

Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Ramraj Chandradevan

Date

Training Search and Ranking Models with Minimal Supervision

By

Ramraj Chandradevan
Doctor of Philosophy

Computer Science and Informatics

Eugene Agichtein, Ph.D.
Advisor

Carl Yang, Ph.D.
Committee Member

Fei Liu, Ph.D.
Committee Member

Benjamin Van Durme, Ph.D.
Committee Member

Bhaskar Mitra, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D, MPH
Dean of the James T. Laney School of Graduate Studies

Date

Training Search and Ranking Models with Minimal Supervision

By

Ramraj Chandradevan
BSc, University of Moratuwa, Sri Lanka, 2019

Advisor: Eugene Agichtein, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2024

Abstract

Training Search and Ranking Models with Minimal Supervision By Ramraj Chandradevan

Searching for relevant or useful information in large document collections or the Web is challenging, particularly to support diverse search tasks, domains, or languages. The current State-of-the-Art methods use a Learning to Rank approach, typically using deep Neural Networks pretrained on large ranking datasets such as MS-MARCO. These methods have shown impressive performance without additional training (a.k.a in a zero-shot setting), since acquiring sufficiently large domain specific training data is often not feasible. However, as zero-shot ranking does not take advantage of the target domain information, there is a potential for improvement, especially for specialized domains and tasks. However, to fine-tune a large Neural Net-based ranker for a new domain requires large amounts of labeled training data. The central challenge of this thesis is whether state-of-the-art Neural rankers can be adapted to new domains, with minimal supervision. To solve this challenge, this thesis proposes several unsupervised and weakly-supervised approaches to learning to rank, aiming to fill a critical gap in prior literature, namely how to automatically train a (Neural) ranker with minimal supervision. My research focuses on three primary research questions: (1) Can adapting *query representation* with domain information improve ranking performance? (2) Can *ranking models* be effectively fine-tuned with minimal or no supervision? and (3) Can ranking models be further *adapted for specific downstream tasks*, such as Retrieval Augmented Generation (RAG), using weak or no supervision? To address these questions, this thesis investigates multiple techniques, including LLM self-referencing, pseudo-labeling, synthetic query generation, ensemble-prompting, contrastive fine-tuning, and query enrichment. These approaches leverage diverse target information such as documents, synthetic queries, and weak labels. Lastly, this thesis proposes a list of experiments to evaluate the proposed approaches on ranking benchmarks across multiple domains, such as BEIR, and across multiple languages, such as MIRACL and CLEF, and across retrieval tasks, such as CRAG, to demonstrate their effectiveness and robustness.

Training Search and Ranking Models with Minimal Supervision

By

Ramraj Chandradevan
BSc, University of Moratuwa, Sri Lanka, 2019

Advisor: Eugene Agichtein, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2024

Contents

1	Introduction	1
1.1	Overview of Thesis	3
2	Related Work	5
2.1	Transfer Learning	5
2.2	Domain Adaptation (DA)	6
2.3	Source-Free Domain Adaptation (SFDA)	6
2.4	Test Time Domain Adaptation (TTDA)	7
2.5	Learning-to-Rank Overview	7
2.6	Supervised Training Neural Rankers	8
2.7	Weakly Supervised Training Neural Rankers	9
2.8	Unsupervised Training Neural Rankers	12
2.9	LLM based Neural Rankers	13
3	Unsupervised Test Time Query Representation	18
3.1	Query Expansion (QE)	19
3.2	Pseudo Relevance Feedback (PRF)	19
3.3	Overview of Proposed Query Representation Approaches	20
3.4	Learning to Enrich Query Representation	21
3.5	Unsupervised Ensemble Generative Prompting to Enrich Query Representation	28
4	Unsupervised Ranker Adaptation	35
4.1	Overview of Proposed Ranker Adaptation Approaches	36
4.2	Preliminary Study on Domain Fine-tuning	37
4.3	Diversified Synthetic Query Generation for Ranker Fine-tuning	41
4.4	Multilingual Synthetic Query Generation for Ranker Fine-tuning	53
5	Task-Specific Neural Ranker Training	63
5.1	Overview of Proposed Task Fine-tuning Approaches	65
5.2	Task-Specific Contrastive Fine-tuning	65
5.3	Event-Aware Task Fine-tuning	72
5.4	Task Fine-tuning Neural Ranker in a RAG System	79
6	Conclusions and Future Work	99
6.1	Thesis Contributions	100
6.2	Summary of Results	100
6.3	Limitations	102

6.4	Future Work	103
A	Appendix for TFT-RAG	105
A.1	Reranked References for TFT-RAG Prediction	105
	Bibliography	107

List of Figures

1.1	An overview and classification of my approaches discussed in this thesis.	3
3.1	Architecture of NCLPRF ranker. A query is represented as the aggregation of N query-feedback document encodings. The reranking score of a candidate document is based on the similarity of its document encoding and the query encoding.	23
3.2	Different vector aggregations across PRF depths in (a) NTCIR Chinese and (b) CLEF Persian collections.	27
3.3	The complete flow and algorithm of GenQREnsemble and GenQREnsemble-RF (dotted).	29
3.4	The complete flow of GenQRFusion. GenQRFusion-RF is shown with the dotted line.	30
3.5	The N reformulation instructions used for GenQREnsemble and GenQRFusion	31
3.6	The prompt used for all the Llama-2 Query reformulators.	32
3.7	Effect of PRF with increasing feedback on TP19. BM25 is employed to directly query using the reformulated queries.	33
4.1	DUQGen : an unsupervised domain-adaptation framework for neural ranking.	42
4.2	Prompt template with in-context examples for synthetic query generation for the SCI-DOCS dataset.	45
4.3	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.	52
4.4	Prompt template with Hindi language examples for generating queries.	55
5.1	Pipeline for training a dense retrieval model. An additional pretraining phase is introduced targeting CLIR.	67
5.2	ColBERT models with zero-shot transfer (ZS) and translate-train (TT) approaches using XLM-RoBERTa-base on HC4 Chinese test set. The dashed line demonstrate the nDCG@100 value for XLM-RoBERTa-large with both approaches.	70
5.3	Overview of fine-tuning an event ranker model on unsupervised and weakly-supervised training data.	72
5.4	An example showing event extraction (IE) and query generation.	74
5.5	TFT-RAG : task-specific fine-tuning for neural ranking.	81
5.6	Prompt template used for self-referencing to annotate (a) positive and (b) negative grounded documents during the training stage. Black and blue font colored portions indicate system and user prompt messages, respectively.	82
5.7	Sample output generated by LLM during hard negative generation process	83
5.8	Prompt template used for chain-of-thought answer generation (annotate-then-generate) during the inference stage. Black and blue font colored portions indicate system and user prompt messages, respectively.	85
5.9	Reranking pipeline illustrating the complete retrieval, chunking, and reranking processes for (a) the training-stage and (b) the inference-stage.	86

5.10	Prompt template used for automatic model-based evaluation. (a) and (b) are combined during evaluation (split here for space constraints). Black and blue font colored portions indicate system and user prompt messages, respectively.	96
5.11	Example questions, ground truth answers, and incorrect gpt-4o predictions. These gpt-4o predictions are generated by my RAG system, which includes a task fine-tuned neural ranker and the annotate-then-generate prompting technique.	97
5.12	Prompt template used for chain-of-thought answer generation (annotate-then-generate) during the inference stage only for llama3-8b model. Black and blue font colored portions indicate system and user prompt messages, respectively.	98
A.1	An example question, ground truth answer, ranked references, and incorrect generator predictions. Prediction includes the entire thought process of ratings, thought, and final answer.	106

List of Tables

3.1	Examples of query types used across different retrieval applications.	20
3.2	Train (CLIR-Matrix), dev (HC4), test (CLEF and NTCIR) collections statistics.	24
3.3	Comparison of NCLPRF against baselines. All models at the bottom half of the table rerank the top 100 documents retrieved by BM25 (without RM3). BM25 uses machine-translated queries and neural rerankers use the provided English queries. Percentages in parentheses are relative improvements over the BM25 + RM3 with 2 feedback documents.	26
3.4	Comparison of PRF performance on the TP19 benchmark using queries generated from <code>flan-t5-xxl</code> . With BM25 Retriever utilizes the reformulated query directly for initial retrieval, while With Neural Reranking further re-ranks the results obtained from the BM25 first-stage retrieval.	32
4.1	Comparison of zero-shot point-wise rankers measured in nDCG@10. The best scores are highlighted in bold.	38
4.2	Comparison of zero-shot point-wise and list-wise rankers measured in nDCG@10. The best scores are highlighted in bold.	39
4.3	Evaluation of supervised fine-tuning on a subset of BEIR datasets measured in nDCG@10.	40
4.4	Main results comparing nDCG@10 scores between 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.	48
4.5	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.	48
4.6	Comparison of R@100 scores across baselines and . 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 ([†]).	48
4.7	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.	49
4.8	nDCG@10 performances across different LLMs for query generation ($K = 1, N = 5000$) with MonoELECTRA re-ranker fine-tuned on the generations	50

4.9	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.	51
4.10	Performance comparison of mDUQGen against strong and state-of-the-art baselines on MIRACL dev datasets using nDCG@10 and R@100 metrics. The model names with (+) denote that a first-stage retrieval top-100 results are re-ranked. Hybrid is the score fusion between BM25 and mDUQGen-retriever	56
4.11	Comparison of different neural re-rankers pre-trained using different source data measured in nDCG@10. mDUQGen-reranker is the fine-tuned Mono-mT5-base. They re-rank the top-100 documents from mDUQGen-retriever	57
4.12	Performances of mContriever fine-tuned with generated queries ($K = 1, N = 1000$) using different LLMs evaluated on dev datasets: Swahili and Arabic.	59
4.13	Performances of mContriever fine-tuned using different training sizes (N) and evaluated on dev datasets: Swahili and Arabic, where queries are generated using gpt-3.5-turbo and $K = 1000$	60
5.1	Reranking effectiveness of ColBERT and DPR models with and without our C3 pretraining. The top shows XLM-RoBERTa-base models; the bottom shows XLM-algin-base models. Symbols indicate statistically significant differences at $p < 0.05$ by a two-tailed paired t -test with Bonferroni correction for 6 tests, either with and without C3 (*) or between C3 and original BM25 results (†). Δ shows the mean relative improvement from C3 across the 6 collections.	69
5.2	Ablation study on different similarity function used in contrastive learning with and without the Condenser head (Cond.). The values showed in the table is nDCG@100 on HC4 Chinese test set.	71
5.3	CLEF, NTCIR, and BETTER collections statistics.	76
5.4	Retrieval performance of statistical and neural retrieval models for varying languages and tasks. * indicates the difference between ColBERT-X (ZS) and the full model is statistically significant with 95% confidence using the paired t-test with Bonferroni correction of two tests (across collections within the tasks).	77
5.5	Experimental results involve tuning chunk hyper-parameters such as the number of chunks and their maximum lengths during training and inference stages. These experiments were conducted on the validation data split using a RAG system incorporating a MonoELECTRA re-ranker and an LLaMA3-8B-instruct generator.	87
5.6	Performance comparison between TFT-RAG and baseline methods. The RAG (Reproduced) employs MiniLM-L6-v2 as the re-ranker, while TFT-RAG incorporates fine-tuned MonoELECTRA re-ranker. -Inference indicates the addition of self-referencing instructions to the prompt during the inference stage. * indicates a prompt variation (refer Figure 5.12) due to limitations present in llama3-8b.	90
5.7	Performance comparison of various inference-stage prompt configurations. Prompts consist of three main components: Passage Evaluation, Answer Generation, and Few-shot Examples. An X indicates the absence of a specific component. RG-*/4L refers to Relevance Grading across four levels, with 2,3 and 1,2,3 denoting the selection of evidence based on those respective grades for final answer generation. Q refers to questions.	91

5.8 End-to-end RAG performances measured across different question types on test split. For each row, the baseline is taken from the row immediately above, skipping one. Specifically, rows 2, 4, and 6 are considered, with the baselines being rows 1, 3, and 5, respectively. In the Accuracy sub-table, (+%) and (-%) indicate the rounded integer relative improvement or decrease in Accuracy compared to the baseline. In the Hallucination sub-table, (+%) and (-%) represent the rounded integer relative increase or decrease in Hallucination compared to the baseline. Positive changes in Accuracy and negative changes in Hallucination are highlighted in blue, while negative changes in Accuracy and positive changes in Hallucination are highlighted in red, with blue representing better performance and red indicating worse performance. 93

List of Algorithms

1
to Acquire Hard Negatives84

1 Introduction

In rapidly advancing digital era, the seamless storage, retrieval, and manipulation of information have become indispensable facets of human-computer interaction. As the volume of data — encompassing text, images, videos, and audio — continues to burgeon, the necessity for expansive storage infrastructures and sophisticated search systems has become paramount. Despite the strides made in developing efficient information storage systems, the precise retrieval of pertinent data from digital repositories remains an ongoing challenge. This challenge stems primarily from the intricate nature of articulating user needs or intents within the digital realm. Searching through text collections is relatively easier than other modalities because of its structured nature. An optimal text-based search system returns an ordered list of documents in response to a user query, determined by a relevancy measure. However, searching for relevant or useful information in a large document collection is challenging, particularly across diverse domains, languages, and tasks [1].

For many decades, researchers delved into constructing a potent search system, initially focusing on rule-based and statistical retrieval systems. The introduction of machine learning and neural networks led to widespread adoption of feature-based learning-to-rank algorithms. More recently, the rise of pretrained language models (PLMs) and large language models (LLMs) has significantly enhanced semantic understanding and reasoning capabilities, especially in discerning relevant documents by efficient search systems. This thesis centers on investigating neural search systems built on PLMs or LLMs, aiming to elucidate how such effective rankers can be trained with minimal human intervention for different domains and applications.

State-of-the-art neural rankers, pretrained on large datasets, such as MS-MARCO [2], have shown impressive performance without additional training (a.k.a. zero-shot setting), since acquiring sufficiently large domain-specific training data is often not feasible. However, as zero-shot ranking does not take advantage of the target domain information, there is a potential for improvement, especially for specialized domains, languages, and tasks. Fine-tuning dataset collections often differ significantly from pretrained collections in various aspects including topics, linguistic attributes, language, structure, and format [3]. Consequently, a pretrained neural ranker lacks awareness of target domain-specific vocabularies, knowledge, data distributions, and user intents. Hence, it is crucial for a pretrained neural ranker to effectively adapt to individual domains and tasks to achieve substantial performance enhancements.

The central challenge of this thesis is whether state-of-the-art neural rankers can be adapted to new domains, with minimal supervision. To solve this challenge, this thesis proposes several unsupervised and weakly-supervised approaches, aiming to fill a critical gap in prior literature, namely how to automatically train a ranker with minimal supervision. Past studies have explored training neural ranking models with weak supervision or no supervision, yet they often fail to strike a balance between effectiveness, robustness, scalability, computational cost, and adaptability. For instance, unsupervised domain fine-tuning techniques, such as InPars [4], DocGen-RL [5], and Promptagator [6], rely on expensive frameworks, and their performance gains are often inconsistent across out-of-distribution datasets. Building upon Approximate nearest neighbor Negative Contrastive Estimation (ANCE) [7] learning, the extension ANCE-PRF [8] approach faces performance constraints on target query representation due to context length limitations and difficulties in generalizing across different languages. Task fine-tuning approaches like REPLUG [9] and ARL2 [10], which aim to align a neural ranker with a downstream system such as an LLM, often fail to fully capture task relevance and require multiple costly LLM calls to generate task-specific training data. Hence, in light of these challenges and significant gaps in domain adaptation, my research is centered around addressing three key research questions:

RQ1: Can adapting query representation with domain information improve ranking performance?

RQ2: Can ranking models be effectively fine-tuned with minimal or no supervision?

RQ3: Can ranking models be further adapted for specific downstream tasks, such as Retrieval Augmented Generation (RAG), using weak or no supervision?

To address these questions, this thesis investigates multiple techniques, including LLM self-referencing, pseudo-labeling, synthetic query generation, ensemble-prompting, contrastive fine-tuning, and query enrichment. These approaches leverage diverse target information such as documents, synthetic queries, and weak labels. In summary my primary thesis contributions can be listed as:

1. Proposed unsupervised, scalable query expansion techniques to integrate domain-specific information from target documents.
2. Introduced unsupervised, efficient ranker fine-tuning methods designed to incorporate domain-specific information from target queries.
3. Developed weakly supervised, task-specific fine-tuning strategies that integrate domain knowledge from target relevance labels.

The thesis begins with an overview of background literature (next chapter). Subsequent chapters will delve into each of the research questions (RQs) raised, presenting detailed solutions supported by experimental evidence. The thesis will conclude with a summary and discussion of limitations. Before exploring my thesis in detail, I will present an overview of the key contributions to offer a comprehensive understanding of my work.

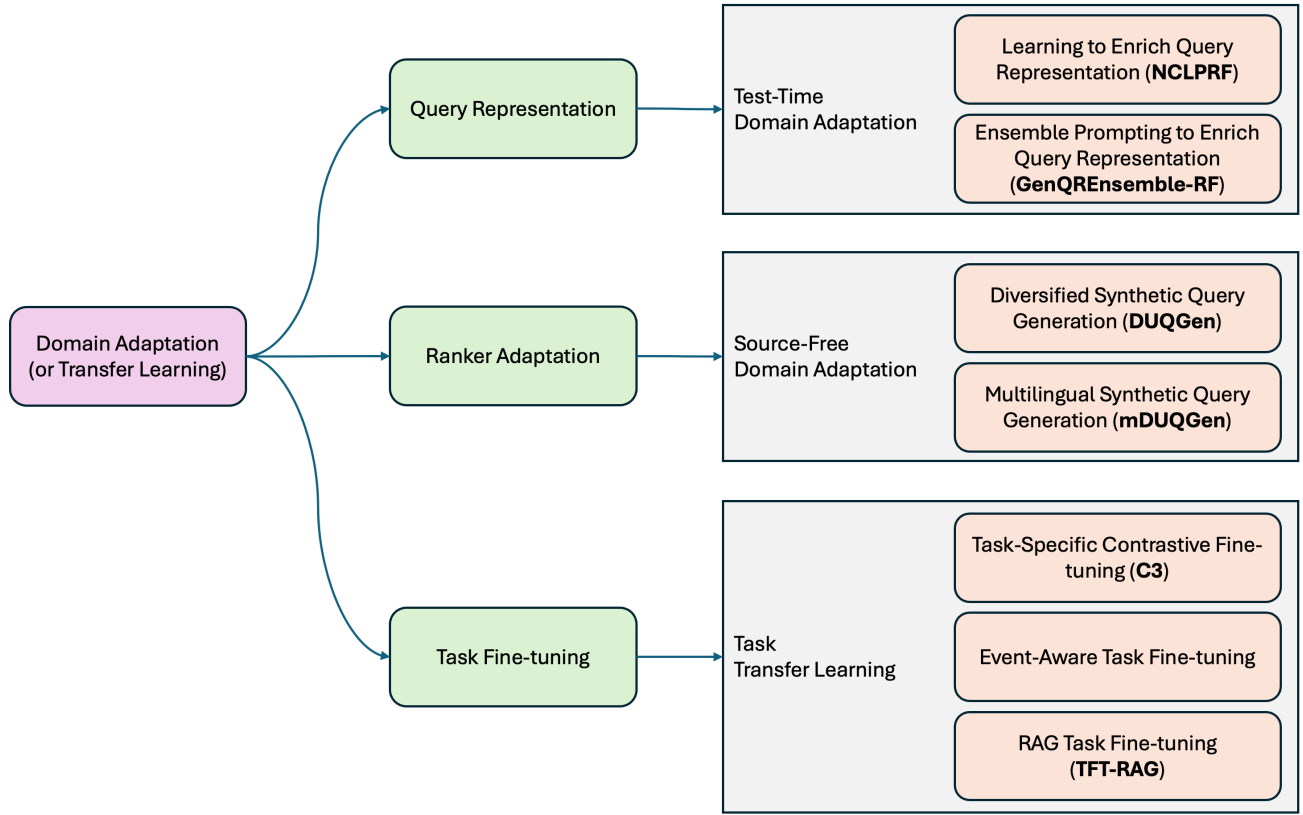


Figure 1.1: An overview and classification of my approaches discussed in this thesis.

1.1 Overview of Thesis

My thesis focuses on training search and ranking models with minimal or no supervision. I approach the research problem through the lens of transfer learning and domain adaptation, aiming to develop models that can be effectively applied across new domains, languages, and tasks. In real-world applications, a wide variety of natural language tasks require search and ranking models that are scalable, robust, modular, and cost-efficient for vertical deployment. As a result, it is essential to dedicate significant time and effort to researching innovative methods that prioritize scalability, robustness, and cost-effectiveness.

The information retrieval community has experienced numerous innovations, including learning-to-rank, neural ranking, and more recently, generative AI-based ranking. My thesis focuses on neural ranking models, particularly those based on transformer architectures and large language models. The primary research questions I presented in the Introduction highlight key areas that require further exploration: **query representation**, **ranker adaptation**, and **task fine-tuning**.

First, effective query representation is crucial for understanding user needs and intent, necessitating the adaptation of user queries in the target domain with minimal effort. Second, ranker adaptation is essential for fine-tuning a neural ranking model to a specific domain, enabling it to extract domain-specific information and understand data distributions from queries and documents to enhance matching signals. Third, task fine-tuning aligns a neural ranking model with the corresponding downstream system, such as a large language model, to improve overall system performance. Consequently, it is vital to develop novel and effective approaches that address these three objectives in a scalable and cost-effective manner.

Each of my RQs uniquely contributes to improving neural ranker performance in domain and task

transfer learning. Specifically, RQ1, RQ2, and RQ3 focus on adjusting query representation towards the target domain, adapting the neural ranker to the target domain or language, and aligning the relevance task with downstream applications, respectively. RQ1 achieves domain adaptation without requiring training, simply modifying input representation at inference time. For a retrieval task, modifying the input query is an effective solution for achieving domain adaptation across different datasets.

In contrast, RQ2 and RQ3 require model retraining with full weight updates to transfer knowledge to the target domain or task. While both RQ2 and RQ3 focus on fine-tuning a neural ranker, they address distinct problem settings. RQ2 tackles domain adaptation, where the task remains consistent across source and target domains (such as retrieval task) but the domains themselves vary, such as across datasets or languages. Thus, RQ2 emphasizes **adapting the ranker to a target domain** (*e.g.*, specialized fields like finance, biomedical, and healthcare, or languages like Spanish, French, and Russian) using domain-specific query-document pairs. This line of works help the neural ranker become more corpus-aware and develop a holistic understanding of the target domain.

In contrast, RQ3 addresses task transfer learning, where the tasks differ between the source and target (*e.g.*, from pretraining language modeling to classification, summarization, or question answering). RQ3 focuses on **aligning the ranker with a target downstream task or application** (such as question answering, event extraction, or retrieval-augmented generation (RAG)) using task-specific contrastive query-document pairs. This task-specific fine-tuning enables a neural ranker, initially trained for question answering (*e.g.*, MS-MARCO), to be more effective for application-specific retrieval tasks.

I propose seven novel approaches, categorized according to the three previously outlined RQs, with an overview provided in Figure 1.1. The first two approaches, **NCLPRF** and **GenQREnsemble-RF**, present effective, efficient, and scalable techniques for query expansion and enrichment aimed at enhancing query representation during the inference stage, without the need to retrain the neural ranking model on the target domain. These methods fall under the category of *test-time domain adaptation (TTDA)*, where the input is modified based on target information during inference.

The next two approaches, **DUQGen** and **mDUQGen**, focus on effective, efficient, and scalable ranker adaptation across various domains, languages, and retrieval tasks. They introduce a novel, cost-effective, and sample-efficient method for generating diversified synthetic queries, thereby augmenting the training data to fine-tune neural ranking models. These approaches fall within the traditional *domain adaptation* category, where source pretrained models are fine-tuned using synthetic training data.

The final three approaches, **C3**, **Event-aware Task Fine-tuning**, and **TFT-RAG**, offer effective and efficient techniques for task fine-tuning across new tasks. They introduce three distinct weakly-supervised pseudo-labeling strategies—contrastive learning, pseudo-label filtering, and pseudo-label generation—to enhance the quality of training labels during the fine-tuning process. These approaches are classified under the task transfer category (transfer learning), where source pretrained models undergo task-specific fine-tuning. Notably, TFT-RAG demonstrates that aligning a neural ranker with a downstream system (such as an LLM) using that system as a weak annotator (or teacher) yields high-quality annotations that significantly improve end-to-end system performance.

In summary, these seven novel approaches represent highly effective, efficient, and scalable techniques for training search and ranking models across new domains, languages, and tasks, surpassing current state-of-the-art methodologies. The next chapter 2, will review previous studies relevant to my research questions to further substantiate the innovations presented in my thesis.

2 Related Work

This Chapter provides the foundational aspects of my thesis research, outlining key works related to transfer learning, domain adaptation, learning-to-rank, and neural ranking models. I begin by providing a broad overview of transfer learning, discussing various classes such as traditional domain adaptation, source-free domain adaptation, and test-time domain adaptation. The discussion then transitions to the background of information retrieval, starting with an introduction to learning-to-rank approaches, before progressing into the realm of neural ranking techniques. The organization of the information retrieval literature is based on the scale of training data required: learning-to-rank models typically need smaller datasets, while supervised neural ranking models demand more extensive data. This scale continues to increase with weakly supervised and unsupervised neural ranking systems, which involve large-scale pretraining and fine-tuning phases. Finally, the Chapter concludes with a discussion on large language models (LLMs) as rankers, which necessitate massive datasets for both pretraining and fine-tuning stages.

2.1 Transfer Learning

Adapting a learning-to-rank framework to new domains, languages, or applications presents challenges due to the differences between the source and target domains. Since this is fundamentally a learning problem, it aligns with the broader machine learning challenge of how to adapt models to different domains or tasks. This motivated me to explore general transfer learning and domain adaptation methods, including test-time adaptation, data selection, fine-tuning, and pseudo-label filtering and generation, in order to apply on specific learning-to-rank problem. Through this investigation, I introduced innovative approaches to neural ranker adaptation, particularly in addressing domain shifts. Consequently, this section will first cover transfer learning and domain adaptation in general, then delve into their prior works to neural ranking specifically.

One of the most extensively researched topics in the machine learning and deep learning community is how to adapt the machine learning models to various specialized applications, scenarios, and environments. This challenge is commonly referred to as "transfer learning". In transfer learning, the goal is for the model to perform well on related but different data distributions. It is formally defined as learning with minimal or no supervision using the previously acquired knowledge. In both supervised and semi-supervised learning settings, it is typically assumed that the training and testing data come from the same distribution [11]. However, in real-world applications, this is often not the case, and similar performance on test data is not always guaranteed.

Pan and Yang [12] classified the surveyed transfer learning approaches into two categories: *domain* and *task*. A domain includes a feature space and a marginal probability distribution, while a task consists of a label space and a predictive objective function [13]. Thus, transfer learning can involve transferring knowledge from a source domain to a target domain, from a source task to a target task, or a combination of both. My thesis research broadly covers transfer learning methods that rely on minimal or no supervision. Specifically, my solutions to **RQ1** and **RQ2** fall under *domain* transfer learning, while the solution to **RQ3** falls under *task* transfer learning.

2.2 Domain Adaptation (DA)

Domain adaptation is commonly described as transferring the learnt knowledge from a source domain to a target domain. It is a specific class of transfer learning where the source and target domains are related but have different data distributions. Domain adaptation is necessary when there is a divergence or distribution shift between the two domains. The goal of domain adaptation is to maintain or enhance the performance on the target domain by utilizing any available information from both the source and target domains, ultimately reducing the discrepancy between their distributions.

The main challenge in domain adaptation is minimizing the domain shift under the *independent homogeneous distribution assumption*. This assumption states that while the feature and label spaces remain consistent across domains, the data distributions differ. For instance, in a binary image classification task between cats and dogs, the feature space consists of pixel values in both the source A and target B domains, and the label space includes the classes "cat", "dog". Domain A might contain high-resolution images taken by professional cameras, while Domain B consists of images captured by mobile phones with varying resolutions and lighting conditions. Despite these data distribution differences due to the image qualities, the independent homogeneous distribution assumption still holds because the feature and label spaces are the same. Therefore, addressing the distribution discrepancies between domains is critical when deploying machine learning models in real-world applications.

As deep learning and generative models continue to advance and demonstrate strong data representation capabilities at scale, they often struggle to perform well in specialized real-world applications, such as healthcare, finance, and medicine. To address the distribution gap in these scenarios, several approaches have been proposed. These include, but are not limited to, learning domain-invariant features to align source and target distributions, mapping between domains, designing ensemble methods, and building target-discriminative models. These strategies aim to reduce the domain shift between source and target domains.

2.3 Source-Free Domain Adaptation (SFDA)

Domain adaptation techniques have been applied across a wide range of machine learning applications, including computer vision, speech processing, and natural language processing. While domain adaptation typically assumes the access to both source and target data, the raw source data is often unavailable for various reasons, such as data privacy concerns, confidentiality, copyright issues, storage and transmission costs, and computational limitations [14]. As a result, much of the research has focused on the *Source-Free Domain Adaptation (SFDA)* category, which operates under the assumption that while source data is not available, target data is provided for the purpose of domain adaptation.

To train machine learning models effectively, it is essential to acquire new labeled examples from a target domain. However, obtaining such labels can be expensive in many real-world applications. Consequently, SFDA approaches have primarily aimed to eliminate the dependence on supervised labeled data from the target domain by exploring methods such as unsupervised, weakly-supervised, and semi-supervised learning. As a result, my thesis research has focused on developing strategies within the realm of source-free unsupervised (**RQ1** and **RQ2**) and weakly-supervised (**RQ3**) domain adaptation.

The SFDA approaches can be classified into two categories: (1) data-centric and (2) model-centric [14]. Firstly, the data-centric approaches primarily aim to reconstruct the source domain using the available data from the target domain. Examples of these approaches include neighborhood clustering and local structure clustering, which operate under the assumption that the intrinsic structure in both the source and target data is represented by their neighborhood relationships [14]. As a result, target domain samples can

be mapped to the source domain distribution by examining these neighborhood connections [14].

In addition, with the advent of recent language models, generating synthetic training data (or augmenting target data) has become another category of data-centric approaches. This method enhances the target domain representation by creating additional data, helping to reduce the distribution gap between the source and target domains. My two solutions to **RQ2** fall under the data-centric approaches: **DUQGen** and **mDUQGen** both leverages the LLM based synthetic training data generation to address the domain and language distribution differences.

Secondly, the model-centric approaches rely on the principle of self-training. In these methods, a model trained on the source domain is sequentially fine-tuned using its own predictions on the target domain. This approach assumes that the source-trained model can generalize to the target domain at certain extent due to their relatedness. Previous research in model-centric approaches has primarily focused on techniques such as pseudo-labeling, entropy minimization, and contrastive learning.

Pseudo-labeling approaches are widely used in model-centric SFDA. In these methods, pseudo-labels are assigned to unlabeled target domain examples, and the source-trained model is then fine-tuned using these pseudo-labeled examples. These approaches primarily differ in how the pseudo-labels are obtained, which can involve variations in prototype generation, pseudo-label assignment, or pseudo-label filtering [14]. My three solutions for **RQ3** fall under model-centric approaches: **C3** utilizes contrastive learning, **IE-IR** focuses on pseudo-label filtering, and **TFT-RAG** applies pseudo-label assignment.

SFDA approaches can also be classified into black-box and white-box categories [15], depending on the availability of the source pretrained model weights. All of my solutions, except for **GenQREnsemble-RF**, fall under the white-box approaches, as they involve sequentially fine-tuning the source pretrained model. **GenQREnsemble-RF**, on the other hand, does not fine-tune any models but adapts the feature representations during the inference.

2.4 Test Time Domain Adaptation (TTDA)

Test Time Domain Adaptation (TTDA) refers to the process of adapting a machine learning model to a new domain or distribution at the time of inference, without retraining the model beforehand [16]. In traditional domain adaptation, models are trained on labeled data from both source and target domains. However, in TTDA, the model is trained only on the source domain, and during testing, it adapts to the target domain using unlabeled test data.

TTDA methods typically aim to make adjustments based on the distributional differences between the source and target domains that are observed at test time. This could involve updating model parameters, modifying input representations, or dynamically altering predictions based on characteristics of the test data. The goal is to improve performance on the target domain while avoiding full model retraining. My approaches for **RQ1** fall under TTDA. Both approaches, **NCLPRF** and **GenQREnsemble-RF**, involve modifying the input representations during testing, but they operate on different feature spaces—one at the semantic level and the other at the surface lexical level. These approaches are particularly useful in scenarios where the target domain data is available only at inference and labeled data is limited or absent.

2.5 Learning-to-Rank Overview

Ranking has posed a significant challenge for search systems over many decades. The emergence of the machine learning paradigm has proven effective in addressing predictive and decision-making tasks such as classification, regression, and ranking. The ranking task, also known as "relevance ranking," involves

predicting the most optimized order of documents for a given query based on their relevance to the query. Learning-to-rank approaches view ranking as a learning problem, optimizing a feature-based learning objective towards improving a performance measure. Ideally, a learning-to-rank algorithm should produce an effective order for a set of documents in response to a query.

A variety of popular machine learning methods have been proposed for learning-to-rank algorithms, and they have been deployed in numerous real-world applications for an extended period. The task of relevance ranking differs from traditional classification and regression tasks in that it is focused on optimizing the permutation order of a list of documents, whereas classification and regression aim to optimize predictions of scores or class labels. Consequently, directly applying existing machine learning techniques to learning-to-rank is not straightforward, requiring efforts from information retrieval (IR) experts to adapt these techniques. Popular learning-to-rank algorithms make use of machine learning methodologies such as logistic regression [17], Support Vector Machines (SVM) [18], Random Forest [19], and Boosting [20]. Several surveys [21, 22, 23, 24] have been conducted to explore these approaches and provide evidence of their effectiveness in relevance learning tasks.

2.6 Supervised Training Neural Rankers

In recent times, pretrained language models based on transformer architectures have exhibited remarkable effectiveness in neural rankers [25, 26]. A neural ranker returns an ordered list of documents given a query, where a relevancy is calculated based on the query and document dense **embedding** representations. This section discusses the existing neural rankers trained with large-scale supervised relevance data (*i.e.*, query-document pairs). Neural rankers can be categorized into two use-cases: (1) first-stage retrievers and (2) re-rankers. Examples of neural first-stage retrievers include DPR [27], ANCE [7], ColBERT [28], and Condenser [29], while examples of neural re-rankers include MonoBERT and DuoBERT [30]. First-stage retrievers are optimized for faster inference and are expected to achieve higher recall rates in finding relevant documents. Conversely, re-rankers aim for higher precision in identifying the top relevant documents at the cost of computational overheads. Recently, multi-stage rankers, built with combining neural first-stage retrievers followed by neural re-rankers, have shown significant performance advantages over traditional methods like BM25 [31] and learning-to-rank frameworks.

2.6.1 Supervised Neural Dense Retrievers

Firstly, dense retrievers for neural first-stage retrieval undergo extensive pretraining and fine-tuning using large-scale training data, resulting in competitive and often superior performance compared to traditional baselines. These retrievers utilize pretrained language models like BERT [32] to encode both queries and documents into dense embeddings. An efficiency-focused dual-encoder architecture is employed, allowing for separate or a shared encoder for encoding queries and documents. The dense embeddings of documents can be pre-computed and stored offline (referred to as an index), while for a given online query, the query dense embedding is inferred real-time and a Maximum inner-product search (MIPS) tool, such as FAISS [33] or ScaNN [34], is utilized for returning top-k relevant documents.

During the early stages, fine-tuning BERT-based dense retrievers on vast amount of relevance pairs, such as MS-MARCO [2] question-answer pairs, yielded impressive in-domain performance and competitive out-of-domain performance (*e.g.*, DPR and ANCE). Subsequent architectural refinements, such as ColBERT’s contextualized late-interaction [28] and Condensor’s conditioned LM prediction on dense representations [29], further improved effectiveness and efficiency. Overall, prior research on dense retrievers emphasizes their competitive edge in enhancing high recall and achieving faster efficiency.

Recent progress in unsupervised pretraining on large amounts of data emphasizes that increasing the size of pretraining results in improved performance. For instance, Contriever, contrastive pretrained on extensive unsupervised data like Wikipedia and CCNet [35], followed by sequential pretraining on supervised data like MS-MARCO, demonstrates significant performance improvements across standard question answering benchmarks and competitive performance in the transfer learning setting. However, challenges persist in generalizing dense retriever performances across out-of-domain datasets [1]. To address this, orthogonal research introduces GTR [36], highlighting the potential of scaling up the pretrained language model (dual-encoders) sizes with multi-stage training for improved generalizability across out-of-domain retrieval datasets.

2.6.2 Supervised Neural Dense Re-rankers

Secondly, cross-encoders used in neural re-rankers, pretrained on large-scale relevance pairs, exhibit significantly enhanced performance gains across both in-domain and out-of-domain datasets. The effectiveness of cross-encoders lies in their ability to leverage cross-interactions between contextualized tokens from the query and document. A basic cross-encoder architecture comprises a pretrained language model like BERT, followed by a linear layer that maps the hidden representation of the last layer’s first token (*e.g.*, [CLS]) to a relevance score. Concatenating a query and a candidate document into a unified text representation with special tokens, the cross-encoder calculates a relevance score, facilitating the sorting of top-k candidate documents for a given query. The first proposed cross-encoder, known as MonoBERT [30], utilizes pretrained BERT and fine-tunes it using MS-MARCO training pairs. Subsequently, various alternative pretrained language models, such as RoBERTa [37] and ELECTRA [38] have been explored as replacement base models, leading to further performance enhancements.

2.7 Weakly Supervised Training Neural Rankers

Fine-tuning with weak supervision enhances the retrieval performance in the absence of human labels, relying on a provided set of real queries. Previous research has focused on selecting top-k weak positive candidates and challenging hard negative candidates efficiently. However, a common oversight among practitioners has been improving ad-hoc retrieval performance in isolation, neglecting downstream tasks or systems. In actual applications, prioritizing a retriever aware of downstream systems holds greater significance than a standalone retriever. Therefore, the key distinction among various weakly supervised training algorithms lies in the methods used to obtain pseudo-labels for training neural rankers.

2.7.1 Relevance Pseudo Labeling with Weak Annotators

Several weak-supervision and semi-supervision approaches have been suggested to enhance the relevance labeling quality for training neural rankers. Early methods relied on first-stage retrievers like BM25 to provide weak positive candidates for training neural rankers [39]. Later, more advanced techniques emerged to improve the existing weak-supervision approaches. For instance, GPL [40] introduced a pseudo labeling method where a pretrained cross-encoder offers soft labels for synthetic queries. Another approach, BERT-PR [41], employs an ensemble technique to aggregate labels from various weak labeling functions, aiming to enhance label quality.

Contrary to these approaches, FWL [42] introduces an out-of-the-box annotator system (Gaussian process) as a teacher to estimate weak label quality (fidelity). The authors first fine-tune a learning-to-rank model using weak labels and then continually fine-tune it with fidelity-weighted samples, leading to

significant improvements over strong baselines. These prior efforts underscore the importance of selecting high-quality positive candidates for fine-tuning neural rankers.

2.7.2 Relevance Pseudo Labeling with LLM Annotators

Recent advancements in LLMs have sparked numerous studies exploring the use of LLMs as "**assessors**" and "**annotators**." By carefully crafting prompts, LLMs can be used to perform large-scale automatic annotations or evaluations. Recent research has shown that LLMs, when used as annotators, can predict relevance labels with greater accuracy than third-party human annotators [43]. This development has enabled the effective fine-tuning of neural ranking models with reduced reliance on human judgments.

In recent years, benchmarks and evaluation methods have evolved significantly, as the standard benchmarks, evaluation scripts, and metrics are no longer sufficient to measure the performance of advanced LLMs. Human judgment is often needed to compare outputs from different LLMs or to evaluate an LLM's output against ground truth. However, due to the cost and scalability issues associated with human evaluations, there is a growing interest in automated model-based evaluation. Powerful LLMs like GPT-4o or LLAMA3-70B-instruct can be prompted to compare an LLM's output against a reference. Studies have shown that the judgments of these stronger models often closely align with human evaluations.

2.7.3 Relevance Pseudo Labeling in RAG

Retrievers have become an essential component in RAG applications. The non-parametric nature of RAG allows for flexibility in choosing and plugging-in various retrieval or ranking models to improve the quality of retrieve-then-generate systems. However, even the state-of-the-art neural rankers often struggle to improve the quality of the LLM generated responses, despite the relevant evidence being available in external data source. These challenges stem from the misalignment between the tasks of the ranker and the LLM. Both are trained on different data sources with distinct training objectives. The concept of relevance for a neural ranker in a RAG setting differs from that in a traditional ad-hoc search optimization. Therefore, fine-tuning a pretrained neural ranker to align with the LLM is essential for better RAG performance.

Earlier research has focused on training retrievers and LLMs jointly to enhance the alignment between the two. For instance, the Atlas [44] model was pretrained on a large corpus to handle knowledge-intensive tasks in a few-shot learning setting. Atlas uses Contriever [45] as its retriever and the T5 [46] sequence-to-sequence model with a fusion-in-decoder [47] modification as its LLM. Each evidence document is encoded separately, and cross-attention is applied at the decoder to improve scalability across multiple documents. The authors introduced four loss functions—Attention Distillation (ADist), End-to-End Training of Multi-Document Reader and Retriever (EMDR2), Perplexity Distillation (PDist), and Leave-One-Out Perplexity Distillation (LOOP)—to train the joint system more effectively. These loss functions aim to provide feedback from the generator to the retriever, improving their alignment.

Recent RAG optimization efforts have shifted towards freezing the LLM and focusing on training the retriever component, largely due to the black-box nature of many closed-source LLMs (such as OpenAI's GPT-4, Anthropic's Claude, and Google's Gemini 1.5). These approaches rely on direct and indirect supervision from the LLMs, using annotations or attention-scores. AAR [48] is one of the early works proposing a RAG system where retrievers function as flexible, generic plug-ins. It combines ground-truth passages with top-k passages ranked by the LLM's average cross-attention scores to create positive training pairs, and follows ANCE [7] to sample negative training pairs for retriever training. The AAR, with a fine-tuned retriever, demonstrates improved zero-shot generalization across various LLMs.

REPLUG [9] is another recent advancement aimed at enhancing the alignment between the retriever and generator by training the retriever with a distillation loss. During the training phase, REPLUG calculates the retrieval likelihood scores from the retriever component alongside the language model likelihood scores by passing each document in conjunction with the query to the LLM, which yields average perplexity scores corresponding to the ground-truth answer tokens. The retriever is then trained using the KL divergence loss to minimize the difference between these two score distributions (retrieval and language model likelihoods). For inference, REPLUG performs similar individual query-document passes to the LLM and aggregates the output probabilities from these different passes. This distillation-based alignment better tunes the retriever to align with LLM preferences.

Building on REPLUG, a subsequent work called ARL2 [10] advances the alignment strategy for closed-source (black-box) LLMs. This study emphasizes the alignment between these opaque LLMs and retrievers. The authors introduce a self-guided adaptive relevance labeling technique that generates training data to fine-tune the retriever. They utilize the LLMs to annotate evidence documents, thereby curating high-quality training data. ARL2 demonstrates superior performance compared to REPLUG on evaluation benchmarks, thanks to the high quality of its pseudo label annotations using direct LLM prompting.

In a similar vein, W-RAG [49] presents a weakly supervised method for fine-tuning a neural ranker by leveraging the ranking capabilities of LLMs to generate the training data based on the assessments of the top- k results from BM25. However, this work does not address multi-document question-answering systems explicitly. Another related study by Ferguson et al. [50] emphasizes identifying relevant documents based on their utility in deriving answers. The authors argue that various pathways (sets of document sequences) can lead to an answer, and it is essential for the retriever to supply these pathways to the LLM to facilitate effective extraction and reasoning for generating responses.

In summary, the approaches discussed above utilize weak supervision from downstream LLMs via annotations, attention scores, or perplexity scores to train retrievers for better alignment with these models. However, they share several common shortcomings: (1) many require passing each query-document pair to the LLM for annotations, which is prohibitively costly; (2) most do not tailor their training towards multi-document question answering, resulting in evaluations primarily based on single-document benchmarks; (3) many rely on standard negative mining methods, which can introduce noise due to the imperfect quality of positive annotations, ultimately limiting the end-to-end performance; and (4) the generated pseudo labels do not strongly reflect the LLM preferences. I specifically address the fourth limitation, noting that previous works often fail to accurately prompt the LLM for precise task-specific pseudo-labels — specifically, grounded evidence documents needed for answer generation. Therefore, in Section 5.4, I propose the **TFT-RAG** method, which employs a self-referencing technique to enhance the quality of task-specific labels for fine-tuning the neural ranker.

Additionally, my **TFT-RAG** approach offers several advantages over the previous works. It simultaneously passes the query along with the top- k evidence documents to the LLM to obtain annotations, which is more cost-effective and allows the LLM to assess the groundedness of information across all documents. Furthermore, I introduce a prompting-based method for hard negative mining to minimize the noise in the generation of negative training pairs. Finally, I conduct evaluations on a comprehensive RAG benchmark, known as CRAG [51], to demonstrate the effectiveness of **TFT-RAG** in multi-document question answering tasks. In Section 5.4, I will provide a detailed discussion of the motivation and implementation.

2.8 Unsupervised Training Neural Rankers

Despite the remarkable ranking performances demonstrated by recent pretrained 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* [52] or *Domain Divergence* [53]. Domain-Shift has been a subject of exploration for decades, including recent investigations in domain adaptation [54]. Traditionally, it is assumed that the source and target domains share samples drawn from the same distribution. Previous studies have addressed this issue by quantifying domain divergence through various measures, such as geometric measures, information-theoretic measures, and higher-order measures [53]. 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 [55, 56] or pretraining a zero-shot ranker. On the other hand, data selection assumes that not all samples contribute equally to domain representation [57], 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 pretraining. In chapter 4, my thesis research centers on the issue of improper representation of the target domain leading to diminished performances in neural rankers and proposes an effective and robust solution.

2.8.1 Neural Query Expansion

The topics of query expansion and Pseudo-Relevance Feedback (PRF) for reranking have been explored for many decades [58]. Multiple works have been formulated to use lexical and semantic PRF information to expand the query in the past, such as Rocchio’s algorithm [58], RM3 [59], and Axiomatic matching [60]. Particularly in the challenging application of Cross-lingual Information Retrieval (CLIR) setting, there has been prior work on selecting inter- and intra-language PRF [61] and aligning token representation for query expansion [62]. Since the beginning of applying neural word embedding to capture semantic nature of the terms and sentences, more works have been explored to utilize PRF and expand queries in monolingual [63] and cross-lingual settings [64].

Lately, multiple neural rerankers were proposed by contextualizing the token and text representations [28, 65, 66, 67] using pretrained language models (*e.g.* BERT [32]). Although there has been transformer-based PRF approaches in the past that took advantage of multiple token representation [68, 69], my thesis research in Section 3.4 of chapter 3 narrows down the scope to single text representation of query and document for reranking.

My approach to the problem identified in Section 3.4 is inspired by ANCE-PRF [8], which is another recent monolingual neural reranker that runs on top of ANCE [7], a dense retrieval framework. They represent the query and document as contextualized embeddings and then apply a distance measure to generate a relevance score for each query-document pair. They modify the query encoder by appending multiple feedback documents following query text. However, they did not consider the nature of long document text, which is the general use case in most CLIR problems. Typically, only one feedback document barely fits in a query encoder along with query text. Therefore, my approach extends ANCE-PRF to fit more than one feedback document to allow query-PRF document cross-attention without scarifying the context available in a each long document.

2.8.2 Synthetic IR Data Generation

The increasing power of Large Language Models (LLMs) 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 [40], InPars [4], InPars-v2 [70], DocGen-RL [5], GenQ [1], and Promptagator [6]. 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[40], for instance, combines a T5-based [46] 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 [71]. GenQ, on the other hand, fine-tunes TAS-B [72] 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 can essentially be avoided. These methods fine-tuned rankers using large-scale synthetic data, ranging from 100k to 1M examples. In contrast, my thesis research in Section 4.3 of chapter 4 centers on the challenge of improper data sampling and argues 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.

2.9 LLM based Neural Rankers

Recent advancements in large language models (LLMs) have significantly enhanced tasks related to both natural language understanding and generation. Distinction between LLMs and pretrained language models (PLMs) or supervised neural rankers can be examined from various perspectives, including training size, model size, and model architecture. However, for the purpose of this thesis report, I define LLMs as language models capable of following instructions with zero-shot prompting and trained on large-scale unsupervised corpora. While the previous sections explored using LLMs to train neural rankers (such as synthetic data generation and knowledge distillation), this section will delve into prior approaches that effectively utilize LLMs directly for ranking purposes. The remarkable performances achieved by these LLMs stem from extensive pretraining on unstructured corpora. Prior works demonstrate a strong positive correlation between model scaling and extensive pretraining with performance improvements [36]. In addition, LLMs employing **sequence-to-sequence** architectures pretrained with auto-regressive objectives offer several advantages, including zero-shot prompting, the ability to follow instructions, learn from in-context demonstrations (examples), and handle larger input context compared to encoder-based **sequence-**

prediction models like BERT.

2.9.1 LLM based Neural Dense Retrievers

Recent findings highlight the substantial performance gains achieved through scaling up pretraining data and model size. GTR [36] highlights that the potential of scaling up the pretrained language model sizes with multi-stage training for improved generalizability across out-of-domain retrieval datasets. Very recently, RepLLaMA [73], a relevance task fine-tuned LLaMA [74] on MS-MARCO training pairs, showcase state-of-the-art in-domain performance and competitive out-of-domain generalization. Nevertheless, LLM-based neural dense retrievers such as RepLLaMA present challenges due to the significant computational power needed for indexing large-scale corpora and the high latency observed during query inference. These challenges impede the direct development of neural dense retrievers using LLMs.

2.9.2 LLM based Neural Re-rankers

Because of their computational efficiency, LLMs have been widely embraced as re-ranking in comparison to first-stage retrieval. Given the substantial existing research on generative LLMs for re-ranking, it is convenient to categorize them into three groups based on how they rank documents: point-wise, pair-wise, and list-wise. A point-wise ranker generates a relevance score for a query-document pair, treating it as a classification or regression task. A pair-wise ranker distinguishes a highly relevant query-document pair from another pair. Lastly, a list-wise ranker takes a query and a set of candidate documents, returning a reordered list of documents based on their relevance to the query.

2.9.2.1 Point-wise LLM Re-rankers

LLMs are designed for sequence-to-sequence generative modeling, while IR practitioners have succeeded on extracting a relevance score to re-rank a set of top documents for a given query. While a cross-encoder employs an encoder-only architecture with bidirectional attention mechanisms, generative models come in either decoder-only or encoder-decoder architecture trained with causal auto-regressive generative objectives. Straightforward adaptation of generative models for point-wise neural re-ranking is not trivial, thus prompting prior research to propose three key techniques: (1) query likelihood modeling, (2) interpreting the underlying logits of target words as relevance probabilities, and (3) utilizing linear score projection. Next sections will discuss the existing point-wise LLM re-rankers based on the aforementioned distinction.

2.9.2.1.1 Query Likelihood Modeling for LLM Re-ranking

In the past, query likelihood has been extensively utilized for re-ranking, assessing the probability of generating a query from a document as a measure of query-document relevance. UPR [75] was the first to propose leveraging LLMs for ranking tasks based on query likelihood. In this approach, given a document, an LLM is prompted to generate a query using this below prompt template,

Please write a question based on this passage : $\{d\}$

where d is a candidate document text. The likelihood between the generated query and the actual query is computed as a re-ranking score. Another recent work [76] conducts a comprehensive benchmark of

available LLMs in a zero-shot manner. The authors also emphasize that instruction fine-tuned LLMs perform less effectively than zero-shot LLMs for re-ranking tasks.

2.9.2.1.2 Interpreting Target Word Logits for LLM Re-ranking

Query likelihood models have limitations due to the assumptions that they make. MonoT5 was introduced as an encoder-decoder generative model designed for ranking tasks. The authors argue that instead of training a cross-encoder with a ranking objective, a sequence-to-sequence model can be trained with a next token prediction objective to generate a relevance label as a target word. MonoT5 was pretrained using MS-MARCO training pairs with a specific prompt template,

Query : $\{q\}$ Document : $\{d\}$ Relevant :

where q and d are query and candidate document text. The model learned to generate "true" or "false" target words based on whether the document was relevant to the query or not. During inference, the logit corresponding to the "true" or "false" token was leveraged to re-rank the list of candidate documents. MonoT5-3B emerged as a robust and competitive state-of-the-art neural re-ranker compared to rerankers based on LLaMA [74] and GPT4 [77].

2.9.2.1.3 Linear Projecting into Scores for LLM Re-ranking

Generative models can also be utilized for neural re-ranking in a manner similar to encoder-only models by harnessing the hidden representations from the decoder component. RankLLaMA adopts the same linear score projection approach to LLaMA by inputting a concatenated form of the query and candidate document text with special tokens. Unlike cross-encoders, which utilize the hidden representation of the first special token, RankLLaMA utilizes the hidden representation of the last special token due to the functionality of auto-regressive language generation. As RankLLaMA undergoes fine-tuning on MS-MARCO, it surpasses the performance of zero-shot GPT3.5 on standard benchmarks [78]. Nevertheless, zero-shot GPT4 (RankGPT4) remains at the forefront of state-of-the-art neural re-rankers [78].

2.9.2.2 Pair-wise LLM Re-rankers

Lately, pair-wise ranking PRP [79] is proposed to address the challenges found in both point-wise and list-wise ranking establishing a common ground between both ranking setting. PRP only takes input as query and a pair of documents to rank. Then progressively it compares all the candidate documents in a pair-wise fashion. PRP achieves competitive results to zero-shot GPT4 (RankGPT4) [78] and often surpassing.

2.9.2.3 List-wise LLM Re-rankers

LLMs demonstrate exceptional proficiency in following instructions, resulting in strong generative and reasoning capabilities. Consequently, a natural way of re-ranking a list of candidate documents based on a query involves prompting an LLM with instructions. RankGPT4 [78] and LRL [80] were introduced as a straightforward yet highly effective solution to achieve state-of-the-art neural re-ranking performance across various standard ranking benchmarks. By simply instructing GPT4 with a query and a list of candidate document texts along with their corresponding IDs, RankGPT4 can generate an ordered list of document IDs, with the order determined by the relevance of the candidate documents to the query.

The benefits of this LLM prompting approach include its zero-shot applicability (*i.e.*, no fine-tuning or in-context demonstrations required), largely reduced number of LLM calls, and utilization of the LLM’s reasoning ability to compare multiple documents simultaneously for relevance to a query.

Even though LLMs have been trained with next token prediction auto-regressive tasks, they excel in ranking tasks without any additional pretraining or fine-tuning. Enormous amount of pretraining data with vast world semantics and knowledge help to better model the relevance prediction in the ranking task. Moreover, there has been several works follow the knowledge-distillation fine-tuning of RankGPT4 into comparatively smaller and open-source LLMs, such as RankVicunna [81] and RankZephyr [82].

I have provided extensive related work on domain adaptation and neural ranking models in this Chapter. The key gaps identified in these studies include inadequate feature representations and the reliance on significant human involvement during the domain adaptation. These limitations prevent neural rankers from efficiently adapting to diverse vertical applications across different domains, languages, and tasks. Previous research has not adequately explored the proper representation of queries, tasks, and domains. Furthermore, earlier methods do not capitalize on large-scale resources and often require retraining models, which incurs higher costs and relies heavily on human annotations. Therefore, addressing the aforementioned challenges can effectively narrow down the performance gap in target adaptation across domains, languages, and tasks.

In Chapter 3, I focus on improving query representation for effective domain adaptation without the need for model retraining, using input modifications at inference time. Queries reflect the user needs and intents, so enhancing the query representation across domains can significantly boost ranking performance. Previous works have shown a gap in achieving optimal query representation using available target data, such as pseudo-relevance feedback for query expansion. Many recent approaches still rely on retraining the neural ranking models to improve query expansion or enrichment. To address this, I propose approaches **NCLPRF** and **GenQREnsemble-RF** in Sections 3.4 and 3.5, which focus on test time domain adaptation via effective pseudo-relevance feedback, closing the gap in target domain representation.

In Chapter 4, I emphasize efficient fine-tuning of neural rankers with minimal human intervention. By leveraging LLMs, I generate target representative queries across various domains and languages to facilitate unsupervised fine-tuning of neural ranking models. Prior approaches often suffer from overfitting during the fine-tuning due to inadequate target representation. Additionally, they involve costly LLM calls and resource-intensive model fine-tuning. Furthermore, adapting neural rankers for multiple languages fails to consistently achieve robust performance comparable to supervised fine-tuning. To address these challenges, I introduce approaches **DUQGen** and **mDUQGen** in Sections 4.3 and 4.4, which focus on diversifying synthetic query generation across domains and languages for more effective fine-tuning of neural rankers.

In Chapter 5, I focus on task-specific fine-tuning of neural rankers for downstream applications. Since the concept of "task" varies across retrieval systems, ensuring appropriate task representation across domains helps bridge the domain representation gap. Previous research on weakly supervised fine-tuning of neural rankers and ranker alignment in RAG has not fully captured the essence of task, particularly regarding pseudo relevance labels. Additionally, the methods used to generate these labels often fall

short in addressing specific downstream needs, such as multi-document groundedness in RAG and event-awareness in event extraction system. Therefore, I introduce approaches **C3**, **IE-IR**, and **TFT-RAG** in Sections 5.2, 5.3, and 5.4, aimed at improving pseudo label generation, target tasks representation, and task-specific fine-tuning of neural rankers through effective weakly supervised techniques.

In the next three chapters, I present effective solutions to address both common and specific challenges found in the existing literature. Specifically, these chapters introduce seven novel approaches across three **RQs** aimed at capturing target representation with minimal or no human supervision, addressing the performance gap in prior work. Finally, in Chapter 6, I provide a summary of results to clearly highlight the key contributions of my thesis research.

3 Unsupervised Test Time Query Representation

Queries represent both the user’s information need and intent. Enhancing query representation to improve the search experience has been a major focus in the information retrieval community. Traditional search systems typically represent queries as bag-of-words vectors [31], while more recent neural search systems use dense embeddings. Bag-of-words representations rely on lexical tokens matching, whereas dense embeddings capture semantic similarities. Hybrid systems combine these approaches, first using bag-of-words matches to retrieve high-recall candidates, and then applying semantic similarity scores to improve precision for the top-ranked results. As a result, optimizing query representation has been a long-standing challenge in search technology.

Adapting search and ranking systems to multiple applications presents several challenges. These systems are used widely in various contexts, including search, recommendation, question answering, fact-checking, and more, and span multiple domains such as web (*e.g.*, Google and Bing), e-commerce (*e.g.*, Amazon and eBay), email (*e.g.*, Gmail and Outlook), entertainment (*e.g.*, Spotify and Netflix), news (*e.g.*, CNN and BBC), and finance (*e.g.*, Bloomberg and Robinhood). This broad horizontal and vertical deployment makes it difficult to apply a single query representation method universally due to the varying user intents and query formats. Across these applications and domains, users seek different types of information—such as web links, answers to questions, evidence for claims, or personalized recommendations—and express their queries in forms like keywords, questions, claims, or interactions. Additionally, queries can be multilingual or multimodal (including images and audio). Table 3.1 illustrates these query variations across diverse applications. Consequently, effective techniques should adapt query representation to the specific needs of each application and domain to enhance search quality and user experience.

Capturing query distributions is a challenge for search and ranking models. Neural ranking models, compared to traditional search systems, more effectively capture query distributions by training on query-document relevance pairs specific to the target domain. Accurately modeling the query distribution improves user query representation across different domains. In each domain, queries and documents follow distinct data distributions. While it is often easier to capture document data distribution since target data is usually accessible, effectively capturing query data distribution requires real user queries, such as query logs from Amazon product searches. In a domain adaptation scenario, a ranking model trained on a source domain using its query distribution may struggle when applied to a related but different target domain due to shifts in query distribution. Therefore, neural ranking models need access to both user query collections and human relevance judgments to be properly trained for the target domain.

Neural rankers need to be retrained to better capture query representations in the target domain. These models require large-scale training data specific to the target domain to effectively learn user and domain-specific features. Previous research has shown that pretrained language models exhibit strong transfer learning performance across various domains and tasks. As a result, several studies have leveraged large-scale, task-specific datasets (such as relevance tasks) to pretrain language models for ranking purposes. One notable example is MS-MARCO [2], which contains 20 million question-answer pairs with human relevance judgments. This dataset is often used to pretrain language models for ranking tasks, which are then transferred in a zero-shot manner to other domains and retrieval tasks [1], such as biomedical retrieval, news retrieval, and fact-checking. However, the performance of source pretrained models on target domains is often limited and can sometimes fail significantly. This is primarily due to differences in query representation and mismatches in query distribution between the source and target domains. For instance, MS-MARCO queries are usually short questions, making it difficult for pretrained

rankers to adapt well to other domains like biomedical or finance, or to tasks like fact-checking and argument retrieval. As a result, the "one-size-fits-all" approach to neural rankers often proves inadequate for different domains and datasets.

Modifying the input query is the most effective and efficient way to adapt user query representation across domains. Various efficient methods have been employed to improve query representation, such as query expansion, query enrichment, and query reformulation. When domain divergence occurs, either (1) the query representation must be adjusted or (2) the neural ranking model must be retrained. Retraining is more difficult due to the need for human-labeled data and the costs involved in updating the model. Consequently, most previous approaches focus on modifying the input query, particularly during inference, which falls under *test-time domain adaptation*. Query expansion and enrichment adjust the query representation, but in different representation spaces, while query reformulation rewrites the input query entirely at inference. Since query reformulation can introduce errors that may alter user intent or the query topic, query expansion (or enrichment) is more commonly used in many applications. The next section will therefore explore query expansion in detail.

3.1 Query Expansion (QE)

Query expansion enhances query representation by adding additional terms to improve search results. This can be done at the token level or in the embedding space to bridge the gap between the user's query and relevant documents. Query expansion offers two key benefits: (1) it addresses the vocabulary mismatch between the query and documents, and (2) it incorporates target domain information from relevant documents into the query representation. These factors can significantly improve search quality. In this chapter, the focus is on the second benefit—incorporating target domain information through query expansion. By gathering relevant terms from the target domain, query expansion aids effective domain adaptation without requiring model retraining. For this reason, query expansion has been widely used in both research and industrial applications for many years, particularly in large-scale commercial systems. Examples of query expansion techniques include synonym expansion, relevance feedback, and pseudo-relevance feedback. The most common approach is pseudo-relevance feedback, which will be detailed in the following section.

3.2 Pseudo Relevance Feedback (PRF)

Pseudo Relevance Feedback (PRF) is a type of Relevance Feedback (RF) that assumes the top-k results from an initial search are relevant and uses them to refine the search results. Traditional relevance feedback involves a user manually identifying relevant documents, which are then fed back into the system to improve the query representation or search outcome. However, due to the effort required from users, automatic approaches like PRF are typically preferred. PRF operates by making two search calls: the first retrieves the top-k documents, which are automatically assumed to be relevant. Expansion terms are then extracted from these documents to enhance the input query representation. The second search call retrieves the final set of top-k documents for the expanded input query. Since the expansion terms are drawn from the top-ranked target domain documents, they help improve the query representation, leading to better search performance.

Many retrieval systems incorporate PRF to handle ambiguous queries. Search queries can range from ambiguous and vague to complex or straightforward, with ambiguous queries posing the greatest challenge. It is crucial for a search system to meet user information needs, even when their queries are

unclear. PRF is an effective method for addressing search system failures with ambiguous queries. It enhances search results by automatically including relevant terms related to the query context, without requiring any user feedback. This helps significantly improve search performance, especially in the case of ambiguous queries.

However, PRF comes with certain challenges, such as computational overhead and query topic drift. It is important to consider these issues before integrating PRF into a search system to avoid poor performance. First, since PRF requires two search calls and the extraction of expansion terms, this added complexity can increase latency in real-time systems. Second, assuming that the top-k search results are relevant can introduce noise, potentially distorting the query representation. Of these challenges, topic drift is particularly problematic—if the top-ranked documents are irrelevant, the query expansion process may introduce irrelevant or noisy terms, significantly degrading search quality. Several strategies can help mitigate the risk of topic drift, such as limiting the number of feedback documents, filtering terms, applying term weighting (*e.g.*, TF-IDF or BM25), and using semantic models for query expansion. By effectively addressing these concerns, PRF can be incorporated into a search system to better handle ambiguous queries and enhance recall. In the next section, I will propose two effective PRF approaches, focusing on incorporating target domain information through query representation.

Query Type	Applications
question	question-answering, duplicate question retrieval
headline	news retrieval, tweet retrieval
entity-oriented query	entity retrieval
argument	argument retrieval
title (article)	citation prediction
claim	fact checking
domain-specific query	bio-medical and financial domains
cross-language query	cross-lingual information retrieval

Table 3.1: Examples of query types used across different retrieval applications.

3.3 Overview of Proposed Query Representation Approaches

In the following section of this chapter, I propose two approaches to effectively represent user queries in a target domain. Specifically, I address the primary research question (**RQ1**): *Can adapting query representation with domain information improve ranking performance?* My contributions to **RQ1** are twofold: (1) **NCLPRF**: Learning to Enrich Query Representation with Pseudo-Relevance Feedback for Cross-lingual Retrieval [83], published as a short paper in **SIGIR 2022**, and (2) **GenQREnsemble-RF**: Generative Query Reformulation Using Ensemble Prompting, Document Fusion, and Relevance Feedback [84] in **Arxiv 2024**. The preliminary study for GenQREnsemble-RF, namely GenQREnsemble [85], is published in ECIR 2024. I was the lead author of the NCLPRF project and collaborated on the GenQREnsemble-RF project, which was primarily led by Kaustubh Dhole. In the GenQREnsemble-RF project, my contributions were focused on prompt tuning and experiments related to integrating PRF into the GenQREnsemble framework. Additionally, I conducted experiments related to ranking and PRF, while the lead author handled experiments focused on LLMs. Next, I will compare and contrast these two approaches before providing detailed descriptions in the following sections.

Although both of my approaches enhance target query representation, they offer distinct advantages depending on the use case. These differences can be understood across three key dimensions: (1) query representation space, (2) context length limitations, and (3) the performance vs. latency trade-off. First, both NCLPRF and GenQREnsemble-RF fall under test-time domain adaptation (TTDA), meaning they modify the input query representation to better fit the target domain during inference, without requiring model retraining or weight updates. However, they differ in the space where the query modification occurs: NCLPRF modifies the query representation in the embedding space, while GenQREnsemble-RF operates in the lexical space.

Second, NCLPRF is constrained by the context length limitations of pretrained language models (*e.g.*, 512 tokens for BERT), whereas GenQREnsemble-RF benefits from a larger context length (*e.g.*, 4,000 tokens for Llama-2-7b-chat-hf).

Third, NCLPRF uses a smaller model (220M parameters), providing low latency but with limited performance, while GenQREnsemble-RF uses a much larger model (7B parameters), achieving higher performance but with significantly increased latency. Given these trade-offs across the three dimensions, practitioners can select the most suitable approach based on their specific use case. Next, I will provide a detailed discussion of my first approach: NCLPRF.

3.4 Learning to Enrich Query Representation

Incorporating pseudo-relevance feedback (PRF) into dense retrieval models to enhance query representation is typically done at the term level. To further boost retrieval effectiveness using target information, queries can be expanded with PRF during query time. Many methods, such as RM3 [59], follow the classic Rocchio algorithm [58], modifying queries with terms found in top-ranked documents. However, expanding—or more accurately, enriching—query representation at the embedding level remains largely underexplored. In this work, I focus on proposing query enrichment through embedding representation to address this gap.

Compared to monolingual retrieval, query enrichment can be especially effective in cross-lingual settings due to the larger representation gap between languages. Cross-lingual information retrieval (CLIR) aims to provide access to information across different languages. While recent pretrained multilingual language models have significantly improved various natural language tasks, including cross-lingual ad-hoc retrieval, PRF has not been extensively explored with neural CLIR models. The additional information provided by feedback documents in the target language can enhance query representation, aligning it more closely with relevant documents in the embedding space. Therefore, I explore the challenges of query enrichment in the CLIR context to propose an effective solution.

3.4.1 Problem Statement

Neural query enrichment for dense retrieval models presents several challenges. Simply expanding query terms has proven less effective than modifying query representations using feedback documents, as seen in approaches like ColBERT-PRF [68]. Additionally, incorporating long feedback documents poses a significant issue when working with dense retrieval models that rely on pretrained language models with limited context lengths (*e.g.*, ANCE-PRF [8]). Including more than a few feedback documents can result in information loss during the query enrichment process, and this problem is further exacerbated when the documents themselves are long.

Neural query enrichment in CLIR faces unique challenges due to cross-language transfer. Using PRF in CLIR is particularly difficult because it amplifies the vocabulary mismatch between the source language

query and PRF documents written in a foreign language. While sequence embeddings in pretrained transformer models have proven effective for monolingual retrieval [29], for CLIR, token representations across languages can diverge further without additional fine-tuning [86, 87]. Aligning query and document representations across languages is already complex, and enriching query representations in one language with text from another adds an extra layer of difficulty. This raises a natural question: can query representation also be enriched effectively for CLIR in dense retrieval? Therefore, it can be hypothesized that enriching query representations through PRF will improve cross-lingual retrieval effectiveness.

3.4.2 NCLPRF

To address these challenges, I propose training a joint encoder for both the query and feedback documents. Specifically, I introduce a new neural CLIR model, **NCLPRF**, which extends the capabilities of the previous state-of-the-art monolingual PRF model, ANCE-PRF [8]. Unlike ANCE-PRF, which is limited to encoding a fixed amount of feedback information, NCLPRF can encode multiple feedback documents, helping to bridge the language gap between queries and documents. NCLPRF aggregates representations that jointly encode the query alongside each feedback document, effectively narrowing the representation gap. The resulting enriched query representation encapsulates the user’s information need, conditioned by the feedback signals. While NCLPRF was originally designed for CLIR, it is flexible enough to be applied in monolingual or multilingual retrieval scenarios as well. The model is trained using weakly supervised cross-language queries and documents from Wikipedia, established through the wiki-links in CLIRMatrix [88]. Performance evaluations on three CLIR test collections in Chinese, Russian, and Persian show significant improvements over traditional and state-of-the-art neural CLIR baselines without requiring retraining across different datasets. Next, I will provide a detailed explanation of my approach, starting with a formal method description, followed by an overview of the experimental setup, and concluding with a discussion of the results and analysis.

3.4.3 Proposed Work

The method adapts the Siamese architecture [89] for measuring text similarity, known as dual-encoders in the neural IR context [25], such as ANCE [7] and DPR [27]. Two encoders generate dense contextualized vector representations of the input query and documents separately that summarize the semantics of the text. Figure 3.1 shows my model architecture.

3.4.3.1 Query representation

Query is encoded using a pretrained transformer multilingual language model, i.e., *universal encoder*. To provide the feedback documents as additional inputs, query and each feedback document form a paired input sequence separated by the separation token for the encoder (SEP). The resulting sequence representation $\mathbf{q}_k^{CLS}$ is the query representation contextualized on the feedback document d_k . Specifically the representation of the first token, i.e., the $\langle s \rangle$ or CLS token, from the last layer of the encoder in the input sequence is used as the representation of the query.

The additional context from the feedback document enriches the query representation over the original query representation $\mathbf{q}^{CLS}$ created based on the short text. Such modifications to the query representation align the original spirit of query expansion with PRF that adjusts queries with relevant terms. However, multiple feedback documents yield multiple query representations, requiring an aggregation function which I will discuss later in the section.

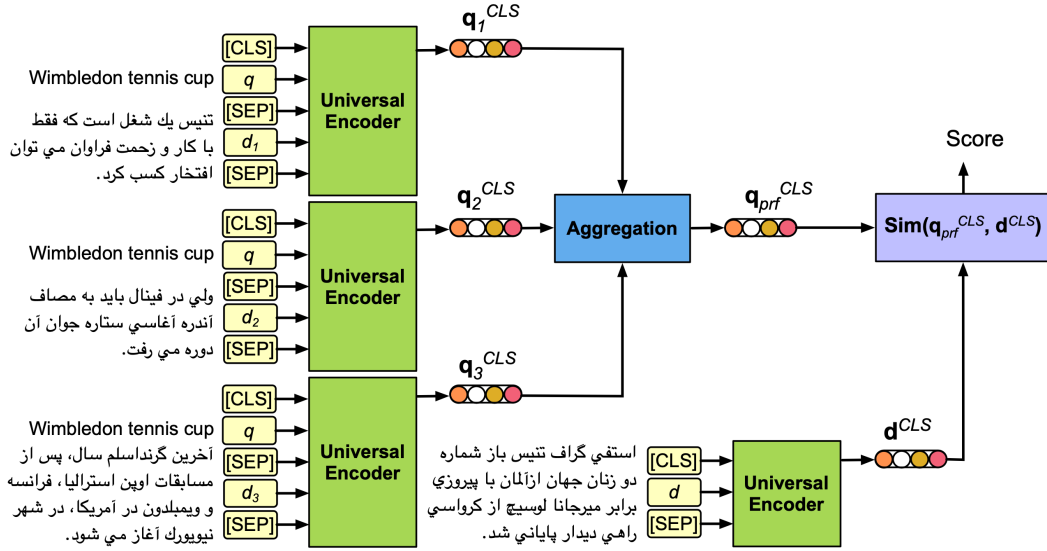


Figure 3.1: Architecture of NCLPRF ranker. A query is represented as the aggregation of N query-feedback document encodings. The reranking score of a candidate document is based on the similarity of its document encoding and the query encoding.

3.4.3.2 Document Representation

Candidate documents d_i are encoded using the same backbone network as the query encoder without the feedback information and denoted as d_i . The feedback information is only manipulating the queries as the candidate documents already contain enough context by themselves. Similarly to the query representation, the representation of the `CLS` token from the last layer of the encoder is used as the representation of the document. The shared encoders between query and documents benefit the query encoder from the document side when encoding the feedback documents. Furthermore, such a decoupling still admits offline encoding of the candidate documents similar to other denser retrieval models, allowing for approximated nearest neighbor search.

3.4.3.3 Vector Aggregation

Inside the query encoder, contextualized vectors corresponding to separate feedback documents have to be fused together to generate a single vector for the enriched query. Specifically, given a query representation q_k^{CLS} conditioned on feedback document d_k , I compute the aggregated vector q_{prf}^{CLS} as $\sum_k w_k q_k^{CLS}$, where $\sum_k w_k = 1$. Vector averaging is implemented as the aggregation method either by uniform weights, BM25 weights, or reciprocal rank (RR) weights (i.e., $w_k = \frac{1}{r_{w_k}}$ where r_{w_k} is the rank of feedback document d_k in the initial rank list). The comparison of each aggregation function is discussed in the results section.

3.4.3.4 Reranking framework

The overall reranking framework consists of query-feedback encoder with aggregation, document encoder, and a vector similarity function f between the representations to generate a reranking score for each candidate document given the query conditioned on the feedback documents. For simplicity, a dot product between the vectors is used as the similarity function. The final score for candidate document d_i can be

		Persian	Russian	Chinese
Train	query-document pairs	410,128	576,940	410,322
	avg query length	2	2	2
	avg document length	77	144	18
Dev	queries	118	118	118
	documents	486,486	4,721,064	646,305
	avg query length	6	5	20
	avg document length	377	208	621
Test	queries	100	62	100
	documents	166,774	16,716	308,832
	avg query length	4	3	17
	avg document length	395	269	525

Table 3.2: Train (CLIR-Matrix), dev (HC4), test (CLEF and NTCIR) collections statistics.

expressed as the following,

$$f(q, d_i) = \mathbf{q}_{prf}^{CLS} \cdot (\mathbf{d}^{CLS})^T \quad (3.1)$$

I train the model with negative log-likelihood loss in a point-wise fashion where the negative documents for each query are randomly sampled within the positive documents in the training batch. Based on my pilot study and given the quick prototyping and resource limitations, one negative document per query was used during training. This concludes the details for my approach, and next I will explain my experimental setup.

3.4.4 Experimental Setup

In this Section, I give more details about my experimental setup, especially the training, validation, evaluation, and the datasets used for each.

3.4.4.1 Dataset

I used multiple datasets and collections for training, validation, and testing. I primarily focused on high-resource languages in my evaluation: Persian, Russian, and Chinese. However, NCLPRF can still be applied to medium- and low-resource languages in a zero-shot transfer fashion. The statistics of the data are shown in Table 3.2. My datasets consist of English queries-foreign documents for three language pairs.

3.4.4.1.1 Training

My dual-encoder based transformer architecture requires large number of query-document pairs to train. Therefore, I used large scale bilingual query-document pairs from CLIRMatrix [88] on three language pairs. The dataset comes with automatic relevance annotations from judgements 0 to 6, where 0 is irrelevant and 6 is highly relevant. I selected the judgement 6 pairs for my baseline training. In addition, I pooled judgements 5 and 4 to provide feedback documents for each queries. In a typical PRF setting, feedback documents are selected from the initial retrieval result. However, I argue that using labeled

relevance documents for feedback signals during training is reasonable since top-k documents are also implicitly assumed relevant during inference.

3.4.4.1.2 Validation

To help picking the best performing model across the training steps and tune the free parameters, I used CLIR Common Crawl Collection (HC4) [90] as the development dataset. In this dataset, the exact same English topics are queried over 3 languages and relevant judgements are annotated for each. I used Patapsco [91] as my initial retrieval tool to query the English topic-title (queries were machine translated to cast for monolingual initial retrieval) for top-100 documents.

3.4.4.1.3 Testing

I reported NCLPRF performance on corresponding three languages from CLEF (Persian and Russian) [92] and NTCIR (Chinese) [93] collections. English topic-title is machine translated to query top-100 initial documents using Patapsco. Conventionally, PRF approaches operate on the same initial retrieval. However, I apply PRF on reranking in this study, and PRF on initial retrieval can be directly extended in future study.

3.4.4.2 Training and Evaluation

3.4.4.2.1 Implementation

I used PyTorch and HuggingFace APIs to train and evaluate my framework. For training, all the models trained with 160k query-document pairs with and without feedback document. XLM-R pretrained LM is used as my base transformer. Given a short amount of time for a quick implementation, I only completed runs on XLM-R large for Russian and XLM-R base for the other two languages. Since my goal is to fine-tune an XLM-R to transfer knowledge between source and target languages, I shared the model parameters between query and document encoders. The parameter sharing also promotes the query to get closer to the document representation with the help of feedback documents. To enforce the XLM-R to learn the cross-language relevance, I did not pass the XLM-R’s contextual output vector to a dense layer. I used batch size of 16 to train for 20 epochs with learning rate of $5e-6$. During inference time, I applied windowing method to cover the long document text. Each documents are chunked into 80 white-space tokens length with stride of 40 tokens. The document relevance score is determined by the maximum score across each chunk relevance scores. XLM-R models were trained on a single RTX 6000 GPU for 200k steps.

3.4.4.2.2 Evaluation

During evaluation, I used Patapsco [91] to pool top-100 documents per query, and then reranked the list using baselines and my models. I used nDCG@5 and nDCG@10 to measure the model reranking performances. My baselines are rerankers without leveraging PRF (BM25 with transformer-based reranker) and with leveraging PRF (RM3 with variable number of feedback documents 1-10, and ANCE-PRF). In the next Section, I will showcase my **NCLPRF** experimental results on evaluation datasets.

3.4.5 Results and Discussion

In this Section, I provide an overview of the experimental results and conduct an analysis of how PRF affects the re-ranking performance. First, I compare my **NCLPRF** approach against robust baseline

Query	Models	CLEF Persian		CLEF Russian		NTCIR Chinese	
		nDCG@5	nDCG@10	nDCG@5	nDCG@10	nDCG@5	nDCG@10
MT	BM25	0.404 (0.0%)	0.392 (0.4%)	0.327 (−6.3%)	0.336 (−9.0%)	0.270 (12.2%)	0.278 (11.5%)
	BM25 + RM3 (w/2 FD)	0.404 (−)	0.390 (−)	0.349 (−)	0.369 (−)	0.241 (−)	0.249 (−)
	BM25 + RM3 (Tuned)	0.423 (4.7%)	0.405 (3.9%)	0.358 (2.6%)	0.370 (0.1%)	0.278 (15.3%)	0.279 (11.9%)
Eng.	ANCE-PRF (XLM-R based)	0.424 (4.9%)	0.413 (6.0%)	0.369 (5.7%)	0.382 (3.4%)	0.281 (16.5%)	0.286 (14.9%)
	NCLPRF (w/o PRF)	0.405 (0.3%)	0.393 (0.7%)	0.327 (−6.3%)	0.345 (−6.6%)	0.272 (12.7%)	0.277 (11.0%)
	NCLPRF (w/2 FD)	0.459 (13.6%)	0.425 (9.0%)	0.374 (7.2%)	0.380 (2.9%)	0.295 (22.3%)	0.298 (19.5%)

Table 3.3: Comparison of NCLPRF against baselines. All models at the bottom half of the table rerank the top 100 documents retrieved by BM25 (without RM3). BM25 uses machine-translated queries and neural rerankers use the provided English queries. Percentages in parentheses are relative improvements over the BM25 + RM3 with 2 feedback documents.

models. Following that, I delve into the impact of the number of feedback documents in PRF on query representation.

3.4.5.1 Reranker Comparisons

The Table 3.3 shows the effectiveness of my models against the traditional baselines (BM25 + RM3) and neural baseline (ANCE-PRF). NCLPRF w/o FD (feedback document) input, performs similarly to BM25, implying that NCLPRF w/o FD only learns the lexical features present in the query and document text in the reranking setting. However, when the model sees one input feedback document (the ANCE-PRF row), it learns better query representation to find more relevant documents on top-10 reranked list. These improvements are substantial across all three languages on both metrics relative to RM3 (w/2 FD). In addition, when two feedback documents are inputted, NCLPRF w/2 FD further improves the scores against ANCE-PRF, which only uses one feedback document. More than one feedback document input provides more diverse context for the encoder to learn better query representation. This is observed by the relative improvements on nDCG@5 (ranging 7.2% to 22.3%) and nDCG@10 (ranging from 2.9% to 19.5%) scores against RM3 (w/2 FD). Even against a RM3 tuned with optimum number of feedback documents, NCLPRF w/2 FD shows a large margin of robust improvement.

The early work, ANCE-PRF is designed to contextualize query based on multiple feedback documents. Their transformer (*BERT*) encoder can only take maximum 512 token length, including query, feedback document, and special tokens. This imposes a limitation on their design choice in the presence of long document collection. It is infeasible to append more than one feedback document along with query into their encoder. To overcome this challenge, my design allows scaling up more than one long feedback document to contextualize the query. In comparison with ANCE-PRF, NCLPRF w/2 FD improves the nDCG@5 scores relatively by 8.5%, 1.4%. and 5.0% on all three collections.

3.4.5.2 Variations on the number of feedback documents

With the increasing number of feedback documents (i.e., depth, k), RM3 expands the query and reweights the candidate documents. However, semantic rerankers that leverage PRF documents aim for better query representation and contextualization. Increasing the PRF depth captures more diverse context from multiple top- k feedback documents and alleviates the impact of noisy signals [8] from a single feedback document.

To measure the performances across varying PRF depth, Figure 3.2a shows the consistent improvement to the nDCG@5 scores in NTCIR Chinese collection. But reciprocal rank weights are not improving

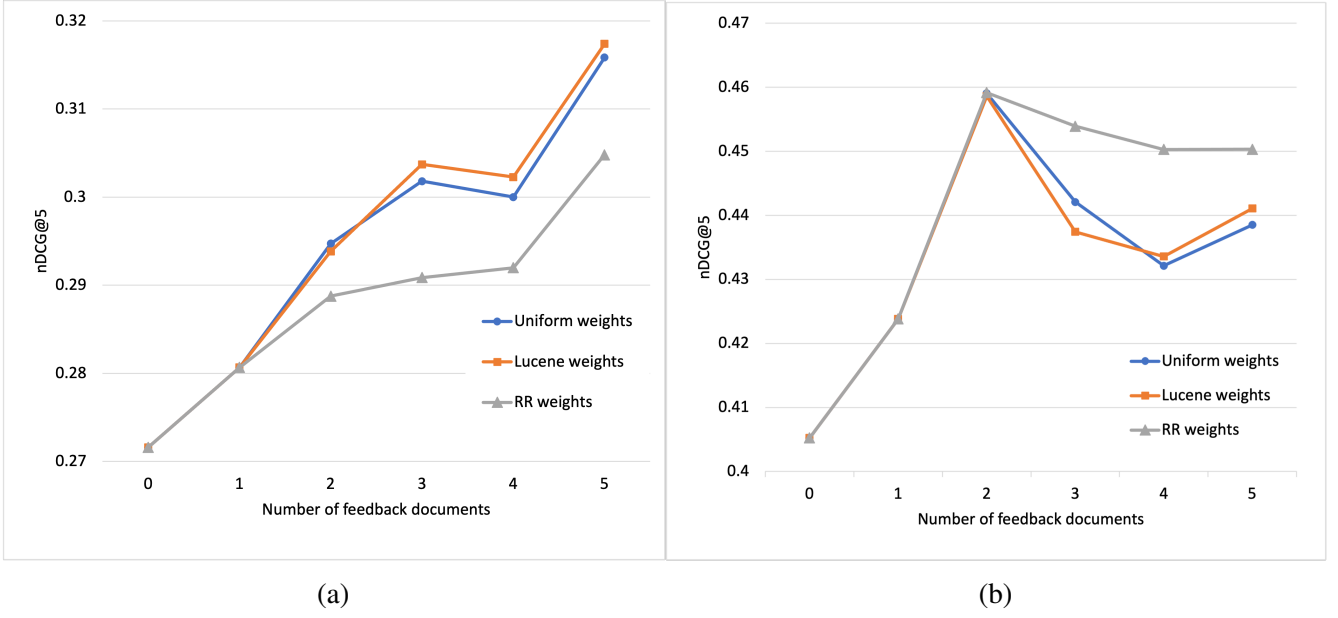


Figure 3.2: Different vector aggregations across PRF depths in (a) NTCIR Chinese and (b) CLEF Persian collections.

as much compared to uniform and Lucene weights. The performance drop at $k = 4$ documents the introduction of non-relevant document in the top- k list. However, reciprocal rank weights show a monotonical improvement across PRF depth. In contrary, Figure 3.2b shows that in CLEF Persian collection the performance improvements peaked at $k = 2$, and when k increases further, the nDCG@5 scores decrease surprisingly. This could be resulted from two reasons: presence of semantically dissimilar feedback documents and suboptimal vector aggregation function. The first reason origins from the fact that I used a neural reranker for an initial lexical-based retrieval system. A more typical practice is to use feedback given by a neural system to a neural system (*e.g.* ANCE-PRF). Therefore, semantically similar documents found in the top- k list provide feedback for a consequent semantic reranker. However, in this study I provide feedback from a lexical system to a semantic reranker. Thus, the feedback documents do not necessarily provide the relevant information to contextualize the query. The second reason is inherent to my simple design choice of vector aggregation. The weights I used to aggregate the vectors are not robust enough to underweight the semantically far feedback documents.

To overcome the problems discussed above in future, a simple solution would be to precede NCLPRF with a dense retrieval framework, which provides more diverse context information from top- k feedback documents. An operator to discard irrelevant documents or give negligible weights to those during aggregation is also worth investigating. To promote this, Figure 3.2 clearly shows that regardless of varying PRF depth and weighting schemes, NCLPRF outperforms ANCE-PRF, which only uses one long feedback document.

3.4.5.3 Variations of Vector Aggregations

To generate a single query representation vector from multiple query-feedback document pairs, I introduce a vector aggregation function. I experimented with three weighted average operators. Uniform and BM25 weights follow a similar pattern and close performances since top ranked documents are likely to have closer BM25 scores. In this weighting scheme, the models prone to semantically irrelevant documents appear on top ranks. However, from Figure 3.2 reciprocal rank weights curve does not show

large performance drop and establishes an evidence for potential robustness of NCLPRF to increasing PRF depth. In the next Section, I will summarize the experimental results and address any limitations in my approach.

3.4.6 Summary and Limitations

I presented a novel method for enriching query representation for CLIR, NCLPRF. The method uses the feedback documents and brings the query representation close to relevant documents. My approach is simple but effective in capturing the information from multiple long documents as compared to the previous work. I showed that NCLPRF outperforms both statistical and Neural CLIR baselines. Thus, NCLPRF effectively enhances the query representation not only for CLIR but also for monolingual and multilingual retrieval across various domains and retrieval tasks. NCLPRF often has certain limitations that might help towards the future extension or exploration.

NCLPRF has certain limitations due to suboptimal design choices that hinder its potential performance. By adopting stronger design strategies, its performance can be improved compared to current state-of-the-art (SOTA) retrieval models. Since NCLPRF was introduced in 2022, several elements—such as its choice of PRF sources, aggregation methods, and training objectives—are outdated compared to more recent SOTA techniques. For instance, leveraging lexical-based PRF sources and using a more advanced aggregation model could help mitigate the noise from non-relevant documents in initial retrieval results. Although NCLPRF was designed for both neural re-rankers and dense retrievers, its performance as a dense retriever may fall short of current SOTA models due to its weaker training objective, specifically pairwise negative log likelihood loss. Current dense retrievers are trained using contrastive learning over large datasets, and applying a similar training strategy to NCLPRF, while incorporating feedback documents into the query, could improve its performance. Hence, by integrating these advanced techniques, NCLPRF could achieve superior results compared to its counterparts. This concludes the discussion on my NCLPRF approach.

3.5 Unsupervised Ensemble Generative Prompting to Enrich Query Representation

Recently, the success of large language models (LLMs) [94, 95] has spurred the development of numerous query expansion (QE) techniques. Generative abilities of LLMs have been leveraged to create new queries [96] and generate valuable keywords to enhance users’ original queries [97]. Given the vast amount of text these models are exposed to during pretraining, prompting has emerged as a promising strategy for tapping into the knowledge embedded in LLMs to improve various downstream tasks [98], particularly for QE [99, 100].

Unlike training or few-shot learning, zero-shot prompting does not require any labeled examples. The key benefit of a zero-shot approach is its simplicity, as a standalone generative model can be prompted with a predefined template and the original query to expand or reformulate the query. Specifically, zero-shot query expansion (QE) can generate keywords by providing the user’s original query along with a natural language instruction that describes the query reformulation task, such as `Generate useful search terms for the given query: "List all the pizzerias in New York"`.

3.5.1 Problem Statement

However, this zero-shot prompting approach relies heavily on the specific instructions given in the prompt, leaving room for potential improvements. LLMs are known to exhibit significant performance differences depending on the prompts used [101, 102] and the generation settings [103]. Many natural language tasks have capitalized on this variation by ensembling multiple prompts or generating diverse reasoning paths [104, 105, 106]. Whether these improvements can be applied to tasks like query expansion (QE) remains unclear. It is therefore plausible to hypothesize that QE could benefit from prompt variation — an ensemble of zero-shot reformulators with varied paraphrastic instructions could generate diverse query expansions by approaching the input query from different perspectives.

3.5.2 GenQREnsemble-RF

To tackle the aforementioned challenge and build on the motivation, an ensemble-based generative prompt query expansion approach, called **GenQREnsemble-RF**, is introduced. I collaborated on the GenQREnsemble-RF project, which was primarily led by Kaustubh Dhole. In the GenQREnsemble-RF project, my contributions were focused on prompt tuning and experiments related to integrating PRF into the GenQREnsemble framework. Additionally, I conducted experiments related to ranking and PRF, while the lead author handled experiments focused on LLMs. **GenQREnsemble** [85] introduces an efficient method that leverages the variability (or instability) in LLM outputs, transforming it into an asset for generating diverse query expansion terms. By incorporating pseudo-relevance feedback (PRF) documents into the prompt, the generated expansion terms are enriched with target-specific information. The expanded query is then used to retrieve the top-k relevant documents from the index. When the LLM generates expansion terms without PRF, the terms tend to be more generalized to the query topic. However, with PRF, the terms become tailored and specialized to the target domain. This domain-specific adaptation enhances retrieval performance without requiring PRF-based fine-tuning. The following section first introduces the **GenQREnsemble** approach, then details the extended **GenQREnsemble-RF** approach, which integrates PRF, followed by empirical evaluations to demonstrate the effectiveness of GenQREnsemble-RF.

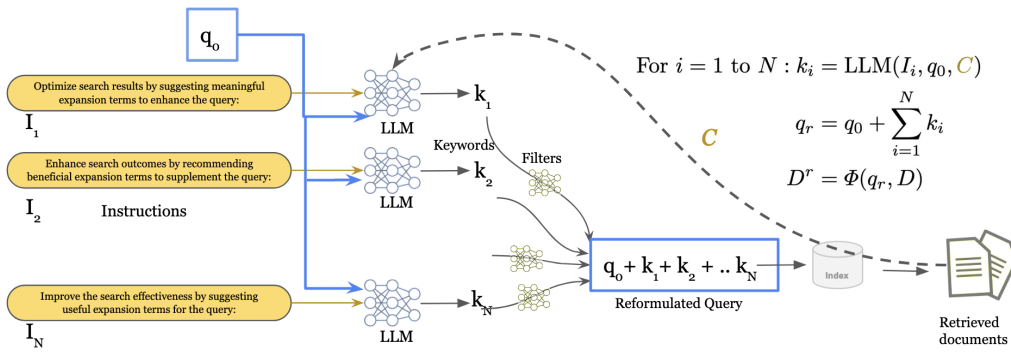


Figure 3.3: The complete flow and algorithm of GenQREnsemble and **GenQREnsemble-RF** (dotted).

3.5.3 Proposed Work

The approach is divided into two stages: the pre-retrieval stage and the post-retrieval stage. In the pre-retrieval stage, the query reformulation process is represented as $q_r = R.q_0$, where the reformulation

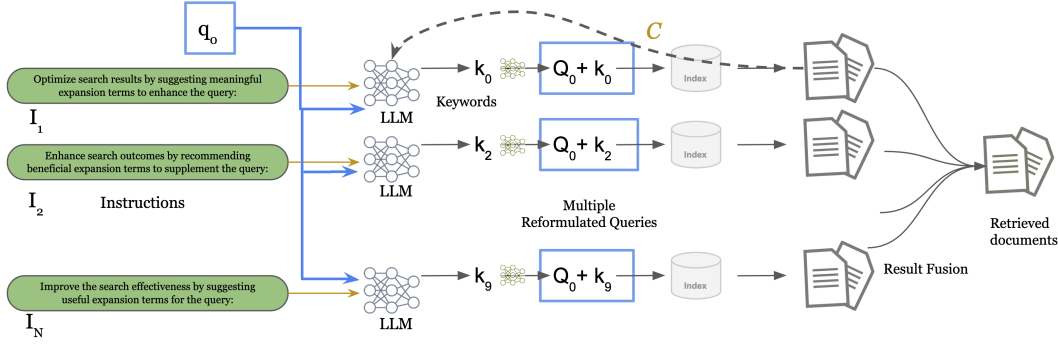


Figure 3.4: The complete flow of GenQRFusion. GenQRFusion-RF is shown with the dotted line.

function R transforms the user’s original query q_0 into a new reformulated query q_r , enhancing retrieval effectiveness for tasks such as passage or document retrieval. Additionally, a post-retrieval stage can be employed, where the reformulator integrates feedback, such as document or passage-level information, as supplementary context to further refine the retrieval process.

3.5.3.1 Pre-retrieval Stage

At this stage, two ensemble-based query reformulation methods, **GenQREnsemble** and **GenQRFusion**, are introduced, both leveraging N diverse paraphrased versions of a query reformulation (QR) instruction to improve retrieval. Initially, an LLM is employed to paraphrase the original instruction I_1 , creating N distinct variations (I_1 to I_N). This step is performed offline. However, each repetition of this step may produce different sets of paraphrased instructions, and this is acceptable because the approach leverages the LLM’s inherent stochastic nature to generate different paraphrases, resulting in diverse expansion terms. Each paraphrased instruction is then paired with the user’s query q_0 to generate instruction-specific keywords. Optionally, these generated keywords are passed through another LLM to filter out irrelevant terms. The key differences between these two approaches are outlined below.

1. **GenQREnsemble**: All generated keywords are appended to the original query, forming a reformulated bag-of-words query. This query is then executed against a document index D to retrieve relevant documents, denoted as D^r . The entire process and corresponding algorithm are depicted in Figure 3.3.
2. **GenQRFusion**: Keywords generated from each instruction are appended individually, creating N separate reformulated queries. Each of these queries is executed against the document index D , producing N sets of relevant documents D_i . These sets are then combined to generate a single ranked list of documents D^r using methods such as score fusion (*e.g.*, BM25 score fusion or reciprocal rank fusion). The process is illustrated in Figure 3.4.

3.5.3.2 Post-retrieval Stage

To evaluate how incorporating additional context, such as document feedback, improves query reformulation, a PRF module is sequentially integrated into the pre-retrieval stage variants: **GenQREnsemble-RF** and **GenQRFusion-RF**. The overall search process involves two retrieval calls. The first call, in the pre-retrieval stage, collects PRF documents using the ensemble prompting technique. The second call, in the post-retrieval stage, incorporates these PRF documents into the query reformulation. Specifically, the N instructions mentioned earlier are prepended with a fixed context string, “Based on the given

context information $\{C\}$,” as used in Wang et al. [97]. It was experimentally found that prepending this string performs better than appending it at the end. Here, $\{C\}$ represents a space-delimited concatenation of feedback documents, $C = d_1 + \dots + d_m$, which are either pseudo-relevance feedback from the initial retrieval or human feedback.

3.5.4 Experimental Setup

In this subsection, the emphasis is on evaluating the incorporation of PRF into the GenQREnsemble approach. The version of GenQREnsemble [85] published in ECIR 2024 only introduced the ensemble prompting method for query reformulation, and the GenQRFusion variant was not explored. Detailed evaluations of GenQREnsemble’s effectiveness can be found in the publication [85]. In this actual work, however, GenQRFusion is introduced and tested. The main focus is on integrating PRF into both variants. Therefore, this thesis centers around PRF incorporation into both GenQREnsemble and GenQRFusion approaches. The following section will cover the experiments conducted and the analysis across different retrieval settings.

3.5.4.1 Prompts

Based on Wang et al. [97], to instruct the LLM to generate query reformulations, initially a base QR instruction I_1 is empirically chosen. This instruction is used to generate N paraphrases ($N = 10$). To this aim, the ChatGPT API [107] is invoked with the paraphrase generating prompt, namely, $I_p = \text{“Generate 10 paraphrases for the following instruction:”}$ – and the base QR instruction I_1 to obtain I_2 to I_{10} . These paraphrases, shown in Figure 3.5 serve as the instruction set for all subsequent experiments.

#	Instruction
1	Improve the search effectiveness by suggesting expansion terms for the query
2	Recommend expansion terms for the query to improve search results
3	Improve the search effectiveness by suggesting useful expansion terms for the query
4	Maximize search utility by suggesting relevant expansion phrases for the query
5	Enhance search efficiency by proposing valuable terms to expand the query
6	Elevate search performance by recommending relevant expansion phrases for the query
7	Boost the search accuracy by providing helpful expansion terms to enrich the query
8	Increase the search efficacy by offering beneficial expansion keywords for the query
9	Optimize search results by suggesting meaningful expansion terms to enhance the query
10	Enhance search outcomes by recommending beneficial expansion terms to supplement the query

Figure 3.5: The N reformulation instructions used for GenQREnsemble and GenQRFusion

3.5.4.2 Generation Models

For generating the query reformulations, two models, `flan-t5-xxl` and `Llama-2-7b-chat-hf` are employed. FlanT5 [108] is a set of models are created by fine-tuning the text-to-text transformer model, T5 [46] on instruction data of a variety of NL tasks. The checkpoint¹ provided through HuggingFace’s Transformers library(HF) [109] is used. Nucleus sampling is performed with a cutoff probability of 0.92 keeping top 200 tokens (`top_k`) and a repetition penalty of 1.2.

An investigation also conducted to assess the use of `Llama-2-7b-chat-hf` [110], an autoregressive language model, which is RLHF fine-tuned and optimized for dialog use cases. The LLama2

¹huggingface.co/google/flan-t5-xxl

series of models are chosen as they have shown state-of-the-art performance across multiple benchmarks. The HF checkpoint² is used, keeping the same generation settings as above with a repetition penalty of 2.1. The prompt template shown in Figure 3.6 is used where the `instruction` variable is the concatenation of the actual instruction and the query provided at run-time. Additional instruction of “And do not explain yourself.” is appended to minimize the conversational jargon that the model could generate.

You are a helpful assistant who directly provides comma separated keywords or expansion terms.
Provide as many expansion terms or keywords as possible related to the query. And do not explain yourself.
instruction: query

Figure 3.6: The prompt used for all the Llama-2 Query reformulators.

3.5.4.3 Retrieval Evaluation and Baselines

For evaluation, standard **TP19** is used: TREC 19 Passage Ranking which uses the MS-MARCO dataset [99, 111] consisting of search engine queries through IRDataset [112]’s interface.

Our work is compared against the following using the Pyterrier [113] framework. For all the post-retrieval experiments, 5 documents are used as feedback.

1. BM25: Raw queries are used for retrieval without any reformulation.
2. BM25+RM3 [59]: BM25 retrieval with RM3 expanded queries (#feedback terms=10).
3. BM25+GenPRF [97]: BM25 retrieval with GenPRF expanded queries.
4. **GenQREnsemble-RF**: The corresponding PRF variant of ensemble query reformulation introduced by Dhole and Agichtein [85] which used a FlanT5 generator.

3.5.5 Results and Discussion

Setting	With BM25 Retriever				With Neural Reranking			
	nDCG@10	nDCG@20	MAP	MRR	nDCG@10	nDCG@20	MAP	MRR
BM25	.480	.473	.286	.642	.718	.696	.477	.881
RM3	.504	.496	.311	.595	.716	.699	.480	.858
GenPRF	.576	.553	.363	.715	.722	.703	.486	.874
GenPRF [97]	.628		.404	.809				
GenQREnsemble-RF	.585(+2%)	.560(+1%)	.373(+3%)	.753(+5%)	.729(+1%)	.706(+1%)	.501(+3%)	.894(+2%)
GenQRFusion-RF	.566	.548	.368	.725	.718	.707	.488	.882
GenPRF (Oracle)	.753	.728	.501	.936	.742	.734	.545	.881
GenQREnsemble-RF (Oracle)	.820^α(+9%)	.773(+6%)	.545(+9%)	.977(+4%)	.756(+2%)	.751(+2%)	.545	.897(+2%)
GenQRFusion-RF (Oracle)	.708	.672	.465	.938	.748	.731	.532	.887

Table 3.4: Comparison of PRF performance on the TP19 benchmark using queries generated from `flan-t5-xxl`. **With BM25 Retriever** utilizes the reformulated query directly for initial retrieval, while **With Neural Reranking** further re-ranks the results obtained from the BM25 first-stage retrieval.

First, Table 3.4 investigates if GenQREnsemble-RF and GenQRFusion-RF can effectively incorporate PRF. While GenQRFusion-RF improves recall, GenQREnsemble-RF improves retrieval performance

²huggingface.co/meta-llama/Llama-2-7b-chat-hf

across all metrics as compared to other PRF approaches and is able to incorporate feedback from a BM25 retriever better than RM3 as well as its zero-shot counterpart. To assess if GenQREnsemble-RF and GenPRF can at all benefit from incorporating relevant documents, an oracle testing is performed by providing the highest relevant gold documents as context. Finding showed that GenQREnsemble-RF is able to improve over GenQREnsemble (without feedback) showing that it is able to capture context effectively as well as benefit from it. It can incorporate relevant feedback better than its single-instruction counterpart GenPRF. Additionally, the improvements even under the neural setting as GenQREnsemble-RF outperforms RM3 and GenPRF. Besides, the oracle improvements are higher with only a BM25 retriever as compared to when a neural reranker is introduced.

Furthermore, the effect of varying the number of feedback documents from 0 to 5 is also evaluated. Resorting to an ensemble approach is highly beneficial as seen in Figure 3.7. In the BM25 setting, the ensemble approach seems always preferable. Under the neural reranker setting too, GenQREnsemble-RF almost always outperforms GenPRF.

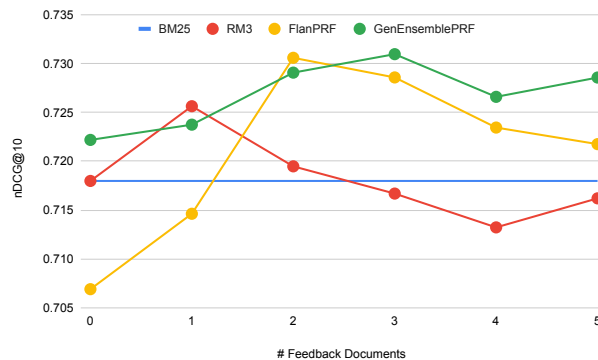


Figure 3.7: Effect of PRF with increasing feedback on TP19. BM25 is employed to directly query using the reformulated queries.

3.5.6 Summary and Limitations

Zero-shot QR is advantageous since it does not rely on any labeled relevance judgments and allows eliciting pretrained knowledge in the form of keywords by prompting the model with the original query and appropriate instruction. By introducing GenQREnsemble and GenQRFusion, the experiments validated that that zero-shot performance can be further enhanced by using multiple views of the initial instruction, both as a unified query and through document fusion. Moreover, the experiments showed that the PRF extension GenQREnsemble-RF can effectively incorporate relevance feedback, either automated or from users. A final filtration step to convert messy keywords to their fluent counterparts helps increase interpretability. The ensemble approach improves upon the state-of-the-art zero shot reformulation and can be applied to a variety of settings, for example, to address different aspects of queries or metrics to optimize, or to better control the generated reformulations, or for improving queries for retrieval augmented generation.

While generative QR greatly benefits from the ensemble approach, these methods come at a cost of potentially increased latency, but this is becoming less problematic with the increased availability of batch inference for LLMs.

The interpretability of not only query reformulations but even of other language phenomena is often highly subjective [114] and it could vary according to the intended application. Besides, it could be

argued that natural language might not always be the best mode for interpretability. While natural language expressions could communicate the precise intent, keywords could also be useful for clustering or visualization – and hence both being useful for interpretability. This work closely adheres to the former definition of interpretability.

4 Unsupervised Ranker Adaptation

Pretrained neural ranking models often struggle or perform suboptimally when faced with domain shifts. A domain shift occurs when there is a mismatch in data distribution between the training and testing samples, particularly between the source and target domain data. A domain can be characterized by various attributes, which fall into three main categories: query, document, and label. For instance, query and document attributes may include topics, linguistic features, language, structure, modality, and format [3], while label attributes may involve different relevance criteria, such as topic relevance, paraphrasing, presence of a correct answer, or counter-argument [3]. Domain shift refers to any change in these attributes from the source to the target domain. Due to these changes, a pretrained neural ranker might not perform optimally across the target domain. To ensure consistent and robust performance across different domains, it is crucial to adapt pretrained neural rankers to the target domain.

Domain-invariant feature learning is a commonly used approach for adapting neural ranking models across target domains. These models are typically pretrained on multiple related, yet distinct, source domain collections to learn robust and generalizable features. As a result, domain-invariant pretrained rankers demonstrate strong zero-shot performance across target domains without requiring domain-specific fine-tuning. This approach is particularly favored in scenarios where generalization, efficiency, and reduced overfitting are important, especially when labeled data for the target domain is scarce. It also provides greater flexibility and scalability for large real-world deployments. Various methods have been employed to achieve domain-invariant feature learning in neural rankers, including adversarial learning, feature alignment, correlation alignment, contrastive learning, domain-invariant regularization, and large-scale training. Among these, large-scale pretraining is often the preferred method due to its broad coverage across many target domains.

Large-scale source domain pretraining is another form of domain-invariant learning. This method enables language models to generalize effectively across various domains and tasks. Language models like BERT [32], RoBERTa [37], and T5 [46] are pretrained in an unsupervised manner on massive, unstructured datasets without human-provided labels. Through this process, they acquire extensive semantic understanding and knowledge about the world. These models can then undergo supervised task-specific pretraining or domain-specific fine-tuning, both of which contribute to domain adaptation. For instance, BERT is further pretrained for relevance tasks on MS-MARCO [111] question-answering data, a variant known as MonoBERT [65]. Likewise, BERT or MonoBERT can be fine-tuned directly on specific target domains such as finance, biomedicine, or movies. Both unsupervised language pretraining and supervised task pretraining help these models learn domain-invariant features. However, in the context of neural ranking domain adaptation discussed in this thesis, I refer to the "source" training as "task-specific supervised pretraining" and the "target" training as "domain-specific supervised fine-tuning". Pretrained neural rankers, including models like MonoBERT, MonoT5 [115], ColBERT [28], and Contriever [45], have demonstrated impressive zero-shot performance across various domains and retrieval tasks, as shown in the BERT [1] benchmark.

However, zero-shot neural ranking may be less effective because it does not leverage target domain information. Fine-tuning on the target domain allows a neural ranker to learn more specialized, rather than generic, features. By sequentially fine-tuning a pretrained neural ranker, the model can acquire domain-specific knowledge, including document and query distributions as well as task features unique to that domain. Key domain-representative features typically come from the target query, document, and label representations. Therefore, it is crucial for a neural ranker to capture and utilize this information to enhance relevance matching within the target domain. While state-of-the-art domain-invariant neural

rankers are often the preferred choice for real-world applications, a cost-effective, minimally effortful domain fine-tuning approach that delivers notable performance gains would be highly valuable.

Several challenges emerge when fine-tuning a neural ranking model across domains. If domain-specific features are not properly captured, performance may be limited. The main challenges of domain fine-tuning neural rankers include the high cost of fine-tuning and the scarcity of target domain data. Transformer-based neural ranking models are data-intensive, requiring large-scale datasets, which increases the cost of data collection (queries), annotation (labels), and computational resources. Moreover, domain fine-tuning typically demands extensive human-labeled supervised data, which can be hard to acquire. Most previous works on domain fine-tuning rely on unsupervised or weakly-supervised approaches, leveraging unlabeled target data. Unsupervised fine-tuning is particularly common due to its scalability and reduced dependence on noisy labels. In response to these challenges, I propose two novel unsupervised domain fine-tuning approaches for neural rankers, which I will outline next.

4.1 Overview of Proposed Ranker Adaptation Approaches

In this chapter, I focus on investigating the effectiveness and resilience of adapting neural rankers for different retrieval tasks and domains. This leads to the central research question (**RQ2**): *Can ranking models be fine-tuned effectively with minimal or no supervision?* My contributions as the lead author addressing **RQ2** include **DUQGen** and **mDUQGen**, both offering solutions for unsupervised ranker adaptation across domains and languages, respectively. DUQGen introduces an efficient method for unsupervised domain adaptation of neural rankers by diversifying synthetic query generation, and it was published as a main conference paper at **NAACL 2024**. mDUQGen presents a cost-effective approach for language adaptation in multilingual retrieval. I aim to submit mDUQGen work to ACL 2025.

Both of my approaches share a common data augmentation framework but differ in their components for domain versus language fine-tuning. I introduce an innovative, efficient sampling-based method to effectively represent the target domain or language. Leveraging the generative capabilities of large language models (LLMs), I generate high-quality queries that are representative of the target domain or language, creating unsupervised training data. This domain- and language-specific training data allows for effective fine-tuning of neural ranking models without increasing costs or requiring human annotations. The main difference between DUQGen and mDUQGen lies in the components used for adapting to target domains and languages, such as the document encoder, query generator, and neural ranker. I will delve into the detailed implementation and specific differences in their respective sections next.

Next, I will present a motivating analysis to demonstrate the importance of performing domain fine-tuning on top of zero-shot neural ranking models before introducing my unsupervised approaches. This preliminary analysis evaluates the impact of supervised fine-tuning using labeled training data, highlighting the performance gains that domain-specific fine-tuning can offer. Experimental results from state-of-the-art neural ranking models based on LLMs, such as MonoT5-3B [115], RankLLaMA [73], and RankGPT [78], provide an upper limit on the performance achievable through supervised fine-tuning. In contrast, my approaches operate without supervision, using an unsupervised strategy. It will be intriguing to see how closely my unsupervised methods can match or even surpass the performance of supervised approaches.

4.2 Preliminary Study on Domain Fine-tuning

Several methods have been proposed to address the challenge of unsupervised domain adaptation, particularly focusing on domain-invariance feature learning. Recent advancements in pretrained language models have shown promise in learning robust contextual features and facilitating transfer learning across diverse domains. The pretraining of these language models on extensive training data is a form of domain-invariance learning since the pretraining data encompasses a wide range of topics. In the upcoming section, I will delve into the transfer learning capabilities of state-of-the-art zero-shot language models, particularly LLMs.

4.2.1 Benchmarking Zero-shot LLM Rankers

As outlined in the literature review section (Section 2.9), recent years have seen the emergence of multiple approaches to utilize LLM for zero-shot ranking. These approaches do not involve domain fine-tuning or incorporate in-context demonstrations during prompting, hence termed as zero-shot. Notably, all three types of ranking families (point-wise, pair-wise, and list-wise) have been explored using LLM. However, point-wise ranking methods have employed varied techniques. Therefore, in the next subsection, I initially evaluate the zero-shot ranking performance of point-wise rankers and then proceed to compare the zero-shot ranking across point-wise, pair-wise, and list-wise rankers.

4.2.1.1 Point-wise Zero-shot Rankers

To compare various types of point-wise neural rankers, I chose SOTA neural re-rankers from each architectural family. My evaluation was performed using the BEIR [1] collection, which encompasses diverse datasets across multiple domains and retrieval tasks. Specifically, I selected MonoELECTRA, MonoT5-3B, and RankLLaMA as representatives of SOTA rankers from encoder-only, encoder-decoder, and decoder-only architectures, respectively. These re-rankers were pretrained on MS-MARCO query-passage pairs but have not undergone fine-tuning on the evaluation dataset, making them zero-shot neural rankers in my analysis.

Table 4.1 clearly demonstrates that generative models outperform discriminative models in point-wise ranking task. The effectiveness of generative language models has notably increased with larger model sizes and multi-stage pretraining. Notably, across all 17 datasets, MonoELECTRA falls short of generative neural rankers. Across the average of 17 BEIR datasets, the nDCG@10 score of MonoELECTRA is 9% lower than RankLLaMA and 13% lower than MonoT5-3B. From Table 4.1, it is evident that MonoT5-3B emerges as the superior choice due to its model size and extensive fine-tuning on the MS-MARCO dataset. Despite being larger than MonoT5-3B, RankLLaMA (7B) lags behind by 4% in relative nDCG@10 scores. In conclusion, Table 4.1 underscores the significance of generative language models in developing future state-of-the-art point-wise rankers.

4.2.1.2 Comparison between Point-wise and List-wise LLM Rankers

To compare point-wise and list-wise rankers, I chose SOTA zero-shot LLM based neural re-rankers. The evaluation was conducted using the BEIR collection across various out-of-distribution datasets. Specifically, MonoT5-3B was chosen as the SOTA point-wise ranker, while three list-wise rankers — RankVicunna, RankGPT(gpt-3.5-turbo), and RankGPT(gpt-4)—were selected to demonstrate competitive performance achievable with list-wise ranking. Once again, these neural re-rankers were not fine-tuned on the domain dataset, making them zero-shot rankers in my analysis.

#Test Queries	Dataset	BM25	Zero-shot Models		
			Encoder	Encoder-Decoder	Decoder
			MonoELECTRA	MonoT5-3B	RankLLaMA
3,452	NQ	0.329	0.540	0.579	0.545
7,405	HotpotQA	0.603	0.691	0.718	0.698
648	FiQA	0.236	0.370	0.462	0.403
6,666	FEVER	0.753	0.816	0.849	0.837
323	NFCorpus	0.325	0.280	0.373	0.350
300	SciFact	0.665	0.684	0.760	0.722
500	BioASQ	0.465	0.502	0.559	0.532
1,000	SCIDOCS	0.158	0.162	0.193	0.182
1,406	Arguana	0.315	0.233	0.316	0.334
1,535	Climate-FEVER	0.213	0.246	0.278	0.222
10,000	Quora	0.789	0.730	0.848	0.773
57	TREC-NEWS	0.398	0.445	0.473	0.484
249	Robust04	0.407	0.440	0.566	0.471
50	TREC-COVID	0.656	0.730	0.830	0.811
97	Signal1M	0.330	0.297	0.321	0.298
49	Touche	0.367	0.278	0.311	0.413
400	DBPedia	0.313	0.278	0.408	0.411
	Average	0.408	0.437	0.500	0.482

Table 4.1: Comparison of zero-shot point-wise rankers measured in nDCG@10. The best scores are highlighted in bold.

#Test Queries	Dataset	BM25	Zero-shot Models			
			Point-wise	List-wise		
				RankVicunna	RankGPT(gpt-3.5-turbo)	RankGPT(gpt-4)
3,452	NQ	0.329	0.579	0.520	-	-
7,405	HotpotQA	0.603	0.718	0.721	-	-
648	FiQA	0.236	0.462	0.324	-	-
6,666	FEVER	0.753	0.849	0.815	-	-
323	NFCorpus	0.325	0.373	0.335	0.356	0.385
300	SciFact	0.665	0.760	0.707	0.704	0.750
500	BioASQ	0.465	0.559	0.516	-	-
1,000	SCIDOCS	0.158	0.193	0.178	-	-
1,406	Arguana	0.315	0.316	0.290	-	-
1,535	Climate-FEVER	0.213	0.278	0.267	-	-
10,000	Quora	0.789	0.848	-	-	-
57	TREC-NEWS	0.398	0.473	0.471	0.489	0.529
249	Robust04	0.407	0.566	0.476	0.506	0.576
50	TREC-COVID	0.656	0.830	0.832	0.767	0.855
97	Signal1M	0.330	0.321	0.330	0.321	0.344
49	Touche	0.367	0.311	0.317	0.362	0.386
400	DBPedia	0.313	0.408	0.421	0.445	0.471
	Average	0.408	0.505	0.486	0.494	0.537

Table 4.2: Comparison of zero-shot point-wise and list-wise rankers measured in nDCG@10. The best scores are highlighted in bold.

With the recent advancements in LLMs, featuring longer context lengths and enhanced reasoning abilities, list-wise ranking has become more cost-efficient and effective when compared to point-wise rankers. Point-wise rankers assess each query-document pair individually to compute relevance scores, while list-wise rankers evaluate a query alongside a list of documents to determine their relative ordering of relevance. During the BERT-era, neural rankers faced limitations due to their limited context lengths. However, with the advent of LLMs, I can process multiple documents simultaneously for re-ranking tasks. As a result, the number of API calls to LLMs has significantly decreased during list-wise ranking compared to point-wise ranking task.

The Table 4.2 demonstrates the significant effectiveness of a list-wise ranker compared to the SOTA point-wise ranker. According to Table 4.2, RankGPT using gpt-4 achieves a 6% relative improvement on average nDCG@10 scores across shared evaluation datasets. Additionally, RankGPT with gpt-3.5-turbo, which is slightly cost-effective than gpt-4, achieves similar average nDCG@10 scores compared to MonoT5-3B. While there has been prior work on pair-wise ranking, such as PRP [79], the authors did not report zero-shot evaluation scores on BEIR datasets. In summary, list-wise neural re-rankers like RankGPT(gpt-3.5-turbo) achieve state-of-the-art effectiveness with fewer API calls, leading to minimized costs.

4.2.2 Supervised Fine-tuning of Zero-shot Rankers

Before delving into unsupervised domain fine-tuning, it is crucial to present evidence of the benefits of domain fine-tuning in enhancing zero-shot neural ranking performance. Thus, I devised a controlled experimental setup to quantify the performance improvements achievable through supervised fine-tuning, leveraging human-labeled datasets where available. For the experiments, a subset of BEIR datasets, containing more than 256 training queries, was selected. If a dataset already had a train split available,

Train Source	#Train Size	Datasets	BM25	Encoder-only		Encoder-Decoder		Decoder-only	
				MonoELECTRA (110M)		MonoT5-3B		RankLLaMA (7B)	
				Zero-shot	Fine-tuned	Zero-shot	Fine-tuned	Zero-shot	Fine-tuned
Train Split	10000	NQ	0.329	0.540	0.325 (-40%)	0.579	0.581 (+0%)	0.545	0.579 (+6%)
	10,000	HotpotQA	0.603	0.691	0.740 (+7%)	0.718	0.751 (+5%)	0.698	0.716 (+2%)
	5,498	FiQA	0.236	0.370	0.421 (+14%)	0.462	0.491 (+6%)	0.403	0.420 (+4%)
	10,000	FEVER	0.753	0.815	0.903 (+11%)	0.849	0.914 (+8%)	0.837	0.827 (-1%)
	2,590	NFCorpus	0.325	0.280	0.350 (+25%)	0.373	0.374 (+0%)	0.350	0.327 (-7%)
	809	SciFact	0.665	0.684	0.740 (+8%)	0.760	0.767 (+1%)	0.722	0.715 (-1%)
X-validation split	800	SCIDOCS	0.156	0.174	0.194 (+11%)	0.211	0.230 (+9%)	0.199	0.188 (-6%)
	1,120	Arguana	0.291	0.226	0.543 (+141%)	0.290	0.411 (+42%)	0.306	0.313 (+2%)
	1,200	Climate-FEVER	0.226	0.241	0.351 (+45%)	0.279	0.386 (+39%)	0.232	0.246 (+6%)
	8,000	Quora	0.792	0.739	0.887 (+20%)	0.860	0.910 (+6%)	0.778	0.825 (+6%)
	398	Robust04	0.395	0.428	0.449 (+5%)	0.572	0.559 (-2%)	0.475	0.430 (-9%)
	640	DBPedia	0.335	0.375	0.388 (+3%)	0.411	0.426 (+4%)	0.403	0.405 (+0%)
		Average	0.425	0.464	0.524 (+13%)	0.530	0.567 (+7%)	0.496	0.499 (+1%)

Table 4.3: Evaluation of supervised fine-tuning on a subset of BEIR datasets measured in nDCG@10.

then it was directly used for training; otherwise, I employed cross-validation to select a training set and a held-out test set with a 1:4 ratio for test:train splits. The training dataset comprised unique queries paired with corresponding positive query-document pairs in a 1:1 ratio. Utilizing a Contriever dense retriever, I obtained the top-100 documents per training query and selected the bottom 4 documents as negative query-document pairs. Subsequently, I evaluated the performance among point-wise rankers, namely MonoELECTRA, MonoT5-3B, and RankLLaMA.

Table 4.3 underscores the importance of fine-tuning using high-quality training data. It shows the nDCG@10 scores for the SOTA neural re-rankers fine-tuned with supervised training data compared to zero-shot baselines. Supervised fine-tuning leads to significant improvements over zero-shot counterparts across most datasets, particularly with MonoELECTRA and MonoT5-3B neural re-rankers. However, RankLLaMA experiences a notable performance drop across 4 out of 12 datasets and some insignificant changes post fine-tuning. I hypothesize that utilizing parameter-efficient fine-tuning (*e.g.*, LoRA) on RankLLaMA overly regularizes the model weights, limiting performance gains and often leading to deterioration. Further insights into the limitations of LoRA fine-tuning should be experimented as a future work. In summary, a medium-sized MonoT5-3B achieves the highest nDCG@10 average scores both before and after fine-tuning, with fine-tuning notably improving the zero-shot re-ranker by 7% relatively. This evidence strongly encourages exploration of an unsupervised domain fine-tuning approach with minimal supervision.

Table 4.3 provides insights into the most challenging and easiest source-target pairs for transfer learning and domain adaptation, highlighting where performance gains are promising or limited. Specifically, when there are changes in task between the source (question answering) and target, such as in argument retrieval (Arguana), fact-checking (Climate-FEVER), and duplicate question retrieval (Quora), these pairs pose the greatest difficulty for transfer learning. However, fine-tuning a neural ranker (MonoELECTRA) on supervised data significantly boosts performance, resulting in relative improvements of 141%, 45%, and 20%, respectively. Conversely, tasks more closely aligned with pretrained question-answering, such as news retrieval (Robust04), entity retrieval (DBPedia), and citation prediction (SCIDOCS), yield limited improvements, with relative gains of just 5%, 3%, and 11%.

When examining domain adaptation—where the source and target share the same retrieval task but differ in domain or collection—similar patterns emerge. Significant improvements are seen when domains are dissimilar, as with financial data (FiQA) and biomedical data (NFCorpus), which show relative gains of 14% and 25%. However, for question-answering collections similar to MS-MARCO, such as HotpotQA

and NQ, performance can stagnate or even drop, with overfitting leading to a decrease of 40% in some cases (NQ). Overall, we find that the most challenging and promising source-target pairs tend to involve task differences, which enable more substantial improvements, whereas closely aligned task and domain pairs see limited or even negative gains in performance.

Having established the motivation, the next two sections will delve into my effective approaches for ranker adaptation across domains and languages.

4.3 Diversified Synthetic Query Generation for Ranker Fine-tuning

Large-scale pretraining on relevance data has demonstrated impressive transfer learning performance. A common approach is to train pretrained language models on large-scale general ranking tasks, such as MS-MARCO passage or document ranking [2] or Wikipedia retrieval [88], to learn task-specific features that are often transferable across different domains and datasets. These trained rankers can then be applied without further adaptation (in a zero-shot manner) to a variety of ranking tasks. For instance, the BEIR [1] benchmark has shown that several zero-shot neural rankers achieve state-of-the-art or near-state-of-the-art performance across a diverse set of retrieval tasks. However, without incorporating target domain information, the performance of these neural rankers can be suboptimal.

4.3.1 Problem Statement

Zero-shot neural ranking may be suboptimal because it does not leverage target domain information. When transitioning to specialized domains such as finance or scientific documents, the performance of zero-shot ranking can be improved by incorporating domain-specific data. To effectively fine-tune a neural ranker, a large-scale supervised dataset from the target domain is typically needed. However, obtaining sufficiently large, high-quality training data for modern neural rankers is both costly and time-consuming. As a result, there has been growing interest in domain adaptation methods for neural rankers, ranging from unsupervised to weakly-supervised approaches. These unsupervised methods often utilize synthetically generated queries, documents, or query-document pairs [4, 5, 6, 70, 75].

Unfortunately, most of the previously reported results from these synthetic training data generation approaches have not outperformed current state-of-the-art (SOTA) zero-shot models, as evaluated on the BEIR benchmark [1]. To the best of my knowledge, no unsupervised ranking adaptation method has consistently improved upon large neural SOTA zero-shot rankers. In fact, these adaptation methods often underperform compared to zero-shot models, highlighting their lack of robustness across different domains. These performance drops and limitations reveal a significant gap in the literature, underscoring the need for a more robust and effective approach to unsupervised domain adaptation.

Additionally, previous approaches require a large number of LLM calls and fine-tuning examples, which significantly increases costs. For instance, prior works like InPars and Promptagator require around 1 million LLM calls to generate a sufficient number of training examples. These approaches often rely on powerful, closed-source LLMs (such as OpenAI’s GPT-3 Curie and Google’s 137B FLAN), whose costs rise with the number of input tokens. Furthermore, they also demand millions of training examples to fine-tune a neural ranker, requiring extensive computational resources. The high cost and computational demands of these methods make them difficult to adapt easily across various real-world applications. Therefore, it can be hypothesized that diversified efficient sampling based synthetic query generation will enhance the effectiveness of ranker adaptation.

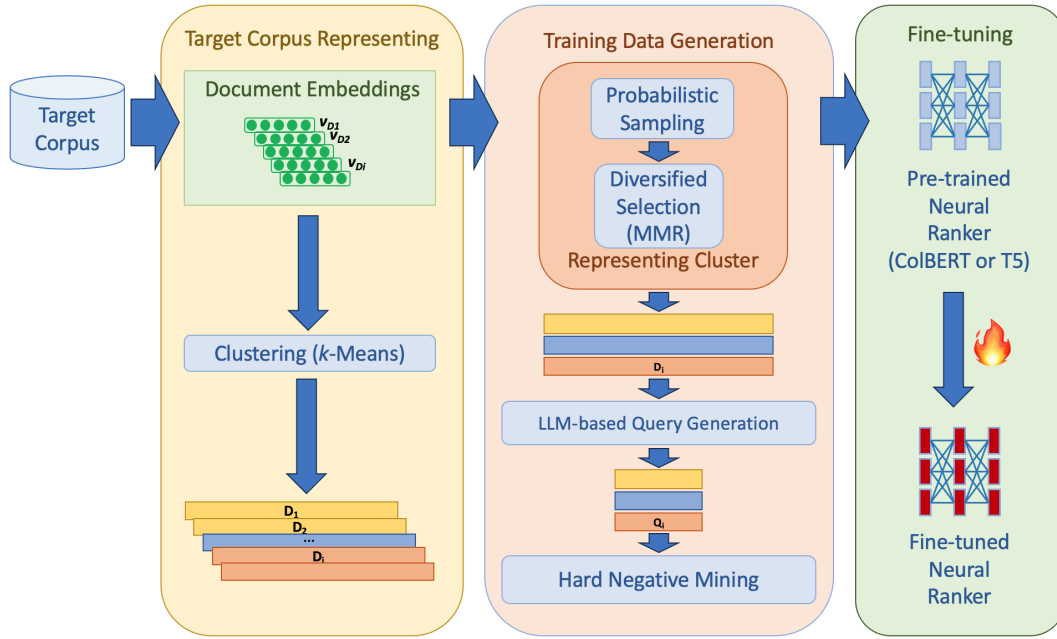


Figure 4.1: **DUQGen**: an unsupervised domain-adaptation framework for neural ranking.

4.3.2 DUQGen

To address these challenges, I introduce a new method called **DUQGen**, which stands for **D**iversified **U**nsupervised **Q**uery **G**eneration. DUQGen presents a general approach for ranking domain adaptation, focusing on selecting a representative and diverse set of document-query pairs for training a neural ranker. The method is based on a key insight: for synthetic training data to be effective for ranker adaptation, it must be both *representative of the target domain* and *sufficiently diverse*. This ensures that the ranking model adapts appropriately at the representation level, without leading to overfitting or catastrophic forgetting (i.e., reduced performance on original tasks).

DUQGen requires only partial access to the target document collection to be searched and can enhance any pretrained neural ranker. The method is illustrated in Figure 4.1 and introduces several innovations compared to previous unsupervised ranking adaptation approaches: (1) representing the target document collection as document *clusters*; (2) diversifying the synthetic query generation by probabilistically sampling over these clusters; and (3) using in-context prompting of a large LLM to generate queries from the selected documents. As shown in my experiments, these innovations consistently improve performance over previous SOTA baselines for ranking adaptation across nearly all BEIR benchmarks, as well as outperforming zero-shot neural rankers. Next, I will provide an in-depth explanation of my approach, beginning with a formal description of the method, followed by a detailed overview of the experimental setup, and concluding with a discussion of the results and analysis.

4.3.3 Proposed Work

DUQGen, shown in Figure 4.1, consists of four components – domain document selection, domain query generation, negative pairs mining, and fine-tuning. Each of the components is explained in detail in the following sections.

4.3.3.1 Domain Document Selection

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

4.3.3.1.1 Collection Document Clustering

Representing a large-scale target collection of documents with limited training data is challenging. Therefore, I propose to divide the collection into portions, and then sample documents within each portion. A clustering approach can be used for collection representation. Moreover, diverse topical documents can be achieved to represent the domain. Starting with a preprocessing step on a full collection of documents, short span documents are discarded by filtering out noisy documents. Then each of the documents are encoded by a SOTA text encoder, viz. *Contriever* [45]. A clustering (e.g., *K*-Means) technique can be applied using the document embeddings v_{D_i} , where K is a hyper-parameter to tune.

4.3.3.1.2 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 $(1 \leq c_k \leq C)$. Ideally, more number of documents can be sampled 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 $Pr(D_i|cluster_k) \propto c_k \forall D_i \in cluster_k$.

I propose to sample N number of synthetic training examples from K number of clusters, where $N \geq K$. Therefore, I 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 the sample size for each clusters are determined, I define my sampling approach. Let's take d_i as the similarity (e.g. *cosine similarity*) between document D_i and its corresponding cluster centroid. An exponential value e^{d_i} is defined 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:

$$Pr(D_i|cluster_k) = \frac{e^{d_i/T}}{\sum_{j=1}^{c_k} e^{d_j/T}} \quad (4.1)$$

$$d_i = \text{cosine}(v_{D_i}, \frac{1}{c_k} \sum_{j=1}^{c_k} v_{D_j}) \quad (4.2)$$

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.

4.3.3.1.3 Diversified Document Selection

Now N_k number of documents are sampled from each cluster $cluster_k$ and pooled 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, a diversity measure is applied, namely Maximal Marginal Relevance (MMR) [116]. First, the sampling process is iterated m times ($m = 5$) to obtain different sample sets. Then MMR is applied on the pooled documents from m sets to select top- N_k documents for $cluster_k$ as shown:

$$\arg \max_{D_i \in R \setminus S} \left[\lambda Sim_1(D_i, D_k) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \right] \quad (4.3)$$

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 I used the *cosine* similarity for both instances.

4.3.3.2 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, an LLM is used to generate a synthetic in-domain query for each sampled document. The LLM is few-shot prompted to generate such training queries similar to the existing work of Bonifacio et al. [4]. However, my contribution in this work 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, a handful (e.g., 3) human generated queries is created for the few-shot example documents with minimal human effort, and an example prompt is shown in Figure 4.2.

4.3.3.3 Negative Pairs Mining.

After obtaining the domain specific documents and queries, both positive and negative query-document pairs should be generated. 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 [7, 27, 45]. The synthetic queries are parsed to any first-stage retrievers, such as BM25 [31], ColBERT [28], or Contriever [45], to get top- x documents. Then the bottom- num_{neg} documents from the top- x are picked to map against the synthetic queries, where $1:num_{neg}$ is the *positive:negative* document pair ratio.

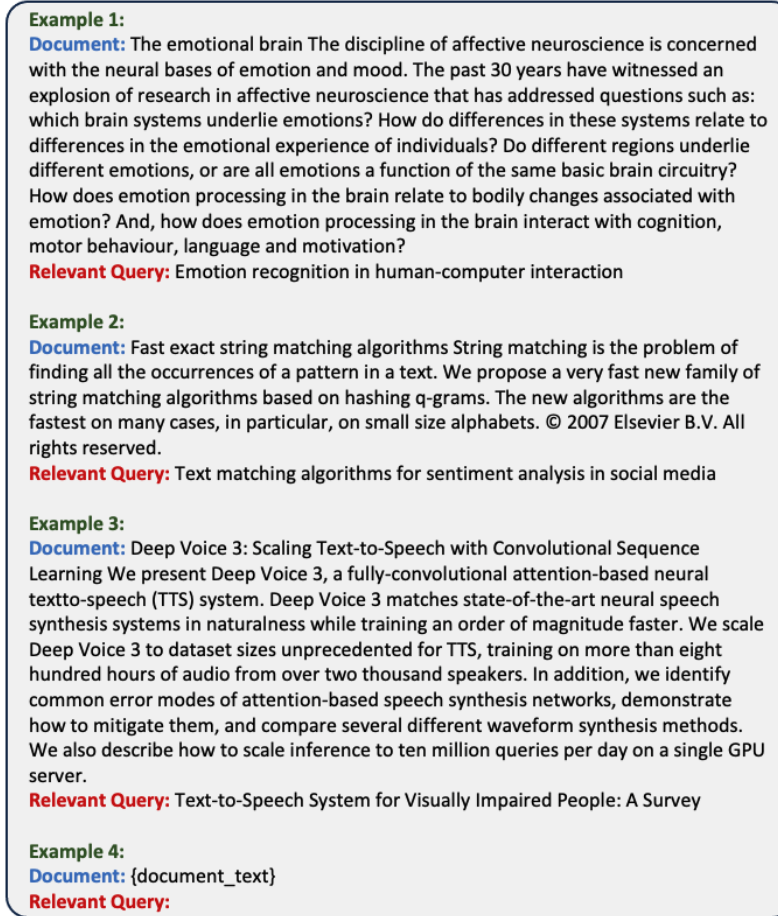


Figure 4.2: Prompt template with in-context examples for synthetic query generation for the SCIDOCS dataset.

4.3.3.4 Fine-tuning with Synthetic Data.

My domain adaptation framework can be applied on any ranking models with any weights initialization. To establish a strong competitor, the task pretrained model (on MS-MARCO) is leveraged, and sequentially fully fine-tune with DUQGen generated synthetic data. The same hyper-parameter settings used in the MS-MARCO pretraining stage are adapted for fair deliverable.

4.3.4 Experimental Setup

In this section, I provide details of my experimental setup to demonstrate the effectiveness of **DUQGen**.

4.3.4.1 Datasets and Metrics

I employed all 18 datasets from BEIR collection, ranging on diverse retrieval tasks, to assess the effectiveness of my domain adaptation framework on standard out-of-distribution datasets. Utilizing the *multi-field* index from Pyserini [117] for all datasets, I retrieved the top-100 and top-200 documents from lexical and dense first-stage retrievers respectively. Subsequently, I restricted re-ranking to top-100 BM25 documents and top-200 dense retriever documents. Since I evaluate my approach on both first-stage retrieval and re-ranking, I measured both nDCG@10 and R@100.

4.3.4.2 Ranking Models

I fine-tuned ColBERT¹ [28] and MonoT5-3B² [115], namely **DUQGen-retriever** and **DUQGen-reranker**, to show the effectiveness in both dense retrieval and re-ranking. During evaluation, I 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.

4.3.4.3 Baselines

I chose strong competitive rankers as baselines to highlight the effectiveness of my domain adapted ranker.

1. **BM25**: Traditional lexical sparse retrieval. I replicated the BM25 scores from scratch.
2. **Zero-shot (ZS) Models**: A fine-tuned ranker on MS-MARCO dataset, includes MonoT5-3B and ColBERT.
3. **InPars** [4]: 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. [5], I do not compare against InPars-v2.
4. **DocGen-RL** [5]: 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.
5. **Promptagator++** [6]: As the SOTA methods closest to my work, I 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 [118] to create 8 queries per document. Following consistency filtering, 1 million queries are selected to train a GTR-Base dual-encoder and cross-encoder [36].

I directly utilized the scores reported by authors for DocGen-RL and Promptagator++. For the remaining baselines, I employed their corresponding HuggingFace [119] models to re-run the inference.

4.3.4.4 Tools and Implementation

Various tools were employed for distinct stages in my pipeline, utilizing Contriever [45] for text encoding, Faiss [33] for k -Means clustering, and Llama2-7B-Chat [74] for query generation, Pyserini for BM25 baseline and hard negative mining, and PyTorch for standard fine-tuning. Throughout my 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.

¹<https://github.com/stanford-futuredata/ColBERT>

²[castorini/monot5-3b-msmarco-10k](https://github.com/castorini/monot5-3b-msmarco-10k)

4.3.4.5 Hyper-Parameter Tuning

In my methodology section, I introduced several hyper-parameters, all of which underwent tuning to determine the optimal values. These include the temperature $T = 1$ (Equation 4.1), 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. I 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, I discovered that a simple InPars-style template, displayed in Figure 4.2, consistently yields superior retrieval performance across datasets. For the process of hard negative mining, I set the first stage retriever hits $x = 100$ and the number of negatives per positive pair $num_{neg} = 4$.

I 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, I adapted its official pretraining 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 I optimize in the subsequent subsections.

4.3.4.5.1 Clustering Optimization

To represent target domain, I employed K -Means algorithm, where K denotes the number of clusters. I identified the optimal K for each dataset through an unsupervised method, known as the Elbow method [120]. 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.

4.3.4.5.2 Optimal Training Sample Size Discovery.

By fixing the optimum number of clusters K at 1000, I determined an optimal training sample size N , that proved effective across all datasets. To tune for N , I 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 4.5 displays nDCG@10 values for various instances of N , with K fixed at 1000. My analysis led us to select optimum $N = 1000$ for ColBERT and both $N = 1000$ and 5000 for MonoT5-3B fine-tuning across the datasets.

4.3.5 Results and Discussion

In this section, I present my main experimental results and delve into the key observations. I first describe my primary findings, reported using nDCG@10 in Table 4.4 comparing between baselines and my approach within each ranking setting. Second, I report the first-stage retrieval performances, measured using R@100 in Table 4.6.

4.3.5.1 Re-ranking Results

In Table 4.4, 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 re-rankers, showcasing average relative enhancements of

Datasets ($\rightarrow$)	covid	nfc	bio	nq	hotpot	fiqa	signal	news	robust	arg	touché	stack	quora	dbp	scidocs	fever	climate	scifact	avg
Models ($\downarrow$)	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^{†*}	.382[†]	.588[†]	.593^{†*}	.748^{†*}	.458*	.333	.483	.591^{†*}	.393 [†]	.320	.439[†]	.895[†]	.422^{†*}	.200 [†]	.890^{†*}	.310[†]	.757	.537
DUQGen-reranker(5k)	.836^{†*}	.376	.590[†]	.588^{†*}	.740^{†*}	.465*	.300	.449	.571*	.427[†]	.269	.439[†]	.894 [†]	.421^{†*}	.202[†]	.891^{†*}	.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 4.4: Main results comparing nDCG@10 scores between 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 4.5: 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.

Datasets ($\rightarrow$)	covid	nfc	bio	nq	hotpot	fiqa	signal	news	robust	arg	touché	stack	quora	dbp	scidocs	fever	climate	scifact	avg
Models ($\downarrow$)																			
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 4.6: Comparison of R@100 scores across baselines and . 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 ([†]).

		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 4.7: 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.

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

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 my 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 4.7). 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 my approach. We initially set $N = 5,000$ by tuning on two dev datasets, as shown in Table 4.5, and then applied this value consistently across all 18 datasets. However, Table 4.4 reveals that using 5,000 examples led to a slight performance decrease on three datasets—signal, news, and touch  —compared to the zero-shot model. These declines are not statistically significant given the small number of test queries (97, 57, and 49 queries, respectively). In contrast, when we reduced N to 1,000 examples, **DUQGen** achieved consistent improvements across all 18 datasets, with no performance drops. Thus, a promising direction for future work would be to develop a self-adaptive method to automatically determine the effective minimum training size (N) needed for each dataset or task.

4.3.5.2 First-Stage Retrieval Results

In Table 4.6, similar to nDCG@10 scores, R@100 also demonstrates more substantial improvements for larger domain-shifts (7.1%³ on TREC-COVID and 3.8%³ on Touch  -2020) and limited improvements for smaller domain-shifts (.4%³ on NQ). On average, **DUQGen** enhances zero-shot ColBERT by 2.1%³ on BEIR datasets.

³denotes absolute percentage improvement.

Number of Training Examples (N)	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%)	-	-	-	-
		1	100	500	1k	5k	10k

Zero-shot MonoELECTRA score on FiQA is 0.370

Number of Training Examples (N)	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%)	-	-	-	-
		1	100	500	1k	5k	10k

Zero-shot MonoELECTRA score on NQ is 0.540

Table 4.9: 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.

4.3.6 Analysis

In this section, I report my 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.

4.3.6.1 Effect of Clustering for Domain Adaptation

I employ clustering to represent the target domain and number of training samples to force diversity during fine-tuning. However, I question whether clustering genuinely contributes to the process and, if so, how it influences the overall performance. Additionally, I take the training sample size N , into account. In Table 4.9, I 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 4.3.6.1 and 4.3.6.2 in order to measure the amplified performance improvements in a smaller model, as described in the previous section.

Table 4.9 confirms my 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.

4.3.6.2 Effect of Query Generators

I conducted an ablation study on query generation to assess how the quality of generated queries impacts overall retrieval performance. Table 4.8 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 [77].

In comparison to the zero-shot re-ranking scores, LLAMA-2 7B was deemed the optimal choice for my 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 [121]. 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 potential for further prompt engineering to enhance performance on each dataset. My second main

contribution in this approach involves using in-domain 3-shot prompts to generate queries over the ms-marco prompt, showcasing notable improvements on LLAMA-2 7B model.

4.3.6.3 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 4.3: 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, I have evaluated the effectiveness of **DUQGen** using quantitative measures and are now shifting the focus to examining the actual queries produced by my method. Figure 4.3 presents ten example queries generated from the Quora and TREC-Covid datasets, each representing distinct tasks and domains. In Figure 4.3, the synthetic queries are sampled across different clusters with different probability scores $Pr(D_i|cluster_k)$. For instance, in Figure 4.3a, I 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 my approach in generating a diverse and representative set of high-quality queries.

4.3.7 Summary and Limitations

I introduced 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. I 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. I 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.

DUQGen can have certain limitations. DUQGen methodology involves two pivotal steps: (1) clustering; and (2) query generation. First, Contriever is employed as the text encoder to produce embeddings for clustering. While it is anticipated that this encoding will produce high-quality document representation and prove to be useful in my work, I did not assess other document embeddings. Future work could directly address the question of choosing the appropriate embedding for clustering.

Secondly, I 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 [121], I 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 [121] and perform corresponding studies to see the impact on reranking.

In the previous Section 4.3, I introduced DUQGen, an effective sampling-based method for fine-tuning neural ranking models across different domains in English monolingual settings. Building on that foundation, the next step is to extend domain adaptation to multiple languages. Therefore, in the following section, I will present **mDUQGen**, a similarly effective sampling-based fine-tuning approach designed specifically for neural rankers in a multilingual context.

4.4 Multilingual Synthetic Query Generation for Ranker Fine-tuning

Multilingual information retrieval (MLIR) aims to seek information from a corpus in a specific language using queries in that same language. Multilingual pretrained language models (MPLMs), such as mBERT [122], XLM-RoBERTa [123], and mT5 [46], have recently demonstrated superior performance in multilingual ranking tasks [64, 124, 125]. Developing MLIR systems can create numerous opportunities for native speakers worldwide to access information in their own languages. MPLMs have served as the foundation for building straightforward MLIR solutions. MPLMs are pretrained in an unsupervised manner on large-scale multilingual text corpora. Recently, these models have been fine-tuned using large-scale relevance pairs (*e.g.*, mMARCO [124]) across multiple languages, showcasing impressive cross-lingual transfer capabilities.

4.4.1 Problem Statement

Recently, large language models (LLMs) have been effectively utilized to synthesize training data for fine-tuning neural ranking models. Previous works, such as UPR [75], InPars [4], DocGen-RL [5], and Promptagator [6], have generated synthetic training data using LLMs especially for English-only ranking

models. These methods require a substantial number of LLM calls (100k - 1M), making them expensive, and they employ a sophisticated consistency filtering stage to remove any low-quality data. However, their application to multilingual retrieval and ranking has been largely unexplored.

The recently introduced SWIM-X [125] generates multilingual synthetic training data; however, it requires a significant number of LLM calls and does not isolate the impact of multilingual language adaptation from domain adaptation. In contrast, DUQGen [126] utilizes an effective sampling-based approach that requires only 1,000 training examples for fine-tuning, making it more cost-effective and consistent in improving ranking performance. This work extends DUQGen to the multilingual retrieval setting to measure the effects of both domain adaptation and isolated language adaptation.

Training a neural MLIR model presents several challenges. The required number of training samples, N , varies depending on the target language. High-resource languages like Spanish and Russian may necessitate a smaller N for fine-tuning as compared to low-resource languages like Tamil and Swahili due to their uneven distribution in language model pretraining. Additionally, the complex syntax and semantics of the target language also impact the convergence of MLIR fine-tuning. Therefore, developing an algorithm that enhances the performance of fine-tuned MLIR models with fewer training data is crucial.

Secondly, the *translate-train* paradigm used for training MLIR models has inherent limitations. This approach translates English training queries and documents into the target language before fine-tuning the MLIR model, which can introduce translation errors that impact the learning process. Additionally, for low-resource languages, reliable translation systems are often unavailable or lack accuracy due to insufficient supervised training data. However, LLMs pretrained on large multilingual corpora, including vast amounts of unstructured text without labels from low-resource languages, can often generate high-quality data in the target language when prompted. Therefore, it can be hypothesized that an efficient sampling based synthetic query generation (similar to DUQGen 4.3) will enhance the effectiveness of multilingual ranker adaptation.

4.4.2 mDUQGen

To address these aforementioned challenges, I propose **mDUQGen**, an extension of DUQGen to address the multilingual retrieval problem. First clustering is used to represent the target language and diversified sampling to capture each cluster representation. Then few-shot LLM prompting is used with target language query-document example pairs to generate new queries for the sampled documents. The evaluation of mDUQGen on standard multilingual retrieval collections demonstrates consistent and substantial improvements across all languages for both first-stage retrievers and re-rankers. Next, I will provide an in-depth explanation of my approach, beginning with a formal description of the method, followed by a detailed overview of the experimental setup, and concluding with a discussion of the results and analysis.

4.4.3 Proposed Work

My approach directly adapts the synthetic training data generation framework from DUQGen while changing the strictly monolingual components to multilingual extensions.

4.4.3.1 Target Language Document Sampling

To represent the target language, the collection documents are effectively sampled. First, clustering is applied to the collection and then diversified document sampling is used to achieve a target language representation. Specifically, the documents in the target language collection is encoded using a multilingual

pretrained encoder (*e.g.*, *mContriever*[45]). The document collection is then applied with clustering, especially into K clusters (*e.g.*, using K -Means). From each cluster, documents are sampled based on a probability distribution and diversified selection as described in the DUQGen Section 4.3.3.1. This process yield N sampled documents from the K clusters in the collection. These N sampled documents are highly representative of the target language and diversified to ensure the ranking model learns essential language-specific information.

4.4.3.2 Synthetic Multilingual Query Generation

Generate a short relevance query for a given document in {lang} language.

Example 1:

Document: पूजा कुमार . पूजा कुमार ने पहली बार 1997 में निर्देशक केआर की तमिल फिल्म कधलै रोजवे में जॉर्ज विष्णु के साथ अभिनय करने के लिए साइन किया, लेकिन फिल्म के निर्माण में देरी का मतलब था कि यह केवल 2000 में रिलीज़ हुई और तमिल फिल्म उद्योग में एक अभिनेत्री के रूप में उनका ध्यान नहीं गया। उन्हें पहले साइन किया गया था और फिर 1997 की फिल्म वी। आई। पी। से बाहर कर दिया गया था, जबकि उन्होंने कार्तिक के साथ चित्रा लक्ष्मण की फिल्म चिन्ना राजा में भी अभिनय करने के लिए साइन किया था, लेकिन अंततः उन्हें फिल्म से हटा दिया गया।

Relevant Query: तमिल फिल्म उद्योग में पूजा कुमार की पहली फिल्म कौन सी थी?

Example 2:

Document: सिरसा . एक सितंबर 1975 को बना जिला, हरियाणा के अंतिम छोर पर पश्चिम में बसा हुआ सिरसा धर्म, राजनीति और साहित्य के क्षेत्र में अपनी पूरी पहचान बना चुका है। आज आर्थिक रूप से संपन्न सिरसा का जन्म महाभारत काल और आजादी से भी पहले हुआ था। सरस्वती नदी के तट पर बसा होने के कारण पहले सिरसा का नाम सरस्वती नगर ही हुआ करता था। डेरा सरसाई नाथ के नाम पर इसका गहरा संबंध बताया जाता है।

Relevant Query: सिरसा का क्या ऐतिहासिक और सांस्कृतिक महत्व है?

Example 3:

Document: चन्\200dद्रदेव राम यादव . चन्\200dद्रदेव राम यादव,भारत के उत्तर प्रदेश की पंद्रहवीं विधानसभा सभा में विधायक रहे। 2007 उत्तर प्रदेश विधान सभा चुनाव में इन्होंने उत्तर प्रदेश के आजमगढ़ जिले के मुबारकपुर विधान सभा निर्वाचन क्षेत्र से बसपा की ओर से चुनाव में भाग लिया।

Relevant Query: 2007 उत्तर प्रदेश विधान सभा चुनाव में चंद्रदेव राम यादव ने किस निर्वाचन क्षेत्र का प्रतिनिधित्व किया था?

Example 4:

Document: {document_text}

Relevant Query:

Figure 4.4: Prompt template with Hindi language examples for generating queries.

Prompting an LLM with target language example (query-document) pairs can produce high-quality queries. Studies such as InPars and Promptagator have demonstrated that few-shot prompting an LLM with a handful of examples (*e.g.*, 3 to 8 etc.) can generate effective queries. Promptagator and DUQGen emphasize that selecting few-shot examples from the target domain ensures the improved quality of the generated queries. Similarly, for multilingual fine-tuning, the LLM is prompted with example pairs from the target language, as shown in the Figure 4.4. An instruction-following LLM can generate a relevant query in the target language for a given document in the same language without needing an explicit

Datasets (→)	avg	ar	bn	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh
Models (↓)	nDCG@10															
BM25	.396	.481	.508	.319	.333	.551	.183	.458	.449	.369	.419	.334	.383	.494	.484	.180
mDPR	.417	.499	.443	.478	.480	.472	.435	.383	.272	.439	.419	.407	.299	.356	.358	.512
BM25 + Cohere-API	.547	.667	.634	.507	.484	.675	.443	.573	.505	.516	.546	.477	.543	.638	.606	.389
SWIM-X	.476	.602	.571	.334	.363	.406	.643	.330	.395	.408	.433	.497	.400	.559	.563	.633
mContriever	.444	.538	.501	.411	.226	.604	.327	.295	.390	.434	.475	.386	.568	.554	.525	.424
mDUQGen-retriever	.479	.614	.516	.464	.377	.647	.372	.370	.406	.455	.480	.421	.580	.519	.556	.405
+ Mono-mT5-base	.627	.718	.695	.546	.552	.728	.491	.575	.502	.623	.640	.593	.680	.754	.753	.550
+ mDUQGen-reranker	.641	.744	.710	.562	.565	.753	.514	.579	.521	.626	.649	.621	.703	.761	.755	.555
Hybrid + mDUQGen-reranker	.655	.752	.724	.583	.577	.757	.510	.625	.540	.637	.670	.633	.709	.767	.770	.566
BGE-M3 (Supervised)	.664	.785	.797	.577	.576	.786	.520	.669	.621	.690	.668	.596	.647	.771	.790	.470
	R@100															
BM25	.785	.889	.909	.702	.731	.891	.653	.868	.904	.805	.783	.661	.701	.831	.887	.560
mDPR	.789	.841	.819	.864	.898	.788	.915	.776	.573	.825	.737	.797	.616	.762	.678	.944
SWIM-X	.804	.892	.878	.700	.763	.916	.758	.725	.743	.776	.768	.779	.878	.849	.929	.699
mContriever	.857	.926	.931	.835	.662	.950	.817	.652	.796	.881	.861	.838	.909	.970	.931	.902
mDUQGen-retriever	.871	.941	.915	.848	.799	.951	.868	.758	.791	.879	.815	.857	.912	.959	.928	.844
Hybrid	.920	.965	.966	.914	.858	.970	.861	.909	.931	.928	.890	.884	.920	.958	.966	.887
BGE-M3 (Supervised)	.785	.889	.909	.702	.731	.891	.653	.868	.904	.805	.783	.661	.701	.831	.887	.560

Table 4.10: Performance comparison of **mDUQGen** against strong and state-of-the-art baselines on MIRACL dev datasets using nDCG@10 and R@100 metrics. The model names with (+) denote that a first-stage retrieval top-100 results are re-ranked. **Hybrid** is the score fusion between BM25 and **mDUQGen-retriever**.

translation.

4.4.3.3 Hard Negative Mining

To train neural ranking models, positive and corresponding negative pairs are required. Effective document sampling and synthetic query generation produce a positive query-document pair. Additionally, effective hard negative mining can further enhance the discriminative feature learning of neural rankers. Therefore, hard negative pairs are then generated using the hard negative mining technique described in the DUQGen Section 4.3.3.3. For multilingual ranking, the key difference is the use of a multilingual dense retriever (*e.g.*, *mContriever*) for generating candidate documents.

4.4.3.4 Fine-tuning

The generated training data can be used for effective end-to-end fine-tuning of any multilingual ranking models. The multilingual rankers are initialized with zero-shot task pretrained multilingual model weights, such as those from mMARCO pretraining. During sequential fine-tuning, the hyper-parameters are kept unchanged.

4.4.4 Experimental Setup

In this section, I provide the detailed description of my evaluation setting.

Datasets ($\rightarrow$)	avg	ar	bn	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh
mDUQGen-retriever	.479	.614	.516	.464	.377	.647	.372	.370	.406	.455	.480	.421	.580	.519	.556	.405
English MS-MARCO																
+ Mono-mMiniLM-L6	.579	.662	.653	.498	.533	.672	.442	.542	.477	.566	.634	.546	.605	.644	.699	.518
+ Mono-mT5-base	.627	.718	.695	.546	.552	.728	.491	.575	.502	.623	.640	.593	.680	.754	.753	.550
+ mDUQGen-reranker	.641	.744	.710	.562	.565	.753	.514	.579	.521	.626	.649	.621	.703	.761	.755	.555
Helsinki Translate of MS-MARCO																
+ Mono-mMiniLM-L6	.494	.603	.527	.451	.445	.630	.440	.409	.438	.418	.521	.514	.539	.470	.557	.448
+ Mono-mT5-base	.576	.670	.633	.485	.514	.695	.472	.486	.454	.579	.562	.569	.645	.670	.717	.494
+ mDUQGen-reranker	.590	.705	.673	.495	.529	.728	.470	.495	.455	.599	.579	.588	.645	.659	.740	.487
Google Translate of MS-MARCO																
+ Mono-mMiniLM-L6	.536	.631	.614	.454	.486	.632	.451	.506	.445	.527	.580	.529	.546	.515	.635	.485
+ Mono-mT5-base	.602	.684	.679	.496	.540	.711	.479	.540	.462	.616	.598	.581	.653	.724	.737	.526
+ mDUQGen-reranker	.618	.721	.697	.507	.550	.737	.496	.552	.469	.640	.619	.608	.666	.725	.750	.537

Table 4.11: Comparison of different neural re-rankers pre-trained using different source data measured in nDCG@10. **mDUQGen-reranker** is the fine-tuned Mono-mT5-base. They re-rank the top-100 documents from **mDUQGen-retriever**.

4.4.4.1 Datasets and Metrics

To evaluate the effectiveness of mDUQGen in the multilingual retrieval context, I utilized the MIR-ACL [127] benchmark. I excluded the English collection, as the target language had already been evaluated in the DUQGen work (refer to Section 4.3.5) using the BEIR [1] collection. Therefore, my experiments focused solely on 15 question-answering datasets, encompassing both high-resource and low-resource target languages. For indexing and evaluation, I used Pyserini [117].

I evaluated the ranking performance using both precision and recall metrics, specifically nDCG@10 and R@100 respectively.

4.4.4.2 Ranking Models

I fine-tuned the state-of-the-art multilingual neural ranking models, specifically focusing on mContriever for first-stage retrieval and Mono-mT5-base [128] for re-ranking. These fine-tuned models are referred to as **DUQGen-retriever** and **DUQGen-reranker**, respectively.

4.4.4.3 Baselines

I compare the performance of mDUQGen against the following baselines, encompassing both traditional methods and state-of-the-art neural models.

1. **BM25**: Traditional lexical retriever.
2. **mDPR** [129]: mBERT encoder based dense passage retriever [27].
3. **Hybrid**: Combining the rank lists from BM25 and mDPR (score-fusion with 0.5 equal weights on normalized scores).
4. **Cohere-API** [130]: API performs re-ranking over top-100 BM25 rank list.
5. **SWIM-X**: mT5-base encoder based DPR dense retriever. It was fine-tuned with 180k training examples generated synthetically through summarize-then-ask PaLM 2 (S) [131] prompting.

6. **mContriever**⁴: mBERT encoder based dense retriever. Initially pretrained on the CCNet [35] multilingual corpus, followed by fine-tuned on the English MS-MARCO dataset.
7. **Mono-mT5-base**⁵ [124]: mT5-base multilingual pretrained model fine-tuned on English MS-MARCO pairs.
8. **BGE-M3**⁶ [132]: The BGE [133] family of models is fine-tuned using large and diverse datasets, including the MIRACL train split, employing a self-distillation technique. This supervised baseline is reported to compare my approach against an upper bound.

Official scores of mDPR, Hybrid, Cohere-API, and SWIM-X are reported directly. However, BM25, Contriever, Mono-mT5-base, and BGE-M3 scores are reproduced either from scratch or HuggingFace checkpoints.

4.4.4.4 Tools and Implementation

I employed Pyserini for evaluating BM25 and dense retrieval methods. I trained mContriever using Tevatron [134]. Query generation was performed using gpt-3.5-turbo. Documents were represented by concatenating both the title and text fields. Initial filtering of collection documents involved removing any texts shorter than 300 characters in length.

4.4.4.5 Hyper-Parameter Tuning

Similar to the hyper-parameter tuning conducted for DUQGen in Section 4.3.4.5, I found out that the number of clusters (K) is 1000 for all MIRACL datasets using Elbow [120] method. Consequently, I selected the number of training size (N) to 1000 for all datasets by tuning on two dev datasets (a low-resource language Swahili and a high-resource language Arabic).

4.4.5 Results and Discussion

In this section, I highlight the main results from my mDUQGen experiments in Table 4.10.

4.4.5.1 First-stage Retrieval Results

Fine-tuning a dense retrieval model improves both recall and precision scores. In Table 4.10, the **DUQGen-retriever** (fine-tuned mContriever) shows 8% and 2% average relative improvements over the zero-shot counterpart in nDCG@10 and R@100, respectively. The substantial 8% improvement in nDCG@10 demonstrates the efficient sampling strategy in identifying diverse and informative target domain pairs, enabling the mContriever to effectively identify highly relevant documents at the top of the rank list. Additionally, the consistent improvements in nDCG@10 across 13 out of 15 datasets highlight the robustness and generalizability of mDUQGen across different domains and languages.

⁴facebook/mcontriever-msmarco

⁵unicamp-dl/mt5-base-en-msmarco

⁶BAAI/bge-reranker-v2-m3

	R@100		nDCG@10		RR@100	
	sw	ar	sw	ar	sw	ar
Zero-shot	.909	.926	.568	.538	.574	.550
LLaMA2-7B	.914	.934	.571	.551	.580	.560
LLaMA3-8B	.926	.922	.580	.514	.596	.518
gpt-3.5-turbo	.912	.941	.580	.614	.593	.635

Table 4.12: Performances of mContriever fine-tuned with generated queries ($K = 1$, $N = 1000$) using different LLMs evaluated on dev datasets: Swahili and Arabic.

4.4.5.2 Re-ranker Results

Fine-tuning a re-ranker with synthetic training data generated using the mDUQGen framework significantly improves the ranking performance, surpassing all previous state-of-the-art baselines. As shown in Table 4.10, **DUQGen-reranker** achieves a 2% relative improvement in nDCG@10 over zero-shot Mono-mT5-base, highlighting the effectiveness of my mDUQGen approach. When combined with Hybrid, **DUQGen-reranker** delivers substantial gains, narrowing the performance gap with supervised fine-tuning using BGE-M3 to just 1%. This small gap underscores the power of my unsupervised mDUQGen framework to nearly match the performance of supervised methods. The top-performing **Hybrid + DUQGen-reranker** outperforms the state-of-the-art baselines like Cohere-API and SWIM-X by 20% and 38% relative average improvements on nDCG@10, respectively. The superior improvements over SWIM-X suggests that fine-tuning with fewer, carefully selected training examples is more effective than using a large dataset (180k examples). Notably, these performance improvements are consistent across all 15 datasets, demonstrating the robustness of mDUQGen. Overall, **Hybrid + DUQGen-reranker** sets a new state-of-the-art baseline for multilingual retrieval, with superior nDCG@10 scores.

Interestingly, the nDCG@10 scores in Table 4.11 indicate that sequentially fine-tuning (mDUQGen) a large-scale English fine-tuned model avoids the need for *resource-intensive large-scale multilingual intermediate translate-train stages*. Surprisingly, the zero-shot Mono-mT5-base and Mono-mMiniLM-L6 models pretrained using either Google or Helsinki translations of MS-MARCO perform worse than their counterparts pretrained on the original English MS-MARCO. This underperformance may be attributed to the differences in training configurations or translation errors that fall outside the focus of my work or my hypothesis. However, all three versions of **DUQGen-reranker**—whether pretrained on the English MS-MARCO or the two translated versions—demonstrate consistent 4% relative average improvements, suggesting that mDUQGen fine-tuning effectively adapts neural rankers to both domain and language in a single stage. Notably, using only 1000 training examples allows the re-ranker to achieve the same performance improvement as an expensive intermediate mMARCO fine-tuning stage. This finding indicates that a "translate-train" approach is unnecessary for intermediate sequential fine-tuning stages if the English only fine-tuned ranker is sequentially fine-tuned with the mDUQGen framework.

4.4.6 Analysis

To further support my experimental results, I performed analyses on the choice of query generators, the minimum training size, and a review of the generated queries.

	R@100		nDCG@10		RR@100	
	sw	ar	sw	ar	sw	ar
Zero-shot	.909	.926	.568	.538	.574	.550
1k	.912	.941	.580	.614	.593	.635
5k	.906	.943	.570	.597	.588	.612
10k	.924	.943	.582	.607	.599	.627

Table 4.13: Performances of mContriever fine-tuned using different training sizes (N) and evaluated on dev datasets: Swahili and Arabic, where queries are generated using gpt-3.5-turbo and $K = 1000$.

4.4.6.1 Selection of Query Generation

I experimented with different LLMs for query generation across various languages. Table 4.12 compares LLaMA2-7B, LLaMA3-8B, and gpt-3.5-turbo as query generators for synthesizing queries in target languages to fine-tune the mContriever model. Although the use of LLaMA3-8B performs competitively with gpt-3.5-turbo, the retrieval performance often falls below zero-shot scores, whereas gpt-3.5-turbo consistently improves over zero-shot. Therefore, I selected gpt-3.5-turbo for query generation throughout this work.

4.4.6.2 Minimum Training Size

Estimating an efficient training size to fine-tune a multilingual ranking model is crucial for budget-constrained scenarios. The number of training examples needed for convergence can vary across target languages. Table 4.13 highlights the impact of increasing training sizes on two dev datasets for fine-tuning the mContriever model. While fine-tuning with 10k training examples occasionally outperforms zero-shot model using 1k examples, a minimum of 1k training examples always consistently improves over the zero-shot model. Therefore, I conducted my experiments using only 1k training examples.

4.4.7 Summary and Limitations

I proposed an extension of DUQGen, called mDUQGen, to efficiently generate synthetic training data for fine-tuning multilingual ranking models. Unlike previous methods that used random document sampling, mDUQGen utilizes document clustering followed by diverse sampling to generate queries across multiple languages. My experiments demonstrate that mDUQGen not only outperforms zero-shot models but also surpasses current state-of-the-art rankers using just 1000 training examples. Consequently, mDUQGen is sample efficient, highly effective, and scalable across multiple languages. My comprehensive analysis further confirms that the core idea behind the DUQGen and mDUQGen approaches is generalizable across various retrieval tasks, domains, and languages.

The effectiveness of mDUQGen may stem from a combination of task transfer learning, domain adaptation, and language adaptation. Initially, task transfer learning is achieved by task fine-tuning a pretrained multilingual model (mT5) on the MS-MARCO dataset. This fine-tuned model is then further refined on MIRACL synthetic training data to incorporate domain and language adaptation. Currently, we lack a task-specific fine-tuning resource like mMARCO (a machine-translated version of MS-MARCO) because task fine-tuning on machine-translated data tends to underperform compared to English fine-tuning due to translation errors. This limitation makes it challenging to isolate and verify the

language adaptation component amid domain and language adaptation, highlighting a potential area for future research—specifically, conducting ablation studies to separate the effects of language and domain adaptation. Nevertheless, it is clear that mDUQGen’s synthetic fine-tuning incorporates both domain and language adaptation.

Additionally, the role of task transfer learning through MS-MARCO could be further examined by directly fine-tuning a pretrained multilingual model on mDUQGen synthetic training data.

$$B = \text{finetune}_{\text{MSMARCO}}(A) \quad (4.4)$$

$$C = \text{finetune}_{\text{mDUQGen}}(B) \quad (4.5)$$

$$C = \text{finetune}_{\text{mDUQGen}}(\text{finetune}_{\text{MSMARCO}}(A)) \quad (4.6)$$

$$D = \text{finetune}_{\text{mDUQGen}}(A) \quad (4.7)$$

Here, A , B , C , and D represent neural models, while MSMARCO and mDUQGen refer to training datasets—specifically, large-scale supervised relevance pairs and synthetic unsupervised domain training data. For instance, A , B , C , and D might correspond to mT5-base, Mono-mT5-base, mDUQGen-reranker_{msmarco}, and mDUQGen-reranker, respectively. In my mDUQGen experiments, I only evaluated Equation 4.6. However, another experiment based on Equation 4.7 could be performed by directly fine-tuning model A using only mDUQGen synthetic training data, skipping MS-MARCO pretraining. This would help to identify the additional task transfer learning impact that MS-MARCO provides as a pretraining stage.

The comparison of results from Equations 4.6 and 4.7 would help to determine whether an extensive pretraining stage is necessary in the transfer learning process. If the performance difference is minimal, we could conclude that mDUQGen sufficiently covers domain adaptation, language adaptation, and task transfer with just 1,000 training samples. However, a larger gap would suggest that MS-MARCO pretraining adds valuable improvements. Therefore, conducting an additional training with Equation 4.7 is essential to improve the effectiveness of task transfer learning in our neural ranking framework.

At a high level, mDUQGen follows a similar pipeline to DUQGen, involving clustering and sampling. As a result, the limitations discussed for DUQGen also be applied to mDUQGen. For instance, using mContriever as a universal language encoder to generate text embeddings before clustering may provide limited document representation quality for certain target languages, since mContriever may not be sufficiently trained as an LLM for all target languages. A potential solution could be to use the embedding from a more robust LLM for an improved document representation, though this would come at a higher cost.

Secondly, the scalability challenges associated with K -Means in DUQGen are also relevant to mDUQGen. However, as previously mentioned, the scalability can be addressed by sampling the entire collection and training the K -Means algorithm only on those samples.

Thirdly, the LLMs have certain limitations in generating high-quality queries across different languages. Not all LLMs excel in every language. For example, LLaMA3-8B produced high-quality queries in certain languages, while gpt-3.5-turbo performed better in others, as indicated by the retrieval performances. Although LLaMA3.1-8B was shown to be effective in multilingual settings across several benchmarks in its official report, its performance on the MIRACL retrieval benchmark was still inferior to that of gpt-3.5-turbo.

Lastly, the LLMs show inconsistencies in query generation across different target languages. Even when prompted correctly, the LLM may produce irrelevant text, such as gibberish text, English or non-target language output, or low-quality content. Repeated prompts or different few-shot examples do

not always resolve this, leading to variable query generation quality and ultimately affecting retrieval performance. While mDUQGen has managed to minimize these issues through its effective clustering and sampling components, these concerns should be thoroughly addressed in the future research.

5 Task-Specific Neural Ranker Training

In information retrieval, the term "relevance" refers to the extent to which a retrieved document satisfies a user's query. Essentially, it measures how well a document aligns with the user's needs or intent. Relevance is a crucial metric when evaluating the performance of retrieval and ranking models. A ranking model performs better when highly relevant documents appear at the top of the results, while its performance diminishes when irrelevant documents rank higher. According to the Cranfield paradigm [135], relevance between a query and a document can be evaluated independently of other potential candidates. Although this assumption may not always hold, it is effective for training neural rankers to optimize relevance matching.

Neural rankers are typically trained using positive and negative pairs to learn how to match relevance effectively. This relevance matching is achieved by contrasting a positive example with its negative counterpart, helping the model capture essential relevance features for the task. For instance, given the query, *"What are the states heavily impacted by Hurricane Helene?"*, a positive document would state, *"Florida, Georgia, North Carolina, South Carolina, Tennessee, and Virginia were devastated by Hurricane Helene on September 26-27, 2024."* A negative document might say, *"Northern and Northeastern states were cleared from Hurricane Helene's impact,"* which does not address the query about impacted U.S. states. In this case, the positive document directly answers the query, while the negative document does not. Various training objectives, like point-wise, pair-wise, list-wise, and contrastive loss, can be used to train neural rankers, with pair-wise and contrastive loss being the most common due to their strong discriminative learning capabilities.

Hard negative documents are more effective than random negative ones when training neural rankers. Hard negatives push the model to learn more discriminative features compared to random negatives. For instance, consider the query about states impacted by Hurricane Helene. A random negative document might be something like, *"Hurricane disasters typically occur at the end of summer in North America,"* which talks about the general timeline of hurricanes but provides no direct answer to the query. In contrast, a hard negative document could be, *"Northern and Northeastern states were cleared from Hurricane Helene's impact,"* which rules out specific regions but still does not directly answer the question. The improved quality of hard negative examples forces the neural ranker to capture fine-grained relevance distinctions, leading to better domain adaptation and enhanced performance across target domains and retrieval tasks.

In addition to the quality of hard negatives, the representation quality of positive labels also plays a critical role in fine-tuning neural rankers for specific tasks. In this thesis, **"task"** refers to the "relevance label notion." Effectively capturing task-specific features can significantly enhance the performance of neural rankers across different target domains. Labels for query-document pairs vary depending on domains, languages, and tasks, particularly based on user intent or the needs of downstream systems. For instance, a document may be topically relevant to a query without directly addressing the user's specific question. Consider the query, *"Where is the CDC located?"*—a document discussing what the CDC does or its functions is irrelevant to the user's intent. In contrast, a document stating, *"The CDC headquarters is in Atlanta, GA,"* directly answers the query, making it highly relevant. Therefore, it is essential to train neural rankers to improve relevance representation tailored to the target domain, language, or task.

One of the recent applications of neural rankers in the context of Generative AI is Retrieval-Augmented Generation (RAG). RAG has demonstrated better performance compared to traditional LLM-only generations that do not use a retriever. It operates on a retrieve-then-generate paradigm, where a neural ranker first retrieves relevant documents, and then a generator (LLM) uses those documents as evidence to

respond to user queries. This mechanism differs from traditional information retrieval, where a ranking system provides the top-k ranked documents, and the user assesses their relevance based on their informational needs. In RAG, however, the generator replaces the role of the user, determining the relevance of the retrieved documents for producing a grounded response. "**Grounded**" refers to ensuring that the generated response is based on factual information retrieved from a document collection. By compelling the LLM to ground its responses on the retrieved documents, RAG enhances the accuracy, reliability, and contextual correctness of the answers. The relevance of the retrieved documents is assessed by the LLM based on their "groundedness," which indicates whether the document contains or supports the essential information needed to generate an accurate response. Therefore, fine-tuning a neural ranker is crucial for optimizing end-to-end RAG response generation.

Task fine-tuning is a valuable method for adapting neural ranking models to new tasks. These models can be integrated with downstream systems to carry out a variety of natural language tasks, including question answering, language generation, event extraction, and fact-checking. To optimize end-to-end performance, it is essential for the ranker to be aligned with the specific task of the downstream system, which may involve an LLM. Consequently, task fine-tuning enhances the alignment of the neural ranker with the intended application. However, several challenges emerge in the effective task fine-tuning of these neural rankers.

Task fine-tuning for neural ranking models is difficult due to the scarcity of training data. Gathering labels across different domains, languages, and tasks is both expensive and time-consuming. High-quality relevance judgments or labels are essential for effectively aligning a ranker with the downstream system to optimize overall performance. For instance, in low-resource areas such as finance or healthcare, third-party annotators may struggle to deliver high-quality annotations. Consequently, human expert annotations are often necessary, which can be costly. Additionally, in cross-lingual retrieval contexts, it is challenging to find human experts who are fluent in both the source and target languages. In a RAG scenario, annotating the top-k candidate documents for each query can be prohibitively costly and time-consuming. To address the challenges of obtaining high-quality task labels, previous research has investigated unsupervised and weakly supervised approaches.

Unsupervised fine-tuning typically does not yield high-quality task labels. Without high-quality task labels, training neural rankers have difficulty in aligning with downstream systems. Previous research has investigated generating synthetic queries and designating the corresponding source original documents as positive examples, while employing hard negative mining to gather negative documents. However, the assumptions underlying these methods often fall short. For instance, a query generated by an LLM does not always correlate with the original document due to inconsistencies and hallucinations in LLM outputs. Additionally, for well-known topics like COVID-19, there may be numerous relevant documents presenting various perspectives, making it inadequate to assign just one positive document for a given query; this limits the neural ranker's ability to learn about multiple facets of relevance. Moreover, hard negative mining can be problematic and may introduce noise. For example, when top-k retrieved documents are compiled and negatives are chosen randomly, those randomly selected documents might not actually be negative at all. Furthermore, the random documents selected from the bottom of the top-k rank are unlikely to represent the most challenging negatives necessary for extracting the most discriminative features during the training of the neural ranker. Given these challenges, relying on unsupervised fine-tuning may not effectively facilitate task alignment between the ranker and the downstream system.

Consequently, a viable solution for task-specific fine-tuning is weakly-supervised fine-tuning. Weak labeling can yield better quality pseudo labels than unsupervised methods. While any teacher model or system can provide weak supervision, using the corresponding downstream system ensures high-quality pseudo labels for task fine-tuning the neural ranker. This improved fine-tuning facilitates better alignment of the neural ranker with the downstream system. In this section, I present three innovative weakly-

supervised approaches designed to task fine-tune neural ranking models across various target domains, languages, and tasks.

5.1 Overview of Proposed Task Fine-tuning Approaches

In this chapter, I explore how to obtain high-quality training labels for neural rankers using effective weakly supervised methods. Thus, I present the primary research question (**RQ3**): *Can ranking models be further adapted for specific downstream tasks using weak or no supervision?* My contributions to RQ3 include (1) **C3**: Continued Pretraining with Contrastive Weak Supervision for Cross-Language Ad-Hoc Retrieval [136], (2) a project on Scalable Cross-Lingual Event Retrieval through Weakly-Supervised Fine-Tuning of Ranking Models, and (3) Task Fine-Tuning of Neural Rankers within a RAG System. The collaborative project **C3**, with Dr. Eugene Yang as the lead author, was published as a short paper at **SIGIR 2022**. My specific contributions involved preparing the cross-lingual relevance pairs (estimating data quality) and developing data loading scripts that support multiple cross-language training datasets and objectives. The other two projects are led by me as the primary author. The second project on Event Retrieval was completed in February 2023 but has yet to be published in a peer-reviewed venue. The third project, which focuses on task-specific fine-tuning of a neural ranker, is a recent initiative, and the manuscript is currently in progress. I aim to submit TFT-RAG work to ACL 2025.

In this chapter, I focus on the effective transfer of knowledge from a source task to a target task within a weakly supervised learning framework 2.1. My methods aim to improve the quality of weakly supervised training labels. The first study, C3, presents an intermediate weakly supervised pretraining framework utilizing contrastive learning, providing a novel approach to transfer learning. Instead of merely acquiring pretrained knowledge for the target domain, C3 introduces an intermediate source domain with a task related to the target domain. By pretraining a neural ranker on this intermediate domain and subsequently fine-tuning it on the target domain, the learning process for the neural ranker is enhanced.

The second study centers on fine-tuning a neural ranker using high-quality weak labels annotated by an information extraction system. This approach directly improves the quality of weak labels for fine-tuning on the target domain or task. This concept is examined in the context of event retrieval and extraction, showcasing the effectiveness of the fine-tuned neural ranker. Finally, in the third study, I expand the application of event extraction to encompass general knowledge-intensive question answering within the RAG framework. Together, these two studies enhance the quality of weak training labels, allowing the neural ranker to adapt more effectively to the target domain or task.

The first two projects deliver two significant findings: (1) weakly supervised labeling improves the performance of fine-tuned ranking models, and (2) a downstream system can act as a high-quality weak annotator to further enhance fine-tuning performance. Building on these insights, my third project explores a generalized approach, examining how to utilize a generator (downstream system) to deliver effective annotations that enhance retriever performance within a RAG framework.

5.2 Task-Specific Contrastive Fine-tuning

Recently, neural ranking models have been fine-tuned by training pretrained language models on domain-specific data. During this domain fine-tuning process, neural ranking models learn both domain-specific and task-specific features. The number of training examples required for fine-tuning can vary depending on the characteristics of the target domain to achieve good performance in learning both domain knowledge and relevance tasks. In Chapter 4, I introduced the DUQGen approach (refer to Section 4.3), which

effectively learns domain knowledge using only 1,000 training examples. However, learning task-specific information is more challenging, especially for smaller pretrained models like BERT [32] and MiniLM [137]. A straightforward solution for task fine-tuning is to use large-scale unsupervised or weakly-supervised unstructured data. Weakly-supervised data can offer better supervision (pseudo-labels) compared to unlabeled data. However, learning task-specific information from weakly-supervised data requires a novel technique. Therefore, the focus of this work is on developing an effective method for task fine-tuning from large amounts of weakly-supervised data.

Compared to monolingual retrieval, task-specific fine-tuning using large-scale weakly-supervised data can be particularly effective in cross-lingual retrieval, due to the added benefit of cross-language transfer. In cross-lingual information retrieval (CLIR), a system returns relevant documents in a foreign language for a query in a source language. In Chapter 4, I introduced the mDUQGen approach (refer to Section 4.4), which efficiently learns both domain and language knowledge across different domains and languages. However, task-specific language features are more subtle in a CLIR setting, making it more difficult to train a neural CLIR ranker. Therefore, it is important to assess the nuanced impact of cross-language task-specific feature learning from large-scale weakly-supervised data in CLIR. Designing a technique that effectively fits this application can be challenging.

5.2.1 Problem Statement

Designing a framework for task-specific fine-tuning of pretrained language models for ranking in cross-lingual information retrieval (CLIR) presents several challenges. First, acquiring domain- or language-specific data necessary for fine-tuning a neural CLIR ranker can be difficult. As a solution, intermediate large-scale pretraining can be beneficial, enabling the training of a zero-shot ranker capable of transfer learning across multiple domains and languages without requiring retraining. Second, there is an abundance of weakly linked data sources, such as Wikipedia articles, which can be leveraged to some extent for warm-start training in relevance ranking.

Thirdly, using translate-train methods to train neural CLIR rankers is a costly process. Recent neural rankers based on pretrained multilingual language models, such as mBERT [122] and XLM-R [138], have demonstrated effectiveness for CLIR when trained on monolingual query-document pairs (*e.g.*, Tamil-Tamil, Chinese-Chinese, and Russian-Russian), allowing for zero-shot transfer [139, 140, 141]. Although training models with translated MS MARCO data (translate-train) is more effective, it is significantly more expensive [139, 142].

Fourthly, many pretrained language models do not inherently ensure that the representations of related text pairs are similar [143]. This necessitates a task-specific fine-tuning process to adjust the representations generated by the pretrained model, bringing them closer together for related or relevant texts. Such processes can be complicated and resource-intensive, as seen in RocketQA [144]. Consequently, an efficient multi-stage training approach, which includes “continued pretraining” in the pipeline for monolingual retrieval, has been proposed [145, 146]. This approach occurs before conducting task-specific fine-tuning with retrieval objectives (as shown in Figure 5.1).

Fifthly, recent studies have indicated that extending the pretraining of a language model with auxiliary objectives before fine-tuning it for the retrieval task can enhance retrieval effectiveness. In contrast to monolingual retrieval, creating a suitable auxiliary task for cross-language mappings presents significant challenges.

Lastly, by construction, the representations for texts conveying similar information in different languages are not inherently similar, as multilingual pretrained models like mBERT and XLM-R do not utilize parallel text during pretraining. In other contexts, integrating alignment information into the retrieval model has proven beneficial for cross-lingual information retrieval (CLIR) [147, 148, 149]. Thus,

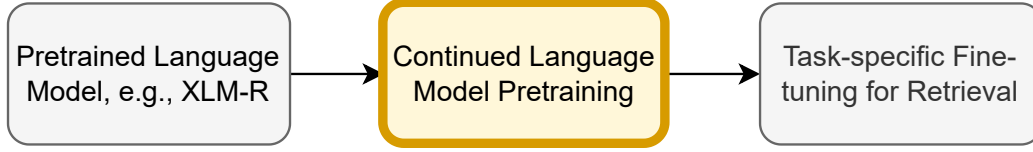


Figure 5.1: Pipeline for training a dense retrieval model. An additional pretraining phase is introduced targeting CLIR.

it can be posited that explicitly fostering token-level similarity during the pretraining phase will improve the effectiveness of CLIR models. Therefore, it can be hypothesized that task fine-tuning with large-scale weakly-supervised cross-language pairs will enhance the effectiveness of CLIR ranker.

5.2.2 C3

To address the challenges mentioned earlier, this work introduces **C3**, a continued pretraining method that utilizes weak supervision with document-aligned comparable corpora. This collaborative project **C3**, with Dr. Eugene Yang as the lead author, was published as a short paper at **SIGIR 2022**. My specific contributions involved preparing the cross-lingual relevance pairs (estimating data quality) and developing data loading scripts that support multiple cross-language training datasets and objectives. This approach aims to make the representations of texts with similar meanings in different languages more comparable through contrastive learning. The continued pretraining phase adjusts an existing pretrained model before it undergoes fine-tuning for the actual retrieval objective, as depicted in Figure 5.1. Specifically, the similarity between pairs of texts is modeled using contrastive learning with token-level embeddings to help the model capture token-level similarities and alignment information. The method employs Wikipedia articles for the relevant language pairs, leveraging cross-language links found within Wikipedia to establish connections. As these Wikipedia articles are based on weak links, this approach falls under weakly-supervised learning. To the best of my knowledge, this is the first work of applying contrastive learning to CLIR pretraining in this manner. The work is tested using high-resource languages that have strong evaluation resources available. However, the ultimate goal is to adapt this weakly-supervised approach for lower-resource languages, where traditional methods relying on parallel text may be impractical. Next, I will present a detailed explanation of the approach, starting with a formal description of the method, followed by a comprehensive overview of the experimental setup, and concluding with a discussion of the results and analysis.

5.2.3 Proposed Work

This section introduces a continued pretraining approach, named C3, which uses contrastive learning to align similar representations of text pairs across different languages. The language model learns to establish a semantic space containing the two languages of interest with meaningful similarity by training with this objective.

Specifically, consider a comparable corpus with linked document pairs (d_i^S, d_i^T) in languages $\mathcal{S}$ and $\mathcal{T}$ (i.e., pairs of documents in different languages containing similar information). Given a list of such document pairs $[(d_1^S, d_1^T), (d_2^S, d_2^T), \dots, (d_n^S, d_n^T)]$, a list of spans $[s_1^S, s_1^T, s_2^S, s_2^T, \dots, s_n^S, s_n^T]$ is constructed by randomly sampling one span from each document.

Let h_i^L be the sequence of token representations of span s_i^L where $L \in \{\mathcal{S}, \mathcal{T}\}$, its SimCLR [150] contrastive loss is constructed as

$$\mathcal{L}_{iL}^{co} = -\log \frac{\exp\left(f\left(h_i^{\mathcal{S}}, h_i^{\mathcal{T}}\right)\right)}{\sum_{j=1}^n \sum_{k \in \{\mathcal{S}, \mathcal{T}\}} \mathbb{1}(i \neq j \wedge L \neq k) \exp\left(f\left(h_i^L, h_i^k\right)\right)}$$

with $\mathbb{1}(\bullet)$ being the indicator function and $f(h_1, h_2)$ being the similarity function between representations h_1 and h_2 . This contrastive loss is similar to the one proposed in coCondenser [145] but encourages the model to learn different knowledge. Instead of sampling pairs of spans from the same document, the pair is constructed by sampling one span from each side of the linked documents. Equation 5.2.3 promotes the representation $h_i^{\mathcal{S}}$ and $h_i^{\mathcal{T}}$ to be closer while discouraging representations of spans in the same language from being similar (since k can be the same as L). This construction pushes the encoder away from clustering text in the same language in the semantic space and pulls the text across languages with similar meanings closer, while retaining distributional robustness by randomly matching the spans in the documents.

To promote token-level similarities, the MaxSim operator proposed in ColBERT [151] is applied as the similarity function $f(h_1, h_2)$. Specifically, the function can be written as

$$f(h_1, h_2) = \sum_{i \in |h_1|} \max_{j \in |h_2|} h_{1i} \cdot h_{2j}^T$$

where $|h_\bullet|$ denotes the number of tokens in the corresponding span and $h_{\bullet k}$ denotes the representation of the k -th token in $h_\bullet$. With this similarity function, the contrastive learning loss flows into the token representation to explicitly promote token alignment in the semantic space.

Finally, $\mathcal{L}_{iL}^{co}$ is combined with the masked language modeling loss $\mathcal{L}_{iL}^{mlm}$ and $\mathcal{L}_{iL}^{cdmlm}$ on span s_i^L from the transformer network and the Condenser head [143], respectively, to train the bottom half of the network more directly. Therefore, the total loss $\mathcal{L}$ can be expressed as

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^n \sum_{L \in \{\mathcal{S}, \mathcal{T}\}} \left[\mathcal{L}_{iL}^{co} + \mathcal{L}_{iL}^{cdmlm} + \mathcal{L}_{iL}^{mlm} \right]$$

5.2.4 Experimental Setup

The experiment follows the workflow in Figure 5.1. In this specific study, English is used as the pivot language for the queries and Chinese, Persian, French, and German as the document languages. However, it is arguable that C3 is generalizable to other language pairs. The rest of the section discusses the experiments' data, models, setup, and results.

5.2.4.1 Datasets

To continue pretraining the off-the-shelf pretrained models with C3, linked Wikipedia articles on the same topic are assembled in different languages. Specifically, CLIRMatrix [152] is leveraged as a retrieval collection that uses the article titles as the queries to retrieve documents for 19,182 language pairs. For each language pair, all query and document pairs with relevance score 6 are extracted, which are the Wikipedia pages on the same topic as asserted by inter-wiki links (one query only has one document with a score of 6 given a specific language pair). These documents are linked to construct the comparable corpus. Document pairs in the number of 586k, 586k, 1,283k, and 1,162k are extracted for Chinese, Persian, French, and German, respectively.

		nDCG@100							nDCG@10							
Retrieval Model	With C3	HC4		NTCIR	CLEF				HC4		NTCIR	CLEF				
		Chinese	Persian	Chinese	Persian	German	French	Δ		Chinese	Persian	Chinese	Persian	German	French	Δ
QT + BM25		0.362	0.354	0.264	0.336	0.419	0.563			0.258	0.251	0.229	0.407	0.379	0.505	
XLM-RoBERTa (base)																
ColBERT	$\times$ $\checkmark$	0.352 *0.444	0.385 0.391	0.249 0.278	0.283 $\dagger$ *0.286	0.510 $\dagger$ 0.521	0.590 0.574	+8%		0.248 *0.345	0.277 0.274	0.223 0.255	0.325 0.337	0.513 $\dagger$ 0.535	0.514 0.482	+11%
DPR	$\times$ $\checkmark$	0.330 *0.395	0.319 0.341	0.218 0.255	0.259 $\dagger$ 0.266	0.467 $\dagger$ 0.503	0.531 0.562	+10%		0.223 *0.287	0.220 0.226	0.184 0.231	0.299 $\dagger$ 0.302	0.449 $\dagger$ *0.523	0.449 0.491	+15%
XLM-align (base)																
ColBERT	$\times$ $\checkmark$	0.425 $\dagger$ *0.483	0.399 0.400	0.303 $\dagger$ 0.330	0.252 0.275	0.523 $\dagger$ 0.528	0.579 0.588	+4%		0.332 $\dagger$ *0.408	0.294 0.280	0.283 $\dagger$ 0.316	0.285 0.321	0.532 $\dagger$ 0.536	0.478 0.499	+6%
DPR	$\times$ $\checkmark$	0.385 0.421	0.366 0.403	0.260 0.286	0.235 $\dagger$ 0.244	0.480 $\dagger$ 0.503	0.581 0.586	+6%		0.300 0.324	0.256 0.312	0.239 0.264	0.265 $\dagger$ 0.279	0.482 $\dagger$ 0.520	0.503 0.506	+8%

Table 5.1: Reranking effectiveness of ColBERT and DPR models with and without our C3 pretraining. The top shows XLM-RoBERTa-base models; the bottom shows XLM-align-base models. Symbols indicate statistically significant differences at $p < 0.05$ by a two-tailed paired t -test with Bonferroni correction for 6 tests, either with and without C3 (*) or between C3 and original BM25 results ($\dagger$). Δ shows the mean relative improvement from C3 across the 6 collections.

For domain-specific fine-tuning, the “small” training triples provided in MSMARCO-v1, consisting of 39 million triples of query, positive, and negative passages, is used

The final retrieval models are evaluated on HC4 [90], a newly constructed evaluation collection for CLIR, for Chinese and Persian, NTCIR [153] for Chinese, CLEF 08-09 for Persian [154, 155], and CLEF 03 [156] for French and German. HC4 consists of 50 topics for each language. NTCIR and CLEF 08-09 Persian contains 100 topics, and CLEF 03 French and German contains 60 topics. The title in English is used as the evaluation queries.

Despite experimenting with relatively high resource language pairs, there is no language-specific component in C3. Therefore, it is arguable that C3 is applicable to language pairs that have similar amount of linked Wikipedia pages.

5.2.4.2 Implementation

The approach is tested with XLM-R-base [138] and XLM-align-base [157], which is a variant of XLM-R-base pretrained with parallel text in 14 language pairs and multilingual text in 94 languages. All text in the experiments is tokenized by Sentence BPE [138], which XLM-R uses.

The spans with a window of 180 tokens are constructed from document pairs. The model is pretrained with C3 for 100,000 gradient update steps with an initial learning rate set to 5×10^{-6} using 4 GPUs with 24GB of memory each. Gradient Cache [158] is leveraged to run with batches of 64 document pairs (16 per GPU).

Two dense retrieval models are tested, namely ColBERT [151] and DPR [159]. After pretraining, each model is fine-tuned with the retrieval objective (either ColBERT or DPR) for 200,000 steps also using 4 GPUs with a learning rate set to 5×10^{-6} for each query-document language pair. For fine-tuning of ColBERT, its original implementation is used, and for fine-tuning of DPR with a shared encoder for queries and documents, Tevatron [160] is used. Both retrieval models are tested in a reranking setting, where all models rerank the top-1000 documents retrieved by BM25 with machine translated queries. The machine translation models for Chinese and Persian were trained using AWS Sockeye v2 Model [161] with 85M and 12M general domain parallel sentences for each language pair respectively. Google Translation

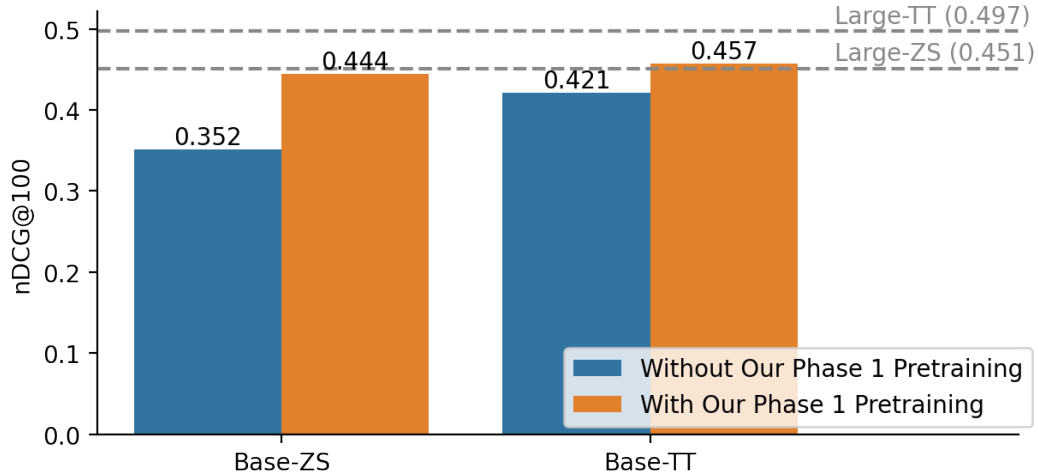


Figure 5.2: ColBERT models with zero-shot transfer (ZS) and translate-train (TT) approaches using XLM-RoBERTa-base on HC4 Chinese test set. The dashed line demonstrate the nDCG@100 value for XLM-RoBERTa-large with both approaches.

is used for German and French.

5.2.5 Results and Discussion

Table 5.1 summarizes the main results of the experiments, which indicate that building dense retrieval models using C3 yields better effectiveness. When starting from XLM-R, C3 provides an 8% relative improvement in nDCG@100 (and 11% in nDCG@10) over directly fine-tuning a ColBERT model. The model benefits from the *warm start* before training with relevance labels by pretraining with a similar objective (MaxSim) with weakly supervised text. On the other hand, a slightly larger gain is observed on DPR, suggesting even retrieval models that score documents with sequence representations (i.e., embeddings of CLS tokens) benefit from a task that promotes token-level similarity.

The improvement in the retrieval effectiveness by C3 is less substantial when starting from XLM-align (at most 6% in nDCG@100 compared to 10%). Since XLM-align is trained with parallel text, its ability to create relationships between text across languages is better than XLM-R, resulting in a diminishing return from investing computation resources in pretraining. Nevertheless, C3 still provides more effective retrieval results across languages.

Among the evaluated language pairs, English-French is particularly interesting. Applying C3 yields negative “improvements” in some cases. As English and French have a close relationship linguistically, it can be suspected that the original XLM-R model, which is not trained with parallel text, already establishes an effective cross-language semantic space. Continued pretraining with C3 may simply not be necessary in such a case. Notably, XLM-align, which initialized its parameters by XLM-R, also yields worse retrieval results (0.590 to 0.579 in nDCG@100 and 0.514 to 0.478 in nDCG@10), which further supports the observation.

Note that all the fine-tuned reranking models underperform BM25 on CLEF Persian collection. After evaluating separately on topics generated in CLEF 2008 and 2009, a discovery is made that the topic characteristics are different between the two (nDCG@100 of 0.421 on 08 and 0.250 on 09 for BM25). Models pretrained with C3 underperform BM25 in 2008 topics, but are at least on par with BM25 on 2009 topics. While this effect deserves further investigation, it was noted that queries for

Lang. Model	Ret. Model	Cond.	Contrastive Similarity		
			None	CLS	MaxSim
XLM-R	ColBERT	✗	0.352	0.389	0.410
		✓	–	0.391	0.444
	DPR	✗	0.330	0.382	0.381
		✓	–	0.368	0.395
XLM-A	ColBERT	✗	0.425	0.482	0.474
		✓	–	0.457	0.483
	DPR	✗	0.385	0.406	0.406
		✓	–	0.408	0.421

Table 5.2: Ablation study on different similarity function used in contrastive learning with and without the Condenser head (Cond.). The values showed in the table is nDCG@100 on HC4 Chinese test set.

this collection were originally created in Persian and then translated into English, possibly initially by nonnative speakers [154, 155]. Perhaps the English queries in 2009 better match the English for which the models have been trained. Nevertheless, C3 still improves the pretrained language models in these cases.

Comparing the average relative improvements (over all six test collections) that result from applying C3, somewhat consistent stronger relative improvements are demonstrated with nDCG@10 than with nDCG@100. From this it can be concluded that the effects of the improved modeling are particularly helpful nearer to the top of the ranked list, where interactive users might be expected to concentrate their attention.

To investigate the initially raised hypothesis regarding the utility of token-level similarity, evaluation on models in which different similarity functions were used as a basis for contrastive learning in continued pretraining is conducted. Using the CLS token in this way is similar to the coCondenser model. Results in Table 5.2 suggest that with the Condenser head, as implemented in the coCondenser model, pretraining with MaxSim similarity as the contrastive learning objective produces better retrieval models. The improvement is minimal without the Condenser head, indicating that token-level similarity benefits from routing information directly to the bottom half of the network. Interestingly, the second-best approach among the four combinations is CLS-based contrastive learning without using the Condenser head, which contradicts the original proposal of coCondenser. However, any continued pretraining is rewarding. Despite the competition among the variants, all language models with continued pretraining outperform their original off-the-shelf version.

Finally, a natural question arises: what if translation of MS-MARCO is affordable so that translate-train model can be used? To investigate, the Chinese translation of the MSMARCO-v1 training triples is utilized from ColBERT-X [139], which can also be accessed via `ir_datasets` [112] with the dataset key `neumarco/zh`¹. Figure 5.2 shows that without C3, the ColBERT model improves from 0.352 to 0.421, which is still worse than zero-shot transfer models trained with C3 for CLIR, suggesting allocating effort to C3 rather than training a translation model when computational resources are limited. When both are affordable, the effectiveness (0.457) is on par with zero-shot transfer a ColBERT model with XLM-R-large (0.451), which is even more expensive to train. With translate-train, ColBERT with XLM-R-large achieves close to 0.5 in nDCG@100 but requires more computational resources to run.

¹<https://ir-datasets.com/neumarco.html#neumarco/zh>

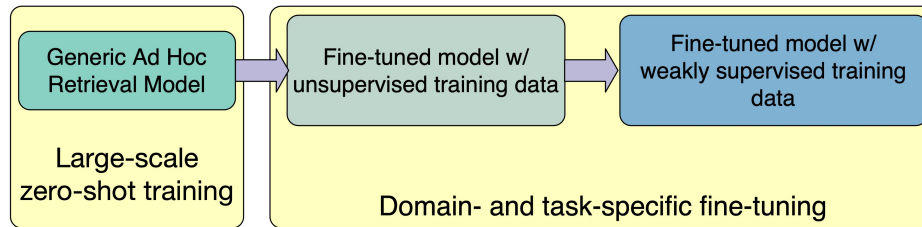


Figure 5.3: Overview of fine-tuning an event ranker model on unsupervised and weakly-supervised training data.

5.2.6 Summary and Limitations

A continued pretraining framework C3 is proposed on a weakly supervised corpus with a contrastive learning objective. It was experimentally showed that the final retrieval models that are fine-tuned from models trained with C3 are more effective than off-the-shelf multilingual models. Further analysis suggests that translate-train can further improve retrieval models fine-tuned from C3-pretrained models. Evaluating with larger models such as XLM-R-large can also provide insight into the robustness of the C3 approach.

The C3 approach has certain limitations regarding the quality of cross-lingual pairs and the selection of hyper-parameters. The weakly labeled pairs are collected from CLIRMatrix, which crawls Wikipedia and automatically connects pages with weak links through a pipeline. The quality of weakly labeled pairs can also affect the training of a neural ranker. Depending on the stronger or weaker labeled annotations, the neural ranker may struggle to learn discriminative relevance signals from positive and negative pairs.

Secondly, the contrastive learning benefits from larger batch sizes for optimal performance. However, due to hardware limitations, smaller batch sizes had to be used and rely on empirical hyper-parameter values based on similar prior experiments. This can be problematic, as the weakly-supervised algorithm may be sensitive to these hyper-parameter choices when learning from noisy data.

Moreover, it would be interesting to extend C3 on some lower-resource languages where Wikipedia articles are limited in such languages. Beyond that, despite being motivated by CLIR problems, C3 might also be applied to monolingual retrieval in cases where documents appear on the same topic that may use different writing styles.

5.3 Event-Aware Task Fine-tuning

Human activities and global events can largely be categorized as events, including everything from organized sports and social gatherings to weather phenomena and natural disasters. Searching for events or temporal information is a crucial and common task in both general search engines and specialized retrieval systems [162, 163, 164, 165]. A traditional system for event retrieval and extraction typically comprises two components: a retrieval system followed by an extraction system. The information retrieval system identifies and returns the top-k relevant documents that contain the events related to the user’s query. These retrieved documents may then be processed further by an information or event extraction (IE) system, which organizes and aggregates the information for analysis or other uses. The IE system extracts key spans, entities, and trigger words that represent the events found in the retrieved documents. Such systems often depend on identified mentions of events within the documents to accurately match the queried event. Therefore, to task-align a retriever against an IE system it is essential to train a neural ranking model that understands event semantics and knowledge to locate documents with rich

event-related information, ensuring that the extraction system can effectively identify the user’s event search.

Numerous significant information needs pertain to events, ranging from organized human activities to natural occurrences or disasters, and can increasingly be expressed in any language. Cross-lingual event information retrieval [166] has gained prominence due to the necessity of collecting information from around the globe, such as in tourism and breaking news, where documents may be written in languages unfamiliar to the searcher. The goal of the cross-lingual event retrieval task is to enable users to specify their events of interest in their native language while retrieving documents or passages that describe those events in other languages. This capability allows users to express their event queries in one language and obtain information in another, thus expanding the pool of accessible information. Consequently, investigating task fine-tuning, particularly event-aware fine-tuning, of neural ranking models within cross-lingual systems proves to be more effective and impactful than in a monolingual retrieval context.

5.3.1 Problem Statement

Cross-lingual event retrieval presents significant challenges, particularly when there is insufficient training data for the specific events or target language documents needed. Obtaining this event data through human annotation is often costly and impractical. Instead, practitioners typically depend on Information Extraction (IE) models to automatically gather this information. However, running an IE model over a large dataset can still be computationally demanding, rendering it impractical for extensive collections or the wider web [167, 168] to precompute, index, and retrieve results offline for faster inference.

Recent developments in cross-lingual event retrieval [166] have enhanced effectiveness by integrating IE components only when reranking results from a sparse retrieval model, such as BM25, using machine-translated documents. However, these initial sparse retrieval models are not specifically designed for cross-lingual event retrieval, leading to a suboptimal set of documents for the subsequent reranker to process.

Substituting the sparse retrieval model in the pipeline with a cross-lingual dense retriever can enhance overall performance [64, 169]. Nevertheless, these dense retrievers are usually trained on English or translated MS-MARCO [64], which consists of web search queries and documents. While these models have demonstrated greater effectiveness compared to earlier retrieval systems, they may experience a mismatch between the training data and the event-search queries used, falling into the domain adaptation problem. Thus, it can be hypothesized that event-aware task fine-tuning of dense retrievers, using a downstream IE system as a weak annotator, will improve the overall effectiveness of an end-to-end IE-IR system.

5.3.2 Event Fine-tuned IE-IR System

To address these challenges, I propose a scalable and effective method for training an event-aware dense retrieval model using weak supervision, eliminating the need for costly IE systems when responding to queries. This approach is designed for scenarios where an IE system, such as a variant of the Semantic Role Labeling (SRL) model, is employed to extract relevant events. It enables the acquisition of event attributes—serving as training data—without requiring human annotation, thus providing machine-generated labels during training. With advancements in natural language processing, SRL can accurately extract events from sentences [170, 171]. The generated event annotations and their associated confidence scores are utilized to create weakly-supervised query-passage pairs that include event signals. Additionally, the training queries are augmented with event types to aid the neural ranker in understanding complex event semantics. Figure 5.3 illustrates the fine-tuning process.

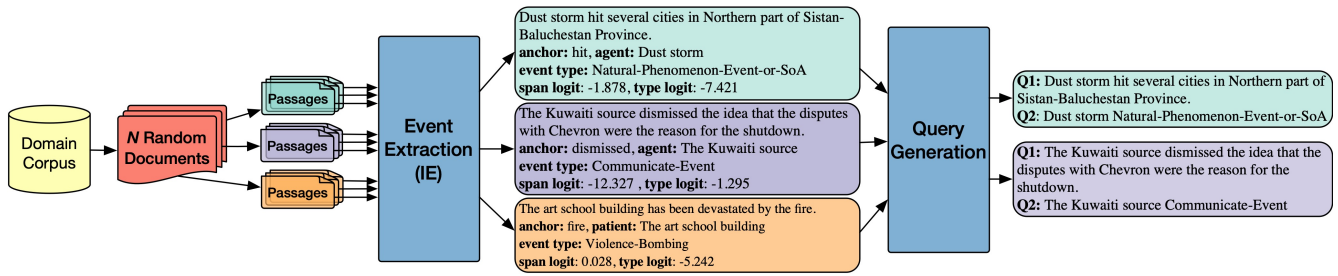


Figure 5.4: An example showing event extraction (IE) and query generation.

Specifically, my approach employs a curriculum learning strategy, where a ranker is progressively fine-tuned using both unsupervised and weakly-supervised training labels. These weakly-supervised labels are generated by annotating a small subset of the corpus with events and entities using a targeted event extraction system. Experiments conducted on four publicly available cross-lingual collections demonstrate that even with a limited amount of training data, sequentially fine-tuning a neural ranker significantly enhances both recall and precision compared to state-of-the-art baselines. This method facilitates rapid adaptation and generalization of event retrieval across new languages, domains, and corpora. Next, I will present a detailed explanation of my approach, starting with a formal description of the method, followed by a comprehensive overview of the experimental setup, and concluding with a discussion of the results and analysis.

5.3.3 Proposed Work

My approach involves fine-tuning a retrieval model using in-domain training data created through unsupervised and weakly-supervised methods. Specifically, the weakly-supervised data is generated by an IE system, which assigns calibrated scores to the data. This section begins with a brief overview of the IE system and the process for generating event scores, followed by an explanation of my methods for generating queries and training data.

5.3.3.1 Information Extraction System

Event Extraction or Information Extraction (IE) systems transform unstructured text into structured data by identifying and extracting key entities, elements, and relationships, while providing a confidence estimate. In my approach, an IE system is utilized to annotate events and compute confidence estimates to assess the quality of downstream annotations. The following sections will cover details about the IE system (Spanfinder) used and its calibrated confidence scores to provide a foundational understanding of these components.

5.3.3.1.1 Spanfinder

The IE system used in this approach is based on the Spanfinder model [170, 172], which includes a contextualized encoder, a BiLSTM CRF tagger, and an MLP head for joint span extraction and labeling. Starting from a virtual root node for each sentence, the Spanfinder model first extracts event *trigger* spans and assigns event labels from a predefined ontology. For each identified event trigger, the model then extracts *argument* spans, assigning labels such as *agent* or *patient*. To provide the downstream retrieval model with more information about prediction confidence, the original Spanfinder model is modified to output logits for span extraction and labeling [173]. A separate IE model is trained for each corpus

configuration. The system is trained on human-labeled in-domain BETTER English data [174], which is further enhanced with machine-translated and aligned "silver" data in the target language [175].

5.3.3.1.2 Confidence Calibration

Neural IE systems often tend to overestimate their confidence levels [176]. While this might not pose an issue when individual confidence scores are used separately, it can create problems when combining these scores, as the ranking may become skewed by the most sensitive components. To address this, for each IE model, a distinct set of calibrators is trained for both label prediction confidence and span extraction confidence using temperature scaling [177] on a held-out development set. During inference, the raw scores from the IE model are adjusted by these calibrators, resulting in more accurate and calibrated confidence scores.

5.3.3.2 Event Score Generation

In an event retrieval task, a query must include at least one identified *event*, with the span mentioning the event trigger referred to as the *anchor*. Entities associated with this event anchor are categorized as either *agent* or *patient*, and each event belongs to a predefined *event type* class.

For a sentence with labeled event anchor and type, the IE system generates two separate probability logits: 1) **span logit**: the likelihood of a span in the sentence being identified as an event anchor, and 2) **type logit**: the likelihood of a given event anchor span being assigned a specific event type. To produce a unified event extraction confidence score for a sentence, a joint probability of these two logits is introduced, since they are calculated independently.

5.3.3.3 Unsupervised In-Domain Training

As an initial step in the framework, an unsupervised fine-tuning phase using the in-domain corpus is carried out, with positive training examples generated as follows. From a randomly selected set of document passages, a sentence is chosen from each passage based on the assumption that passages (*e.g.*, paragraphs) are written *coherently*, meaning that all sentences within a passage are likely to share the same topic and be related. This assumption of local coherence has been noted in prior tasks, such as language modeling [178]. Therefore, the resulting pairs consist of selected sentences and their source passages, which serve as positive examples of relevant query-document pairs. Although this simple query-passage pairing may be noisy, it could assist the ranker in learning domain-specific vocabulary embeddings and topical relevance, which will be explored experimentally. However, the method's benefits are expected to be limited since this unsupervised approach does not account for target events or event-specific queries in retrieval. To address this, a novel weakly-supervised training data generation approach is introduced to enable scalable fine-tuning for event matching and retrieval. A similar approach has been recently used in the unsupervised pretraining of the Contriever [45] model, although my work predates and was independently proposed from the Contriever.

5.3.3.4 Weakly-Supervised Training

I will now outline the main contribution of this approach: a scalable, weakly-supervised approach for training event retrieval models. First, I explain the process of generating automated queries and relevance data using the IE system to identify relevant sentences. This is followed by a generalization step, which

	queries	documents	avg doc len.
CLEF Persian	100	166,774	573
NTCIR Chinese	100	308,832	354
BETTER Arabic	54	864,971	636
BETTER Persian	53	856,167	664

Table 5.3: CLEF, NTCIR, and BETTER collections statistics.

exposes the ranker to a broader set of potential queries. Together, these steps are used to fine-tune the retrieval model.

5.3.3.4.1 Query Generation

The training queries for the weakly-supervised dataset are generated in two forms:

Q1 : a sentence containing an event.

Q2 : a concatenation of its event arguments and event type.

First, a random set of N documents is selected from the target language corpus. Each document is divided into multiple passages, and the sentences within these passages are parsed using the IE system in the target language, with a confidence calibrator applied to refine extractions. For each output sentence, the system checks for the presence of an event and its calibrated score. The event score, as described in Section 5.3.3.2, reflects the confidence in how well the event has been extracted and represents the event request in the sentence. By applying a threshold to the event score, high-confidence event sentences are selected as training queries for Q1.

To help the first-stage retrieval model learn event semantics, the generated queries are enhanced with a generalized query form, **Q2**. This augmented query is essentially a template that combines the event’s argument and type, where the agent or patient spans and the event type are concatenated into a single string. For example, consider the sentence: *"Dust storm hit several cities in the northern part of Sistan-Baluchestan Province."* Here, the anchor, argument, and event type are identified as *"hit"*, *"Dust storm"*, and *"Natural-Phenomenon-Event-or-SoA"*, respectively. Therefore, the generalized query Q2 becomes *"Dust storm Natural-Phenomenon-Event-or-SoA"*, where the event anchor (e.g., *"hit"*) is replaced with the broader event type (e.g., *"Natural-Phenomenon-Event-or-SoA"*).

5.3.3.4.2 Document Selection

Each weakly-labeled event query is matched with its corresponding source passage to create a positive training pair. For each sentence, two positive pairs are generated: one using **Q1** and the other using **Q2**. These pairs demonstrate how events are described in the documents during training, highlighting the model’s ability to recognize different forms of event mentions. Training with Q2 pairs, in particular, helps the model learn how different types of events vary within documents. For each positive pair, a random passage from the corpus, which is not the source passage, is selected as the negative passage.

Retrieval Models	BETTER Persian			BETTER Arabic			CLEF Persian			NTCIR Chinese		
	R@1K	nDCG	RPrec	R@1K	nDCG	RPrec	R@1K	nDCG	RPrec	R@1K	nDCG	RPrec
MT + BM25	0.740	0.507	0.285	0.571	0.298	0.101	0.625	0.488	0.247	0.605	0.388	0.213
MT + BM25 + RM3	0.756	0.536	0.351	0.648	0.329	0.153	0.667	0.521	0.274	0.652	0.408	0.230
ColBERT-X (ZS)	0.817	0.657	0.419	0.481	0.289	0.129	0.701	0.546	0.289	0.688	0.460	0.259
+ Unsupervised	0.834	0.683	0.479	0.580	0.370	0.180	0.740	0.583	0.311	0.717	0.476	0.275
+ Weakly Supervised	0.853	0.703*	0.507*	0.598*	0.375*	0.193*	-	-	-	-	-	-

Table 5.4: Retrieval performance of statistical and neural retrieval models for varying languages and tasks. * indicates the difference between ColBERT-X (ZS) and the full model is statistically significant with 95% confidence using the paired t-test with Bonferroni correction of two tests (across collections within the tasks).

5.3.3.5 Ranker Fine-tuning

My fine-tuning method can be applied to train any learning-to-rank model. To demonstrate its effectiveness, the strong CLIR dense retrieval model, ColBERT-X, is fine-tuned as an event retriever using this scheme. During inference, passage-level relevance can be aggregated into document-level relevance predictions using the MaxP technique [179].

5.3.4 Experimental Setup

Four publicly available cross-lingual collections are used for the evaluation purpose: two general ad-hoc CLIR retrieval collections (CLEF 2001 Persian [92] and NTCIR-8 Chinese Adhoc Retrieval Task [93]) and two event retrieval CLIR tasks (BETTER Farsi (Persian) and Arabic collections). Table 5.3 summarizes the statistics of the respective collections.

The IARPA BETTER (Better Extraction from Text Towards Enhanced Retrieval) program² provides two large collections extracted from CommonCrawl News collections in Arabic and Persian [174]. This collection has a set of event-oriented topics with graded relevance judgments provided by human annotators. The collection is constructed for cross-lingual event retrieval with both queries and examples.

All provided queries are in English. For CLEF and NTCIR (ad-hoc retrieval tasks), the topic title is used as the evaluation query. For BETTER collections (event retrieval tasks), the concatenation of *req-text* and *task-title* is used as the queries.

ColBERT-X[64] source code³ is used to train and evaluate my approach in an end-to-end fashion. The maximum query and document length is kept to 32 and 180, respectively. A zero-shot retrieval ColBERT-X model is trained on MS-MARCO translated cross-lingual pairs for 200k iterations with a learning rate of 3×10^{-6} and a batch size of 32 per GPU. The event retrieval fine-tuning is applied on zero-shot trained models for an additional 1000 iterations each with a batch size of 8 per GPU, where the learning rate was decreased to 1×10^{-6} . Training and evaluation are conducted using four NVIDIA RTX A6000 GPUs.

Due to the focus on the first-stage retrieval, the models are evaluated with recall-oriented measures: Recall at 1000 (R@1K) and R-Precision (RPrec). In addition to the two, nDCG@1K is also reported to demonstrate the improvement in the ranking. To compare with the alternative of using ColBERT-X as the first-stage retriever, the models trained with my fine-tuning scheme are compared against BM25+RM3 with machine-translated queries, which is a strong CLIR baseline [64].

²<https://www.iarpa.gov/index.php/research-programs/better>

³<https://github.com/hltcoe/ColBERT-X>

5.3.5 Results and Discussion

Table 5.4 presents a summary of the dense retrieval performance across the four collections. As the CLEF Persian and NTCIR Chinese datasets are limited to evaluating ad-hoc retrieval tasks, the performance is reported for models that are fine-tuned using *ZS + Unsupervised* methods. The upper portion of the table displays statistical retrieval baselines, while the lower portion features neural CLIR dense retrieval models at various fine-tuning stages.

Each fine-tuned method is incrementally added on top of the previously trained model. When compared to MT+BM25+RM3, the zero-shot trained ColBERT-X model demonstrates significant enhancements across all languages, domains, and metrics, with the exception of R@1k for BETTER Arabic. This finding reinforces the case for further fine-tuning the zero-shot ColBERT-X using the weak labels from the Information Extraction (IE) system during the fine-tuning stages. The unsupervised fine-tuning consistently enhances ZS ColBERT-X, particularly showing relative improvements in R@1k (from 1.7% to 9.9%), nDCG@1k (from 1.6% to 8.1%), and RPrec (from 1.6% to 6.0%) across all languages and domains. This suggests that unsupervised fine-tuning is beneficial for retrieval models. Additionally, incorporating weakly-supervised training also yields consistent advantages, with relative improvements in R@1k (from 1.8% to 1.9%), nDCG@1k (from 0.5% to 2.0%), and RPrec (from 1.3% to 2.8%) across all languages in event retrieval. When comparing the overall performance changes between ZS ColBERT-X and the complete model, the relative improvements are notable in nDCG (ranging from 4.6% to 8.6%) and RPrec (from 6.4% to 8.8%), although BETTER Persian does not exhibit significant improvement in R@1k.

In conclusion, when comparing MT+BM25+RM3 with the sequentially fine-tuned ColBERT-X, the fine-tuned model shows a relative advantage over the traditional baseline, particularly in nDCG@1k (an improvement of 4.6%) and RPrec (ranging from 4.0% to 15.6%). While the traditional BM25+RM3 achieves the highest R@1k in the BETTER Arabic collection, combining the ranks of MT+BM25+RM3 with the sequentially fine-tuned ColBERT-X results in an enhanced R@1k, increasing from 64.8% to 76.1%.

5.3.6 Summary and Limitations

I propose a novel weakly-supervised fine-tuning approach to train a cross-lingual event retrieval model for a given language and domain without requiring human supervision. Specifically, I propose a method for sequentially fine-tuning a ranker using a curriculum of training data with increasing specificity for the event retrieval task. I introduced and evaluated a scalable and effective method for unsupervised in-domain pretraining, followed by fine-tuning the ranker by using calibrated IE extractions to automatically generate weakly supervised, high-quality training data. I empirically showed that my sequential fine-tuning approach outperforms zero-shot SOTA retrieval model significantly on most metrics across languages with only a limited amount of training data. Despite the substantial improvements achieved, further gains may be possible incorporating small amounts of human supervision (*e.g.*, with human-in-the-loop), and by experimenting with multi-task fine-tuning approaches combining gains across target tasks, suggesting promising avenues for future work.

My solution of pseudo-label filtering to enhance the fine-tuning of a neural ranker has certain limitations related to the downstream Information Extraction (IE) system and the quality of the generated training data. First, running the IE system on the top-k ($k=100$) documents for a large number (1000+) of training queries is resource-intensive and expensive. Additionally, in certain applications or cross-lingual settings, a well-trained and calibrated IE system may not be readily available or could be difficult to acquire.

Secondly, when annotating weak candidate documents for training queries using an IE system, it can be challenging to gather the required number of training examples after filtering. For queries related to popular events or topics, many retrieved candidates may be annotated with high-confidence event types by the IE system. However, for more difficult queries, the IE system may struggle to accurately identify events with high confidence. Since training a neural ranker requires a certain number of gradient update steps (*e.g.*, 1000 steps), and a sufficient number of training examples (*e.g.*, a batch size of 32 requires 32,000 examples), it may be difficult to obtain even a single high-confidence positive document for some queries. As a result, there is no guarantee that parsing IE annotations across a large number of queries and documents will yield the desired number of positive training examples.

Lastly, hard negative mining is a critical step in generating training data. A poor selection of hard negative pairs can significantly affect the training of ranking models. Errors in identifying positive documents using an IE system may result in incorrectly classifying them as negative documents, leading to noisy labeling that negatively impacts the training process. Acquiring stronger negative documents could enhance the ranking performance; however, due to the limitations of downstream systems, obtaining such high-quality hard negative labels is challenging. Consequently, it becomes difficult for my weakly-supervised fine-tuning to achieve the performance levels of fully-supervised fine-tuning.

5.4 Task Fine-tuning Neural Ranker in a RAG System

In recent years, Large Language Models (LLMs) have gained increasing popularity for handling a wide range of natural language tasks. However, they often struggle with knowledge-intensive and complex reasoning tasks due to their limited internal knowledge. Retrieval Augmented Generation (RAG) [180] has emerged as a widely explored technique to incorporate external knowledge into LLMs, enhancing their performance on such tasks. Beyond knowledge-intensive challenges, RAG offers additional benefits, such as reducing hallucinations, eliminating the need to retrain LLMs with updated information, and enhancing privacy by keeping proprietary data separate from the model’s internal memory. RAG operates on a retrieve-then-generate approach, where a retriever selects top-k relevant evidence documents, and the LLM generates a response based on the user query and these evidence documents. This method has demonstrated strong performance in overcoming the aforementioned challenges.

5.4.1 Problem Statement

Most state-of-the-art (SOTA) LLMs are large in size and function as black-box systems, accessible primarily through APIs. This necessitates improvements in components like prompt design and evidence retrieval and reranking. For Question-Answering (QA) tasks in particular, the quality of the evidence documents plays a critical role in the LLM’s output. If irrelevant documents dominate the top-k results, it can negatively affect the generation process, leading to more factually incorrect answers. Therefore, it is crucial to implement a more effective ranking system to provide supportive documents that better ground the LLM’s generated responses.

The SOTA neural ranking models are often pretrained on large-scale QA datasets, such as MS-MARCO [2]. While these models demonstrate superior transfer-learning performances across various domains and tasks [1, 126], they still limit the performance of RAG systems, particularly due to the domain and task shifts. These rankers are originally trained for ad-hoc search tasks, where relevance is based on human preferences of raw evidence documents. However, in a RAG system, where the position of human is replaced by LLM, relevance is defined by how effectively the retrieved evidence supports the LLM’s response generation. Task-specific fine-tuning of neural rankers is challenging due to the scarcity

of supervised training data, which is costly and time-consuming to obtain. Although previous studies have proposed weakly supervised and unsupervised training methods, a performance gap persists due to the task misalignment between the ranker and the LLM.

Recently, several studies have focused on pseudo-label distillation and aligning LLMs with rankers [9, 10, 44]. These methods utilize various techniques to generate pseudo labels for fine-tuning ranking models. For instance, they have employed different scoring functions for pseudo-labeling, including attention scores [44], perplexity scores [9, 44], and relevance annotations [10]. While these pseudo-labeling techniques enhance the ranking performance, they do not fully capture the task relevance from the LLM’s perspective. Additionally, these methods require individual query-document passes through the LLM, which is both expensive and time-consuming. Therefore, it can be hypothesized that effective and efficient task fine-tuning a neural ranker will enhance RAG question-answering accuracy.

5.4.2 TFT-RAG

To address the previously discussed challenges and enable task-specific fine-tuning of a neural ranker, I propose a novel approach, called **Task Fine-Tuned RAG (TFT-RAG)**, which introduces a novel technique called **LLM self-referencing**, applied during both the training and inference stages. In training, the ground-truth answer is used to identify the evidence documents as grounded or non-grounded, assigning positive and negative labels accordingly. During inference, the LLM is prompted to self-reference the grounded evidence documents before generating an answer. Additionally, Chain-of-Thought (CoT) [181] prompting ensures that the selected evidence is explained. Unlike the previous methods that require multiple individual query-document passes to the LLM, my approach processes the question and evidence documents in a single pass, significantly reducing the costs. I evaluated my RAG system on the CRAG [51] benchmark and demonstrated a substantial accuracy improvement over the RAG baselines.

5.4.3 Proposed Work

My RAG framework consists of both training and inference stages. I introduce a novel self-referencing technique in both stages, contributing independently to the two main components — the ranker and the LLM — to enhance the overall quality of answer generation.

5.4.3.1 Training-Stage

Training a ranker to improve a QA system is more cost-effective and practically feasible than jointly training a ranker with an LLM. Therefore, during the training stage, a pretrained zero-shot ranker R is sequentially fine-tuned. The goal is to fine-tune the neural ranker for better alignment with the LLM, resulting in a stronger ranker capable of providing more grounded and supportive evidence documents. This, in turn, maximizes the quality of the LLM’s answer generation. To achieve this fine-tuning, I adopt a weakly-supervised approach, using the downstream LLM as a teacher model to generate high-quality task-specific labels for training the neural ranker, and this approach is illustrated in Figure 5.5.

5.4.3.1.1 Self-referencing

LLMs have become adept at identifying reference evidence documents that support and ground their generated answer. Consequently, the performance bottleneck in a RAG system lies in the ability of a ranker to supply high-quality, grounded evidence documents to the LLM. This motivates us to improve the task-specific label generation process during training. In particular, using the LLM itself to annotate

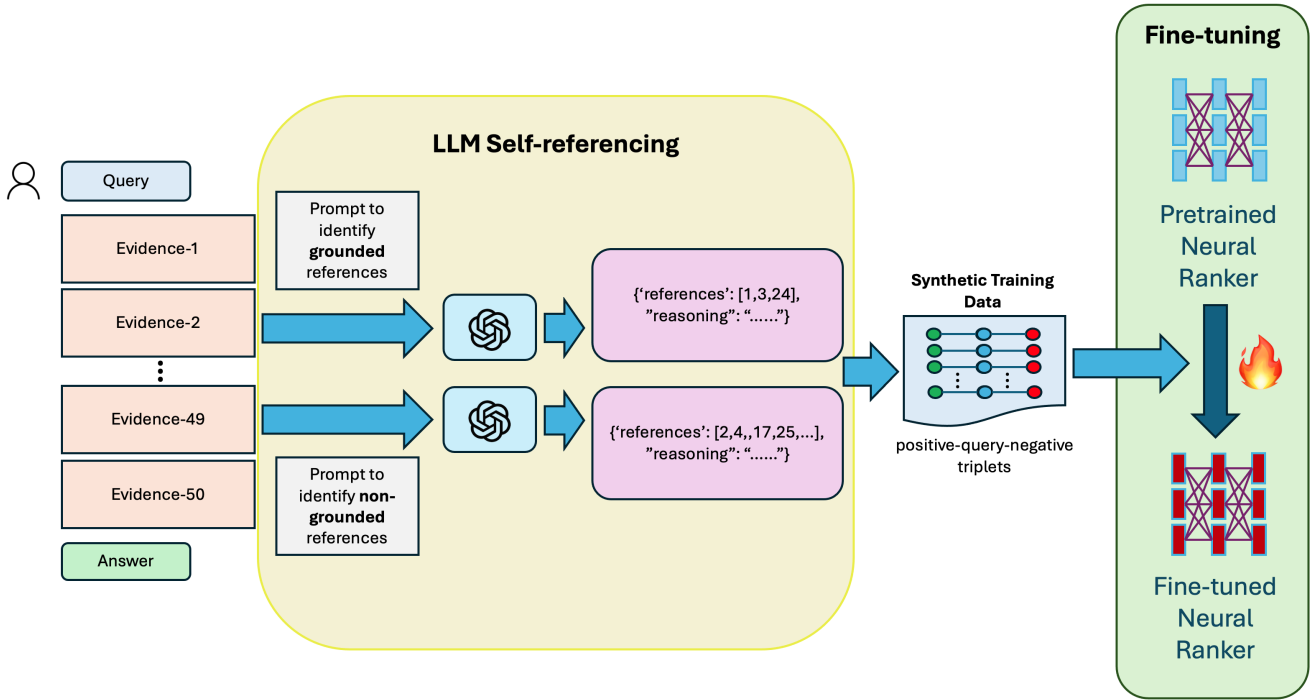


Figure 5.5: **TFT-RAG**: task-specific fine-tuning for neural ranking.

candidate evidence documents proves to be more effective than other pseudo-labeling methods, such as perplexity scores, attention scores, or binary relevance annotations. Moreover, the top-k evidence documents are collectively fed to the LLM in order to generate a grounded answer. Presenting these top-k documents together leads to higher-quality annotations compared to feeding individual document-query pairs, followed by prior works. I refer to this approach as **"self-referencing,"** as it prompts the LLM to identify the evidence documents that maximize the quality of its answer generation.

I present my novel self-referencing technique to generate positive labels for training a neural ranker. In this approach, an LLM is prompted with a question, a ground-truth answer, and top-k evidence documents in a format shown in Figure 5.6a, to obtain label annotations in a single pass. Additionally, the LLM is instructed to provide an explanation for selecting the grounded evidence documents, using CoT prompting to ensure high-confidence label annotations and enhance interpretability. The evidence documents identified by the LLM are then marked as positive labels for fine-tuning the neural ranker.

5.4.3.1.2 Hard Negative Mining

The effectiveness of training a neural ranker heavily relies on the quality of the positive and negative labels in the training data. A hard negative mining approach is typically preferred over random negative pair selection due to the advantages of contrastive learning. In the hard negative mining approach, a first-stage retrieval is performed for each training query to gather top-k candidate documents, and n_{neg} random documents within this set are chosen as hard negatives. However, since the grounded documents can appear at any rank in the top-k, randomly selecting negatives introduces the risk of incorrectly labeling grounded documents as negatives, which can adversely affect the ranker fine-tuning.

To address this potential issue, I propose using an inverse version of the self-referencing technique to identify hard negatives. An algorithm of obtaining hard negatives is described in Algorithm 1. First, LLM is prompted in the same manner as self-referencing but ask it to strictly identify non-grounded

You are given with a question, an answer, and several references, each with an associated reference id. Your task is to determine which references contain the relevant and specific information needed to support the answer.

- Multiple references may contribute to answering the question.
- The identified references should provide concrete, well-supported, and grounded information, not vague statements.

You should mention the reference ids and the explanation in JSON format and nothing else. The JSON format should look like this:

```
{\"references\": [1,3], \"explanation\": \"explanation if any\"}
```

Current Time: {query_time}

Question: {query}

Answer: {answer}

For the question and answer given above, you need to identify the reference ids from the references listed below.

References

id 1 – {document_1}

id 2 – {document_2}

{...}

id 50 – {document_50}

(a)

You are given with a question, an answer, and several references, each with an associated reference id. Your task is to determine which references do not contain the specific information needed to support the answer.

- The identified references may include non-grounded, vague, unsupported, or unclear statements.

You should mention the reference ids and the explanation in JSON format and nothing else. The JSON format should look like this:

```
{\"references\": [2,4,,5,6,7,...], \"explanation\": \"explanation if any\"}
```

Current Time: {query_time}

Question: {query}

Answer: {answer}

For the question and answer given above, you need to identify the reference ids from the references listed below.

References

id 1 – {document_1}

id 2 – {document_2}

{...}

id 50 – {document_50}

(b)

Figure 5.6: Prompt template used for self-referencing to annotate (a) positive and (b) negative grounded documents during the training stage. Black and blue font colored portions indicate system and user prompt messages, respectively.

documents. Figure 5.6b shows the prompt template used for hard negative referencing. And Figure 5.7 presents a sample output generated by the LLM. The identified documents are topically relevant to the user query but may contain non-grounded, vague, unsupported, or unclear information, offering high-quality discriminative signals for training the ranker. An explanation is also requested from the LLM to minimize gibberish or random text in the output. However, preliminary experiments showed that asking for an explanation when generating hard negatives had no effect on the selection of references. Additionally, incorporating self-referencing in hard negative mining can help to reduce hallucination effects, and this argument has been validated in my preliminary experiments. By formatting the LLM output, hard negative candidates can be collected for a query. After the LLM identifies the potential hard negatives, a consistency filtering verification step ensures that no positive labels are mistakenly included, excluding any inconsistent identifiers. Finally, the top- n_{neg} candidates from the remaining negative documents are designated as hard negatives to improve the training of the neural ranker.

```

[[
  "references": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
  36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50],
  "explanation": "None of the provided references contain information about the average number of 3-point attempts per game
  during the seasons Steve Nash made the 50-40-90 club."
},
{
  "references": [2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 41,
  42, 43, 44, 45, 47, 48, 49, 50],
  "explanation": "These references do not contain specific information confirming that Tenet won the Oscar for Best Visual Effects in
  2021."
},
{
  "references": [2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
  40, 41, 42, 43, 45, 46, 47, 48, 49, 50],
  "explanation": "These references do not provide information indicating whether SGML has distributed dividends or not"
},
{
  "references": [13, 29, 31, 34, 38, 39, 41, 45, 48, 50],
  "explanation": "These references do not contain specific information about the gross of Pixar movies that can be used to calculate
  the average gross for the top 3 Pixar movies."
},
{
  "references": [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39,
  40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50],
  "explanation": "These references do not mention the Atlantic Ocean as the cooling source of the Koeberg nuclear power station."
}]

```

Figure 5.7: Sample output generated by LLM during hard negative generation process

5.4.3.1.3 Ranker Fine-tuning

The task-specific fine-tuning data generated through the self-referencing approach can be used to train any neural ranker, enhancing the accuracy of a RAG system. The positive and negative labels generated by prompting an LLM, as outlined in previous subsections, provide the required training data. By applying a pairwise loss, a neural ranker can be fine-tuned to more effectively distinguish between task-specific grounded documents and non-grounded documents.

5.4.3.2 Inference-Stage

CoT prompting has proven effective for solving complex reasoning tasks. As a result, CoT is incorporated into the training stage self-referencing technique to obtain high-quality annotations. During the inference stage, a modified version of self-referencing, using only the user query and top-k evidence documents, can still help guiding the LLM. Since not all top-k evidence documents will ground the generated answer, this simpler referencing approach helps preventing the LLM from being distracted by irrelevant documents. Additionally, the user queries can often contain factual errors (false premises), and ranking highly relevant but incorrect documents could mislead the LLM into generating an incorrect response instead of stating "Invalid question." To address this, an inference prompt, as illustrated in Figure 5.8, is introduced. This prompt asks the LLM to identify and explain the supportive documents before answering the question.

Algorithm 1 Steps to Acquire Hard Negatives

Require: Q : A question

Require: E : A set of candidate references

Require: E^{Pos} : A set of positive labels

Ensure: E^{Neg} : A set of negative labels

- 1: $\text{llm_output} \leftarrow \text{LLM}(Q, E)$
 - 2: $\text{hard_negative_candidates} \leftarrow \text{formatting}(\text{llm_output})$
 - 3: $\text{hard_negative_labels} \leftarrow \text{consistency_filtering}(\text{hard_negative_candidates}, E^{\text{Pos}})$
 - 4: $E^{\text{Neg}} \leftarrow \text{Pick top-}n_{\text{neg}}(\text{hard_negative_labels})$
 - 5: **return** Q, E^{Neg}
-

The Step1 is extracted from LLMJudge [182, 183] benchmark for evidence annotations. Requesting an explanation not only enhances interpretability but also prompts the LLM to safely respond with "I don't know" when unsure, avoiding factually incorrect answers.

5.4.3.3 Tools and Implementation

I built my RAG system on top of the original CRAG⁴ codebase, which utilizes the HuggingFace API for LLM calls and PyTorch for reranker training and inference. The LLaMA3-8B-instruct model was run locally on a NVIDIA H100 GPU with 80GB of memory.

5.4.4 Experimental Setup

In this section, I provide detailed description of my evaluation setting, including datasets, metrics, models chosen for ranking and generation, baselines, and parameter-tuning.

5.4.4.1 Dataset

To evaluate the effectiveness of my RAG system and validate my hypothesis on task fine-tuning, I chose the recently introduced comprehensive RAG benchmark, CRAG [51]. CRAG introduces three distinct tasks:

1. Task-1: Retrieval Summarization
2. Task-2: KG and Web Retrieval Augmentation
3. Task-3: End-to-end RAG

Although all three tasks share the same evaluation questions and ground-truth answers, they differ in the limitation of external data accessibility. Task 1 relies solely on the top 5 web documents for each query, Task 2 includes both the top 5 web documents and access to knowledge graphs, and Task 3 expands to the top 50 web documents plus knowledge graph access. My focus was specifically on enhancing Task 1 (Retrieval Summarization) with my system. Since the benchmark did not include a separate training split, I applied a stratified 50% split across the major categories of examples in the validation and public test sets, totaling 2,706 examples. As a result, I utilized 1,422 examples for training and 1,284 for testing. A random 100 questions are sampled from the training split as a validation split to tune the hyper-parameters.

⁴<https://gitlab.aicrowd.com/aicrowd/challenges/meta-comprehensive-rag-benchmark-kdd-cup-2024>

You are a search quality rater evaluating the relevance of passages and generating an answer based on those rated passages.

Step 1: Passage Evaluation

Given a question and a set of passages, you must provide a score on an integer scale of 0 to 3 for each passage with the following meanings:

- 3 = Perfectly relevant: The passage is dedicated to the question and contains the exact answer.
- 2 = Highly relevant: The passage has some answer for the question, but the answer may be a bit unclear, or hidden amongst extraneous information.
- 1 = Related: The passage seems related to the question but does not answer it.
- 0 = Irrelevant: The passage has nothing to do with the question.

Assume that you are writing an answer to the question:

- If the passage seems related but does not include an answer, mark it 1.
- If you would use any information from the passage in your answer, mark it 2.
- If the passage primarily addresses the question or contains vital information, mark it 3.
- Otherwise, mark it 0.

Step 2: Generate the Answer

After evaluating the passages:

Think step by step and generate the final answer using only the passages rated 1, 2, and 3.

Ignore any information from passages rated 0.

If the question contains factual errors, you MUST reply `invalid question`.

If the question requires real-time, fast-changing, or slow-changing information or if you don't know the answer, you MUST respond with `I don't know`.

- Real-time: The answer to the question changes over seconds (e.g., "What's Costco's stock price today?").
- Fast-changing: The answer to the question changes no more than daily (e.g., "When is Laker's game tonight?").
- Slow-changing: The answer to the question changes no more than yearly (e.g., "Who won the Grammy award last year?").

First, provide your thought process regarding the question. Then, give your final answer using as few words as possible.

Output Format

You should only output both the ratings and the thought process along with the final answer in the following JSON format:

```
{"ratings": "1,3,0,2,3,3,1,0,2,1,2,4,3,0,1,3,2,3,0,1", "thought": "your thinking process", "answer": "your final answer"}
```

Current Time: {query_time}

Question: {query}

Answer the above question using only the passages listed below:

```
-----
# Passages
- {document_1}
- {document_2}
{...}
- {document_50}
```

Figure 5.8: Prompt template used for chain-of-thought answer generation (annotate-then-generate) during the inference stage. Black and blue font colored portions indicate system and user prompt messages, respectively.

5.4.4.2 Evidence Retrieval and Reranking Pipeline

The RAG framework includes both retrieval (with ranking) and generation stages. The CRAG benchmark baselines follow a strict retrieve-then-generate pipeline, as shown in Figure 5.9. The process begins with a user query, which is sent to a first-stage retriever, retrieving the top-5 web documents (5 is imposed by the CRAG organizers). These documents are typically long, so each is divided into smaller chunks using the *BlingFire*⁵ library `text_to_sentences_and_offsets` function. Although some previous works [64] suggest using a windowing method to process long documents, I chose sentence chunking instead, as encoding spans typically produce lower-quality embeddings compared to full sentence embeddings. However, using sentence-wise chunking may cause some degradation in sentence representation quality due to the disruption of the original document structure. Each document yields a variable number of chunks with different chunk sizes. For each query, a reranker then reorders the pooled chunks (pool size also varies across queries) based on their relevance to the query. Finally, the top- K chunks are selected as the final output of the retrieval stage and provided as evidence to the LLM for generation.

⁵<https://github.com/microsoft/BlingFire>

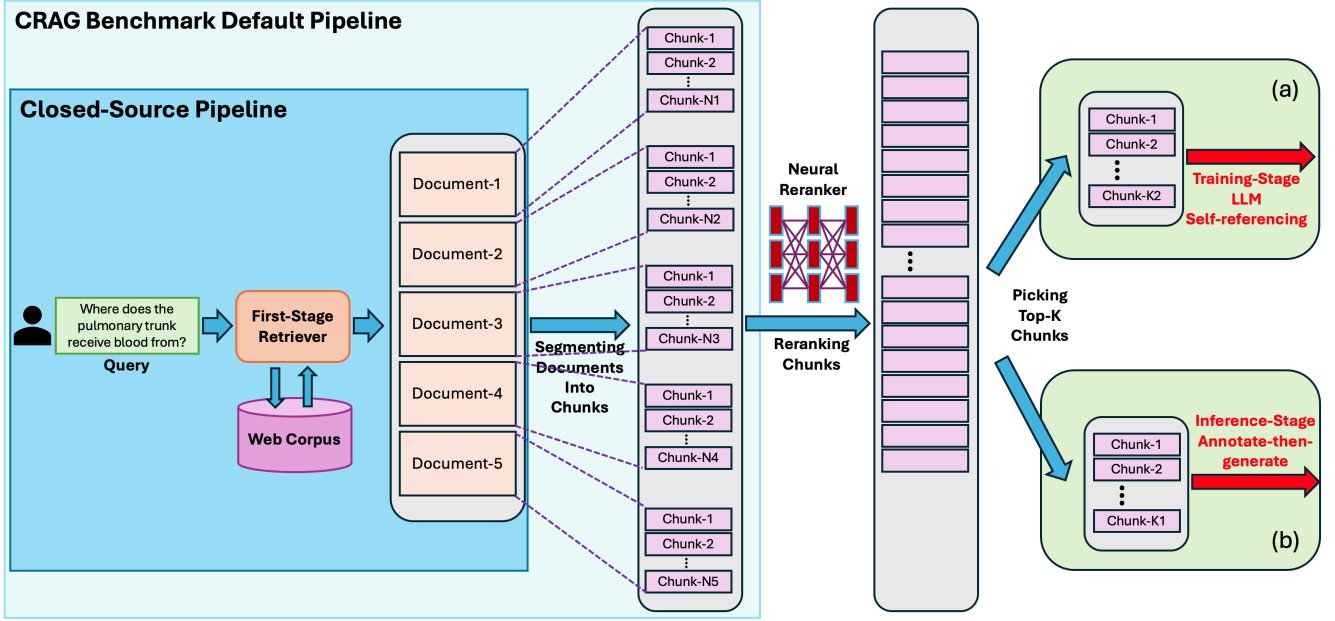


Figure 5.9: Reranking pipeline illustrating the complete retrieval, chunking, and reranking processes for (a) the training-stage and (b) the inference-stage.

Since Task 1 has a limited set of evidence documents (top-5), document-level reranking is not suitable, making chunk-level reranking more appropriate. Additionally, one or a few lengthy documents may exceed the context window of smaller LLMs like LLaMA3, preventing remaining documents information from being utilized in LLM generation. Therefore, chunk-wise reranking is the most effective approach for CRAG Task 1.

5.4.4.2.1 Inference-Stage Reranking Pipeline

For each of the three CRAG tasks, the entire web corpus is not accessible; instead, only the top 5 (for Task 1 and Task 2) or top 50 (for Task 3) web documents are provided per query. This constraint, shown in Figure 5.9 (with component (b)), limits experimentation to reranking rather than retrieval. Consequently, our main experiment focused on assessing end-to-end RAG performance by replacing the reranker after fine-tuning it. The top- $K1$ chunks, post-reranking, were passed to the LLM for an annotate-then-generate process at the inference-stage. To maximize the diversity of chunks in the limited context window of a smaller LLM like LLaMA3 and ensure broader recall coverage, a maximum length cap, L_{max} , was applied to each chunk. This entire inference-stage reranking process introduced a few hyper-parameters to tune: (1) maximum chunk length (L_{max}) and (2) the number of top- $K1$ chunks chosen after inference reranking. Table 5.5 shows the end-to-end RAG performance results after tuning these hyper-parameters on a held-out validation set, with the optimal values found to be $L = 400$ and $K1 = 20$.

5.4.4.2.2 Training-Stage Reranking Pipeline

The training stage adapts the inference-stage pipeline to select top chunks for the self-referencing technique. As shown in Figure 5.9 (with component (a)), for each training question and its given top 5 documents, the documents are first segmented into chunks, reranked with a pretrained neural ranker, and the top- $K2$ chunks are selected for the LLM's self-referencing stage. The pretrained neural ranker is

	Training-stage			Training and Inference-stage				Inference-stage			
	Number of Chunks ($K2$)			Maximum Chunk Length (L_{max})				Number of Chunks ($K1$)			
	30	50	80	100	150	200	400	5	10	20	40
Score _a	3.4	4.0	3.5	3.4	1.9	2.4	6.1	-4.3	-1.9	-1.7	-6.0
Accuracy	33.1	33.6	32.5	33.1	31.8	31.9	33.6	33.6	37.2	39.6	39.4
Hallucination	29.7	29.6	29.0	29.7	29.8	29.4	27.6	37.9	39.2	41.4	45.4
Missing	37.2	36.8	38.6	37.2	38.4	38.7	38.8	28.6	23.6	19.0	15.2

Table 5.5: Experimental results involve tuning chunk hyper-parameters such as the number of chunks and their maximum lengths during training and inference stages. These experiments were conducted on the validation data split using a RAG system incorporating a MonoELECTRA re-ranker and an LLaMA3-8B-instruct generator.

used to identify highly relevant chunks, ensuring to obtain high-quality positive and negative reference annotations from the LLM. Limiting the selection to the top- $K2$ chunks addresses the context window constraints of smaller LLMs like LLaMA3. The same maximum chunk length, L_{max} , used in inference is applied during training. This reranking process in training data generation stage introduces an additional hyper-parameter, the number of top- $K2$ chunks chosen after reranking for self-referencing. As shown in the Table 5.5, I tuned $K2$ on the held-out validation set, with end-to-end RAG performance indicating an optimal value of $K2 = 50$.

5.4.4.3 Automatic Model-based Evaluation

I revised the original CRAG evaluation prompt to better capture semantic matches between ground-truth and predicted answers. CRAG provided a local evaluation script with a default prompt that used ChatGPT (gpt-3.5-turbo) for automatic model-based evaluation. However, after fine-tuning the neural ranker and incorporating CoT-based annotate-then-generate improvements, error analysis revealed significant issues with the original prompt-based evaluation. For instance, answers like "\$5,000,000" and "\$5 million" were incorrectly marked as 'false.' These similar errors were due to insufficient instructions and a lack of diverse few-shot examples, resulting in ChatGPT evaluating only near-exact string matches accurately. As shown in Figure 5.10, I added clearer instructions and more few-shot examples of both 'true' and 'false' cases. This modified prompt improved the automatic model-based evaluation, leading to better overall scores and significantly fewer errors during analysis.

5.4.4.4 Metrics

I assess the performance of my system using the standard metrics defined by CRAG: Accuracy, Hallucination, Missing, and Score_a. These metrics are described below.

$$\text{Accuracy} = \frac{\text{number of correct predictions}(n_c)}{\text{num of total samples}(N)} \quad (5.1)$$

$$\text{Missing} = \frac{\text{number of "I don't know" predictions}(n_m)}{\text{num of total samples}(N)} \quad (5.2)$$

$$\text{Hallucination} = 1 - \text{Accuracy} - \text{Missing} \quad (5.3)$$

$$\text{score}_a = \text{Accuracy} - \text{Hallucination} \quad (5.4)$$

where N , n_c , n_m denotes total number of samples, number of correct predictions, and number of "I don't know" predictions. Hallucination (Equation 5.3) is derived as the number of incorrect predictions, *i.e.*, $(N - n_c - n_m)/N$. My results are based solely on automatic model-based evaluation scores.

5.4.4.5 Models Choices

Since RAG consists of both a ranker and a generator, various combinations of model choices can be explored in experimentation. However, in training, only the ranker is task fine-tuned, while the generator model weights remain frozen. By choosing to fine-tune only the neural ranker, we establish a generic RAG framework that is compatible with both open-source and closed-source (black-box) generator LLMs, eliminating the need to fine-tune the LLMs. For generating weak labels in the training data collection stage, gpt-4o was used as the annotator, as prior studies have shown that models in the GPT series beyond ChatGPT (gpt-3.5-turbo) demonstrate superior annotation performance compared to human annotators [43].

5.4.4.5.1 Ranking Models

To explore different ranking models for a RAG pipeline, I selected MonoELECTRA [30] compared to MonoT5-3B [115]. Although MonoELECTRA is considerably smaller in size compared to MonoT5-3B, in the preliminary experiments their performances are both competitive and closely matched. Therefore, I selected MonoELECTRA to show impressive performances that can be achieved by fine-tuning a smaller neural ranker.

5.4.4.5.2 Generative Models

While CRAG primarily compares two generative models, LLaMA3-70B-instruct and GPT-4-turbo, I opted for LLaMA3-8B-instruct and GPT-4o. The 8B version of LLaMA3 was chosen over the 70B model due to the resource constraints, and GPT-4o was selected as it represents the latest or improved version of GPT-4-turbo.

5.4.4.5.3 RAG Models

My approach is called **TFT-RAG**, which includes the task fine-tuned MonoELECTRA as a re-ranker and self-referencing CoT Inference prompting. I also compare against **TFT-RAG (w/o Inference)**, which includes only the task fine-tuned MonoELECTRA to show the effectiveness of the self-referencing based training data generation.

5.4.4.6 Baselines

I present multiple baselines to highlight the significance of task-specific fine-tuning of a neural ranker in a RAG system and demonstrate the effectiveness of my enhancements compared to a standard RAG system.

1. **LLM-only**: An LLM as a generator with no retrieval system. This system only relies on LLM's internal memory answer zero-shot questions.
2. **RAG (Reproduced)**: The reproduced RAG for the task-1 using the similar codebase.
3. **RAG-MonoELECTRA**: The default MiniLM-L6-v2 re-ranker is replaced with MonoELECTRA.

5.4.5 Results and Discussion

The effectiveness of my TFT-RAG approach can be demonstrated through experiments that highlight improvements in Accuracy alongside reductions in Hallucination. In the first subsection, I present the main experimental results using an end-to-end automatic model-based evaluation. In the second subsection, I explore the sensitivity of Hallucination in response to variations in inference-stage prompts. In both sections, I highlight the strength of my framework in improving Accuracy while either keeping the Hallucination rate stable or allowing a slight increase when necessary.

5.4.5.1 Automatic End-to-end Evaluation

Table 5.6 showcases the key outcomes from my neural ranker fine-tuning experiments. These experimental results provide enough evidence to support the hypothesis of task fine-tuning a neural ranker using labels from a downstream LLM. TFT-RAG delivers an 8% and 15% relative improvement in Accuracy compared to the RAG-MonoELECTRA, across both gpt-4o and llama3-8b respectively, while maintaining similar levels of hallucination as the baseline (RAG-MonoELECTRA). Notably, there is no change in hallucination rates with llama3-8b, and only a slight improvement is observed with gpt-4o. These results suggest that task fine-tuning a neural ranker effectively retrieves highly relevant documents to ground the LLM’s answer generation, without introducing irrelevant content that could lead to undesired hallucination effects.

Secondly, introducing self-referencing during the inference stage significantly enhances RAG performance. Larger LLMs, such as gpt-4o, exhibit a strong ability to reason effectively and follow closely to the given instructions. By applying CoT (Chain-of-Thought) instructions, these LLMs can first identify relevant or supportive evidence documents, and then use only that information to generate the final answer, leading to a notable impact. Table 5.6 supports this argument, showing that TFT-RAG-Inference significantly outperforms TFT-RAG, with improvements of 24% in Accuracy and 4% in Hallucination. Most critically, the Score_a increases by 35% compared to TFT-RAG. This observation strongly affirms that CoT prompting, by directing the model to identify supportive evidence documents before generating an answer, is indeed beneficial.

The llama-8b model faces certain challenges due to its relatively smaller model size. Specifically, it tends to generate repetitive, gibberish text and struggles with reasoning during long output sequences compared to the gpt-4o model. For this reason, the exact TFT-RAG-Inference prompt (Figure 5.8) used for gpt-4o did not work well for llama-8b. Instead, I developed a different but similarly structured CoT prompt for llama-8b, shown in Figure 5.12. Table 5.6 highlights that smaller models, like llama-8b, often face difficulties in self-referencing and annotation tasks (find results marked by * in the table). My results show that llama-8b’s relevance annotations are frequently inaccurate, leading to a noticeable decline in performance during self-referencing. Conversely, gpt-4o produces significantly better annotations, aligning with prior research by Thomas et al. [43] and Rahmani et al. [182], which demonstrate that GPT models can match human labelers in predicting search preferences. This suggests that due to its smaller size (<10B) and pretrained nature, LLaMA models may not be ideal for relevance annotation tasks—a finding that has not yet been extensively validated. This limitation encourages further research into improving smaller models’ answer generation through CoT prompting or inference-stage self-referencing.

Aside from the primary experimental results, an intriguing observation reveals that llama3-8b, as an LLM-only model, significantly outperforms gpt-4o, with a 60-point relative improvement in Accuracy. A review of the predicted answers generated by llama3-8b shows that it excels at directly responding from its internal memory when prompted with a question. In contrast, gpt-4o is more cautious and relies heavily on in-context evidence, leading to an increase in missing ("I don’t know") responses. This difference is

	gpt-4o				llama3-8b			
	Score _a	Accuracy	Hallucination	Missing	Score _a	Accuracy	Hallucination	Missing
LLM-only	23.4	29.0	5.5	65.5	21.3	46.5	25.2	28.3
RAG (Reproduced)	22.0	40.0	18.0	42.0	21.7	46.0	24.3	29.8
RAG-MonoELECTRA	29.3	45.2	16.0	38.8	21.0	47.7	26.8	25.5
TFT-RAG	30.9	48.8	17.8	33.4	28.1	54.9	26.8	18.3
TFT-RAG-Inference	41.8	60.4	18.5	21.1	27.1*	52.2*	25.1*	22.7*

Table 5.6: Performance comparison between **TFT-RAG** and baseline methods. The RAG (Reproduced) employs MiniLM-L6-v2 as the re-ranker, while **TFT-RAG** incorporates fine-tuned MonoELECTRA re-ranker. **-Inference** indicates the addition of self-referencing instructions to the prompt during the inference stage. * indicates a prompt variation (refer Figure 5.12) due to limitations present in llama3-8b.

largely attributed to the pretraining and instruction-tuning processes of these models, which is beyond the scope of this paper.

Finally, llama3-8b demonstrates notable performance improvements only after fine-tuning the neural ranker. As shown in Table 5.6, LLM-only, RAG (Reproduced) with MiniLM, and RAG-MonoELECTRA achieve similar results, with Score_a and Accuracy hovering around 21% and 47%, respectively. This observation is despite performance differences between MiniLM and MonoELECTRA in other standard retrieval benchmarks like BEIR [1]. However, once MonoELECTRA undergoes task fine-tuning, performance significantly increases by 34% in Score_a and 15% in Accuracy. This reinforces our hypothesis again that task fine-tuning is essential to tightly align the neural ranker with the downstream LLM in a RAG system.

My TFT-RAG-Inference demonstrates significantly stronger performance compared to the Winning Solution for Meta KDD Cup '24 [184]. Specifically, Xia et al. [184] reports that their approach achieves Score_a of 28.4%, 42.7%, and 47.8% on three respective tasks. Since I only evaluated my approach on Task-1, where the first-place solution scored 28.4%, my TFT-RAG-Inference reaches 41.8%, a relative improvement of 47%. Even with simple fine-tuning of a neural ranker, the Score_a is 30.9%, reflecting a 9% improvement. My best Score_a is even close to Task-2 of the winning solution (42.7%), which leverages a knowledge graph as an external resource while my solution do not utilize a knowledge graph. This highlights that effectively fine-tuning only the neural ranker and modifying prompts through annotate-then-generate can substantially enhance RAG end-to-end performance with minimal costs.

5.4.5.2 Hallucination Sensitivity to Prompt Variations

Beyond just improving accuracy, another key goal is to minimize the LLM hallucinations. LLMs often hallucinate due to various factors, but the primary causes include difficulty grounding an answer in the contextual evidence provided, such as when faced with false premise questions, outdated or incomplete factual knowledge, or contradictions between parametric (pretrained) and non-parametric (external) knowledge. I primarily studied hallucinations stemming from false premises and the absence of recent information. To counteract hallucination, the LLM is explicitly instructed to respond with "I don't know" rather than generating incorrect answers [51].

Firstly, false premise questions significantly contribute to model hallucinations. A false premise arises when a question contains factual errors [51], such as asking, *"What time does the sun rise in the North during the Summer?"* Here, the question assumes an incorrect fact—that the sun rises in the North, whereas it always rises in the East—making the question itself invalid. To help an LLM identify a

Prompt Components	Generate Answer	Original Instructions from CRAG							
	Passage Evaluation	$\times$	RG-{2,3}/4L				RG-{1,2,3}/4L		
	Few-shot Examples	$\times$	$\times$	Invalid Q	Dynamic Q	$\times$	Dynamic Q	Invalid Q	Dynamic Q + Invalid Q
Experiment Name		P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈
Metrics	Score _a	30.9	32.0	37.2	38.6	41.8	37.1	37.9	38.0
	Accuracy	48.8	57.9	59.9	60.3	60.4	58.2	58.0	57.9
	Hallucination	17.8	25.9	22.7	21.7	18.5	21.0	20.2	19.9
	Missing	33.4	16.1	17.4	18.1	21.1	20.8	21.8	22.3

Table 5.7: Performance comparison of various inference-stage prompt configurations. Prompts consist of three main components: Passage Evaluation, Answer Generation, and Few-shot Examples. An $\times$ indicates the absence of a specific component. RG-*/4L refers to Relevance Grading across four levels, with 2,3 and 1,2,3 denoting the selection of evidence based on those respective grades for final answer generation. Q refers to questions.

false premise, it must rely on both its internal knowledge and any available external context or evidence. However, this task can be challenging due to the vast amount of world knowledge an LLM must parse to determine which facts are accurate. To mitigate hallucinations caused by false premise questions, the model can be guided through specific instructions and few-shot examples. These prompts instruct the model to respond with *"invalid question"* when it detects a false premise, or *"I don't know"* when it is unsure.

Secondly, questions requiring dynamic temporal information can lead to increased hallucinations in LLMs. These models might attempt to respond to queries involving constantly changing data, such as real-time, fast-changing, or slow-changing facts. However, if the provided context lacks the most up-to-date information, the LLM may produce inaccurate responses. The CRAG [51] benchmark defines four types of dynamic questions, which are outlined here for reference:

1. **Real-time:** Answers fluctuate by the second (*e.g.*, "What's Costco's stock price right now?").
2. **Fast-changing:** Answers update no more than daily (*e.g.*, "When is the Lakers' game tonight?").
3. **Slow-changing:** Answers change annually or less frequently (*e.g.*, "Who won last year's Grammy?").
4. **Static:** Answers do not change over time, such as a person's birth date.

Real-time and fast-changing questions often require API calls for accurate responses. For slow-changing queries, even though the information updates over a longer period, LLMs can still struggle without recent context. To mitigate the risk of generating incorrect answers, it is advisable to instruct the LLM to respond with *"I don't know"* when lacking recent information.

To evaluate the challenges of false premise and dynamic questions, I conducted experiments with various inference-stage prompt modifications to assess their effect on hallucination sensitivity. The experiments used a RAG system paired with a task fine-tuned MonoELECTRA re-ranker, and the variations in the inference prompt are listed below:

P₁: LLM using the original answer generation prompt.

P₂: LLM using a Chain-of-Thought (CoT) prompt followed by the original answer generation prompt, as illustrated in Step 1 of Figure 5.8, where only passages rated 2 and 3 are considered for generating the final answer.

P₃: Few-shot examples of invalid questions are added to P₂.

P₄: Few-shot examples of dynamic questions are added to P₂.

P₅: A modified version of P₂, where passages rated 1, 2, and 3 are used for generating the final answer, leading to more granular evidence grounded generations.

P₆: Few-shot examples of dynamic questions are added to P₅.

P₇: Few-shot examples of invalid questions are added to P₅.

P₈: Few-shot examples of both dynamic and invalid questions are added to P₅.

These prompt variations, particularly in instructions and few-shot examples, were employed to explore how hallucination sensitivity and accuracy were influenced by re-ranker choices and prompt modifications.

Table 5.7 presents the performance sensitivity of a RAG system across different inference-stage prompt variations. Identifying the optimal prompt is crucial for mitigating hallucination rather than just enhancing accuracy, as tracked by Score_a. Implementing the CoT prompt (*e.g.*, RG-2,3/4L in P₂) improves Accuracy compared to the fine-tuned neural ranker-based RAG (P₁), but also increases Hallucination by 45%. Including few-shot examples for invalid and dynamic questions enhances Accuracy and reduces Hallucination rates. For instance, invalid (P₃) and dynamic (P₄) questions boost Accuracy by 3% and 4%, respectively, while lowering Hallucination by 14% and 20% compared to the baseline (P₂) without few-shot examples.

Additionally, using passages rated 1, 2, and 3 for generating the final answer improves Accuracy by 4% and significantly reduces Hallucination by **40%** compared to only using passages rated 2 and 3. This underscores the impact of finer-grained annotations in providing high-recall information to support more grounded answers. Further experiments with modified CoT prompts (P₅), including few-shot examples for dynamic (P₆), invalid (P₇), or both types (P₈) of questions, showed mixed results, with adverse effects on both Accuracy and Hallucination.

Ultimately, while prompt optimization helps improve RAG performance, Hallucination remains highly sensitive to prompt variations. A clear takeaway from these experiments is that finer-grade annotations and incorporating as much relevant in-context evidence documents during inference significantly improve Accuracy and reduce Hallucination.

In conclusion, mitigating hallucination in LLMs can be effectively achieved through proper prompt engineering unless the model's weights are fine-tuned. Hallucination control depends on two key factors: (1) the effectiveness of the re-ranker in retrieving top-ranked relevant evidence, and (2) the design of prompt instructions. As discussed in the Results 5.4.5 section, fine-tuning a neural ranker significantly improves accuracy while keeping hallucination levels almost stable. However, during the inference stage, altering prompts—such as using Chain-of-Thought (CoT) prompting—can lead a non-instruction-tuned LLM to hallucinate. Despite this, by carefully selecting CoT instructions, providing clear descriptions, and using relevant few-shot examples, I was able to minimize hallucination throughout the RAG system optimization.

Question Types		Simple	Simple w. Condition	Set	Comparison	Aggregation	Multi- hop	Post-processing heavy	False Premise
# of Test Questions		754	407	249	333	315	231	108	309
Accuracy									
llama3-8b	RAG-MonoELECTRA	42.3	49.5	76.9	65.5	52.0	42.2	61.3	12.7
	TFT-RAG	51.4 (+21%)	59.1 (+19%)	77.8 (+1%)	69.6 (+6%)	56.6 (+9%)	52.0 (+23%)	58.1 (-5%)	23.3 (+84%)
gpt-4o	TFT-RAG	47.0	52.5	80.3	61.9	44.7	50.0	77.4	6.0
	TFT-RAG-Inference	52.5 (+12%)	63.1 (+20%)	86.3 (+7%)	60.1 (-3%)	64.5 (+44%)	59.8 (+20%)	80.6 (+4%)	48.0 (+700%)
Hallucination									
llama3-8b	RAG-MonoELECTRA	31.7	29.3	15.4	20.8	25.7	34.3	22.6	24.0
	TFT-RAG	30.3 (-4%)	25.8 (-12%)	17.1 (+11%)	19.0 (-9%)	25.0 (-3%)	38.2 (+11%)	22.6 (-0%)	30.7 (+28%)
gpt-4o	TFT-RAG	21.6	22.7	6.8	8.3	21.1	14.7	6.5	22.7
	TFT-RAG-Inference	24.3 (+13%)	22.2 (-2%)	7.7 (+13%)	6.5 (-21%)	14.5 (-31%)	19.6 (+33%)	6.5 (-0%)	27.3 (+21%)

Table 5.8: End-to-end RAG performances measured across different question types on test split. For each row, the baseline is taken from the row immediately above, skipping one. Specifically, rows 2, 4, and 6 are considered, with the baselines being rows 1, 3, and 5, respectively. In the Accuracy sub-table, (+%) and (-%) indicate the rounded integer relative improvement or decrease in Accuracy compared to the baseline. In the Hallucination sub-table, (+%) and (-%) represent the rounded integer relative increase or decrease in Hallucination compared to the baseline. Positive changes in Accuracy and negative changes in Hallucination are highlighted in blue, while negative changes in Accuracy and positive changes in Hallucination are highlighted in red, with blue representing better performance and red indicating worse performance.

5.4.5.3 Analysis across Question Types

To further demonstrate the effectiveness of my approach, I conducted an analysis of end-to-end RAG performance across different question types. By evaluating Accuracy and Hallucination for each question type, I aimed to identify which types benefit or are challenged by the new techniques: task fine-tuning neural ranker and annotate-then-generate. Table 5.8 displays RAG performance across question types to illustrate the impact of these techniques.

First, task fine-tuning the neural ranker enhances Accuracy across nearly all question types when using a smaller llama3-8b model, without significantly increasing hallucination. In particular, the task-aligned neural ranker provides high-quality, grounded documents to the LLM, resulting in substantial improvements for question types requiring multiple documents. For instance, “Simple with Condition” and “Multi-hop” questions show relative gains of 19% and 23%, respectively. Simple questions requiring only a single evidence document also experience a notable 21% improvement.

Additionally, accuracy for “False Premise” questions improves dramatically by 84%, indicating that enough factual documents are being supplied to enable the LLM to recognize invalid questions. On the downside, fine-tuning the neural ranker can also increase Hallucination rates for “Multi-hop,” “False Premise,” and “Set” questions. Although some highly ranked documents may occasionally confuse the LLM and lead to incorrect predictions, the substantial Accuracy gains generally outweigh these negative impacts on hallucination.

Secondly, Table 5.8 highlights the effectiveness of my annotate-then-generate technique applied at the inference-stage. Using this approach, gpt-4o demonstrates improvements across nearly all question types, often resulting in significant performance gains. For instance, for questions requiring multiple documents, such as “Simple with Condition,” “Aggregation,” and “Multi-hop,” relative accuracy improvements of 20%, 44%, and 20% were observed, respectively. Notably, prompting the LLM to rate evidence documents before answering led to a striking 700% relative Accuracy improvement for “False Premise” questions,

enabling the model to more accurately distinguish invalid queries.

While the annotate-then-generate approach sometimes raised hallucination rates for certain question types, it is possible that errors in the LLM's self-assessment led to misleading evidence that occasionally resulted in incorrect predictions. However, I also saw different other scenarios where my best model makes incorrect predictions.

I analyzed the inaccurate predictions made by the gpt-4o-based TFT-RAG-Inference model on "Set" and "Multi-hop" question types, as these had a notable increase in hallucination. For "Set" questions, an example such as *"In 2022-12, which teams were able to defeat the Miami Heat in head-to-head matchups?"* had the ground-truth answer *"Chicago Bulls, Denver Nuggets, Detroit Pistons, Indiana Pacers, Memphis Grizzlies, San Antonio Spurs,"* while TFT-RAG-Inference (gpt-4o) predicted only *"Denver Nuggets."* Reviewing the top 20 reranked references revealed that no sources mentioned the full list of teams in the ground-truth answer, indicating a performance limitation. Another example, *"What teams are part of Group C in the 2023-24 UEFA Champions League?"* had the ground-truth answer *"Real Madrid, Napoli, Braga, Union Berlin,"* but TFT-RAG-Inference (gpt-4o) incorrectly answered with *"Benfica, Inter Milan, Red Bull Salzburg, and Real Sociedad."* Again, the top 20 references lacked relevant information on the correct teams, instead providing outdated Champions League data. This demonstrates a limitation: when the model lacks accurate references, it may still attempt to answer the question, relying on incorrect information.

Examining "Multi-hop" question errors reveals additional limitations in recall performance. For instance, the question *"On Langeland, what side of the road do people drive on?"* has a ground-truth answer of *"right,"* yet TFT-RAG-Inference (gpt-4o) predicted *"left."* Although a reference in the top 20 mentioned that Langeland is in Denmark, none addressed the driving side, highlighting that even one missing piece of critical information can lead to an incorrect answer. Another example, *"Who starred in the film that won the Best Picture Oscar in 2018?"* has *"Sally Hawkins"* as the ground-truth answer, but TFT-RAG-Inference (gpt-4o) answered *"Frances McDormand."* Here, none of the top 20 references mentioned either name, underlining the importance of high recall for multi-hop questions, where all relevant details are crucial for accuracy.

Overall, this analysis across different question types underscores that both task-specific neural ranker fine-tuning and the annotate-then-generate techniques significantly enhance performance in multi-document question answering and in identifying invalid (False Premise) questions.

5.4.5.4 Examples of Incorrect Predictions

Hallucination occurs when an LLM (generator) generates incorrect answers. These errors may arise from a lack of grounded evidence, conflicting information, outdated internal or external data, or the absence of real-time information. Figure 5.11 illustrates examples of incorrect predictions generated by my best system. Comparing these predictions with the ground-truth answers reveals some noteworthy observations that can build foundations for future explorations.

For instance, in Question-1, the generator's incorrect response stems from a lack of supportive documents. Similarly, in Question-2, the generator guessed "Alabama River" as the longest river in Alabama. This incorrect prediction indicates an absence of supportive evidence from the top 20 reranked references for the correct answer, Tennessee River.

Questions-3 and -4 demonstrate cases of invalid questions or predictions. In Question-3, the generator responds with "Avengers: Endgame" when asked about Captain America's use of a glove to stop Thanos. Here, the generator associates Captain America's fight with Thanos but overlook that the question's mention of a "glove" makes it invalid. In Question-4, although a correct answer exists, the generator marks it as invalid due to its reliance on available information, where a simple "I don't know" would be

more appropriate.

Other questions in the Figure 5.11 require dynamic information access. Questions-5 and -6, for example, involve real-time or rapidly changing information, such as current or past stock prices. Without access to an updated corpus or real-time APIs, the generator predictions are likely to be incorrect, increasing hallucination rates. The CRAG benchmark’s Task-2 and Task-3 specifically address these needs by incorporating external knowledge bases and real-time APIs for such questions. Expanding my RAG approach to cover these tasks should enhance its ability to answer dynamic questions accurately.

5.4.6 Summary and Limitations

I introduced TFT-RAG to efficiently generate high-quality weakly supervised training data for fine-tuning a neural ranker. In contrast to previous approaches, I utilize a self-referencing technique that allows us to directly prompt the LLM for grounded and supportive documents in a single pass, enabling us to obtain both positive and negative training pairs. Additionally, I incorporate a CoT based self-referencing during inference to further enhance the end-to-end performance. My experiments on a comprehensive RAG benchmark underscore the significance of self-referencing and the effectiveness of task-specific fine-tuning for the neural ranker. As a result, I demonstrate that the task-specific self-referencing from an LLM can improve the alignment of retriever and generator within a RAG system.

TFT-RAG has certain limitations due to the pretrained nature of the underlying LLM, particularly its sensitivity to prompt variations. This sensitivity can directly affect the quality of task label generation during the self-referencing process, potentially leading to erroneous annotations, especially for low-confidence labels. However, my approach mitigates this risk by using a two-step prompting process to request positive and negative evidence separately, followed by inconsistency filtering. Another potential solution involves calibrating the LLM’s confidence scores before generating high-confidence labels, which can help improve label accuracy.

Task:

You are given a Question, a model Prediction, and a list of Ground Truth answers, judge whether the model Prediction matches any answer from the list of Ground Truth answers. Follow the instructions step by step to make a judgement.

1. If the model prediction matches any provided answers from the Ground Truth Answer list, "Accuracy" should be "True"; otherwise, "Accuracy" should be "False".
2. If the model prediction appears to be a different format of any provided answers from the Ground Truth Answer, "Accuracy" should be "True". You should not perform an exact string match but a semantic match.
3. If the model prediction appears to be a superset or subset in the meaning (e.g. US and Atlanta) of any provided answers from the Ground Truth Answer, "Accuracy" should be "True".
4. If the model prediction correctly generates one or more items from the Ground Truth Answer list, "Accuracy" should be "True".
5. Even if one of the model prediction is incorrect, but other model prediction answers matches any provided answers from the Ground Truth Answer list, "Accuracy" should be "True".
6. If the model prediction says that it couldn't answer the question or it doesn't have enough information, "Accuracy" should always be "False".
7. If the Ground Truth is "invalid question", "Accuracy" is "True" only if the model prediction is exactly "invalid question".

Output:

Respond with only a single JSON string with an "Accuracy" field which is "True" or "False".

(a)

Examples:

Question: how much does a super bowl ad cost?
Ground truth: "\$7 million for 30 seconds of ad time."
Prediction: "\$7 million"
Accuracy: True

Question: what was the average budget for all movies in the harry potter series?
Ground truth: "\$147,375,000"
Prediction: "\$144.375 million"
Accuracy: True

Question: can you tell me where kanye west is originally from?
Ground truth: "us"
Prediction: "Atlanta, Georgia"
Accuracy: True

Question: cwhat is the name of the movie that has been honored with the most academy awards?
Ground truth: "the lord of the rings: the return of the king"
Prediction: "Titanic, Ben-Hur, The Lord of the Rings: The Return of the King"
Accuracy: True

Question: which female tennis players have won all four grand slam tournaments in a single calendar year?
Ground truth: "maureen connolly, margaret court, and steffi graf have won all four grand slam tournaments in a single calendar year."
Prediction: "Maureen Connolly, Steffi Graf"
Accuracy: True

Question: how many seconds is 3 minutes 15 seconds?
Ground truth: "195 seconds"
Prediction: "3 minutes 15 seconds is 195 seconds."
Accuracy: True

Question: Who authored The Taming of the Shrew (published in 2002)?
Ground truth: "William Shakespeare, Roma Gill"
Prediction: "The author to The Taming of the Shrew is Roma Shakespeare."
Accuracy: False

Question: Who played Sheldon in Big Bang Theory?
Ground truth: "Jim Parsons, Iain Armitage"
Prediction: "I am sorry I don't know."
Accuracy: False

Question: {query}
Ground truth: {ground_truth}
Prediction: {prediction}

(b)

Figure 5.10: Prompt template used for automatic model-based evaluation. (a) and (b) are combined during evaluation (split here for space constraints). Black and blue font colored portions indicate system and user prompt messages, respectively.

Question-1: in 2004, which animated film was recognized with the best animated feature film oscar?

ground_truth: finding nemo

prediction: The Incredibles

Question-2: how long is the longest river in alabama?

ground_truth: the longest river that flows through alabama, the tennessee river, is 652 miles long.

prediction: The Alabama River stretches approximately 631 miles.

Question-3: what movie did captain america use the glove to stop thanos?

ground_truth: invalid question

prediction: Avengers: Endgame

Question-4: for the fall of saigon, can you tell me who was the main director?

ground_truth: michael dutfield

prediction: invalid question

Question-5: did tesla's stock perform better than nvidia's stock today?

ground_truth: no

prediction: Nvidia's stock performed better than Tesla's stock today.

Question-6: what company in the dow jones is the best performer today?

ground_truth: salesforce

prediction: Walmart (WMT) was one of the biggest gainers today, up 1.6%.

Figure 5.11: Example questions, ground truth answers, and incorrect gpt-4o predictions. These gpt-4o predictions are generated by my RAG system, which includes a task fine-tuned neural ranker and the annotate-then-generate prompting technique.

Given a question and a set of passages, assume that you are writing an answer to the question.

Step 1: You must select supportive and helpful passages that contain the relevant and specific information needed to generate an answer.

Step 2: Generate the final answer using only the passages selected in Step 1.

Multiple passages may contribute to answering the question.

The identified passages should provide concrete, well-supported, and grounded information, not vague statements.

Ignore any information from the remaining passages.

You should only mention the passage ids and not the explanation.

If the question contains factual errors, you MUST reply `invalid question`.

If the question requires real-time, fast-changing, or slow-changing information or if you don't know the answer, you MUST respond with `I don't know`.

- Real-time: The answer to the question changes over seconds (e.g., "What's Costco's stock price today?").

- Fast-changing: The answer to the question changes no more than daily (e.g., "When is Laker's game tonight?").

- Slow-changing: The answer to the question changes no more than yearly (e.g., "Who won the Grammy award last year?").

Output Format

You should only output the passage ids along with the final answer, and nothing else. You SHOULD NOT explain.

Your output should be in the following JSON format:

```
{\"passages\": [1,3], \"answer\": \"Steve Jobs\"}
```

Current Time: {query_time}

Question: {query}

Answer the above question using only the passages listed below:

Passages

- {document_1}

- {document_2}

{...}

- {document_50}

Figure 5.12: Prompt template used for chain-of-thought answer generation (annotate-then-generate) during the inference stage only for llama3-8b model. Black and blue font colored portions indicate system and user prompt messages, respectively.

6 Conclusions and Future Work

My thesis research focuses on domain adaptation problem of training neural ranking models with minimal supervision and introduces several innovative unsupervised and weakly-supervised methods for training state-of-the-art neural rankers in new domains. This thesis report is structured into chapters. The first chapter serves as an introduction to the thesis topic and outlines the motivations to pursue this research area. The second chapter provides a comprehensive review of related work, covering key areas such as transfer learning, domain adaptation, neural ranking models, and both unsupervised and weakly supervised techniques for training these rankers. It also explores the role of large language models (LLMs) in ranking systems and delves into the importance of neural rankers in retrieval-augmented generation (RAG) systems. The subsequent three chapters are primarily focused on addressing individual Research Questions (RQs). They lay the foundations for my thesis research by providing detailed discussions on the existing out-of-distribution issues encountered in the current domain adaptation approaches. Additionally, those RQs propose effective, efficient, and robust solutions aimed at enhancing retrieval performance with minimal supervision.

My approaches aim to address the challenges of transfer learning and domain adaptation across various domains, languages, and tasks. Building effective search and ranking models across these diverse areas presents significant challenges, and creating a unified framework that spans all three—domains, languages, and tasks—is even more complex. In terms of adaptation difficulty, transferring knowledge across languages and tasks yields greater performance improvements than domain adaptation, mainly due to substantial shifts in representation attributes between source and target domains, languages, or tasks. Within domain adaptation itself, larger attribute differences between source and target domains lead to more notable gains compared to cases with fewer differences. When the source and target domains are related but still distinct, such as having similar data distributions, techniques like query expansion for domain adaptation at test time can refine query representation to enhance alignment between queries and candidate documents.

My NCLPRF approach significantly boosts ranking performance when trained on a cross-lingual dataset and tested on a similar cross-language collection. Additionally, the GenQREnsemble-RF approach demonstrates that with zero-shot prompting across related domains, query expansion can be improved by combining LLMs with PRF. When source and target domains differ substantially, fine-tuning a neural ranker becomes essential to effectively capture characteristics of the target domain. The DUQGen approach I propose shows that training a neural ranker with a holistic target domain representative training data can deliver substantial performance gains.

For language differences between source and target datasets, such as in multilingual or cross-lingual query-document pairs, fine-tuning a neural ranker is also necessary for effective language adaptation. My mDUQGen approach demonstrates that efficient fine-tuning tailored to the target language can greatly enhance language adaptation on neural rankers. When adapting to different tasks, such as RAG, fact-checking, or argument retrieval, fine-tuning with high-quality task-specific labels significantly improves ranking outcomes. For example, my TFT-RAG approach introduces a novel method for obtaining high-quality weak labels to better train neural rankers for task alignment. In conclusion, my extensive experiments confirm that transfer learning for tasks and languages achieves greater performance improvements compared to domain adaptation.

Next, I will outline the notable contributions of my thesis research and summarize the findings from my research works. Finally, I will address any limitations that need to be tackled in the future.

6.1 Thesis Contributions

In my thesis research, I present numerous contributions to the active information retrieval and RAG community, and those contributions will help to explore the associated domain adaptation problems in the future. These contributions are outlined below:

1. Categorized and highlighted transfer-learning challenges associated with out-of-distribution datasets, including query representation (in Chapter 3), ranker adaptation (in Chapter 4), and task-relevance labeling (in Chapter 5).
2. Conducted a thorough survey on domain adaptation for neural rankers, up to date as of mid-2024 (detailed in Chapter 2).
3. Introduced various effective unsupervised and weakly supervised frameworks that require minimal or no supervision and are cost-effective.
4. Delivered several innovative approaches addressing three key research questions (RQs), including NCLPRF 3.4, GenQREnsemble-RF 3.5, DUQGen 4.3, mDUQGen 4.4, C3 5.2, IE-IR 5.3, and TFT-RAG 5.4.
5. Published several of the above novel approaches in top-tier conferences, including NCLPRF at SIGIR 2022 [83], DUQGen at NAACL 2024 [126], and C3 at SIGIR 2022 [136]. Additionally, I am actively working to publish the remaining approaches as well.
6. Developed a novel self-referencing based RAG system that integrates an enhanced neural ranker with CoT LLM prompting, offering a strong foundation for future RAG research and advancements.
7. Provided access to codes (NCLPRF¹ and DUQGen²), models, publications, and demo websites through open-source platforms.

6.2 Summary of Results

The core focus of my research is leveraging minimal or no human supervision to train neural models, particularly through unsupervised and weakly supervised methods. Various conclusions have been drawn from the experiments conducted on my approaches, which are outlined below.

Chapter 3 explores the challenges of query representation across domain shifts and introduces two effective strategies: NCLPRF [83] and GenQREnsemble-RF [84]. These methods utilize pseudo-relevance feedback (PRF) to improve query representation by integrating information from the target domain. While both approaches rely on PRF, they differ in how they handle the query expansion in different representation spaces and the input context length limitations of the underlying pretrained models. Despite the context length restrictions of NCLPRF, it showed notable improvements when using only two PRF documents across three cross-lingual evaluation datasets (Persian, Russian, and Chinese), outperforming traditional baselines, NCLPRF without PRF, and NCLPRF with only one PRF (competitor approach) enrichment. Interestingly, increasing the number of feedback documents beyond two can degrade performance due to the inclusion of noisy, irrelevant information, which undermines query enrichment. However, employing a robust vector aggregation function, such as a reciprocal rank-based weighted sum, resulted in steadily

¹<https://github.com/emory-irlab/NCLPRF>

²<https://github.com/emory-irlab/DUQGen>

improving performance as PRF depth increases, highlighting the scalability and effectiveness of this query expansion approach across new domains.

The second query expansion technique, GenQREnsemble-RF, offers a greater context length and introduces two variations based on aggregation methods at the post-retrieval stage: GenQREnsemble-RF and GenQRFusion-RF. GenQREnsemble-RF outperformed GenQRFusion-RF in integrating target domain PRF because it employs cross-attention interactions between the query and expansion tokens earlier in the process, shortly after the compact query reformulation stage—showing that early-fusion outperforms late-fusion. Similar to NCLPRF, increasing the number of feedback documents led to consistent performance improvements. An oracle test, using one highly relevant ground truth document as a PRF document, highlighted a substantial performance boost over scenarios with noisy or non-PRF incorporated feedback.

Chapter 4 tackles the issue of ranker adaptation across out-of-distribution datasets and introduces an effective and robust unsupervised approach known as DUQGen [126]. This approach efficiently samples a diverse set of collection documents and generates high-quality synthetic queries to fine-tune a neural ranker. Through comprehensive evaluation across the BEIR [1] benchmark, DUQGen demonstrated consistent and substantial improvements over existing state-of-the-art baselines. Additionally, utilizing few-shot demonstrations with in-domain examples resulted in the generation of high-quality domain-representative queries compared to the usage of generic few-shot demonstrations. Among the various LLMs employed for query generation, LLAMA-2 7B stood out for producing high-quality synthetic queries compared to LLAMA-2 13B, BLOOM, and gpt-3.5-turbo.

Building on DUQGen from Chapter 4, I introduced a multilingual extension for unsupervised domain adaptation, called mDUQGen. This approach follows a similar pipeline to DUQGen but replaces the monolingual English components with multilingual ones. Evaluation on the MIRACL [127] dev benchmark demonstrates that mDUQGen achieved near-supervised and often outperforming supervised fine-tuning performances, with consistent and robust improvements across datasets using only 1,000 training examples. Additionally, I observed that the machine translation errors in translate-train pipelines led to significant performance drops during the pretraining of neural re-rankers in the multilingual setting. Interestingly, mDUQGen achieved the same relative performance improvements over the pretrained neural ranker, regardless of whether it was pretrained using English (MS-MARCO) or translated (mMARCO) training data. This finding suggests that the translate-train paradigm is no longer necessary, and sampling-based fine-tuning approaches like mDUQGen can effectively produce state-of-the-art multilingual models.

Chapter 5 delves into the issue of task fine-tuning of neural rankers across out-of-distribution tasks and retrieval applications and introduces three effective weakly supervised fine-tuning approaches, namely C3 [136], event retrieval fine-tuning, and TFT-RAG. The studies conducted in this chapter underscore the importance of annotating high-quality labels with minimal supervision for robust learning of neural rankers. The first approach, C3, fine-tunes a pretrained multilingual language model with weakly-labeled large-scale cross-lingual pairs, resulting in substantial and often significant performance improvements across various evaluation target languages (Chinese, Persian, German, and French) for both ColBERT and DPR neural rankers.

The second approach of using confidence scores to weakly supervise the fine-tuning of neural rankers has also demonstrated impressive performance, highlighting the importance of high-quality training labels. In particular, this approach introduces a two-stage approach initially suggests a warm-up fine-tuning phase using unsupervised in-domain training examples, which aids in learning domain vocabularies leading to enhanced performance. Subsequently, the sequential fine-tuning with weakly supervised training examples, augmented by synthetic queries and filtered based on downstream confidence score annotations, further improved the performance, often resulting in significant gains.

The third approach, TFT-RAG, involves task-specific fine-tuning of neural rankers using a self-referencing technique, leading to significant improvements in RAG question-answering tasks. The

self-referencing method employed during training stage created high-quality weakly supervised positive and negative labels for fine-tuning the ranker. Moreover, applying self-referencing with explanations during inference, similar to CoT prompting, resulted in notable performance gains. Ultimately, in RQ3, this novel method was inspired by the first two approaches and is highly applicable to a wide range of real-world tasks involving LLM-based question answering.

6.3 Limitations

The unsupervised and weakly supervised approaches presented in this thesis research have already demonstrated significant performance improvements compared to competitive baselines. However, it is crucial to acknowledge that there are several limitations that may be overlooked and should be considered for future extensions.

Firstly, the limitation of context length is evident in query enrichment approaches. For instance, NCLPRF still faces constraints due to its smaller context length, often resulting in the truncation of longer feedback documents and loss of information towards the end of the documents. However, a pretrained language model with larger context length can be an easy replacement. Similarly, while GenQREnsemble-RF has a larger context length, it can accommodate a maximum of only five average-sized passages, leading to truncation of longer documents as well. Additionally, the use of simple heuristic functions, such as first-stage retriever weights and reciprocal rank weights, as vector aggregation functions in NCLPRF, makes it less robust against noisy and irrelevant documents. Therefore, adopting a more sophisticated learning-based vector aggregation approach is essential for the robust fusion of enriched query representations.

Secondly, while DUQGen exhibits significant improvements over current state-of-the-art baselines, it does possess several limitations. Contriever was employed as a text encoder for generating embeddings for clustering. However, the Contriever text embedding may inadequately represent certain out-of-distribution datasets that are far from the pretrained data. Therefore, it is crucial to first evaluate the quality of generated embeddings or pretrain an encoder on target domain before using the corresponding encoder in the DUQGen framework.

Furthermore, I utilized the Faiss library for K -Means clustering. However, for large collections comprising millions of documents, the algorithm may not scale well. Nonetheless, the Faiss library implements a sampling strategy by initially training the K -Means algorithm on a documents sample before clustering the entire collection, which can help mitigate these scalability issues to some extent.

Additionally, LLMs are sensitive to the prompt variations [121], resulting in different outputs even with small changes in prompt instructions, in-context documents, and demonstrations. Some existing works provide calibration techniques [121] before using LLMs for generating sensitive content. However, developing a robust synthetic query generation version of DUQGen is essential for better reproducibility.

The extension of DUQGen in the multilingual retrieval imposes additional limitations. The LLMs might struggle to generate good quality of text in low-resource languages even with few-shot prompting. The inconsistent generations of LLM can often cause to generate gibberish text or text in a different language other than the target language.

Moreover, fine-tuning neural rankers with weak label annotations can introduce limitations due to errors propagated from teacher model used for annotation. For instance, C3 [136] relies on weak links derived from the Wikipedia repository, yet there may be missed or erroneous links that could adversely affect the fine-tuning performance, particularly in low-resource language scenarios. Likewise, the cross-lingual event retrieval system faces constraints stemming from annotations produced by the downstream event extraction system capable of cross-lingual event extraction. However, this system can accumulate

errors during entity recognition, relation extraction, and confidence estimation, which might carry over to the weak labeling phase, resulting in false positives and false negatives in the training data. Overall, weakly supervised training operates under the assumption that while the training data may not be of high quality, it can still yield certain performance improvements.

Task-specific fine-tuning of a neural ranker in a RAG system also presents certain limitations. Inconsistent LLM outputs can affect the label annotation process, such as producing garbage of text, missing grounded or non-grounded evidence annotations, or mistakenly labeling the same documents as both positive and negative, all of which can hinder fine-tuning performance. Additionally, attempting to parse more than 50 evidence documents in an LLM’s input context for self-referencing may be ineffective due to context length truncation or the "lost-in-the-middle" [185] effect. Additionally, smaller LLMs (under 10B parameters), like LLAMA3-8B-Instruct, tend to be less reliable in strictly following detailed instructions with few-shot examples when compared to models like gpt-4o, particularly in CoT (Chain of Thought) reasoning tasks. This is largely due to the smaller model size and its pre-training, which results in weaker reasoning capabilities. Despite the frequent introduction of more advanced and larger LLMs, it remains worthwhile to explore ways to adapt smaller LLMs for budget-constraint applications.

6.4 Future Work

The limitations discussed in the earlier Section indicate promising directions for future extensions of my thesis contributions. Nevertheless, there are specific projects I wish to emphasize as particularly impactful and significant for the advancement of the information retrieval and RAG community.

In Chapter 4, DUQGen was initially designed for optimizing ad-hoc searches, where the relevance between a document and a query is assessed independently from other relevant documents, following the assumptions of the Cranfield paradigm [135]. However, in contemporary RAG settings, where answers can be derived from multiple documents, such as in multi-hop, set-based, and aggregation questions, the fine-tuning of DUQGen for retrievers often results in limited performance. A straightforward extension would involve generating queries that depend on multiple documents, followed by training the neural rankers accordingly. This approach can utilize DUQGen’s clustering and sampling methods for effective domain representation. By carefully selecting a group of similar yet distinct documents, and using appropriate prompts with few-shot demonstrations, diverse queries can be generated (with eight types of multi-document questions detailed in the CRAG report [51]). These complex generated queries would enhance the training of RAG neural rankers, significantly improving end-to-end question-answering performance.

It is widely recognized that LLMs can produce inconsistent responses to the same prompt. Repeating the same prompt often results in varying outputs, and modifying the prompt with different paraphrased instructions or sets of few-shot in-context examples can lead to significant differences in the generated content, causing fluctuations in downstream performance. In Chapter 4, I introduced two novel approaches, namely DUQGen and mDUQGen, to unsupervised domain adaptation for fine-tuning neural rankers using LLM-generated queries. Therefore, new strategies are needed to enhance the self-consistency of LLM-generated queries or to make the fine-tuning of neural rankers more resilient to these variations.

I have been working on a solution to address the inconsistencies in query generation by LLMs. The core idea is to develop a neural ranker training framework that is resilient to these inconsistencies. This involves selecting the most challenging examples from a pool of candidates. Specifically, after sampling documents from a cluster (as in DUQGen) and generating multiple queries with varying information needs via the LLM, a novel scoring mechanism can be introduced to identify the hardest examples for the zero-shot neural ranker—those where the ranker already performs poorly. By focusing on these difficult

examples and their generated queries during training, a more robust system can be built to minimize the impact by LLM inconsistencies.

My thesis introduces seven novel approaches to address three key **RQs** in the context of domain adaptation. These solutions aim to enhance various components of training a neural ranker. Chapter 3 focuses on improving the domain adaptation by modifying the query representations at test time. Chapter 4 presents methods for sequentially fine-tuning neural rankers using synthetic training data, while Chapter 5 proposes fine-tuning neural rankers with high-quality task-specific weak labels. Although these solutions are complementary, it would be valuable to experiment with the combined end-to-end performance of a ranking system incorporating the improvements from all three chapters. Additionally, studying the individual impact of each component could help to identify which contributes most significantly to performance. Ultimately, a comprehensive neural ranking system based on the combined solutions in this thesis could make a significant impact on the information retrieval and RAG communities.

In conclusion, this thesis has explored the challenges of domain adaptation and transfer learning, addressing three key research questions (RQs) to guide the study. I reviewed relevant work in the field, proposed seven novel approaches under the three RQs, and highlighted my contributions, along with the conclusions drawn from each approach. These contributions focus on enhancing query representation, ranker adaptation, and task fine-tuning across new target domains with minimal to no supervision.

For RQ1, my methods that modify query representation at inference time, enabling effective and efficient domain adaptation. For RQ2, I introduced holistic target domain representations to efficiently sample and generate synthetic queries for fine-tuning neural rankers. In RQ3, my pseudo labeling techniques—contrastive learning, pseudo label filtering, and pseudo label generation—enhanced task fine-tuning for neural rankers.

Together, these approaches are scalable, adaptable, and applicable across various domains, languages, and retrieval tasks. My methods set new baselines for future research in the fields of information retrieval and retrieval-augmented generation (RAG), advancing the development of domain-adaptive neural ranking models.

Appendix A

Appendix for TFT-RAG

A.1 Reranked References for TFT-RAG Prediction

To further illustrate examples of incorrect predictions made by my best model, I included the top reranked chunks in Figures A.1. Due to space constraints, I excluded highly irrelevant chunks and highlighted any relevant or misleading information in yellow. For Question-2 in Figure 5.11, the reranked documents referenced in Figure A.1a reveal multiple mentions of the Alabama River. However, none explicitly identify it as the longest river in Alabama. There are also some mentions of the Tennessee River, but these chunks lack descriptive details or supporting evidence to confirm it as the longest river. Consequently, the generator relies on the available information, incorrectly assumes that the Alabama River is the longest, and outputs it. This incorrect prediction likely results from either the absence of supportive evidence in the entire web corpus or the failure of the retriever or reranker to include such evidence in the top-k ranked list.

Another example of incorrect predictions is illustrated in Figure A.1b, supporting Question-3 from Figure 5.11. While it is true that Captain America stopped Thanos with his bare hands, the question asking whether he used a glove makes the query invalid. The reranked chunks include some references to gloves but more frequently mention gauntlets, none of which are directly relevant to the specific event of Captain America stopping Thanos. Notably, Reference ID 5 discusses the event of Captain America trying to stop Thanos (the "Mad Titan") but does not clarify who possesses the gauntlet—whether it is Captain America or Thanos—despite it being true that Thanos holds the gauntlet. This lack of clarity and absence of additional supportive evidence leads the generator to produce an incorrect prediction, further compounded by the invalid nature of the question itself.

["question": "how long is the longest river in alabama?",
 "ground_truth": "the longest river that flows through alabama, the tennessee river, is 652 miles long.",
 "prediction": {"ratings": "1,0,0,0,3,3,3,0,0,1,0,0,2,0,0,1,2,0,0,1", "thought": "The question asks about the length of the longest river in Alabama. To answer this, we need passages that discuss the lengths of rivers in Alabama or specifically address which river is the longest. Passages 5, 6, and 7 provide relevant information about the length of the Alabama River, while passage 13 states that the Alabama Scenic River Trail follows the Alabama River, implying its significance. No other passages provide specific details on the length of the rivers.", "answer": "The Alabama River, the longest river in Alabama, has a navigable length of 305 miles (491 km)."},
 "ranked_results": {
 1: "Its navigable length is 305 miles (491 km), and the river drains 22,800 square miles (59,050 square km).",
 2: "The Alabama is joined 45 miles (72 km) north of Mobile by the Tombigbee to form the Mobile and Tensaw rivers, which flow into Mobile Bay , an arm of the Gulf of Mexico .",
 4: "Alabama River Black Warrior River Cahaba River Chattahoochee River Choctawhatchee River Coosa River Conecuh River Escatawpa River Mobile & Tensaw River Perdido River Tallapoosa River Tennessee River Tombigbee River Yellow & Blackwater Rivers Contact Auburn University Water Resources Center ALFA Agricultural Services and Research Building 961 S Donahue Drive Auburn, AL 36840 1-888-844-4785 wrc@auburn.edu Campus Map Website feedback Webmaster AUWRC Pages AUWRC Home About Us Research Extension & Outreach Education & Resources Alabama Water Resources Conference Links Auburn University A-Z Index Accessibility AU Access Copyright People Finder Privacy @2024 Auburn University Water Resources Center, Alabama Agricultural Experiment Station.",
 5: "Alabama River | Map, Length, & Facts | Britannica Search Britannica Click here to search Search Britannica Click here to search Login Subscribe Now Subscribe Home Games & Quizzes ...",
 6: "Originating just north of Montgomery, the Alabama River is born from the marriage of the Coosa River and the Tallapoosa River near the Fall Line.",
 7: "It is formed by the Coosa and Tallapoosa rivers, 7 miles (11 km) northeast of Montgomery , winds westward to Selma , and then flows southward.",
 8: "Lower Alabama River The Lower Alabama River begins at the Claiborne Lock and Dam and runs south 72.5 miles through Monroe, Clarke, and Baldwin counties where it joins the Tombigbee River.",
 10: "It receives its chief tributary, the Cahaba , about 10 miles (16 km) southwest of Selma.",
 11: "The Alabama Scenic River Trail is a 631-mile boating trail from Weiss Lake down the Coosa River into the Alabama River and through the Mobile Delta to the Gulf of Mexico.",
 12: "Nestled in Alabama's southwest hill country, the lake encompasses over 60 miles of the Alabama River.",
 13: "Alabama River | Outdoor Alabama Skip to main content Top Nav About Us Contact Us Media News Events Translation Sign Up View More Results Outdoor ...",
 14: "Select Citation Style MLA APA Chicago Manual of Style Copy Citation Share Share Share to social media Facebook Twitter URL https://www.britannica.com/place/Alabama-River Feedback External Websites Feedback Corrections?",
 15: "External Websites Fact Monster - World - Alabama River, United States Print Cite verified Cite While every effort has been made to follow citation style rules, there may be some discrepancies.",
 16: "The Editors of Encyclopaedia Britannica Last Updated: Jan 17, 2024 • Article History Table of Contents Alabama River Alabama River, near Selma, Alabama, U.S. (more) Alabama River, Alabama Alabama River, river in southern Alabama , U.S.",
 17: "Select Citation Style MLA APA Chicago Manual of Style Copy Citation Share Share Share to social media Facebook Twitter URL https://www.britannica.com/place/Alabama-River Give Feedback External Websites Feedback Corrections?",
 18: "External Websites Fact Monster - World - Alabama River, United States Written and fact-checked by The Editors of Encyclopaedia Britannica Encyclopaedia Britannica's editors oversee subject areas in which they have extensive knowledge, whether from years of experience gained by working on that content or via study for an advanced degree.",
 20: "US Army Corps of Engineers Mobile District Website • About Mission and Principles Leadership History Commanders of U.S. Army Corps of Engineers, ... Allatoona Lake Alabama River Lakes Apalachicola-Chattahoochee-Flint Black Warrior & Tombigbee (BWT) Lakes Carters Lake Lake Seminole Lake Sidney Lanier Okatibbee Lake Tennessee Tombigbee Waterway ..."}
 }

(a)

["question": "what movie did captain america use the glove to stop thanos?",
 "ground_truth": "invalid question",
 "prediction": {"ratings": "0,0,2,0,3,1,0,0,3,1,1,0,0,2,0,1,0,0,1,3", "thought": "The question asks about the movie where Captain America used the glove to stop Thanos. Several passages mention relevant events involving Captain America and Thanos in the Marvel Cinematic Universe. Specifically, it is mentioned in passages rated 3 that during the final battle of 'Avengers: Endgame', Captain America fought Thanos. A passage rated 2 gives additional context on Captain America's actions against Thanos.", "answer": "Avengers: Endgame"},
 "ranked_results": {
 1: "Thanos: The Infinity Relativity #1 Marvel Cinematic Universe (Earth-199999) [] There are two Infinity Gauntlets: a right-handed replica held in Odin's Vault and a functional left-handed one used by Thanos .",
 2: "Summer Special #1 ↑ Marvel Contest of Champions ; 'Rise of the Black Panther' Event Quest ↑ Infinity Chaos - Marvel Contest of Champions ↑ Marvel Contest of Champions ; 'Infinity Chaos' Event Quest ↑ Marvel Realm of Champions: Elegy S1E02 ↑ Thanos: The Infinity Relativity #1 ↑ Thor ↑ Thor: Ragnarok ↑ 25.0 25.1 Avengers: Infinity War ↑ Avengers: Age of Ultron ↑ 27.0 27.1 27.2 Avengers: Endgame Categories Categories : Items Illuminati (Earth-616)/Items Gloves Magical Items Jim Starlin/Creator Ron Lim/Creator Thanos (Earth-616)/Items Adam Warlock (Earth-616)/Items Magus (Earth-7528)/Items Reed Richards (Earth-616)/Items Anthony Stark (Earth-616)/Items Nebula (Earth-616)/Items Steven Rogers (Earth-616)/Items Earth-616/Items 1990 Item Debuts Odin's Treasure Room Thanos Experiment Languages Português do Brasil Русский Community content is available under CC-BY-SA unless otherwise noted.",
 3: "Galactus , the Watcher, and Thanos sensed the assembly of the gauntlet, and Captain America used it to push away the other Earth and prevent the collision.",
 4: "Marvel Contest of Champions Home to the Living Tribunal (Adam Warlock) (Earth-19141) [] Thanos used the Infinity Gauntlet in order to get Mistress Death 's favours, but Adam Warlock stopped him, stole the Gauntlet and formed the Infinity Watch in order to keep the Stones . [22]",
 5: "When Cap tries to stop Mad Titan from attacking Vision, he holds his hand that has infinity gauntlet .",
 6: "Soon after, he used its power to fight the heroes and Thanos at the Heart of Infinity.",
 7: "Thanos surviving and overcoming the cosmic entities' assault Nigh Omnipotence: The Infinity Gauntlet granted virtual/nigh-omnipotence to its wearer.",
 8: "During Avengers: Endgame final battle, we again saw Captain America fighting the Thanos from 2014.",
 9: "In Avengers: Infinity War , we saw Thanos coming face-to-face with Iron Man, Captain America, Black Panther and others.",
 10: "This Interesting Fan Theory Answers In Avengers: Infinity War, we saw Thanos coming face-to-face with Iron Man, Captain America, Black Panther and others.",
 11: "– A J ♦ May 3, 2019 at 6:41 3 But before Thanos punches her with the power stone, she has a moment with Peter Parker, who hands her the gauntlet .",
 12: "When Thanos arrived the following week, he and Black Panther used their respective gauntlets to engage in a cosmic battle. [18]",
 13: "As stated in the movie, Thanos can only activate the Infinity Gauntlet's power when he closes his fist.",
 14: "This Interesting Fan Theory Answers(Pic scredit: Stills from Avengers: Infinity War) It cost a lot to defeat Thanos and his army in Avengers: Endgame .",
 15: "TAGS Avengers: Endgame Avengers: Infinity War Black Widow Captain America Captain Marvel Hollywood Iron Man Rocket Raccoon Thanos Thor RELATED ARTICLES Entertainment News 'Black Widow' Scarlett Johansson Once Reflected On The Dark Side Of Fame & Expressed How Much She Hated It: "It Can Bring Out The...".",
 16: "When a time-travelling Thanos tried to use it to destroy the whole universe, Tony Stark used his own armor's nanotechnology to create a makeshift gauntlet to accommodate the Stones, which were then used to wipe out the time travelling Thanos and his army. [27]",
 ...
 19: "The villain Thanos used it to kill half of the universe and other activities in an attempt to court Death .",
 20: "He hoped to appease his love to the entity known as Death , Thanos used the Infinity Gauntlet with all of the Infinity Gems to remove half of the population of the Universe . [6"]
 }

(b)

Figure A.1: An example question, ground truth answer, ranked references, and incorrect generator predictions. Prediction includes the entire thought process of ratings, thought, and final answer.

Bibliography

- [1] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 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)*, 2021. URL <https://openreview.net/forum?id=wCu6T5xFjeJ>.
- [2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.
- [3] Helia Hashemi, Yong Zhuang, Sachith Sri Ram Kothur, Srivas Prasad, Edgar Meij, and W. Bruce Croft. Dense retrieval adaptation using target domain description. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '23*, page 95–104, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700736. doi: 10.1145/3578337.3605127. URL <https://doi.org/10.1145/3578337.3605127>.
- [4] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 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, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531863. URL <https://doi.org/10.1145/3477495.3531863>.
- [5] Arian Askari, Mohammad Aliannejadi, Chuan Meng, Evangelos Kanoulas, and Suzan Verberne. Expand, highlight, generate: RL-driven document generation for passage reranking. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10087–10099, Singapore, December 2023. Association for Computational Linguistics. URL <https://aclanthology.org/2023.emnlp-main.623>.
- [6] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples, 2022.
- [7] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *ArXiv*, abs/2007.00808, 2020. URL <https://api.semanticscholar.org/CorpusID:220302524>.
- [8] HongChien Yu, Chenyan Xiong, and Jamie Callan. *Improving Query Representations for Dense Retrieval with Pseudo Relevance Feedback*, page 3592–3596. Association for Computing Machin-

ery, New York, NY, USA, 2021. ISBN 9781450384469. URL <https://doi.org/10.1145/3459637.3482124>.

- [9] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. Replug: Retrieval-augmented black-box language models, 2023.
- [10] LingXi Zhang, Yue Yu, Kuan Wang, and Chao Zhang. ARL2: Aligning retrievers with black-box large language models via self-guided adaptive relevance labeling. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3708–3719, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.203. URL <https://aclanthology.org/2024.acl-long.203>.
- [11] Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees, 2022. URL <https://arxiv.org/abs/2004.11829>.
- [12] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.
- [13] Garrett Wilson and Diane J. Cook. A survey of unsupervised deep domain adaptation. 11(5), July 2020. ISSN 2157-6904. doi: 10.1145/3400066. URL <https://doi.org/10.1145/3400066>.
- [14] Jingjing Li, Zhiqi Yu, Zhekai Du, Lei Zhu, and Heng Tao Shen. A comprehensive survey on source-free domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5743–5762, 2024. doi: 10.1109/TPAMI.2024.3370978.
- [15] Yuqi Fang, Pew-Thian Yap, Weili Lin, Hongtu Zhu, and Mingxia Liu. Source-free unsupervised domain adaptation: A survey. *Neural Netw.*, 174(C), July 2024. ISSN 0893-6080. doi: 10.1016/j.neunet.2024.106230. URL <https://doi.org/10.1016/j.neunet.2024.106230>.
- [16] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts, 2023. URL <https://arxiv.org/abs/2303.15361>.
- [17] William S. Cooper, Fredric C. Gey, and Daniel P. Dabney. Probabilistic retrieval based on staged logistic regression. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992. URL <https://api.semanticscholar.org/CorpusID:125993>.
- [18] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. 1999. URL <https://api.semanticscholar.org/CorpusID:15777237>.
- [19] Clebson C.A. de Sá, Marcos A. Gonçalves, Daniel X. Sousa, and Thiago Salles. Generalized broof-l2r: A general framework for learning to rank based on boosting and random forests. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’16, page 95–104, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340694. doi: 10.1145/2911451.2911540. URL <https://doi.org/10.1145/2911451.2911540>.

- [20] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 391–398, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595935977. doi: 10.1145/1277741.1277809. URL <https://doi.org/10.1145/1277741.1277809>.
- [21] Ashish Phophalia. A survey on learning to rank (letor) approaches in information retrieval. *2011 Nirma University International Conference on Engineering*, pages 1–6, 2011. URL <https://api.semanticscholar.org/CorpusID:29350795>.
- [22] Junior Zilles, Giancarlo Lucca, and Eduardo Nunes Borges. A literature review on methods for learning to rank. In *International Conference on Enterprise Information Systems*, 2022. URL <https://api.semanticscholar.org/CorpusID:248717886>.
- [23] Muhammad Ibrahim and M. Manzur Murshed. From tf-idf to learning-to-rank: An overview. 2016. URL <https://api.semanticscholar.org/CorpusID:64253521>.
- [24] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 129–136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273513. URL <https://doi.org/10.1145/1273496.1273513>.
- [25] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: Bert and beyond, 2021.
- [26] Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, December 2018. URL <https://www.microsoft.com/en-us/research/publication/introduction-neural-information-retrieval/>.
- [27] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 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, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL <https://www.aclweb.org/anthology/2020.emnlp-main.550>.
- [28] Omar Khattab and Matei Zaharia. 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, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL <https://doi.org/10.1145/3397271.3401075>.
- [29] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval, 2021.
- [30] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert, 2019.

- [31] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, apr 2009. ISSN 1554-0669. doi: 10.1561/15000000019. URL <https://doi.org/10.1561/15000000019>.
- [32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- [33] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.
- [34] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization, 2020.
- [35] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL <https://aclanthology.org/2020.lrec-1.494>.
- [36] 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. Large dual encoders are generalizable retrievers. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.669. URL <https://aclanthology.org/2022.emnlp-main.669>.
- [37] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [38] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. URL <https://openreview.net/pdf?id=r1xMH1BtvB>.
- [39] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’17*, page 65–74, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350228. doi: 10.1145/3077136.3080832. URL <https://doi.org/10.1145/3077136.3080832>.

- [40] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *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, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.168. URL <https://aclanthology.org/2022.naacl-main.168>.
- [41] Peng Xu, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Passage ranking with weak supervision, 2019.
- [42] Mostafa Dehghani and J. Kamps. Learning to rank from samples of variable quality. *ArXiv*, abs/1806.08694, 2018. URL <https://api.semanticscholar.org/CorpusID:49392953>.
- [43] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. Large language models can accurately predict searcher preferences. In *2024 International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, July 2024. URL <https://www.microsoft.com/en-us/research/publication/large-language-models-can-accurately-predict-searcher-preferences/>. An earlier version of this paper appeared as arXiv preprint arXiv:2309.10621v1 [cs.IR].
- [44] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models, 2022. URL <https://arxiv.org/abs/2208.03299>.
- [45] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning, 2021. URL <https://arxiv.org/abs/2112.09118>.
- [46] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- [47] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.74. URL <https://aclanthology.org/2021.eacl-main.74>.
- [48] Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. Augmentation-adapted retriever improves generalization of language models as generic plug-in, 2023. URL <https://arxiv.org/abs/2305.17331>.
- [49] Jinming Nian, Zhiyuan Peng, Qifan Wang, and Yi Fang. W-rag: Weakly supervised dense retrieval in rag for open-domain question answering, 2024. URL <https://arxiv.org/abs/2408.08444>.

- [50] James Ferguson, Hannaneh Hajishirzi, Pradeep Dasigi, and Tushar Khot. Retrieval data augmentation informed by downstream question answering performance. In Rami Aly, Christos Christodoulopoulos, Oana Cocarascu, Zhijiang Guo, Arpit Mittal, Michael Schlichtkrull, James Thorne, and Andreas Vlachos, editors, *Proceedings of the Fifth Fact Extraction and VERification Workshop (FEVER)*, pages 1–5, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.fever-1.1. URL <https://aclanthology.org/2022.fever-1.1>.
- [51] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. Crag – comprehensive rag benchmark, 2024. URL <https://arxiv.org/abs/2406.04744>.
- [52] Peide Zhu and Claudia Hauff. Unsupervised domain adaptation for question generation with DomainData selection and self-training. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2388–2401, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.183. URL <https://aclanthology.org/2022.findings-naacl.183>.
- [53] Abhinav Ramesh Kashyap, Devamanyu Hazarika, Min-Yen Kan, and Roger Zimmermann. Domain divergences: A survey and empirical analysis. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1830–1849, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.147. URL <https://aclanthology.org/2021.naacl-main.147>.
- [54] Simon Lupart, Thibault Formal, and Stéphane Clinchant. Ms-shift: An analysis of ms marco distribution shifts on neural retrieval. In Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo, editors, *Advances in Information Retrieval*, pages 636–652, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-28244-7.
- [55] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks, 2016.
- [56] Daniel Cohen, Bhaskar Mitra, Katja Hofmann, and W. Bruce Croft. 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, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210141. URL <https://doi.org/10.1145/3209978.3210141>.
- [57] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In Regina Barzilay and Mark Johnson, editors, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL <https://aclanthology.org/D11-1033>.

- [58] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.
- [59] Victor Lavrenko and W. Bruce Croft. Relevance based language models. SIGIR '01, page 120–127, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133316. doi: 10.1145/383952.383972. URL <https://doi.org/10.1145/383952.383972>.
- [60] Hui Fang and ChengXiang Zhai. Semantic term matching in axiomatic approaches to information retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, page 115–122, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933697. doi: 10.1145/1148170.1148193. URL <https://doi.org/10.1145/1148170.1148193>.
- [61] Chia-Jung Lee and W. Bruce Croft. Cross-language pseudo-relevance feedback techniques for informal text. In *Proceedings of the 36th European Conference on IR Research on Advances in Information Retrieval - Volume 8416*, ECIR 2014, page 260–272, Berlin, Heidelberg, 2014. Springer-Verlag. ISBN 9783319060279.
- [62] Xuwen Wang, Qiang Zhang, Xiaojie Wang, and Junlian Li. Cross-lingual pseudo relevance feedback based on weak relevant topic alignment. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 529–534, Shanghai, China, October 2015. URL <https://aclanthology.org/Y15-1061>.
- [63] Saar Kuzi, Anna Shtok, and Oren Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1929–1932, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340731. doi: 10.1145/2983323.2983876. URL <https://doi.org/10.1145/2983323.2983876>.
- [64] Suraj Nair, Eugene Yang, Dawn Lawrie, Kevin Duh, Paul McNamee, Kenton Murray, James Mayfield, and Douglas W. Oard. Transfer learning approaches for building cross-language dense retrieval models, 2022.
- [65] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020.
- [66] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert, 2019.
- [67] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL <https://aclanthology.org/2020.findings-emnlp.63>.
- [68] Xiao Wang, Craig MacDonald, Nicola Tonellotto, and Iadh Ounis. Pseudo-relevance feedback for multiple representation dense retrieval. *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, 2021.
- [69] Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. Ceqe: Contextualized embeddings for query expansion, 2021.

- [70] Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. Inpars-v2: Large language models as efficient dataset generators for information retrieval, 2023.
- [71] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François 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 Laurence, 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 González Ponferrada, Efrat Levkovich, 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 Froberg, 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 López, 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 François Lavallée, 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, Aurélie Névoul, Charles Lovering, Daniel H Garrette, Deepak R. Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Ekaterina 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 Behrooz, 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, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynek, 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 Fourier, Daniel Le'on Perin'an, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrmann, 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 Sangaroonsiri, 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. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100, 2022. URL <https://api.semanticscholar.org/CorpusID:253420279>.

- [72] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 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, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462891. URL <https://doi.org/10.1145/3404835.3462891>.
- [73] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval, 2023.
- [74] 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. Llama 2: Open foundation and fine-tuned chat models, 2023.

- [75] Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joëlle Pineau, and Luke Zettlemoyer. Improving passage retrieval with zero-shot question generation. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL <https://api.semanticscholar.org/CorpusID:248218489>.
- [76] Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. Open-source large language models are strong zero-shot query likelihood models for document ranking. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8807–8817, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.590. URL <https://aclanthology.org/2023.findings-emnlp.590>.
- [77] 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. Language models are few-shot learners, 2020.
- [78] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents, 2023.
- [79] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting, 2024.
- [80] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model, 2023.
- [81] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models, 2023.
- [82] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze!, 2023.
- [83] Ramraj Chandradevan, Eugene Yang, Mahsa Yarmohammadi, and Eugene Agichtein. Learning to enrich query representation with pseudo-relevance feedback for cross-lingual retrieval. *SIGIR '22*, page 1790–1795, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3532013. URL <https://doi.org/10.1145/3477495.3532013>.
- [84] Kaustubh D. Dhole, Ramraj Chandradevan, and Eugene Agichtein. Generative query reformulation using ensemble prompting, document fusion, and relevance feedback, 2024. URL <https://arxiv.org/abs/2405.17658>.

- [85] Kaustubh D. Dhole and Eugene Agichtein. Genqrensemble: Zero-shot llm ensemble prompting for generative query reformulation. In Nazli Goharian, Nicola Tonellotto, Yulan He, Aldo Lipani, Graham McDonald, Craig Macdonald, and Iadh Ounis, editors, *Advances in Information Retrieval*, pages 326–335, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-56063-7.
- [86] Zewen Chi, Li Dong, Bo Zheng, Shaohan Huang, Xian-Ling Mao, Heyan Huang, and Furu Wei. Improving pretrained cross-lingual language models via self-labeled word alignment. *arXiv preprint arXiv:2106.06381*, 2021.
- [87] Zi-Yi Dou and Graham Neubig. Word alignment by fine-tuning embeddings on parallel corpora, 2021.
- [88] Shuo Sun and Kevin Duh. CLIRMatrix: A massively large collection of bilingual and multilingual datasets for cross-lingual information retrieval. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4160–4170, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.340. URL <https://aclanthology.org/2020.emnlp-main.340>.
- [89] Jane M. Bromley, Isabelle Guyon, Yann LeCun, Eduard Sackinger, and Roopak Shah. Signature verification using a siamese time delay neural network. In *7th Annual Neural Information Processing Systems Conference*, pages 737–744. Morgan Kaufmann Publishers, 1994. URL <http://oro.open.ac.uk/39214/>. Advances in Neural Information Processing Systems 6 Edited by Jack D. Cowan, Gerald Tesauro, Joshua Alspector.
- [90] Dawn Lawrie, James Mayfield, Douglas W. Oard, and Eugene Yang. HC4: A new suite of test collections for ad hoc CLIR. In *Proceedings of the 44th European Conference on Information Retrieval (ECIR)*, 2022.
- [91] Cash Costello, Eugene Yang, Dawn Lawrie, and James Mayfield. Patapasco: A python framework for cross-language information retrieval experiments, 2022.
- [92] Carol Peters and Martin Braschler. European research letter: Cross-language system evaluation: The clef campaigns. *J. Am. Soc. Inf. Sci. Technol.*, 52(12):1067–1072, oct 2001. ISSN 1532-2882. doi: 10.1002/asi.1164. URL <https://doi.org/10.1002/asi.1164>.
- [93] T MITAMURA. Overview of the ntcir-8 aclia tasks: Advanced cross-lingual information access. In *NTCIR-8 Workshop, 2010*, 2010.
- [94] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. volume 33, pages 1877–1901, 2020.
- [95] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.
- [96] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

- [97] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. Generative query reformulation for effective adhoc search. In *The First Workshop on Generative Information Retrieval, SIGIR 2023*, 2023.
- [98] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.
- [99] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*, 2023.
- [100] Orion Weller, Kyle Lo, David Wadden, Dawn Lawrie, Benjamin Van Durme, Arman Cohan, and Luca Soldaini. When do generative query and document expansions fail? a comprehensive study across methods, retrievers, and datasets. *arXiv preprint arXiv:2309.08541*, 2023.
- [101] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR, 2021.
- [102] Kaustubh Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahadiran, Simon Mille, Ashish Shrivastava, Samson Tan, et al. Nl-augmenter: A framework for task-sensitive natural language augmentation. *Northern European Journal of Language Technology*, 9(1), 2023. URL <https://nejlt.ep.liu.se/article/view/4725/3874>.
- [103] Gian Wiher, Clara Meister, and Ryan Cotterell. On decoding strategies for neural text generators. *Transactions of the Association for Computational Linguistics*, 10:997–1012, 2022.
- [104] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. pages 5315–5333, 2023.
- [105] Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re. Ask me anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations*, 2022.
- [106] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. 2022.
- [107] OpenAI. Gpt-4 technical report, 2023.
- [108] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- [109] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 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, 2020.

- [110] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [111] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human-generated machine reading comprehension dataset. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
- [112] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. Simplified data wrangling with ir_datasets. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2429–2436, 2021.
- [113] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. Pyterrier: Declarative experimentation in python from bm25 to dense retrieval. In *Proceedings of the 30th acm international conference on information & knowledge management*, pages 4526–4533, 2021.
- [114] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.
- [115] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL <https://aclanthology.org/2020.findings-emnlp.63>.
- [116] Jaime Carbonell and Jade Goldstein. 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, 1998. Association for Computing Machinery. ISBN 1581130155. doi: 10.1145/290941.291025. URL <https://doi.org/10.1145/290941.291025>.
- [117] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 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, 2021.
- [118] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.
- [119] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 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, 2020.
- [120] Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 18:267–276, 1953. URL <https://api.semanticscholar.org/CorpusID:120467216>.
- [121] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models, 2021.

- [122] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- [123] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [124] Luiz Henrique Bonifacio, Vitor Jeronimo, Hugo Queiroz Abonizio, Israel Campiotti, Marzieh Fadaee, , Roberto Lotufo, and Rodrigo Nogueira. mmarco: A multilingual version of ms marco passage ranking dataset, 2021.
- [125] Nandan Thakur, Jianmo Ni, Gustavo Hernandez Abrego, John Wieting, Jimmy Lin, and Daniel Cer. Leveraging LLMs for synthesizing training data across many languages in multilingual dense retrieval. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7699–7724, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.naacl-long.426>.
- [126] Ramraj Chandradevan, Kaustubh D. Dhole, and Eugene Agichtein. Duggen: Effective unsupervised domain adaptation of neural rankers by diversifying synthetic query generation, 2024.
- [127] Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. Making a miracle: Multilingual information retrieval across a continuum of languages, 2022. URL <https://arxiv.org/abs/2210.09984>.
- [128] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer, 2021. URL <https://arxiv.org/abs/2010.11934>.
- [129] Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. XOR QA: Cross-lingual open-retrieval question answering. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 547–564, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.46. URL <https://aclanthology.org/2021.naacl-main.46>.
- [130] Ehsan Kamalloo, Xinyu Zhang, Odunayo Ogundepo, Nandan Thakur, David Alfonso-hermelo, Mehdi Rezagholizadeh, and Jimmy Lin. Evaluating embedding APIs for information retrieval. In Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 518–526, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-industry.50. URL <https://aclanthology.org/2023.acl-industry.50>.

- [131] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023. URL <https://arxiv.org/abs/2305.10403>.
- [132] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024. URL <https://arxiv.org/abs/2402.03216>.
- [133] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings, 2024. URL <https://arxiv.org/abs/2309.07597>.
- [134] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. Tevatron: An efficient and flexible toolkit for dense retrieval. *ArXiv*, abs/2203.05765, 2022.
- [135] Pedro Rodriguez and Jordan Boyd-Graber. Evaluation paradigms in question answering. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9630–9642, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.758. URL <https://aclanthology.org/2021.emnlp-main.758>.
- [136] Eugene Yang, Suraj Nair, Ramraj Chandradevan, Rebecca Iglesias-Flores, and Douglas W. Oard. C3: Continued pretraining with contrastive weak supervision for cross language ad-hoc retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’22, page 2507–2512, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531886. URL <https://doi.org/10.1145/3477495.3531886>.
- [137] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings*

of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

- [138] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.
- [139] Suraj Nair, Eugene Yang, Dawn Lawrie, Kevin Duh, Paul McNamee, Kenton Murray, James Mayfield, and Douglas W. Oard. Transfer learning approaches for building cross-language dense retrieval models. 2022. URL <https://arxiv.org/abs/2201.08471>.
- [140] Sean MacAvaney, Luca Soldaini, and Nazli Goharian. Teaching a new dog old tricks: Resurrecting multilingual retrieval using zero-shot learning. In *Proceedings of the 42nd European Conference on Information Retrieval Research*, pages 246–254, 2020. doi: 10.1007/978-3-030-45442-5_31. URL https://link.springer.com/chapter/10.1007/978-3-030-45442-5_31.
- [141] Peng Shi and Jimmy Lin. Cross-lingual relevance transfer for document retrieval. *arXiv preprint arXiv:1911.02989*, 2019.
- [142] Peng Shi, Rui Zhang, He Bai, and Jimmy Lin. Cross-lingual training with dense retrieval for document retrieval. *arXiv preprint arXiv:2109.01628*, 2021.
- [143] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253*, 2021.
- [144] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*, 2020.
- [145] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540*, 2021.
- [146] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Towards unsupervised dense information retrieval with contrastive learning, 2021.
- [147] Zhiqi Huang, Hamed Bonab, Sheikh Muhammad Sarwar, Razieh Rahimi, and James Allan. *Mixed Attention Transformer for Leveraging Word-Level Knowledge to Neural Cross-Lingual Information Retrieval*, page 760–770. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450384469. URL <https://doi.org/10.1145/3459637.3482452>.
- [148] Kareem Darwish and Douglas W Oard. Probabilistic structured query methods. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 338–344, 2003.
- [149] Shota Sasaki, Shuo Sun, Shigehiko Schamoni, Kevin Duh, and Kentaro Inui. Cross-lingual learning-to-rank with shared representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 458–463, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2073. URL <https://aclanthology.org/N18-2073>.

- [150] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [151] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020.
- [152] Shuo Sun and Kevin Duh. 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, 2020.
- [153] T MITAMURA. Overview of the ntcir-8 aclia tasks: Advanced cross-lingual information access. In *NTCIR-8 Workshop, 2010*, 2010.
- [154] Eneko Agirre, Giorgio Maria Di Nunzio, Nicola Ferro, Thomas Mandl, and Carol Peters. Clef 2008: Ad hoc track overview. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 15–37. Springer, 2008.
- [155] Nicola Ferro and Carol Peters. Clef 2009 ad hoc track overview: Tel and persian tasks. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 13–35. Springer, 2009.
- [156] Martin Braschler and Carol Peters. CLEF 2003 methodology and metrics. In *Comparative Evaluation of Multilingual Information Access Systems*, pages 7–20. Springer Berlin Heidelberg, 2004.
- [157] Zewen Chi, Li Dong, Bo Zheng, Shaohan Huang, Xian-Ling Mao, Heyan Huang, and Furu Wei. Improving pretrained cross-lingual language models via self-labeled word alignment. *arXiv preprint arXiv:2106.06381*, 2021.
- [158] Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup. *arXiv preprint arXiv:2101.06983*, 2021.
- [159] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.
- [160] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. Tevatron: An efficient and flexible toolkit for dense retrieval. *ArXiv*, abs/2203.05765, 2022.
- [161] Tobias Domhan, Michael Denkowski, David Vilar, Xing Niu, Felix Hieber, and Kenneth Heafield. The Sockeye 2 neural machine translation toolkit at AMTA 2020. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 110–115, Virtual, October 2020. Association for Machine Translation in the Americas.
- [162] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *Proceedings of the international AAAI conference on web and social media*, volume 5, pages 438–441, 2011.
- [163] Donald Metzler, Congxing Cai, and Eduard Hovy. Structured event retrieval over microblog archives. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 646–655, 2012.

- [164] Alan Ritter, Oren Etzioni, and Sam Clark. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112, 2012.
- [165] Nattiya Kanhabua and Avishek Anand. Temporal information retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1235–1238, 2016.
- [166] Sheikh Muhammad Sarwar and James Allan. Query by example for cross-lingual event retrieval. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.
- [167] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, page 563–570, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220646. URL <https://doi.org/10.3115/1220575.1220646>.
- [168] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pages 113–124, 2003. doi: 10.1109/ICDE.2003.1260786.
- [169] Dawn Lawrie, Eugene Yang, Douglas W. Oard, and James Mayfield. Neural approaches to multilingual information retrieval, 2022. URL <https://arxiv.org/abs/2209.01335>.
- [170] Patrick Xia, Guanghui Qin, Siddharth Vashishtha, Yunmo Chen, Tongfei Chen, Chandler May, Craig Harman, Kyle Rawlins, Aaron Steven White, and Benjamin Van Durme. LOME: Large ontology multilingual extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 149–159, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-demos.19. URL <https://aclanthology.org/2021.eacl-demos.19>.
- [171] Sheng Zhang, Rachel Rudinger, and Ben Van Durme. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, Montpellier, France, September 2017.
- [172] Guanghui Qin. *Towards Efficient Long-Context Natural Language Processing*. PhD thesis, 2024.
- [173] Aron Culotta and Andrew McCallum. Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 109–112. Association for Computational Linguistics, 2004.
- [174] Ian Soboroff. The better cross-language datasets. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 3047–3053, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3591910. URL <https://doi.org/10.1145/3539618.3591910>.
- [175] Mahsa Yarmohammadi, Shijie Wu, Marc Marone, Haoran Xu, Seth Ebner, Guanghui Qin, Yunmo Chen, Jialiang Guo, Craig Harman, Kenton Murray, Aaron Steven White, Mark Dredze, and

- Benjamin Van Durme. Everything is all it takes: A multipronged strategy for zero-shot cross-lingual information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1950–1967, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.149. URL <https://aclanthology.org/2021.emnlp-main.149>.
- [176] Zhengping Jiang, Anqi Liu, and Benjamin Van Durme. Calibrating zero-shot cross-lingual (un-)structured predictions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2648–2674. Association for Computational Linguistics, 2022.
- [177] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.
- [178] Jinxi Xu and W Bruce Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems (TOIS)*, 18(1):79–112, 2000.
- [179] Zhuyun Dai and Jamie Callan. 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, SIGIR’19*, page 985–988, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361729. doi: 10.1145/3331184.3331303. URL <https://doi.org/10.1145/3331184.3331303>.
- [180] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL <https://arxiv.org/abs/2005.11401>.
- [181] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL <https://arxiv.org/abs/2201.11903>.
- [182] Hossein A. Rahmani, Nick Craswell, Emine Yilmaz, Bhaskar Mitra, and Daniel Campos. Synthetic test collections for retrieval evaluation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’24*, page 2647–2651, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657942. URL <https://doi.org/10.1145/3626772.3657942>.
- [183] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. Large language models can accurately predict searcher preferences, 2024. URL <https://arxiv.org/abs/2309.10621>.
- [184] Yikuan Xia, Jiazun Chen, and Jun Gao. Winning solution for meta kdd cup’ 24, 2024. URL <https://arxiv.org/abs/2410.00005>.
- [185] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL <https://arxiv.org/abs/2307.03172>.