DUQGen: Effective Unsupervised Domain Adaptation of Neural Rankers by Diversifying Synthetic Query Generation

Ramraj Chandradevan, Kaustubh D. Dhole, Eugene Agichtein
Department of Computer Science
Emory University
Atlanta, USA-30307
`{rchan31,kdhole,yagicht}@emory.edu`

Abstract

State-of-the-art neural rankers pre-trained on large task-specific training data such as MS-MARCO, have been shown to exhibit strong performance on various ranking tasks without domain adaptation, also called zero-shot. However, zero-shot neural ranking may be sub-optimal, as it does not take advantage of the target domain information. Unfortunately, acquiring sufficiently large and high quality target training data to improve a modern neural ranker can be costly and time-consuming. To address this problem, we propose a new approach to unsupervised domain adaptation for ranking, **DUQGen**, which addresses a critical gap in prior literature, namely how to automatically generate both effective and diverse synthetic training data to fine tune a modern neural ranker for a new domain. Specifically, **DUQGen** produces a more effective representation of the target domain by identifying clusters of similar documents; and generates a more diverse training dataset by probabilistic sampling over the resulting document clusters. Our extensive experiments, over the standard BEIR collection, demonstrate that **DUQGen** consistently outperforms all zero-shot baselines and substantially outperforms the SOTA baselines on 16 out of 18 datasets, for an average of 4% relative improvement across all datasets. We complement our results with a thorough analysis for more in-depth understanding of the proposed method's performance and to identify promising areas for further improvements.

1 Introduction

Large Language Models (LLMs) have enabled new state-of-the-art performance in neural ranking (Yan et al., 2019; Kamps et al., 2020; Hu et al., 2022; Nogueira et al., 2020a). An effective approach has been to train the LLMs on a large-scale general ranking task such as MS-MARCO passage or document ranking (Bajaj et al., 2016) or Wikipedia retrieval (Sun and Duh, 2020), to learn task-specific features, which are often shared across other domains and datasets. The resulting rankers can then be used, without any adaptation (or in a zero-shot way) for a wide range of ranking tasks. For example, the BEIR (Thakur et al., 2021) benchmark had demonstrated SOTA or near-SOTA performance of several zero-shot neural rankers on a diverse set of retrieval tasks.

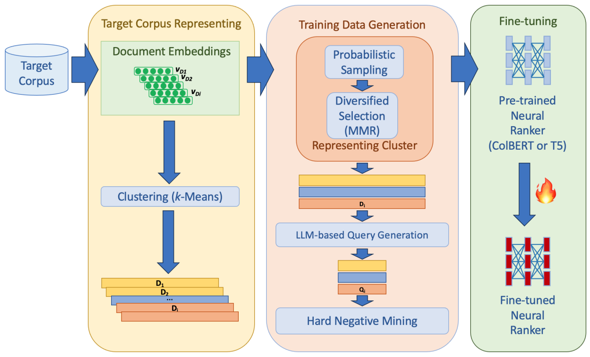


Figure 1: **DUQGen**: an unsupervised domain-adaptation framework for neural ranking.

However, when switching to specialized domains such as finance or scientific documents, zero-shot ranking performance should benefit from additional information for the target domain. For training modern neural rankers, acquiring sufficiently large and high-quality target training data of query-document pairs, to improve a modern neural ranker, can be costly and time-consuming. Hence, there has been significant interest in various approaches to domain adaptation for neural rankers, with varying degrees of supervision, including unsupervised approaches using synthetically generated queries, documents, or both query-document pairs (Sachan et al., 2022; Bonifacio et al., 2022; Jeronymo et al., 2023; Dai et al., 2022; Askari et al., 2023).

Unfortunately, most of the previously reported results did not exceed the ranking performance compared to the current SOTA zero-shot models, as evaluated on the BEIR benchmark (Thakur et al., 2021). In other words, to the best of our knowledge, *no previously reported unsupervised ranking adaptation method demonstrated consistent improvements over large neural SOTA zero-shot rankers.*

In this work, we investigate whether it possible to improve the ranking performance of a pre-trained SOTA neural ranker for a given target domain, through unsupervised domain adaptation (UDA). To address this question, we first identify a critical requirement for the synthetic training data to be effective for ranking adaptation: that the generated training data should be both *representative of the target domain*, and *sufficiently diverse* to force changes to the ranking model at the appropriate representation level, without causing over-fitting or catastrophic forgetting (i.e., degrading performance on the original ranking tasks).

Specifically, we propose a new method **DUQGen**, which stands for **D**iversified **U**nsupervised **Q**uery **Gen**eration. **DUQGen**introduces a general approach for ranking domain adaptation, which focuses on selecting representative and diverse set of documents and query pairs for training a neural ranker. **DUQGen** only requires access to a (part of) target document collection to be searched, and can improve any pre-trained neural ranker. **DUQGen** is illustrated in Figure 1, and introduces the following innovations compared to previous unsupervised ranking adaptation approaches: (1) representing the target document collection as document *clusters*; (2) diversifying the synthetic query generation by probabilistic sampling over the resulting document clusters; and (3) prompting a large LLM for query generation with in-context examples to generate queries from the selected documents. As we show experimentally, these innovations are responsible for consistent improvements over the previous SOTA baselines for ranking adaptation on almost all BEIR benchmarks, as well as consistent improvements over zero-shot performance of SOTA neural rankers. In summary, our contributions include:

1. **DUQGen**, a general and effective unsupervised approach for domain adaption of neural rankers via synthetic query generation for training.

2. A novel and general method for creating representative and diverse synthetic query data for a given collection via clustering and probabilistic sampling.

3. Comprehensive experiments demonstrating that **DUQGen** consistently outperforms all SOTA baselines on 16 out of 18 BEIR datasets, and thorough analysis of the components of **DUQGen** responsible for the improvements. We release all our code and models publicly[1].

Next, we describe prior work on domain adaptation of neural rankers in more detail, to place our contributions in context.

2 Related Work

In this section, we discuss the prior works that help to establish the problem and navigate the solution.

2.1 Neural Rankers

Recently, transformer-based pre-trained language models have demonstrated impressive effectiveness in neural rankers (Lin et al., 2021b). A neural ranker returns an order list of documents given a query, where a relevancy score between query and document dense embedding representations is used for sorting. Extensive studies have been conducted on both dense retrievers and re-rankers (Mitra and Craswell, 2018). In comparison to encoder-based rankers (MonoBERT and DuoBERT) (Nogueira et al., 2019), encoder-decoder (Nogueira et al., 2020b) and decoder-based (Ma et al., 2023) rankers exhibit notably superior performance with a larger margin. While ColBERT (Khattab and Zaharia, 2020), Contriever (Izacard et al., 2021), and GTR (Ni et al., 2022) perform competitively as dense retrievers, MonoT5-3B (Nogueira et al., 2020b) is widely adapted for re-ranking purposes.

2.2 Unsupervised Domain Adaptation for Neural Rankers

Despite the remarkable ranking performances demonstrated by recent pre-trained language models in zero-shot settings, they often encounter catastrophic failures in real-world deployment scenarios. The main factor contributing to these failures is *Domain-Shift* (Zhu and Hauff, 2022) or *Domain Divergence* (Ramesh Kashyap et al., 2021). Domain-Shift has been a subject of exploration for decades, including recent investigations in domain adaptation (Lupart et al., 2023). Traditionally, it is assumed that the source and target domains share samples drawn from the same distribution. Previous studies have addressed this

[1] https://github.com/emory-irlab/DUQGen

issue by quantifying domain divergence through various measures, such as geometric measures, information-theoretic measures, and higher-order measures (Ramesh Kashyap et al., 2021). Ultimately, these measures contribute to the development of novel solutions for domain adaptation in neural rankers.

Solutions addressing domain divergence typically fall into two categories: (1) representation learning; and (2) data selection. Representation learning approaches primarily address UDA, with a focus on learning domain-invariant representations (Bousmalis et al., 2016; Cohen et al., 2018) or pre-training a zero-shot ranker. On the other hand, data selection assumes that not all samples contribute equally to domain representation (Axelrod et al., 2011), highlighting the importance of identifying effective target domain samples. Improper selection of target data during fine-tuning has the potential to undermine the impact of source pre-training. Our research centers on the issue of improper representation of the target domain leading to diminished performances in neural rankers. Therefore, **DUQGen** aims to identify representative and diverse target samples that can be effective during the fine-tuning process.

2.3 Synthetic IR Data Generation

The increasing power of Large Language models have prompted numerous studies to focus on utilizing LLMs for the creation of high-quality training data. Several previous works have explored unsupervised synthetic data generation for fine-tuning ranking models, including GPL (Wang et al., 2022), InPars (Bonifacio et al., 2022), InPars-v2 (Jeronymo et al., 2023), DocGen-RL (Askari et al., 2023), GenQ (Thakur et al., 2021), and Promptagator (Dai et al., 2022). These frameworks use random document sampling or random seed queries to start their pipelines, which leaves room for improvement.

Each of the previously mentioned works utilizes distinct strategies employed alongside their data synthesis processes. GPL(Wang et al., 2022), for instance, combines a T5-based (Raffel et al., 2020) query generator with a pseudo-labeling cross-encoder to enhance robust learning. InPars and InPars-v2 methods utilize GPT-3 and GPT-J query generators along with different filtering strategies to eliminate low-quality synthetic queries. DocGen-RL introduces an RL-driven guided approach combined with document synthesis using BLOOM (Scao et al., 2022). GenQ, on the other hand, fine-tunes TAS-B (Hofstätter et al., 2021) with queries generated from an MS-MARCO fine-tuned T5-base generator. Promptagator employs a pipeline similar to InPars, but with improved components, such as a random million document samples, a 137B FLAN query generator, and a strong consistency filter to prune 8 million synthetic queries through a relatively complicated and expensive process. Notably, none of the methods mentioned above take into consideration the significance of identifying domain-representative documents or diversifying the resulting queries. Consequently, the fine-tuned performances appears to fall short of zero-shot performances in many cases.

The quality of the generated training queries significantly affects the end retrieval performances. Despite the utilization of strong query generators (BLOOM and GPT-3), the domain query representation can still be improved. For instance, InPars employed a prompt containing in-context examples from MS-MARCO training data, yet it still maintains a domain representation gap during in-context generation. Furthermore, their query generation did not address the need for diversity among the generated training samples. Additionally, they incorporated a complex filtering step to prune the generated queries, which we show can be avoided. These methods fine-tuned rankers using large-scale synthetic data, ranging from 100k to 1M examples. In contrast, we argue that judicious selection of training samples can obviate the necessity for such large-scale generation, reducing the required amount of synthetic training data by a factor of x1000.

3 Methodology

DUQGen, shown in Figure 1, consists of four components – domain document selection, domain query generation, negative pairs mining, and fine-tuning. We now cover each component in detail.

3.1 Domain Document Selection

We propose to represent a target domain with clusters and each clusters with its sampled documents. Therefore, in this section we describe them in three stages, namely collection document clustering, probabilistic document sampling, and diversified document selection.

3.1.1 Collection Document Clustering

Representing a large-scale target collection of documents with limited training data is challenging. Therefore, we propose to divide the collection into portions, and then sample documents within each portion. We use a clustering approach for the collection representation. Moreover, we can achieve diverse topical documents to represent the domain. We start with the full collection of documents and apply a preprocessing step, where we discard short span documents, filtering out noisy documents. Then we use a SOTA text encoder, viz. *Contriever* (Izacard et al., 2021), to encode each of the documents. Using the document embeddings v_{D_i}, we apply clustering (*e.g.,* K-Means) technique, where K is a hyper-parameter to tune.

3.1.2 Probabilistic Document Sampling

Representing each cluster within large data collections is challenging since the resultant clusters can often be of imbalanced sizes. Let's take k^{th} cluster size as c_k and collection size as C, where $(1 \leq c_k \leq C)$. We ideally want to sample more number of documents from larger size clusters in proportion to the cluster size. If $cluster_k$ and D_i represent k^{th} cluster and its i^{th} document, the probability of selecting D_i from $cluster_k$ is $Pr(D_i|cluster_k) \propto c_k \forall D_i \in cluster_k$.

We intend to sample N number of synthetic training examples from K number of clusters, where $N \geq K$. Therefore, we design a stratified expression to determine the document sample size N_k for k^{th} cluster, given by

$$N_k^0 = 1 + \left\lfloor \frac{c_k}{C}(N-K) \right\rfloor$$

$$P = N - \sum_{k=1}^{K} N_k^0$$

$$N_k = \begin{cases} N_k^0 + 1 & \text{if } k \in \text{argsort}_{top-P}(c_k) \\ N_k^0 & \text{if } k \notin \text{argsort}_{top-P}(c_k) \end{cases}$$

where N_k^0 and P are intermediate sample size and integer number. $\lfloor * \rfloor$ operation finds the floor integer value.

Now that we determined the sample size for each clusters, we define our sampling approach. Let's take d_i as the similarity (*e.g. cosine similarity*) between document D_i and its corresponding cluster centroid. We define an exponential value e^{d_i} as the representative of how close D_i is to its cluster centroid. Therefore, $Pr(D_i|cluster_k)$ becomes the normalized softmax given by:

$$Pr(D_i|cluster_k) = \frac{e^{d_i/T}}{\sum_{j=1}^{c_k} e^{d_j/T}} \quad (1)$$

$$d_i = cosine(v_{D_i}, \frac{1}{c_k}\sum_{j=1}^{c_k} v_{D_j}) \quad (2)$$

where T is the softmax temperature and v_{D_i} is the i^{th} document embedding. Intuitively, a document likelihood to be selected to generate an associated query is proportional to the document similarity to its cluster centroid.

3.1.3 Diversified Document Selection

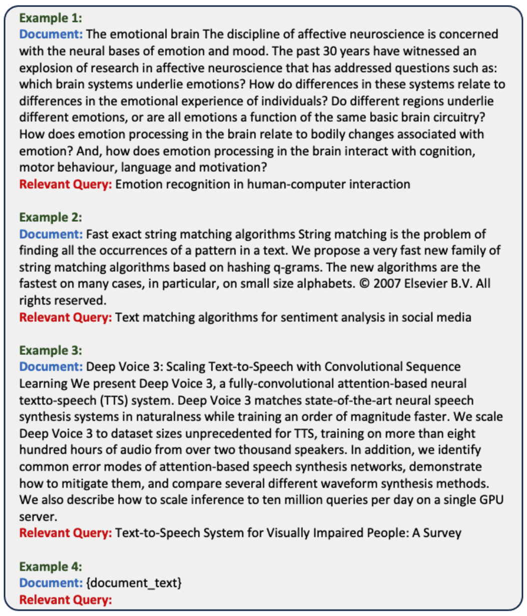
Now we sample N_k number of documents from each cluster $cluster_k$ and pool them to obtain the required training size documents N. Different sample sets can be drawn from the aforementioned sampling approach with different choices of random seed values. Therefore, to improve selection robustness in the sampling process, we apply a diversity measure, namely Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998). We first iterate the sampling process m times ($m = 5$) to obtain different sample sets. Then we apply MMR on the pooled documents from m sets to select top-N_k documents for $cluster_k$ as shown:

$$\begin{aligned} &\operatorname*{argmax}_{D_i \in R \setminus S} \Big[\lambda Sim_1(D_i, D_k) \\ &-(1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \Big] \end{aligned} \quad (3)$$

where D_k is the document closest to the cluster centroid, λ is a trade-off weight (to be tuned) between similarity to cluster centroid and diversity, R is the pooled documents, S is a subset of documents already selected from R, and Sim_1 and Sim_2 that can be same or different, but we used the $cosine$ similarity for both instances.

3.2 Synthetic Query Generation

Query generation is an essential component in an unsupervised data generation pipeline for ranking models. Queries represent a target domain *w.r.t.* the user's information need and the domain-task by taking different types, such as questions, headlines, keywords, or claims. Therefore, we use a LLM to generate a synthetic in-domain query for each sampled document. We few-shot prompt the LLM to generate such training queries similar to the existing work of Bonifacio et al. (2022). However,



Example 1:
Document: The emotional brain The discipline of affective neuroscience is concerned with the neural bases of emotion and mood. The past 30 years have witnessed an explosion of research in affective neuroscience that has addressed questions such as: which brain systems underlie emotions? How do differences in these systems relate to differences in the emotional experience of individuals? Do different regions underlie different emotions, or are all emotions a function of the same basic brain circuitry? How does emotion processing in the brain relate to bodily changes associated with emotion? And, how does emotion processing in the brain interact with cognition, motor behaviour, language and motivation?
Relevant Query: Emotion recognition in human-computer interaction

Example 2:
Document: Fast exact string matching algorithms String matching is the problem of finding all the occurrences of a pattern in a text. We propose a very fast new family of string matching algorithms based on hashing q-grams. The new algorithms are the fastest on many cases, in particular, on small size alphabets. © 2007 Elsevier B.V. All rights reserved.
Relevant Query: Text matching algorithms for sentiment analysis in social media

Example 3:
Document: Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning We present Deep Voice 3, a fully-convolutional attention-based neural textto-speech (TTS) system. Deep Voice 3 matches state-of-the-art neural speech synthesis systems in naturalness while training an order of magnitude faster. We scale Deep Voice 3 to dataset sizes unprecedented for TTS, training on more than eight hundred hours of audio from over two thousand speakers. In addition, we identify common error modes of attention-based speech synthesis networks, demonstrate how to mitigate them, and compare several different waveform synthesis methods. We also describe how to scale inference to ten million queries per day on a single GPU server.
Relevant Query: Text-to-Speech System for Visually Impaired People: A Survey

Example 4:
Document: {document_text}
Relevant Query:

Figure 2: Prompt template with in-context examples for synthetic query generation for the SCIDOCS dataset.

our contribution lies in showing that the in-domain few-shot examples (query-document pairs) help to achieve high-quality of queries compared to out-of-domain generic MS-MARCO examples. On each domain, we create a handful (e.g., 3) human generated queries for the few-shot example documents with minimal human effort, and an example prompt is shown in Figure 2.

3.3 Negative Pairs Mining

After obtaining the domain specific documents and queries, we should generate both positive and negative query-document pairs. First, the positive query-document pairs can be easily generated by mapping the synthetic queries with their corresponding original (seed) documents. Second, the negative query-document pairs can be generated from hard negative mining, described in the standard practices (Izacard et al., 2021; Xiong et al., 2020; Karpukhin et al., 2020). We parse the synthetic queries to any first-stage retrievers, such as BM25 (Robertson and Zaragoza, 2009), ColBERT (Khattab and Zaharia, 2020), or Contriever (Izacard et al., 2021), to get top-x documents. Then we pick the bottom-num_{neg} documents from the top-x to map against the synthetic queries, where 1:num_{neg} is the *positive:negative* document pair ratio.

3.4 Fine-tuning with our Synthetic Data

Our domain adaptation framework can be applied on any ranking models with any weights initialization. To establish a strong competitor, we leverage the task pre-trained model (on MS-MARCO), and sequentially fully fine-tune with our own generated synthetic data. We also adapt the same hyperparameter settings used in the MS-MARCO pre-training stage for fair deliverable.

4 Experiments

In this section, we provide details of our experimental setup to demonstrate the effectiveness of **DUQ-Gen**.

4.1 Datasets and Metrics

We employed all 18 datasets from BEIR collection, ranging on diverse retrieval tasks, to assess the effectiveness of our domain adaptation framework on standard out-of-distribution datasets. Utilizing the *multi-field* index from Pyserini (Lin et al., 2021a) for all datasets, we retrieved the top-100 and top-200 documents from lexical and dense first-stage retrievers respectively. Subsequently, we restricted re-ranking to top-100 BM25 documents and top-200 dense retriever documents. Since we evaluate our approach on both first-stage retrieval and re-ranking, we measured both nDCG@10 and R@100.

4.2 Ranking Models

We fine-tuned ColBERT[2] (Khattab and Zaharia, 2020) and MonoT5-3B[3] (Nogueira et al., 2020b), namely **DUQGen-retriever** and **DUQGen-reranker**, to show the effectiveness in both dense retrieval and re-ranking. During evaluation, we tested two multi-stage ranking pipelines: (1) **DUQGen-reranker**: a fine-tuned MonoT5-3B re-ranking BM25 top-100 and (2) **DUQGen-retriever + DUQGen-reranker**: a fine-tuned MonoT5-3B re-ranking a fine-tuned ColBERT top-200 documents.

4.3 Baselines

We chose strong competitive rankers as baselines to highlight the effectiveness of our proposed domain adapted ranker.

BM25: Traditional lexical sparse retrieval. We replicated the BM25 scores from scratch.

Zero-shot (ZS) Models: A fine-tuned ranker on MS-MARCO dataset, includes MonoT5-3B and ColBERT.

[2]https://github.com/stanford-futuredata/ColBERT
[3]castorini/monot5-3b-msmarco-10k

InPars (Bonifacio et al., 2022): An unsupervised training data generation framework for ranking. Synthetic queries are generated from randomly selected documents using few-shot prompting GPT-3 Curie model. Language model likelihood is used as a filtering step to pick top-10k high-quality synthetic queries before fine-tuning any ranker. Based on the reasons provided by Askari et al. (2023), we do not compare against InPars-v2.

DocGen-RL (Askari et al., 2023): An RL-driven framework to generate documents from queries. Also an iterative approach, based on expand, highlight, and generate stages, generates documents from queries to prepare training data.

Promptagator++ (Dai et al., 2022): As the SOTA methods closest to our work, we evaluate against Promptagator++. This methods operates by randomly selecting 1 million documents from the target collection. It utilizes 8-shot prompting with a 137 billion-parameter FLAN model (Wei et al., 2022) to create 8 queries per document. Following consistency filtering, 1 million queries are selected to train a GTR-Base dual-encoder and cross-encoder (Ni et al., 2022).

We directly utilized the scores reported by authors for DocGen-RL and Promptagator++. For the remaining baselines, we employed their corresponding HuggingFace (Wolf et al., 2020) models to re-run the inference.

4.4 Tools and Implementation

Various tools were employed for distinct stages in our pipeline, utilizing Contriever (Izacard et al., 2021) for text encoding, Faiss (Johnson et al., 2019) for k-Means clustering, and Llama2-7B-Chat (Touvron et al., 2023) for query generation, Pyserini for BM25 baseline and hard negative mining, and PyTorch for standard fine-tuning. Throughout our experiments, documents were represented using their title along with the text. Initially, collection documents were filtered for noise by excluding those with a character length less than 300 (can vary across datasets). Greedy decoding with a temperature of 0.0 was employed for the LLM to generate queries.

4.5 Hyper-Parameter Tuning

In our methodology section, we introduced several hyper-parameters, all of which underwent tuning to determine the optimal values. These include the temperature $T = 1$ (Equation 1), MMR weight $\lambda = 1.0$, number of clusters $K = 1000$, and training sample size $N = 1000$ for ColBERT and $N = 1000$ and 5000 for MonoT5-3B fine-tuning. We tuned the varying number of in-context examples and found the optimal performance with 3-shot prompting (also used in InPars). Additionally, through tuning different prompt templates, we discovered that a simple InPars-style template, displayed in Figure 2, consistently yields superior retrieval performance across datasets. For the process of hard negative mining, we set the first stage retriever hits $x = 100$ and the number of negatives per positive pair $num_{neg} = 4$.

We fine-tuned MonoT5-3B using a batch size of 8, gradient accumulation steps of 16, learning rate of $2e^{-5}$, AdamW optimizer with weight decay of 0.01 and warm-up ratio of 0.1, and epochs of 1. To fine-tune ColBERT, we adapted its official pre-training hyper-parameters, including a batch size of 32, a learning rate of $3e^{-6}$, and a maximum sequence length of 300.

The scale and quality of synthetic data depend on the training examples, N, and number of clusters, K, which we optimize in the subsequent subsections.

4.5.1 Clustering Optimization

To represent target domain, we employed K-Means algorithm, where K denotes the number of clusters. We identified the optimal K for each dataset through an unsupervised method, known as the Elbow method (Thorndike, 1953). The elbow method computes the Sum of Squared Error (SSE) for each value of K, where SSE is calculated as the sum of cosine distances between every collection document and its closest cluster centroid. The optimal K consistently aligns at a fixed point of 1000 across all evaluation datasets, irrespective of variations in corpus size, domain properties, or domain-divergence from MS-MARCO.

4.5.2 Optimal Training Sample Size Discovery

By fixing the optimum number of clusters K at 1000, we determined an optimal training sample size N, that proved effective across all datasets. To tune for N, we utilized FiQA and NQ as dev datasets, referencing prior work (InPars-v2) which demonstrated improved performance on FiQA and a declined performance on NQ compared to the zero-shot scores. Table 2 displays nDCG@10 values for various instances of N, with K fixed at 1000. Our analysis led us to select optimum $N = 1000$ for ColBERT and both $N = 1000$

Datasets (→)	covid	nfc	bio	nq	hotpot	fiqa	signal	news	robust	arg	touché	stack	quora	dbp	scidocs	fever	climate	scifact	avg
Models (↓)	Retriever																		
BM25	.656	**.325**	.465	.329	.603	.236	**.330**	.398	.407	.315	**.367**	.299	.789	.313	**.158**	.753	**.213**	.665	.423
Zero-shot ColBERT	.706	.305	.480	.523	.590	.318	.270	.390	.392	.404	.209	.350	.853	.392	.144	.771	.184	.672	.442
DUQGen-retriever	**.751**	**.325**†	**.497**†	**.530**†	**.614**†	**.336**†	.271	**.399**	**.411**†	**.425**†	.234†	**.363**†	**.857**†	**.401**	.155†	**.805**†	.196†	**.688**†	**.459**
	BM25 Top-100 + Re-ranker																		
Zero-shot MonoT5-3B	.830	.373	.559	.579	.718	.462	.321	.473	.566	.316	.311	.421	.848	.408	.193	.849	.278	.760	.515
InPars	.803	-	-	.313	-	.352	-	-	.510	-	-	-	-	.351	-	-	-	-	-
DocGen-RL	-	-	-	.517	.663	-	-	-	-	-	-	-	-	-	-	.720	-	-	-
DUQGen-reranker(1k)	**.862**$^{\dagger *}$	**.382**†	.588†	**.593**$^{\dagger *}$	**.748**$^{\dagger *}$	.458*	**.333**	**.483**	**.591**$^{\dagger *}$	.393†	**.320**	**.439**†	**.895**†	**.422**$^{\dagger *}$	.200†	**.890**$^{\dagger *}$	**.310**†	.757	**.537**
DUQGen-reranker(5k)	.836$^{\dagger *}$	.376	**.590**†	.588$^{\dagger *}$	.740$^{\dagger *}$	**.465***	.300	.449	.571*	**.427**†	.269	**.439**†	.894†	.421$^{\dagger *}$	**.202**†	**.891**$^{\dagger *}$	.288†	**.761**	.528
	Dense Retriever Top-200 + Re-ranker																		
GTR (base) retriever	.539	.308	.271	.495	.535	.349	.261	.337	.437	.511	.205	.357	.881	.347	.149	.660	.241	.600	.416
Promptagator++	.762	.370	-	-	.736	.494	-	-	-	**.630**	**.381**	-	-	.434	.201	.866	.203	.731	.528$^{\$}$
DUQGen-retriever	.751	.325	.497	.530	.613	.336	.270	.399	.411	.425	.234	.363	.857	.403	.154	.805	.196	.688	.459
+ **DUQGen-reranker(1k)**	**.851**	**.402**	.594	**.671**	**.769**	.511	**.275**	**.477**	**.636**	.511	.331	.462	**.898**	**.484**	**.203**	.901	**.309**	.758	**.558**
+ **DUQGen-reranker(5k)**	.817	.398	**.602**	.661	.768	**.517**	.253	.422	.609	.575	.262	**.463**	.896	.482	.202	**.903**	.284	**.762**	.549

Table 1: Main results comparing nDCG@10 scores between **DUQGen** and baselines on BEIR datasets. The best scores across each ranking setting are highlighted in bold. Avg score marked by $^{\$}$ calculated across only 11 datasets. **DUQGen-reranker(1k)** and **(5k)** represent the MonoT5-3B fine-tuned with 1k and 5k training examples correspondingly. Statistical significance reported using two-tailed paired t-test with Bonferroni correction ($p < 0.05$), against Zero-shot counterparts (†) and best of InPars or DocGen-RL ($*$). Promptagator++ was fine-tuned on GTR base, thus we reported GTR scores for comparison.

	N	FiQA	NQ
	(ZS) 0	.4617†	.5792
MonoT5-3B	1k	.4581	**.5934**
	5k	**.4646**	.5880†
	10k	.4553	.5777
	(ZS) 0	.3183	.5228
ColBERT	1k	.3356†	**.5301**
	5k	**.3388**	.5233†
	10k	.3306	.5171

Table 2: Ranking performances evaluated on nDCG@10 across the scale of training sample size N on dev datasets. Bold and † indicate the best and second-best scores across benchmarks for each ranker.

and 5000 for MonoT5-3B fine-tuning across the datasets.

5 Results and Discussion

In this section, we present our main experimental results and delve into the key observations. We first describe our primary findings, reported using nDCG@10 in Table 1 comparing between baselines and our approach within each ranking setting. Second, we report the first-stage retrieval performances, measured using R@100 in Table 3.

5.1 Re-ranking Results

In Table 1, it is evident that **DUQGen** consistently surpasses the SOTA baselines in most cases, exhibiting notable improvements in performance. Specifically, **DUQGen** consistently and substantially outperforms both InPars and DocGen-RL rerankers, showcasing average relative enhancements of 26% and 17% respectively across the evaluation datasets they share. When compared to Promptagator++, **DUQGen** demonstrates an average relative improvement of 4% across the shared evaluation datasets. Remarkably, **DUQGen** surpasses Promptagator++ in performance, utilizing merely 1000 LLM calls and fine-tuning with only 1000 training pairs, in contrast to Promptagator++'s requirement of generating 8 million queries using a 137B LLM and fine-tuning with 1 million training pairs. This highlights the effectiveness of our efficient and robust approach compared to the complex, resource-intensive, and exhaustive training methods based on reinforcement learning.

In many instances, the performance of the SOTA baselines degraded, compared to zero-shot counterparts. For instance, both InPars and DocGen-RL consistently demonstrate performance decreases relative to the zero-shot MonoT5-3B, with Avg. decrements of 18% and 11% respectively across the evaluation datasets they share (DocGen-RL also underperforms compared to zero-shot MonoT5-base, as shown in Table 6). On the other hand, **DUQGen** consistently surpasses all zero-shot models across all BEIR datasets, whether trained with 1,000 or 5,000 synthetic training examples.

Interestingly, training **DUQGen-reranker** with only 1,000 synthetic examples exhibited a slight performance improvement compared to training with 5,000 synthetic examples on 13 out of 18 datasets, indicating the sample efficiency of our approach. In the future, it may be feasible to au-

Datasets (→) Models (↓)	covid	nfc	bio	nq	hotpot	fiqa	signal	news	robust	arg	touché	stack	quora	dbp	scidocs	fever	climate	scifact	**avg**
BM25	.498	.250	**.714**	.760	.740	.540	**.370**	**.422**	**.375**	**.942**	**.538**	.606	.973	.398	.356	.931	.436	**.908**	.598
Zero-shot ColBERT	.473	.255	.664	.911	.747	.598	.278	.369	.311	.885	.436	.625	.989	.458	.345	.934	.447	.878	.589
DUQGen-retriever	**.544**†	**.272**†	.691†	**.915**	**.769**†	**.615**†	.291	.380†	.321†	.906†	.474†	**.645**†	**.990**	**.493**†	**.356**†	**.948**†	**.465**†	.899†	**.610**

Table 3: Comparison of R@100 scores across baselines and **DUQGen**. The best scores for each dataset are highlighted in bold. Statistical significance reported using two-tailed paired t-test with Bonferroni correction ($p < 0.05$), against Zero-shot counterpart (†).

LLM	Prompt	Size	FiQA	NQ
(Zero-shot) No LLM		-	.3702	.5404
LLAMA-2 7B Chat	ms-marco	7B	.3736	.5371
	in-domain	7B	.3811	.5444
LLAMA-2 13B Chat	in-domain	13B	**.3912**	.5370
BLOOM-3B		3B	.3380	.5193
BLOOM-7B1		7.1B	.3634	.5172
gpt-3.5-turbo		20B	.3742	**.5466**

Table 4: nDCG@10 performances across different LLMs for query generation ($K = 1$, $N = 5000$) with MonoELECTRA re-ranker fine-tuned on the generations

tomatically determine the minimum training size (N) for each dataset or task.

5.2 First-Stage Retrieval Results

In Table 3, similar to nDCG@10 scores, R@100 also demonstrates more substantial improvements for larger domain-shifts (7.1%[4] on TREC-COVID and 3.8%[4] on Touché-2020) and limited improvements for smaller domain-shifts (.4%[4] on NQ). On average, **DUQGen** enhances zero-shot ColBERT by 2.1%[4] on BEIR datasets.

6 Analysis

In this section, we report our analysis of **DUQGen**'s performance, which includes examining the need for clustering, confirming the choice of the query generator, and validating the quality of the generated queries.

6.1 Effect of Clustering for Domain Adaptation

We employ clustering to represent the target domain and number of training samples to force diversity during fine-tuning. However, we question whether clustering genuinely contributes to the process and, if so, how it influences the overall performance. Additionally, we take the training sample size N, into account. In Table 5, we illustrate the combined effect of both the K and N on MonoELECTRA top-100 BM25 re-ranking performances, measured in nDCG@10. MonoELECTRA is used in the analysis Sections 6.1 and 6.2 in order to measure the amplified performance improvements in a smaller model, as described in the previous section.

Table 5 confirms our decision to select $N = 5000$ for MonoELECTRA. Notably, this figure highlights that the most substantial and consistent improvements occur around the values of {K=1000, N=5000} across both datasets. Performances without clustering ($K = 1$) often fall below zero-shot in both datasets, especially NQ exhibiting the poorest performances.

6.2 Effect of Query Generators

We conducted an ablation study on query generation to assess how the quality of generated queries impacts overall retrieval performance. Table 4 displays the performances of MonoELECTRA fine-tuned with queries generated by various LLMs, including LLAMA2-Chat (7B and 13B), BLOOM (3B and 7B), and GPT-3.5-turbo (Brown et al., 2020).

In comparison to the zero-shot re-ranking scores, LLAMA-2 7B was deemed the optimal choice for our query generator. LLAMA-2 7B with 3-shot in-domain prompts exhibits higher improvements on both dev datasets, surpassing gpt-3.5-turbo. While LLAMA-2 13B demonstrates superior performance to 7B on FiQA, it falls below the zero-shot performance in NQ, attributed to its large model capacity and sensitivity to prompts (Zhao et al., 2021). BLOOM generates short queries lacking context, despite having sufficient contextual query examples from 3-shot examples. GPT-3.5-turbo generates high-quality queries, resulting in improved performance over zero-shot, but tends to be unstable with few-shot prompts, suggesting

[4]denotes absolute precentage improvement.

Number of Training Examples (N)	K = 1	100	500	1k	5k	10k
10k	.391 (+2%)	.391 (+2%)	.399 (+3%)	.396 (+3%)	.398 (+3%)	.389 (+2%)
5k	.381 (+1%)	.390 (+2%)	.395 (+3%)	**.392 (+2%)**	.387 (+2%)	-
1k	.382 (+1%)	.385 (+2%)	.390 (+2%)	.384 (+1%)	-	-
500	.381 (+1%)	.381 (+1%)	.381 (+1%)	-	-	-
100	.359 (-1%)	.371 (+0%)	-	-	-	-

Number of Clusters (K)

Zero-shot MonoELECTRA score on FiQA is 0.370

Number of Training Examples (N)	K = 1	100	500	1k	5k	10k
10k	.532 (-1%)	.547 (+1%)	.542 (+0%)	.546 (+1%)	.540 (+0%)	.546 (+1%)
5k	.544 (+0%)	.542 (+0%)	.544 (+0%)	**.551 (+1%)**	.543 (+0%)	-
1k	.546 (+1%)	.547 (+1%)	.550 (+1%)	.548 (+1%)	-	-
500	.537 (-0%)	.548 (+1%)	.548 (+1%)	-	-	-
100	.539 (-0%)	.546 (+1%)	-	-	-	-

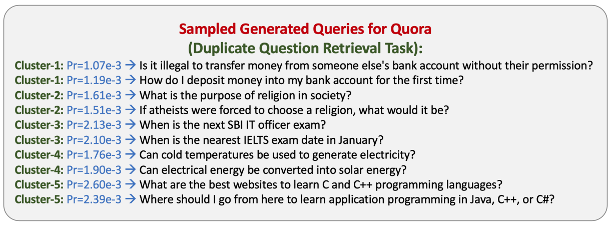
Number of Clusters (K)

Zero-shot MonoELECTRA score on NQ is 0.540

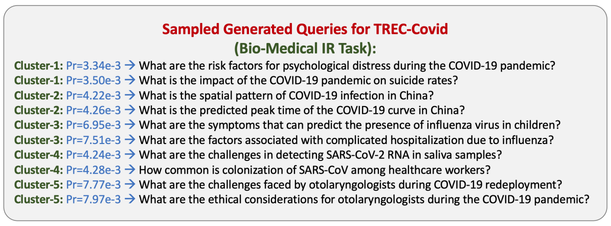
Table 5: Fine-tuned MonoELECTRA re-ranking performances in nDCG@10 for different values of K and N on FiQA (left) and NQ (right). (+%) and (-%) indicate integer rounded superior or inferior performance percentage against zero-shot scores.

potential for further prompt engineering to enhance performance on each dataset. Our second main contribution involves using in-domain 3-shot prompts to generate queries over the ms-marco prompt, showcasing notable improvements on LLAMA-2 7B model.

6.3 Examples of DUQGen Queries


Sampled Generated Queries for Quora
(Duplicate Question Retrieval Task):
Cluster-1: Pr=1.07e-3 → Is it illegal to transfer money from someone else's bank account without their permission?
Cluster-1: Pr=1.19e-3 → How do I deposit money into my bank account for the first time?
Cluster-2: Pr=1.61e-3 → What is the purpose of religion in society?
Cluster-2: Pr=1.51e-3 → If atheists were forced to choose a religion, what would it be?
Cluster-3: Pr=2.13e-3 → When is the next SBI IT officer exam?
Cluster-3: Pr=2.10e-3 → When is the nearest IELTS exam date in January?
Cluster-4: Pr=1.76e-3 → Can cold temperatures be used to generate electricity?
Cluster-4: Pr=1.90e-3 → Can electrical energy be converted into solar energy?
Cluster-5: Pr=2.60e-3 → What are the best websites to learn C and C++ programming languages?
Cluster-5: Pr=2.39e-3 → Where should I go from here to learn application programming in Java, C++, or C#?

(a)


Sampled Generated Queries for TREC-Covid
(Bio-Medical IR Task):
Cluster-1: Pr=3.34e-3 → What are the risk factors for psychological distress during the COVID-19 pandemic?
Cluster-1: Pr=3.50e-3 → What is the impact of the COVID-19 pandemic on suicide rates?
Cluster-2: Pr=4.22e-3 → What is the spatial pattern of COVID-19 infection in China?
Cluster-2: Pr=4.26e-3 → What is the predicted peak time of the COVID-19 curve in China?
Cluster-3: Pr=6.95e-3 → What are the symptoms that can predict the presence of influenza virus in children?
Cluster-3: Pr=7.51e-3 → What are the factors associated with complicated hospitalization due to influenza?
Cluster-4: Pr=4.24e-3 → What are the challenges in detecting SARS-CoV-2 RNA in saliva samples?
Cluster-4: Pr=4.28e-3 → How common is colonization of SARS-CoV among healthcare workers?
Cluster-5: Pr=7.77e-3 → What are the challenges faced by otolaryngologists during COVID-19 redeployment?
Cluster-5: Pr=7.97e-3 → What are the ethical considerations for otolaryngologists during the COVID-19 pandemic?

(b)

Figure 3: Example queries generated by **DUQGen** on (a) Quora and (b) TREC-Covid datasets. Pr denotes the $Pr(D_i|cluster_k)$ where D_i and $cluster_k$ refer to i^{th} document and k^{th} cluster.

So far, we have evaluated the effectiveness of **DUQGen** using quantitative measures and are now shifting our focus to examining the actual queries produced by our method. Figure 3 presents ten example queries generated from the Quora and TREC-Covid datasets, each representing distinct tasks and domains. In Figure 3, the synthetic queries are sampled across different clusters with different probability scores $Pr(D_i|cluster_k)$. For instance, in Figure 3a, we observe that in the Quora duplicate question retrieval task, each cluster corresponds to sub-topics of the target domain representation, such as monetary bank transfers, religion, exams in India, energy, and programming languages. Within each cluster, diverse queries are sampled using different probabilistic scores to aid in learning the domain representation. Additionally, the generated queries contain sufficient context or entities to retrieve pertinent information from its respective collection. This analysis of the generated queries further validates the effectiveness of our approach in generating a diverse and representative set of high-quality queries.

7 Conclusions

We proposed a general unsupervised domain adaptation method **DUQGen**, which can be used to fine-tune any ranking model for a given target domains. **DUQGen** introduced significant innovations over the previously reported unsupervised domain adaptation methods. Specifically, **DUQGen** proposes representing the target domain collection with document clustering; an effective method to diversify the synthetically generated queries, and an effective prompting strategy for using an LLM to generate more effective and representative synthetic training data. We experimentally demonstrated that **DUQGen** is both scalable and effective, as it uses only a few thousands of synthetic training examples, while consistently improves over the SOTA zero-shot rankers, and significantly outperforms the SOTA methods for unsupervised domain adaptation methods in most cases. We complemented the strong empirical performance of **DUQGen** with an in-depth analysis of the components to quantify their contributions. Together, the presented techniques and experimental results significantly advance neural ranking adaptation, establish a new state-of-the-art in neural ranking, and suggest promising directions for future improvements.

8 Limitations

Our proposed methodology involves two pivotal steps: (1) clustering; and (2) query generation. First, we employ Contriever as our text encoder to produce embeddings for clustering. While we anticipate that it will produce high-quality document representation and prove to be useful in our work, we did not assess other document embeddings. Future work could directly address the question of choosing the appropriate embedding for clustering.

Secondly, we employed the Faiss library to implement K-Means clustering. However, as the collection size scales up over the millions, clustering becomes impractical. Consequently, Faiss resorts to sampling the collection and then training their algorithm. This loss of information during sampling could propagate as errors in the final retrieval scores. However, given that large collections typically contain dense clusters, the process of sampling for clustering in such cases may pose less problem.

Akin to many previous studies (Zhao et al., 2021), we often encountered a lack of robustness of LLMs and their sensitivity to minor changes in the prompt affecting subsequent retrieval performance. Future work could explore strategies to mitigate this robustness through techniques like calibration (Zhao et al., 2021) and perform corresponding studies to see the impact on reranking.

9 Ethical Considerations

Retrieval systems may give rise to a variety of ethical issues, such as the potential for bias, which can result in the preferential treatment of specific perspectives, a lack of transparency due to the opaque nature of deep learning models, obscuring the reasons behind the ranking of documents, and, in extreme cases, the facilitation of echo chambers. Therefore, it is essential to conduct thorough testing of these systems both prior to and during their deployment.

As shown by our work, the performance of downstream retrievers can be rightly influenced by the LLMs employed for generating synthetic queries. Given that LLMs can produce content that is inaccurate or entirely fabricated, there's a risk that they might generate problematic queries, especially if applied to sensitive datasets. Although this issue may appear less critical in a scenario such as ours where the generated content is intended solely as training material for a following retriever, there is still a potential for generating harmful and toxic queries. Such queries could lead the retriever towards biased outcomes. Therefore, it is imperative to assess these systems to mitigate these risks against biases of the data generator.

Acknowledgements

The authors are thankful to Harshita Sahijwani and Sergey Volokhin, and the reviewers and meta-reviewers for their valuable comments and suggestions. This work was supported in part by the IARPA BETTER program (2019-19051600005). The views and conclusions contained in this work are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, or endorsements of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Arian Askari, Mohammad Aliannejadi, Chuan Meng, Evangelos Kanoulas, and Suzan Verberne. 2023. Expand, highlight, generate: RL-driven document generation for passage reranking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10087–10099, Singapore. Association for Computational Linguistics.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Unsupervised dataset generation for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2387–2392, New York, NY, USA. Association for Computing Machinery.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind

Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

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*, SIGIR '18, page 1025–1028, New York, NY, USA. Association for Computing Machinery.

Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 113–122, New York, NY, USA. Association for Computing Machinery.

Xiaomeng Hu, Shi Yu, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Ge Yu. 2022. P3 ranker: Mitigating the gaps between pre-training and ranking fine-tuning with prompt-based learning and pre-finetuning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1956–1962, New York, NY, USA. Association for Computing Machinery.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. 2023. Inpars-v2: Large language models as efficient dataset generators for information retrieval.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Jaap Kamps, Nikolaos Kondylidis, David Rau, et al. 2020. Impact of tokenization, pretraining task, and transformer depth on text ranking. In *TREC*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021a. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021b. Pretrained transformers for text ranking: Bert and beyond.

Simon Lupart, Thibault Formal, and Stéphane Clinchant. 2023. Ms-shift: An analysis of ms marco distribution shifts on neural retrieval. In *Advances in Information Retrieval*, pages 636–652, Cham. Springer Nature Switzerland.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval.

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

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020a. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020b. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Abhinav Ramesh Kashyap, Devamanyu Hazarika, Min-Yen Kan, and Roger Zimmermann. 2021. Domain divergences: A survey and empirical analysis. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1830–1849, Online. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joëlle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Conference on Empirical Methods in Natural Language Processing*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris C. Emezue, Christopher Klamm, Colin Leong, Daniel Alexander van Strien, David Ifeoluwa Adelani, Dragomir R. Radev, Eduardo Gonz'alez Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady ElSahar, Hamza Benyamina, Hieu Trung Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jorg Frohberg, Josephine L. Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario vSavsko, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad Ali Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto L'opez, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma Sharma, S. Longpre, So maieh Nikpoor, S. Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-Shaibani, Matteo Manica, Nihal V. Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Févry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiang Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Y Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre Franccois Lavall'ee, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aur'elie N'ev'eol, Charles Lovering, Daniel H Garrette, Deepak R. Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Xiangru Tang, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, S. Osher Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdenvek Kasner, Zdeněk Kasner, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ananda Santa Rosa Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ayoade Ajibade, Bharat Kumar Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David M. Lansky, Davis David, Douwe Kiela, Duong Anh Nguyen, Edward Tan, Emi

Baylor, Ezinwanne Ozoani, Fatim Tahirah Mirza, Frankline Ononiwu, Habib Rezanejad, H.A. Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jan Passmore, Joshua Seltzer, Julio Bonis Sanz, Karen Fort, Lívia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nourhan Fahmy, Olanrewaju Samuel, Ran An, R. P. Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas L. Wang, Sourav Roy, Sylvain Viguier, Thanh-Cong Le, Tobi Oyebade, Trieu Nguyen Hai Le, Yoyo Yang, Zachary Kyle Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Kumar Singh, Benjamin Beilharz, Bo Wang, Caio Matheus Fonseca de Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel Le'on Perin'an, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Iman I.B. Bello, Isha Dash, Ji Soo Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthi Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, María Andrea Castillo, Marianna Nezhurina, Mario Sanger, Matthias Samwald, Michael Cullan, Michael Weinberg, M Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patricia Haller, R. Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo L. Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Pratap Bharati, Tanmay Laud, Th'eo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yashasvi Bajaj, Y. Venkatraman, Yifan Xu, Ying Xu, Yu Xu, Zhee Xao Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100.

Shuo Sun and Kevin Duh. 2020. CLIRMatrix: A massively large collection of bilingual and multilingual datasets for cross-lingual information retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4160–4170, Online. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Robert L. Thorndike. 1953. Who belongs in the family? *Psychometrika*, 18:267–276.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360, Seattle, United States. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *ArXiv*, abs/2007.00808.

Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. Idst at trec 2019 deep learning track: Deep cascade ranking with generation-based document expansion and pre-trained language modeling. In *TREC*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models.

Peide Zhu and Claudia Hauff. 2022. Unsupervised domain adaptation for question generation with DomainData selection and self-training. In *Findings of the Association for Computational Linguistics:*

NAACL 2022, pages 2388–2401, Seattle, United States. Association for Computational Linguistics.

A Appendix

A.1 Effect on Model Sizes

Different model sizes and different ranker families have been shown to exhibit different performances. Therefore, we fine-tuned MonoELECTRA[5] (Clark et al., 2020), MonoT5-base[6], MonoT5-3B, and ColBERT to show the effectiveness of our approach against the model sizes. We pre-trained the MonoELECTRA re-ranker on MS-MARCO using a batch size of 32, learning rate of $2e^{-5}$, AdamW optimizer with weight decay of 0.01 and warm-up ratio of 0.1, regression loss, and a maximum sequence length of 512. We continued using the same hyper-parameters for fine-tuning too. For MonoT5-base fine-tuning we only changed the batch size to 8 and kept the remaining hyper-parameters same as of MonoELECTRA.

The complete performances of **DUQGen** against each zero-shot baseline measured in nDCG@10 is reported in Table 6. **DUQGen** shows consistent improvements over corresponding zero-shot baselines across almost all datasets and across all model sizes. The larger the model size, the higher the performance improvements are, for example, the average score across all datasets increases from .487 to .528 between MonoELECTRA and MonoT5-3B. It is important to note that the performance improvements achieved by **DUQGen** are larger (4% Avg.) in smaller models and smaller (1% Avg.) in larger models. Fine-tuning on ColBERT shows consistent improvements across all datasets with no drop in performance. Thus showcasing the robustness of our approach to deploying dense retrievers in practical systems.

A.2 Running Time of DUQGen

DUQGen is a cost-effective and easily scalable to a large corpus size. **DUQGen** needs to apply the pipeline only once per domain to generate training data to fine-tune a ranker. Then, at inference time, no additional complexity overheard is added. The training dataset generation process is a sequence of three steps, each fully controllable by a small set of parameters, and can be scaled to a large document collection as our experiments show. In fact, the evaluation datasets of NQ, FEVER, BioASQ are 2.5 Million, 5.5 Million, and 14.9 Million documents respectively. To give an idea of the time complexity, single-pass clustering of the full NQ collection required 8 minutes[7] on a single machine; query generation using LLama2-7B to create $N = 5,000$ synthetic training examples was the slowest step which required 1.2 hours[8], and can be increased or reduced easily by modifying N. The fine-tuning of the largest ranker model MonoT5-3B required 38 minutes[8].

A.3 Query Generation Prompts

Using additional prompts can be helpful in showcasing the diversity of tasks in BEIR and highlighting the minimal effort required for any domain using our approach. **DUQGen** can be utilized across a wide range of retrieval tasks with significantly reduced human effort, requiring only a few labeled examples as demonstrations. Thus, in Figures 4a and 4b, we present the additional prompts utilized for dev datasets (NQ and FiQA) to exemplify the diverse tasks and the minimal effort involved.

[5]cross-encoder/ms-marco-electra-base

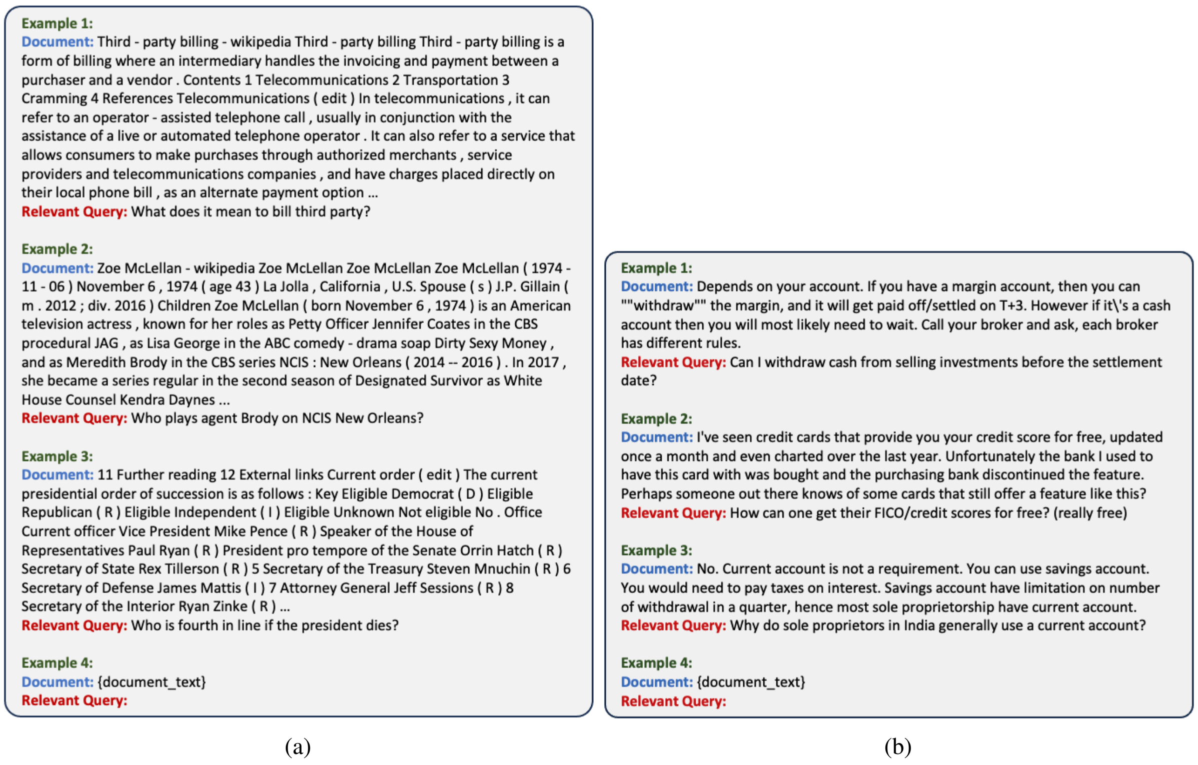
[6]castorini/monot5-base-msmarco-10k

[7]run on Quadro RTX 8000 GPU with 48GB memory.

[8]run on NVIDIA H100 GPU with 80GB memory.

		Dense Retriever		Re-ranker using BM25 Top-100					
Size (→)		110M		110M		220M		3B	
Ranker (→)		ColBERT		MonoELECTRA		MonoT5-base		MonoT5-3B	
Datasets (↓)	BM25	Zero-shot	**DUQGen-retriever**	Zero-shot	**DUQGen-reranker(5k)**	Zero-shot	**DUQGen-reranker(5k)**	Zero-shot	**DUQGen-reranker(5k)**
covid	.656	.706	.751(+6%)	.730	.761(+4%)	.814	**.853**(+5%)	.830	.836(+1%)
nfc	.325	.305	.325(+6%)	.280	.356(+27%)	.357	.368(+3%)	.373	**.376**(+1%)
bio	.465	.480	.497(+4%)	.502	.523(+4%)	.531	.566(+7%)	.559	**.590**(+6%)
nq	.329	.523	.530(+1%)	.540	.551(+2%)	.540	.550(+2%)	.579	**.588**(+2%)
hotpot	.603	.590	.614(+4%)	.691	.709(+3%)	.698	.721(+3%)	.718	**.740**(+3%)
fiqa	.236	.318	.336(+5%)	.370	.392(+6%)	.391	.400(+2%)	.462	**.465**(+1%)
signal	.330	.270	**.271**(+0%)	.297	.284(−4%)	.316	.309(−2%)	.321	.300(−6%)
news	.398	.390	.399(+2%)	.445	.411(−8%)	.459	**.470**(+2%)	.473	.449(-5%)
robust	.407	.392	.411(+5%)	.440	.479(+9%)	.518	.538(+4%)	.566	**.571**(+1%)
arg	.315	.404	.425(+5%)	.233	.327(+40%)	.188	.383(+103%)	.316	**.427**(+35%)
touché	.367	.209	.234(+12%)	.278	.261(−6%)	.305	**.347**(+14%)	.311	.269(−13%)
stack	.299	.350	.363(+4%)	.339	.387(+14%)	.389	.405(+4%)	.421	**.439**(+4%)
quora	.789	.853	.857(+0%)	.730	.873(+20%)	.845	.888(+5%)	.848	**.894**(+5%)
dbp	.313	.392	.401(+2%)	.278	.389(+40%)	.395	.406(+3%)	.408	**.421**(+3%)
scidocs	.158	.146	.155(+6%)	.162	.182(+12%)	.171	.186(+9%)	.193	**.202**(+5%)
fever	.753	.771	.805(+4%)	.816	.867(+6%)	.826	.878(+6%)	.849	**.891**(+5%)
climate	.213	.184	.196(+7%)	.246	**.296**(+21%)	.251	.268(+7%)	.278	.288(+3%)
scifact	.665	.672	.688(+2%)	.684	.727(+6%)	.730	.746(+2%)	.760	**.761**(+0%)
avg	.423	.442	.459(+4%)	.448	.487(+9%)	.485	.516(+6%)	.515	**.528**(+3%)

Table 6: Comparison of nDCG@10 scores across different model sizes and different ranking families. The best scores are highlighted in bold. Blue and red colored percentage values indicate the relative improvements in performance compared to the corresponding zero-shot baseline. Suffix **(5k)** refers to the training size used to fine-tune corresponding models.



Example 1:
Document: Third - party billing - wikipedia Third - party billing Third - party billing is a form of billing where an intermediary handles the invoicing and payment between a purchaser and a vendor . Contents 1 Telecommunications 2 Transportation 3 Cramming 4 References Telecommunications (edit) In telecommunications , it can refer to an operator - assisted telephone call , usually in conjunction with the assistance of a live or automated telephone operator . It can also refer to a service that allows consumers to make purchases through authorized merchants , service providers and telecommunications companies , and have charges placed directly on their local phone bill , as an alternate payment option ...
Relevant Query: What does it mean to bill third party?

Example 2:
Document: Zoe McLellan - wikipedia Zoe McLellan Zoe McLellan Zoe McLellan (1974 - 11 - 06) November 6 , 1974 (age 43) La Jolla , California , U.S. Spouse (s) J.P. Gillain (m . 2012 ; div. 2016) Children Zoe McLellan (born November 6 , 1974) is an American television actress , known for her roles as Petty Officer Jennifer Coates in the CBS procedural JAG , as Lisa George in the ABC comedy - drama soap Dirty Sexy Money , and as Meredith Brody in the CBS series NCIS : New Orleans (2014 -- 2016) . In 2017 , she became a series regular in the second season of Designated Survivor as White House Counsel Kendra Daynes ...
Relevant Query: Who plays agent Brody on NCIS New Orleans?

Example 3:
Document: 11 Further reading 12 External links Current order (edit) The current presidential order of succession is as follows : Key Eligible Democrat (D) Eligible Republican (R) Eligible Independent (I) Eligible Unknown Not eligible No . Office Current officer Vice President Mike Pence (R) Speaker of the House of Representatives Paul Ryan (R) President pro tempore of the Senate Orrin Hatch (R) Secretary of State Rex Tillerson (R) 5 Secretary of the Treasury Steven Mnuchin (R) 6 Secretary of Defense James Mattis (I) 7 Attorney General Jeff Sessions (R) 8 Secretary of the Interior Ryan Zinke (R) ...
Relevant Query: Who is fourth in line if the president dies?

Example 4:
Document: {document_text}
Relevant Query:

(a)

Example 1:
Document: Depends on your account. If you have a margin account, then you can ""withdraw"" the margin, and it will get paid off/settled on T+3. However if it\'s a cash account then you will most likely need to wait. Call your broker and ask, each broker has different rules.
Relevant Query: Can I withdraw cash from selling investments before the settlement date?

Example 2:
Document: I've seen credit cards that provide you your credit score for free, updated once a month and even charted over the last year. Unfortunately the bank I used to have this card with was bought and the purchasing bank discontinued the feature. Perhaps someone out there knows of some cards that still offer a feature like this?
Relevant Query: How can one get their FICO/credit scores for free? (really free)

Example 3:
Document: No. Current account is not a requirement. You can use savings account. You would need to pay taxes on interest. Savings account have limitation on number of withdrawal in a quarter, hence most sole proprietorship have current account.
Relevant Query: Why do sole proprietors in India generally use a current account?

Example 4:
Document: {document_text}
Relevant Query:

(b)

Figure 4: Example prompts used for (a) NQ and (b) FiQA dataset.