.
- [85] Ellen M. Voorhees. 2003. Overview of the TREC 2003 Robust Retrieval Track. In *Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*.
- [86] Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*.
- [87] Ellen M. Voorhees. 2006. Overview of the TREC 2005 Robust Retrieval Track. *TREC 2005 Proceedings* (2006).
- [88] Kuansan Wang and Jan Pedersen. 2011. Review of MSR-Bing web scale speller challenge. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 1339–1340.
- [89] Rey Wiyatno and Anqi Xu. 2018. Maximal jacobian-based saliency map attack. *arXiv preprint arXiv:1808.07945* (2018).
- [90] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).
- [91] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*. 55–64.
- [92] Ying Xu, Xu Zhong, Antonio Jimeno Yepes, and Jey Han Lau. 2021. Grey-box Adversarial Attack And Defence For Sentiment Classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4078–4087.
- [93] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality (JDIQ)* 10, 4 (2018), 1–20.
- [94] Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. Data augmentation for bert fine-tuning in open-domain question answering. *arXiv preprint arXiv:1904.06652* (2019).
- [95] Andrew Yates, Siddhant Arora, Xinyu Zhang, Wei Yang, Kevin Martin Jose, and Jimmy Lin. 2020. Capreolus: A toolkit for end-to-end neural ad hoc retrieval. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 861–864.
- [96] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3481–3487.
- [97] Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 268–276.
- [98] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020).
- [99] Peng Zhang, Dawei Song, Jun Wang, and Yuexian Hou. 2013. Bias-variance decomposition of ir evaluation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 1021–1024.

- [100] Peng Zhang, Dawei Song, Jun Wang, and Yuexian Hou. 2014. Bias–variance analysis in estimating true query model for information retrieval. *Information processing & management* 50, 1 (2014), 199–217.
- [101] Mo Zhou, Le Wang, Zhenxing Niu, Qilin Zhang, Yinghui Xu, Nanning Zheng, and Gang Hua. 2021. Practical Relative Order Attack in Deep Ranking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16413–16422.
- [102] Yun Zhou and W Bruce Croft. 2006. Ranking robustness: a novel framework to predict query performance. In *Proceedings of the 15th ACM international conference on Information and knowledge management*. 567–574.
- [103] Shengyao Zhuang and Guido Zuccon. 2021. Dealing with Typos for BERT-based Passage Retrieval and Ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2836–2842.
- [104] Liron Zigelnic and Oren Kurland. 2008. Query-drift prevention for robust query expansion. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 825–826.