ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

A TWO-STEP LLM-AUGMENTED DISTILLATION METHOD FOR PASSAGE RERANKING

CANDIDATE
Davide Baldelli

SUPERVISOR
Paolo Torroni

CO-SUPERVISORS
Akiko Aizawa
Junfeng Jiang

Academic year 2022-2023

Session 1st

Contents

List of Figures

List of Tables

Abstract

This thesis delves into the exploration and enhancement of passage reranking in Information Retrieval (IR) systems, particularly focusing on the distillation of knowledge from Large Language Models (LLMs) to augment the capabilities of smaller cross-encoders. The research pivots the feasibility of distilling the knowledge of LLMs into smaller models without compromising reranking capabilities, and the impact of the distillation process on the adaptability of the resultant model across diverse scenarios. To navigate through these inquiries, a novel distillation method, termed TWOLAR (TWO-step LLM-Augmented distillation method for passage Reranking), is introduced. TWOLAR is characterized by a new scoring strategy and a distillation process consisting in the creation of a novel and diverse training dataset. The dataset consists of 20K queries, each associated with a set of documents retrieved via four distinct retrieval methods to ensure diversity, and then reranked by exploiting the zero-shot reranking capabilities of an LLM. The ablation study demonstrates the contribution of each introduced component. The experimental results show that TWOLAR significantly enhances the document reranking ability of the underlying model, obtaining state-of-the-art performances on the TREC-DL test sets and the zero-shot evaluation benchmark BEIR, thereby contributing a novel perspective and methodology to the discourse on optimizing IR systems via knowledge distillation from LLMs.

Chapter 1

Introduction

Text IR is the task of finding relevant information, in the form of documents or passages, to user-defined queries, where both the queries and resources are expressed in natural language text.

Since the advent of machine learning and deep learning, we have seen a rapid evolution of text retrieval systems which no longer require hand-crafted features. In particular, the transformer architecture [52] and Pretrained Language Models (PLMs) [11] have signed a significant leap, and the vast majority of recent dense retrieval systems are based on this paradigm.

The state-of-the-art text rerankers are traditional cross-encoders like monoBERT [32], monoT5 [33, 43], and RankT5 [58], and more recently rerankers based on Large Language Models (LLMs) like RankGPT [46], LRL [29] and PRP [39].

Large Language Models (LLMs), such as ChatGPT [37], GPT-4 [36], PaLM [7], LLama[50], and Claude[1] are quickly solving a large diversity of tasks, when previously different and specialized models were needed. The capabilities of those models are the outcome of well-engineered pretraining on large-scale text corpora and alignment fine-tuning to follow human instructions. Thanks to this recipe, LLMs have shown remarkable performances in language understanding, generation, and reasoning. Recently, research has sought to leverage LLMs to improve IR systems [57]. Adapting LLMs for IR

tasks is not straightforward as high efficiency is needed, and the task is not easily expressible in the usual way of language modeling.

Cross-encoders heavily rely on large human-annotated datasets, which renders them not easily scalable and weak in out-of-domain scenarios. On the other hand, the higher performances of LLMs come with higher computational costs. To this end, this work investigates how to mitigate the mentioned limitations. Specifically it focus on the following two questions:

- **(RQ1)** Is it possible to distill the knowledge of LLMs into smaller cross-encoders to match their reranking capabilities?
- **(RQ2)** How does the distillation process affect the adaptability of the student model to different scenarios?

To answer these questions this thesis introduces a new distillation method for passage reranking called **TWOLAR** (a **TWO**-step **L**LM-**A**ugmented distillation method for passage **R**eranking). The distillation consists in exploiting the capabilities of an LLM as a reranker to produce high-quality annotations. The annotations are applied to a dataset of queries generated artificially, either as cropped sentences or again by a specialized language model. In this way I obtain a compact model that ranks among top performing supervised, zero-shot, and LLM-based distillation methods in various popular benchmarks. [1]

The remainder of the thesis is structured as follows: Chapter 2 provides background on text retrieval and ranking methods. Chapter 3 details my approach, subdivided into scoring and distillation strategies. Chapter 4 covers the experimental setup, including datasets, training, baselines, and results. Chapter 5 discusses the results and 6 illustrates the ablation studies. Chapter 7 concludes the thesis.

[1] The work presented in this thesis is currently under review; after publication all the data, code and models will be made available at `https://github.com/Dundalia/TWOLAR`

Chapter 2

Background

IR is a discipline concerned with enabling users to find relevant information from an organized collection of documents [8]. IR systems typically typically start from a user query and return a ranked list of documents based to their relevance to the query.

Formally, given a query and a passage from a large text collection, *text ranking* requires returning a ranked list of the n most relevant texts according to the relevance scores of a model.

Early text retrieval systems are based on the *bag-of-words* assumption, and both the queries and the documents are represented as sparse term-based vectors. Those vectors are built with term weighting rule-based methods, and the relevance can be estimated as the similarity between such text embeddings. Among those methods, *tf-idf* [44] and BM25 [42] still represent strong baselines, and are widely adopted.

Subsequently, statistical language modeling has been widely explored for text ranking [55]. With the development of machine learning, supervised approaches, that still rely on hand-crafted features as well as lexical features, have been proposed [27, 24]. Further progress was made with the adoption of neural networks mapping pieces of text into low-dimensional vectors to obtain better representations [18, 30, 19].

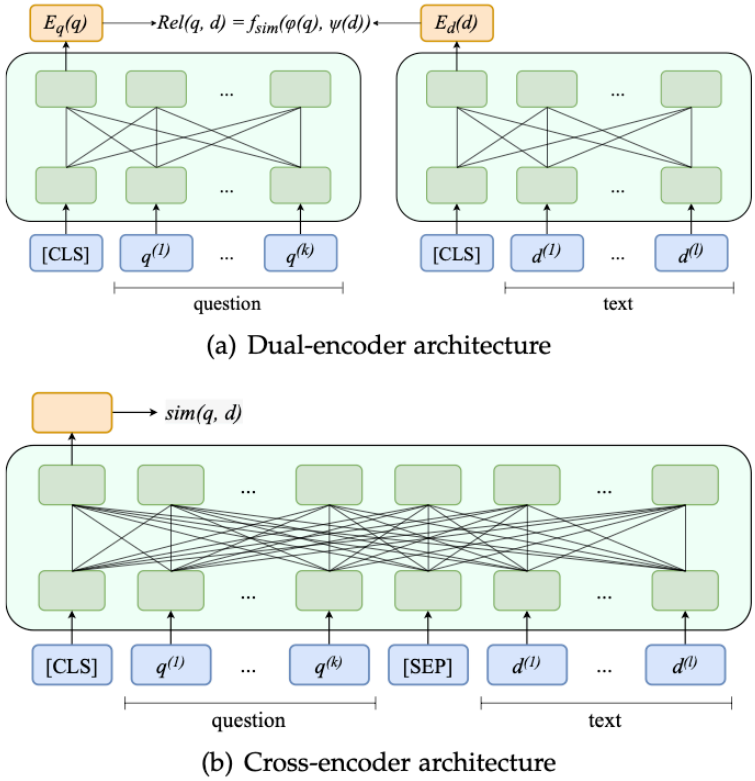

Figure 2.1: Visual illustration of dual-encoder and cross-encoder architectures. Image from [56].

The advent of pretrained language models (PLMs) [11, 41, 40] and large-scale human annotated datasets [31, 22, 54] marked a significant advancement in the field.

The core idea of dense retrieval is to model the *semantic relation* between queries and documents based on the model representations. Those approaches can be divided into two families: bi-encoder architectures and cross-encoder architectures (see Fig. 2.1). In bi-encoder architectures, two separate encoders compute the query embedding and the document embedding. This is accomplished by inputting the language model with a query (respectively, a document) and use as embedding vector the learned representation of a special token (”[CLS]” in BERT). Then the relevance score is computed via some similarity function (e.g., cosine similarity or dot product) between the query embedding and the document embedding. On the other hand, in cross-encoder

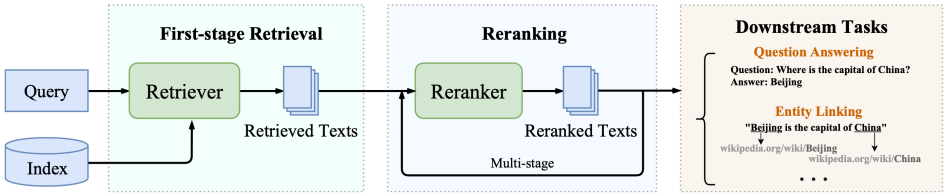

Figure 2.2: The illustration for the overall pipeline of an IR system. Image from [56].

architectures a single PLM is fed with the concatenation of a query and a document. In this way a relevance score can be directly derived by the model's output.

It is important to note that the cross-encoder approach requires every query-document pair to be encoded, and hence does not scale to large collections of documents. On the other hand, with a bi-encoder architecture documents and queries are encoded independently so that the corpus can be processed offline and at inference time I only need to encode the query and compute the similarity score. For an extensive survey on PLM-based dense retrieval models refer to [56].

Recently, a new paradigm emerged [12, 25, 56, 35, 57], consisting of multiple stages (see Fig 2.2): using a first-stage retriever that aims to reduce the candidate space by retrieving a subset of relevant candidates, often numbering in the hundreds or thousands, and then refining these initial results with a second-stage reranker.

With the rapid development of LLMs, recent research has sought to leverage their capabilities to improve IR systems [57].

Most of the neural retrieval systems are trained on large datasets like Natural Questions (NQ) [22] (133k training samples) or MS MARCO [31] (533k training samples), which are composed of a corpus and a set of questions or short keyword-based queries. However, creating a large training corpus is often unfeasible and hence it is crucial for an IR system to perform well in a zero-shot setup. To this aim, the BEIR benchmark [49] has been proposed,

which is composed of 18 retrieval datasets for the evaluation of model generalization. A more detailed review of the datasets is presented in chapter 3.

In recent works [33, 58, 3] the reranker is usually initialized as a T5 model [41]. The Text-to-Text Transfer Transformer (T5) model, developed by Google Research, is a state-of-the-art Natural Language Processing model designed to handle a variety of NLP tasks, including translation, summarization, question-answering, and more. T5 operates under the paradigm of treating every text processing problem as a "text-to-text" problem, i.e. taking text as input and producing new text as output. This model is pre-trained on a large corpus of web crawled text and fine-tuned on downstream tasks. The versatility and effectiveness of T5 have made it a popular choice among researchers and practitioners in the field of NLP. To enhance the capabilities of T5, the flan-T5 model [9] was conceived by scaling the number of tasks and including Chain of Thoughts data.

Now, I am going to present the recent advancements in dense text retrieval and reranking, presenting the State-of-The-Art models, and reviewing the recent approaches to distill LLMs' capabilities.

2.1 SoTA retrievers

Various improvements have been proposed to the classical bi-encoder approach.

SPLADE. The idea behind SPLADE [15, 14, 13] is to estimate the importance of each term of the vocabulary implied by each term of the document, i.e. to compute an interaction matrix between the document or query tokens and all the tokens from the vocabulary. Given an input query or document sequence (after WordPiece tokenization) $t = (t_1, t_2, \ldots, t_N)$ and its corresponding BERT embeddings $(h_1, h_2, \ldots, h_N)$, we consider the importance w_{ij} of the token j (vocabulary) for a token i (of the input sequence):

$$w_{ij} = \text{transform}(h_i)^T E_j + b_j \quad j \in \{1, \ldots, |V|\}$$

where E_j denotes the BERT input embedding for token j, b_j is a token-level bias, and $transform(.)$ is a linear layer with GeLU activation [23] and LayerNorm. The final representation is then obtained by aggregating importance predictors over the input sequence tokens. In the latest version (SPLADEv2) the aggregation is computed as: $w_j = \max_{i \in t} \log(1 + \text{ReLU}(w_{ij}))$.

The transformer encoder is initialized as a DistilBERT [45].

DRAGON. Inspired by the success of curriculum learning, knowledge distillation, and data augmentation, DRAGON [26] has been conceived. The model is trained using only augmented queries, which have been built with two approaches:

- Sentence cropping, which consists in extracting a sentence from a document to use as a query;
- Pseudo query generation, in which, given a document, synthetic queries are generated by a specialized T5 model [34].

The queries are then coupled with positive and negative documents with a progressive label augmentation approach. Given a retrieval model as the source of supervision, positive and negative documents are sampled respectively from the top 10 passages and top 46-50 passages the model has retrieved to form a triplet. At each training iteration, a new source of supervision is added in ascending order of better generalization capability.

2.2 SoTA rerankers

The task of a reranker is to output a fine-grained ordered list of relevant documents, reranking the top retrieved documents from a first-stage retriever, usually the top 100 or top 1000 documents. As the amount of documents to evaluate is far lower than the documents a retriever has to process, rerankers are often implemented as a cross-encoder architecture, so to better model the semantic interaction between any two tokens of a query-document pair.

MonoBERT The most straightforward approach is to adapt BERT model to accomplish this task. In [32] the authors trained a BERT model on the MS MARCO dataset, feeding the model with concatenated query-document pairs. Then, they have inputted the "[CLS]" vector to a single layer neural network to obtain the probability of the document being relevant for the specified query. The model has been fine-tuned using the cross-entropy loss:

$$\mathcal{L} = -\sum_{j \in J_{pos}} \log(s_j) - \sum_{j \in J_{neg}} \log(1 - s_j),$$

where J_{pos} and J_{neg} are respectively the sets of indexes of relevant and non-relevant documents.

It is important to note that the model is trained on a binary classification task, in which the reranker has to classify each query-document pair as relevant and non-relevant. The scores assigned to the positive class is used at inference time as the probability of the document to be relevant for the respective query.

MonoT5 The idea behind MonoT5 [33, 43] is to adapt a pretrained sequence-to-sequence model, T5 [41], to the task of document reranking. Instead of modifying the architecture with a classification head, the model is fine-tuned to produce the words "`true`" or "`false`" depending on whether the document is relevant or not with respect to the query. The input sequence is:

Query: q Document: d Relevant:

where q and d are the query and document texts. To compute the scores for each query-document pair, a softmax is applied to the logits of the "`true`" and "`false`" token, and the probability assigned to the "`true`" token is used as relevance score.

InPars A key challenge in IR is the lack of domain-specific training data. To this aim zero-shot and few-shot learning models are particularly effective. The proposal of InPars [3, 21] is to efficiently use LLMs in reranking, by generating labeled data in a few-shot manner, and finetune reranking models on this synthetic data. As a result, this approach requires different models for each dataset.

The training recipe and the strategy to obtain the scores are analogous to MonoT5.

RankT5 The main contribution of [58] is the application of ranking losses to train T5-based rerankers. Among others they have tested the RankNet loss [5] and the listwise softmax cross-entropy loss [4].

Since the ranking loss functions require a score for each query-document pair at training time, the authors have used as score the logit of a special unused token in T5 vocabulary: "`<extra_id_10>`".

RankGPT In [46] the authors investigate the potential of LLMs in reranking. For this goal, they have developed a *sliding window approach* that enables to generate the permutations of the reranked documents, despite the limited amount of input tokens allowed. Given M passages, a window size w of 20 and a step size s of 10, the LLM ranks the passages from $(M-w)^{th}$ to M^{th}, utilizing w documents per prompt to generate the document permutations. At this point the window is slided in steps of s, so that in the second step the LLM reranks the passages from $(M-w-s)^{th}$ to $(M-s)^{th}$, and the process is repeated until all the passages have been re-ranked. In total the model has to be inputted $(M//s) \times w$ times.

In order to overcome the expenses of utilizing a proprietary model, the authors have proposed an approach to distill the ranking capabilities of ChatGPT into a specialized model. To do so, they have generated a dataset of GPT generated permutations to train a DeBERTaV3 model [20]. To build the dataset, they have randomly sampled 10K queries from MS MARCO, retrieved 20 candidate passages using BM25 for each query, and finally they have prompted ChatGPT to generate the permutations.

PRP. Recently in [39] it has been presented the first application of moderated-sized open-sourced LLMs to text ranking. They proposed a novel approach to reduce the complexity of the task for the LLM: *Pairwise ranking prompting (PRP)*. It consists in asking the LLM to compare which of two given passages is more relevant to a specific query, and using the pairwise comparison as the basic computation unit for three different approaches.
The first approach is to extract a score for each query-document pair prompting the LLM with the comparison between each possible couple of documents. It requires $O(N^2)$ calls to the LLM.
As pairwise comparison is the basic operator of sorting algorithms, the second proposed approach is to use the LLM as the comparator for the Heapsort algorithm, requiring $O(N \log(N))$ LLM calls.
The last approach is analogous to sliding window approach in [46], which corresponds to the Bubblesort algorithm when the comparator is binary. By noticing that ranking usually only cares about Top-K ranking metrics, where K is small, they can perform K passes, leading to $O(N)$ calls to the LLM.

The authors have tried different open-sourced LLMs from the Flan-T5 [9] and Flan-UL [48] families.

2.3 Distilling LLMs

In recent literature we have seen an increased interest in distilling LLMs' capabilities in smaller architectures because although ChatGPT and GPT-4 are highly capable, they are also expensive. One of the first attempt is Self-instruct [53], in which the authors have tried to enhance the capabilities of the OpenAI *davinci* model by artificially building a dataset with the LLM itself.

Furthermore it has been observed that ChatGPT outperforms crowd-worker annotators on text-annotation tasks [16].

Later on, since Alpaca [47], many approaches to distill GPT-3's and GPT-4's capabilities using their own instructional outputs came out [38, 6]. In those studies the student language model is often initialized as LLama [50]. Despite the initial enthusiasm, it has been proven that this form of broad imitation of a proprietary model does not lead to better performances as the student model learns to imitate the *style*, not the *factuality* of the teacher[17]. Nonetheless, the authors found out that training student models on proprietary LLMs remains a promising approach in task-specific settings, and [46] represents a successfull example in text reranking.

Chapter 3

Methodology

The proposed reranking method is based on Flan-T5 [9]. To adapt it to the task, I use the following input template:

Query: [Q] Document: [D] Relevant:

where [Q] and [D] are the query and document texts, respectively, similar to the one adopted in monoT5 [33, 43]

3.1 Scoring Strategy

Flan-T5 can be straightforwardly applied to various tasks due to its text-to-text nature, such as summarization, translation, and classification. However, adapting to the ranking task is not trivial, because for each query-document pair, we usually ask models to answer with a score representing the degree of relevance. The state-of-the-art rerankers, monoT5 [33, 43] and RankT5 [58], which are specialized text-to-text models, suffer from this limitation.

MonoT5. In MonoT5 this is solved by training the model with a binary classification task: given a query-document pair the model is finetuned to produce the words "`true`" if the document is relevant to the query and "`false`" otherwise. At this point the ranking score is obtained from the logits of the "`true`"

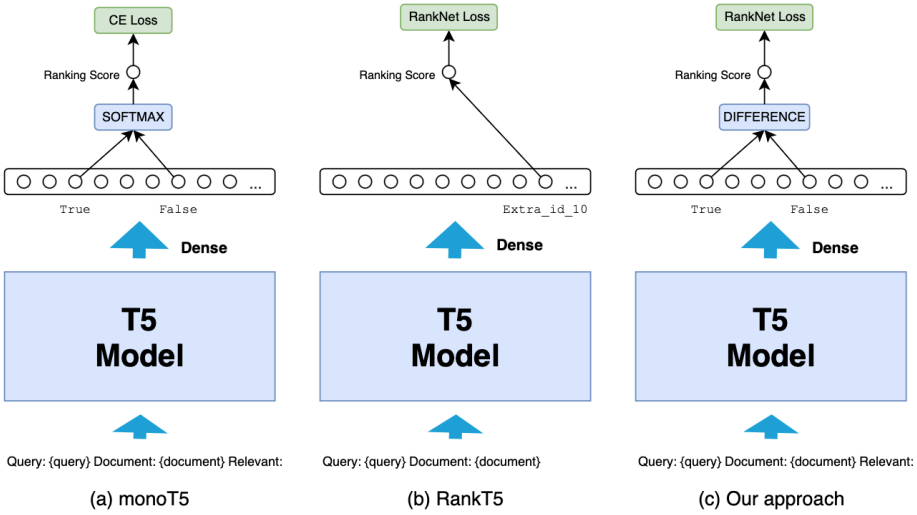

Figure 3.1: Illustration of the different score strategies from monoT5 [33, 43], RankT5 [58] and the proposed approach.

and "`false`" tokens as follows:

$$s = \frac{e^{z_{\texttt{true}}}}{e^{z_{\texttt{true}}} + e^{z_{\texttt{false}}}} \tag{3.1}$$

where $z_{\texttt{true}}, z_{\texttt{false}}$ are the logits of '`true`' and '`false`', respectively.

RankT5. Another approach has been proposed with RankT5 [58]. In this case the model directly learns to rank by optimizing a ranking-based loss function. This family of loss functions requires the model to directly output the ranking score for each query-document pair at training time, so that the unnormalized logit of a special unused token ('`extra_id_10`') in the vocabulary is used as ranking score.

Proposed score strategy. On one hand, monoT5 is not directly finetuned as a ranking model, which may not optimize its ranking performance. On the other hand, RankT5 does not exploit the learned representation in the language modeling head.

To overcome both limitations, I propose a new approach. The idea consists of using the difference between the unnormalized logits corresponding to the ‘`true`’ and ‘`false`’ tokens. In this way, the model is able to output a score directly at training time, and since it is optimized on top of the learned representations of the two tokens, I can make full use of the knowledge from the PLMs.

An illustration of these scoring strategies is shown in Fig. 3.1.

3.2 Datasets

The largest annotated dataset for IR is the MS MARCO passage reranking dataset [31]. It contains around 530K train queries and 6.8K ”dev” queries. The corpus is composed of more than 8.8 millions passages. The labels are binary: for each query, relevant passages are annotated with 1 and others are annotated with 0. For the evaluation I have adopted test sets of the 2019 and 2020 competitions: TREC-DL2019 and TREC-DL2020 [10], which provides dense human relevance annotations for each of their 43 and 54 queries. For the evaluation of zero-shot performances I used the BEIR benchmark [49]. It is an heterogeneus benchmark containing 18 retrieval datasets, covering different retrieval tasks and different text domains.

3.3 Distillation strategy

The proposed distillation strategy aims to capture the reranking capability of LLMs, in this case ChatGPT, through constructing a query-document dataset. The core design principle is the synthesis of suitable artificial queries by query augmentation, and the subsequent use of multiple retrieval models and stages of distillation.

This dataset is characterized by a two-tier distillation process, which incorporates an initial retrieval phase involving four distinct models, followed

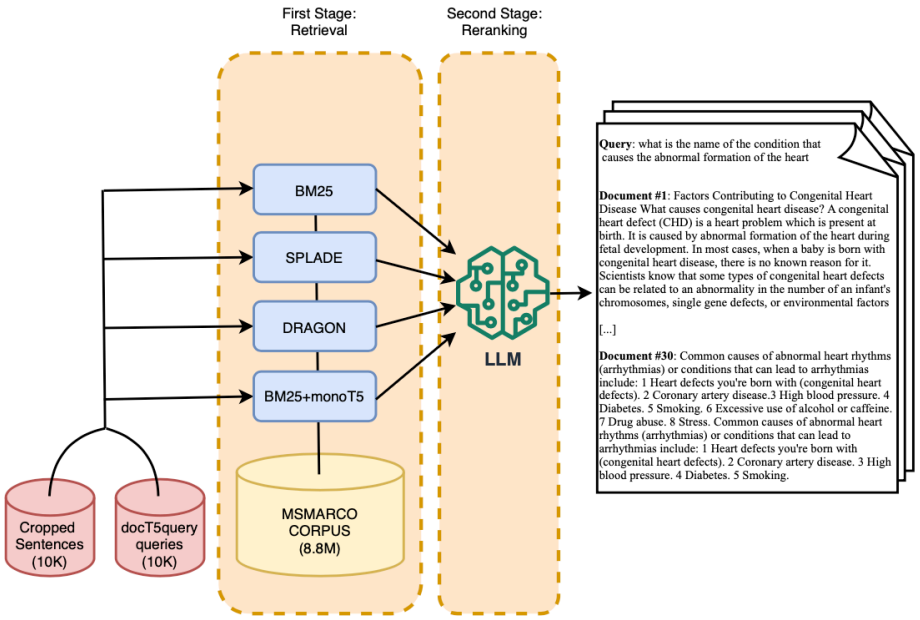

Figure 3.2: Illustration of the proposed two-steps LLM-augmented distillation method used for the construction of the dataset.

by a second phase where ChatGPT is employed. The selection of advanced retrieval models in the first phase is purposeful, intending to introduce a higher level of complexity in the dataset and providing a robust challenge for the subsequent use of ChatGPT.

In parallel, the utilization of two different types of artificially generated queries introduces a layer of diversity to the tasks. This variety encourages a wider spectrum of responses from ChatGPT, furthering the understanding of its capabilities and potential weaknesses across a broader range of situations.

The following subsections will delve into the specifics of each component (see Fig 3.2) in the construction of this distillation dataset, providing a clearer understanding of its design and the underlying rationale.

Query Augmentation. My approach to query augmentation was inspired by the successful application of a method introduced in [26].

According to [26], two common automatic approaches are often adopted to amplify the size of training queries from a given corpus: sentence cropping

and pseudo query generation. The former can readily scale up the size of the query without involving any computationally expensive operation. The latter, on the other hand, generates high-quality but more computationally expensive human-like queries using large language models, and has been utilized in [3, 21] to adapt a ranking model to specific datasets, where a reasonable amount of queries for training is often lacking.

Building on these insights, I developed the query generation process using both sentence cropping and pseudo query generation. I randomly sampled 10,000 queries from DRAGON's collection of cropped sentences, drawn from the MS MARCO corpus consisting of 28 million sentences, originating from 8.8 million passages. Simultaneously, I sampled an additional 10,000 queries from the query pool created by docT5query [34], a specialized T5 model that generates queries based on a given passage.

The combination of these two subsets, each consisting of 10,000 queries, forms a diverse set, boosting the challenge and complexity of the task. My study, details of which will be discussed in subsequent sections, found that a mixed set of queries, including both cropped sentences and docT5query queries, proved to be the most effective approach. This aligns with the findings from the DRAGON paper, thereby validating the choice of a balanced mixture of query types.

First-stage distillation: retrieval. The initial phase of the distillation process involves splitting each of the two sets of 10K queries - one set composed of cropped sentences and the other of docT5query-generated queries - into four subsets of 2.5k queries each. To retrieve documents for these queries, I chose four distinct retrieval models, designed not only to provide high-quality results but also to diversify the types of challenges and contexts presented to ChatGPT in the subsequent distillation stage. The models I have chosen are:

- **BM25** [42]: A state-of-the-art bag-of-words approach that relies primarily on word overlap to match documents to queries. Consequently,

its hard negatives are expected to challenge the language model on lexical-level matches.

- **DRAGON** [26]: A dense retrieval model designed to detect semantic similarity between queries and passages. It pushes the language model towards understanding deeper semantic relations and contexts. I have chosen the DRAGON+ version.

- **SPLADE** [15, 13, 14]: It serves as a kind of midpoint between BM25's focus on word overlap and DRAGON's emphasis on semantic similarity. It introduces a different level of complexity by considering interactions between the tokens of the document or query and all the tokens from the vocabulary. I have chosen the SPLADE++ version.

- **monoT5** [33, 43]: A combination of BM25 and monoT5 where the top-100 documents retrieved by BM25 are re-ranked using monoT5. It introduces negatives that are influenced by the ranking capabilities of a cross-encoder.

In all cases, I retrieve the top 30 documents for each query, resulting in a diverse and challenging set of documents for the next stage of the distillation process.

To substantiate the diversity of the documents retrieved by the four distinct models, I computed the intersection rate between the sets of documents obtained from any two sources of supervision. For each query, I took the two sets of documents retrieved by two different sources and calculated the intersection between them, subsequently dividing by the total number of documents (which in this case is 30).

This process was carried out separately for both types of queries: the cropped sentence queries and the docT5query-generated queries. The mean of the intersection rates was then calculated to provide a comprehensive view of the overall overlap among the retrieved documents from all sources:

Table 3.1: Average intersection rate between each pair of sources. The upper triangular part of the table represents the intersection rate for cropped sentences and the lower triangular part represents the intersection rate for docT5query-generated queries.

doct5query \ sentence (%)	BM25	SPLADE	DRAGON	monoT5
BM25	\	20.0	29.0	49.8
SPLADE	17.8	\	35.8	26.0
DRAGON	25.0	41.0	\	38.4
monoT5	46.4	27.2	38.5	\

$$\frac{\sum_{q \in \mathcal{Q}} \frac{S_q^1 \cap S_q^2}{30}}{|\mathcal{Q}|}$$

where $\mathcal{Q}$ is the whole query set, S_q^1 and S_q^2 represent the retrieved document set from two sources given the query q. The results of these calculations are summarized in the Table 3.1

The table demonstrates a low mean intersection rate, providing clear evidence of the considerable diversity among the retrieved document sets for both types of queries. This substantial range of retrieval contexts further enhances the complexity of the distillation dataset and effectively showcases the variety of retrieval challenges that ChatGPT encounters in the second stage of the distillation process.

Second-stage distillation: reranking. Upon completion of the first stage of document retrieval, I proceed with the reranking process using ChatGPT, in particular the checkpoint "`gpt-3.5-turbo-16k-0613`". I adopted the listwise prompting template developed by [46], illustrated in Figure 3.3.

The reranking method is directly inspired by the approach presented in the work [46], and I have directly adopted their public repository [1]. In this methodology, I input each of the 20,000 queries and their corresponding top 30 retrieved documents into ChatGPT. The language model is then prompted to provide permutations of the indices of these documents, ordered according to their relevance to the associated query.

This approach requires significant computational resources due to the complexity of the task and the vast number of queries and documents involved.

[1] `https://github.com/sunnweiwei/RankGPT`

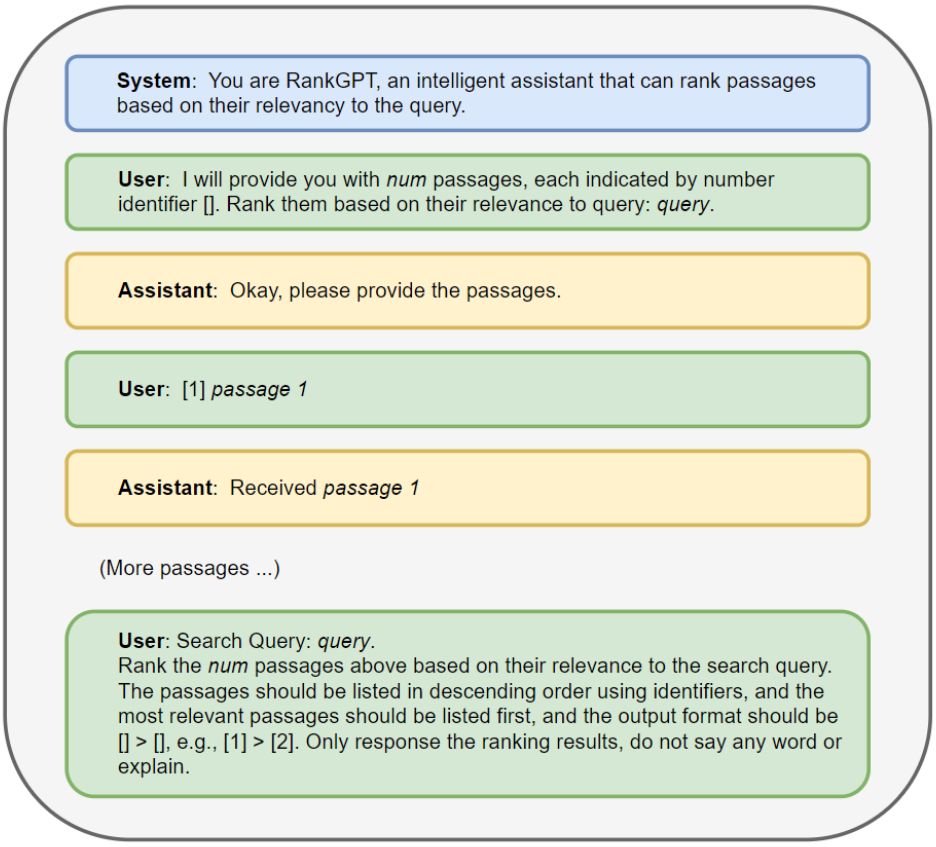

Figure 3.3: Illustration of the permutation generation prompt template developed by [46] .

Notably, the total cost of this reranking operation using the ChatGPT API amounted to $212, demonstrating the feasible financial aspect of employing a large-scale language model in creating such a diverse and complex dataset.

Train-Validation split. The next step in my methodology involved the creation of distinct training and validation sets. Given the importance of maintaining a balanced representation of the different types of queries, I adopted a stratified approach in the partitioning of the dataset. In this process, I reserved a total of 1,000 queries for the validation set, including 500 queries generated by docT5query and 500 queries extracted as cropped sentences. Subsequently, the remaining 19,000 samples were allocated to the training set.

Ranking Loss. In this work, I adopt the RankNet loss [5], a pairwise loss function that models the probability of one document being more relevant than another given a query. RankNet has shown compelling results in the domain of IR and provided a solid foundation for the optimization process.

Given a query q and and M passages (p_1, ..., p_M) ($M = 30$ in my implementation), ChatGPT produces the ranking results of the M passages $R = (r_1, \ldots, r_M)$, where $r_i \in [1, 2, \ldots, M]$ is the rank of the passage p_i: if $r_i = 1$, it means that p_i is the most relevant passage according to ChaGPT. Now the student model works as a cross-encoder which takes as input each query-document pair (q, p_i) and outputs a relevance score s_i.

Therefore, I optimize the model with the following loss function measuring the correctness of relative passage orders:

$$\mathcal{L}_{RankNet} = \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{I}_{r_i < r_j} \log(1 + e^{s_i - s_j})$$

However, it is important to note that the landscape of ranking loss functions is vast and diverse, and the adoption of different loss functions could potentially lead to alternative outcomes. Thus, while I utilize RankNet in the present study, I acknowledge the potential benefits of other ranking loss functions and plan to explore these in future works. This exploration could shed light on the nuanced impact of various loss functions on the fine-tuning of transformer-based ranking models and yield new insights into their performance and learning behavior.

Chapter 4

Experimental setup

4.1 Datasets

All comparisons on TREC-DL2019 and TREC-DL2020 are based on the reranking of top 100 passages retrieved by BM25 [42] for each query. This is the same setting as existing works evaluating zero-shot LLM methods [46, 39].

The evaluation on the BEIR benchmark is based on the the reranking of the top 100 passages retrieved by three different retrievers: BM25, SPLADE++[13], and DRAGON+[26]. The objective is to evaluate the adaptability of the rerankers to different retrievers. I also present the evaluation by reranking the top-1000 documents retrieved by BM25, to give a broad view of the performances on different setting

4.2 Training

I initialized the ranking model with pretrained Flan-T5-xl checkpoint [9]. I set the maximum input sequence length to 500. The batch size is set to 32, meaning that the parameters are updated after computing the score for 30×32 query-document pairs. I utilize the AdamW[28] optimizer with a constant learning rate of $5e-5$. I trained the model for one epoch, requiring approximately 30 hours on a A100 NVIDIA GPU.

4.3 Baselines

I evaluate the model on the TREC-DL2019 and TREC-DL2020 competitions against the following baselines supervised baselines:

- monoBERT [32];
- monoT5-3B [33, 43]
- RankT5 [58]

I also consider the following zero-shot LLM-based baselines:

- RankGPT [46]: the *listwise* prompting based approach using both `gpt-3.5-turbo` and `gpt-4`;
- PRP [39]: the sliding window approach performed only for 10 passes using Flan-T5-xl (3B), Flan-T5-xxl (11B) and Flan-UL2 (20B).

I include in the comparison the distilled model based on DeBertaV2 proposed in [46] as the only other LLM distillaiton method other than ours.

Regarding the zero-shot evaluation on the BEIR benchmark, I evaluate the models against three different rerankers:

- InParsV2 [3, 21];
- monoT5-3B [33, 43];
- the distilled DeBertaV2 model proposed in [46].

4.4 Results

The results on the TREC-DL2019 and TREC-DL2020 benchmarks are summarized in Table 5.1. Tables 5.2 and 5.3 instead summarize respectively the results on the BEIR benchmark by reranking the top-100 and the top-1000 documents.

Chapter 5

Discussion

On TREC-DL. In the evaluation on the TREC-DL2019 and TREC-DL2020 benchmarks, my model demonstrated outstanding performance. When compared against both established supervised methods and LLM-distilled baselines, the proposed approach consistently outperformed them, underlining its robustness and effectiveness. Notably, my model even surpassed the teacher LLM used for the distillation process, i.e. `gpt-3.5-turbo` emphasizing the potency of the proposed distillation strategy. When set against zero-shot LLM baselines, my model either matches or exceeds their performance. The sole model that distinctly outperformed ours was `GPT-4`. This performance difference suggests that leveraging a more advanced LLM for distillation within my methodology might lead to even more enhanced outcomes. Importantly, this is achieved with significantly reduced computational overhead during inference. This dual advantage of superior performance and efficiency positions my method as a compelling benchmark for future reranking tasks.

On BEIR Benchmark. In the evaluation on the BEIR benchmark, TWOLAR consistently surpassed the performance of most existing baselines. This is particularly significant when juxtaposed with the approach taken by models such as InPars. InPars employs a strategy of fine-tuning a monot5-3b on generated, topic-specific data tailored for each of the 18 datasets within the BEIR

Table 5.1: Results on TREC-DL2019 and TREC-DL2020 datasets by reranking top 100 documents retrieved by BM25. The column titled '#Calls' indicates the exact number of inference times of LLM when reranking the top 100 documents. The 'Input Size' column uses the notation $|q|+n|d|$: $|q|$ represents one query and $n|d|$ indicates the number of documents included. For instance, $|q| + 20|d|$ signifies an input of one query with 20 documents. Best model is highlighted in boldface and the second best is underlined for each metric. All the results apart from the LLM distillation Methods are taken from the original papers.

Method	LLM	Size	#Calls	Input Size	TREC-DL2019			TREC-DL2020		
					nDCG@1	nDCG@5	nDCG@10	nDCG@1	nDCG@5	nDCG@10
BM25	-	-	-	-	54.26	52.78	50.58	57.72	50.67	47.96
Supervised Methods										
monoBERT	BERT	340M	100	$\|q\| + \|d\|$	79.07	73.25	70.50	78.70	70.74	67.28
monoT5	T5-xl	3B	100	$\|q\| + \|d\|$	79.07	73.74	71.83	80.25	72.32	68.89
RankT5	T5-xl	3B	100	$\|q\| + \|d\|$	77.38	73.94	71.22	<u>80.86</u>	72.99	69.49
LLM distillation Methods										
RankGPT	DeBertaV2	184M	100	$\|q\| + \|d\|$	78.68	69.77	66.56	59.26	59.83	59.43
TWOLAR-large	Flan-T5-large	783M	100	$\|q\| + \|d\|$	79.84	75.94	72.82	79.94	71.35	67.61
TWOLAR-xl	Flan-T5-xl	3B	100	$\|q\| + \|d\|$	78.29	<u>76.71</u>	<u>73.51</u>	80.25	73.73	**70.84**
Zero-shot LLM Methods										
RankGPT	gpt-3.5-turbo	154B*	10	$\|q\| + 20\|d\|$	<u>82.17</u>	71.15	65.80	79.32	66.76	62.91
RankGPT	gpt-4	1T*	2†	$\|q\| + 20\|d\|$	**82.56**	**79.16**	**75.59**	78.40	<u>74.11</u>	<u>70.56</u>
PRP-Sliding-10	Flan-T5-xl	3B	990	$\|q\| + 2\|d\|$	75.58	71.23	68.66	75.62	69.00	66.59
PRP-Sliding-10	Flan-T5-xxl	11B	990	$\|q\| + 2\|d\|$	64.73	69.49	67.00	75.00	70.76	67.35
PRP-Sliding-10	Flan-UL2	20B	990	$\|q\| + 2\|d\|$	78.29	75.49	72.65	**85.80**	**75.35**	70.46

* OpenAI has not publicly released the amount of parameters and the numbers are based on public estimates [51] [2].
† in [46] gpt-4 reranks the top-30 passages reranked by gpt-3.5-turbo

benchmark. This strategy means that, for each dataset, their model has been exposed to data related to the topic in question.

In contrast, TWOLAR has never been exposed to any topic-specific data, making it genuinely zero-shot when facing new topics and tasks. Furthermore, the efficiency of my method is evident as I utilize a single model that negates the necessity for continuous fine-tuning for different applications.

It's worth noting the performance variations across different datasets within BEIR. In datasets with a specific focus, such as BioASQ, InPars tends to perform better due to its targeted fine-tuning on artificial topic-specific data. However, in datasets where queries are centered around general knowledge, like DBpedia entity, TWOLAR demonstrates a clear advantage over InPars. This marked performance difference in more generalized datasets highlights the strength of TWOLAR's unsupervised approach and its applicability in a broad range of scenarios.

Table 5.2: Results on the BEIR Benchmark by reranking the top 100 documents with different retrievers. Best model is in boldface and second best is underlined for each dataset. Evaluation for InPars on CQADupStack is absent due to its unavailability on the Hugging Face hub.

Retriever	BM25						SPLADE						DRAGON					
Reranker	-	MonoT5-3B	InPars	RankGPT-Deberta	TWOLAR-xl	TWOLAR-large	-	MonoT5-3B	InPars	RankGPT-Deberta	TWOLAR-xl	TWOLAR-large	-	MonoT5-3B	InPars	RankGPT-Deberta	TWOLAR-xl	TWOLAR-large
nDCG@10																		
TREC-COVID	59.5	79.8	82.5	79.4	82.7	84.3	72.8	82.9	85.7	80.1	85.2	**86.9**	75.8	82.8	84.8	82.6	84.6	86.8
NFCorpus	32.2	37.4	35.0	33.3	36.6	35.7	34.8	39.2	38.8	33.2	37.3	35.5	33.9	**39.7**	39.3	33.2	37.9	35.7
FiQA-2018	23.6	46.1	46.2	32.7	41.9	41.1	34.8	50.0	50.0	33.7	44.8	43.8	35.7	**51.2**	50.9	43.1	45.3	44.8
ArguAna	30.0	33.4	32.8	21.1	32.9	34.7	38.8	31.7	31.2	18.6	32.9	34.6	**46.9**	41.5	40.9	25.7	42.8	45.5
Tóuche-2020	**44.2**	31.6	29.6	37.7	37.1	33.4	24.6	29.8	28.7	36.4	35.2	30.4	26.3	30.6	29.4	38.2	36.0	31.5
Quora	78.9	84.1	84.8	78.8	87.2	86.0	83.5	84.3	85.1	80.3	87.4	86.0	**87.5**	83.5	84.4	78.7	87.2	85.7
SCIDOCS	14.9	19.0	19.2	16.1	19.5	18.3	15.9	19.9	**20.9**	16.4	20.2	18.8	15.9	19.8	20.7	16.4	20.2	18.8
SciFact	67.9	76.4	73.5	70.5	**76.5**	75.6	70.2	76.4	76.0	69.1	75.6	74.7	67.8	76.0	75.7	69.4	75.6	74.7
NQ	30.6	56.8	57.8	46.1	58.0	57.7	53.7	65.9	66.4	50.6	66.8	65.8	53.8	65.1	66.6	50.6	**66.9**	66.2
HotpotQA	63.3	74.2	76.5	69.9	76.7	75.9	68.7	74.1	77.1	70.5	**77.7**	76.4	66.2	72.9	75.7	69.8	76.4	75.3
DBPedia	31.8	44.8	44.0	41.9	48.0	47.8	43.6	48.2	51.1	45.9	**52.9**	51.6	41.9	47.2	50.3	44.9	52.1	51.3
FEVER	65.1	83.2	85.5	80.2	84.9	83.4	79.3	85.0	88.0	81.8	87.5	85.4	78.0	84.7	**87.7**	81.7	87.2	85.2
Climate-FEVER	16.5	27.4	30.1	24.2	26.9	26.1	22.9	28.7	**32.8**	25.9	28.9	27.9	22.7	28.6	32.5	25.9	28.6	27.4
CQADupStack	30.2	41.5	-	34.7	41.2	40.6	33.4	43.7	-	35.9	43.6	42.7	35.4	**44.4**	-	36.0	44.2	43.4
Robust04	40.8	56.6	58.7	52.8	57.9	58.3	46.7	62.1	**64.3**	57.3	64.9	65.2	48.1	61.3	63.2	56.6	63.4	63.7
Signal-1M	33.1	32.2	32.9	33.4	**33.8**	33.9	30.0	29.4	30.3	30.0	30.1	30.5	30.0	29.7	30.4	29.4	30.2	30.1
BioASQ	52.3	57.2	**59.8**	53.0	56.2	56.0	49.7	54.1	57.2	49.5	54.6	53.8	43.4	51.9	54.4	48.0	51.9	50.8
TREC-NEWS	39.5	48.5	49.8	51.8	52.7	50.8	41.5	50.0	50.9	53.4	53.3	50.7	44.4	49.5	50.8	52.1	**53.8**	50.0
avg nDCG@10																		
BEIR 18	41.9	51.7	-	47.6	52.8	52.2	46.9	53.0	-	48.3	54.4	53.4	47.4	53.4	-	48.5	**54.7**	53.7
BEIR 17	42.6	52.3	52.9	48.4	53.5	52.9	47.8	53.5	55.0	49.0	55.0	54.0	48.1	53.9	55.2	49.2	**55.3**	54.3

Table 5.3: Results on the BEIR Benchmark by reranking the top 1000 BM25 retrieved documents. Best model is in boldface and second best is underlined for each dataset. All the results, apart from TWOLAR-xl, are from [21].

	BM25	monoT5-3B	InPars-v2	RankT5	TWOLAR-xl
nDCG@10					
TREC-COVID	59.5	80.1	**84.6**	82.3	84.3
NFCorpus	32.2	38.3	38.5	**39.9**	37.3
FiQA-2018	23.6	**50.9**	50.9	49.3	45.2
ArguAna	30.0	37.9	36.9	**40.6**	32.7
Tóuche-2020	44.2	30.9	29.1	**48.6**	35.9
Quora	78.9	83.5	84.5	81.9	**87.3**
SCIDOCS	14.9	19.7	**20.8**	19.1	20.3
SciFact	67.9	**77.4**	77.4	76.0	76.8
NQ	30.6	62.5	63.8	**64.7**	64.2
HotpotQA	63.3	76.0	79.1	75.3	**79.5**
DBPedia	31.8	47.2	49.8	45.9	**52.0**
FEVER	65.1	84.8	**87.2**	84.8	86.7
Climate-FEVER	16.5	28.8	**32.3**	27.5	27.8
CQADupStack	30.2	**44.9**	44.8	-	43.8
Robust04	40.8	61.5	63.2	-	**64.2**
Signal-1M	**33.1**	30.2	30.8	31.9	31.5
BioASQ	52.3	56.6	**59.5**	57.9	56.0
TREC-NEWS	39.5	47.7	49.0	-	**53.2**
avg nDCG@10					
BEIR 18	41.9	53.3	**54.5**	-	54.4
BEIR 15	42.9	53.7	54.9	**55.0**	54.5

Chapter 6

Ablation studies

I conducted an extensive ablation studies in order to validate the design choices. Due to computational constraints, these experiments were performed using the smaller `flan-t5-small` checkpoint, with 77M parameters. Furthermore, I evaluated the models on reranking the top-100 documents from a subset of 8 smallest datasets from the BEIR benchmark, including TREC-COVID, SciFact, NFCorpus, Tóuche-2020, DBPedia, Robust04, Signal-1M, and TREC-NEWS.

The results are summarized in Table 6.1.

Scoring Strategy Effectiveness. I compared the proposed scoring strategy, which utilizes the difference between the '`True`' and '`False`' logits, with the strategy used in RankT5, based on the logit of an extra token in the T5 vocabulary. My analysis indicated superior performance for my proposed strategy, achieving an average nDCG@10 of 46.5, in comparison to 45.7 for the RankT5 scoring approach.

Documents per training samples. I trained models with varying numbers of documents per training sample: 10, 20, and 30. The results suggest a clear advantage in using more than 10 documents per sample. The trade-off between 20 and 30 is less clear, with nDCG@10 scores of 46.3 and 46.5 respectively,

suggesting diminishing returns beyond 20 documents.

Effectiveness of first source of supervision. In an attempt to ascertain the impact of each retrieval strategy in my first stage of distillation, I carried out four individual experiments by strategically excluding each source of supervision (BM25, SPLADE, DRAGON, monoT5) from the training set and training the model on the residual data. This approach allowed us to understand the individual contribution of each retrieval strategy to the overall performance of the model.

Intriguingly, the results demonstrate that BM25, even being a traditional bag-of-words method, still plays a critical role in the model's performance. The exclusion of BM25 led to a decrease in performance, indicating its significance in the retrieval phase. It further underscores the fact that despite the advent of more complex and semantic-based retrieval methods, the importance of traditional retrieval strategies like BM25 in training effective reranking models should not be underestimated.

Impact of Query Type. I also trained models exclusively on cropped sentences, doct5query generated queries, and a mixed subset of both types. The model trained only with doct5query generated queries, which are formulated as natural language questions, demonstrated an overall higher average performance than the model trained only on cropped sentences.

Interestingly, for datasets where the queries were predominantly formed as 'what' or 'how' questions, such as TREC-COVID, the model trained on doct5query queries delivered strong superior performance.

Conversely, the mode trained with cropped sentences performed better in specific datasets where the queries are not expressed as a question in natural language. For example, the queries in SciFact are expert-written claims, aiming to find evidence in annotated abstracts. Here, the model trained with

	TREC-COVID	SciFact	NFCorpus	Tóuche-2020	DBPedia	Robust04	Signal-1M	TREC-NEWS	avg nDCG@10
Score Strategy	effectiveness of score strategy - 19K train samples								
Difference	74.0	67.9	**31.9**	**35.7**	**38.8**	**47.4**	32.5	**43.7**	**46.5**
Extra id	**74.1**	**69.2**	31.5	32.2	36.2	47.0	**34.1**	41.2	45.7
# documents	effectiveness of amount of documents - 19K train samples								
30	**74.0**	67.9	31.9	**35.7**	**38.8**	47.4	**32.5**	**43.7**	**46.5**
20	73.0	**69.8**	**32.5**	31.7	37.9	**47.5**	31.9	40.8	46.3
10	72.3	65.6	29.6	28.4	34.2	43.2	30.4	37.9	42.7
Not used source	effectiveness of first source of supervision - ∼14K train samples								
- BM25	72.7	70.3	31.8	31.8	37.7	47.0	31.9	41.5	45.6
- SPLADE	73.9	**70.9**	**33.6**	32.7	**38.6**	**48.8**	32.4	42.5	**46.3**
- DRAGON	**74.0**	67.7	32.9	**33.9**	37.6	47.7	**33.1**	43.2	46.2
- monoT5	73.9	69.4	31.8	33.0	36.3	46.6	32.5	**43.6**	45.9
Type of query	effectiveness of type of query - 9.5K train samples								
Mixed	**75.5**	67.3	30.4	**34.0**	37.2	**46.2**	**31.8**	41.6	**45.5**
Sentence	67.2	**67.9**	**31.4**	32.7	32.2	44.8	31.7	39.7	43.4
docT5query	74.6	59.4	31.2	33.4	**37.8**	44.9	28.1	**44.0**	44.2

Table 6.1: Ablation studies

cropped sentences achieved an nDCG@10 score of 67.9, significantly outperforming the model trained with doct5query queries, which scored 59.4.

When I trained the model on a mixed subset comprising an equal proportion of both query types, it exhibited the best overall performance. This highlights the benefit of a diverse training regimen incorporating both natural language questions (doct5query) and sentences cropped directly from documents.

These results, summarized in 6.1, underscore the value of the proposed scoring strategy, the importance of incorporating sufficient documents per training sample, the significant contribution of BM25 as a supervision source, and the advantages of a mixed query approach.

Chapter 7

Conclusion

The paradigm shift, enabled by LLMs, suggests that traditional methods relying heavily on handcrafted labeled data might no longer be the most effective or efficient approach for certain machine learning tasks. Indeed, as LLMs continue to showcase their prowess, there is a promising realization that they can be harnessed to provide the needed supervision, reducing the need for manual data labeling. However, tasks that demand efficiency, such as IR, often cannot deploy LLMs directly due to their substantial computational overhead. In such scenarios, distillation enables the retention of the LLM's capabilities in a more computationally amenable format.

In this work, I presented a novel two-step LLM-Augmented distillation approach for passage reranking. My method capitalizes on the strengths of LLMs to enable computationally efficient IR systems, with performance comparable or even superior to that of state of the art baselines and a reduction in size by several orders of magnitude. The experiments, conducted across various benchmarks, demonstrate robustness and generality of my approach across domains. An ablation offers further insight about the crucial elements of my architectural design. All the data, code and models will be made publicly available.[1]

Looking forward, TWOLAR offers promising avenues for scalability. In

[1] `https://github.com/Dundalia/TWOLAR`

the future, I plan to further the experimentation by substituting the 3B model with an 11B version, expanding the number of queries, increasing the sources of supervision, or even refining the quality of the LLM used for distillation, for example by experimenting with more powerful generative language models.

Appendix A

Model Checkpoints

Language Models:

- Flan-T5-small: https://huggingface.co/google/flan-t5-small
- Flan-T5-large: https://huggingface.co/google/flan-t5-large
- Flan-T5-xl: https://huggingface.co/google/flan-t5-xl

Retrievers:

- DRAGON+: https://dl.fbaipublicfiles.com/dragon/checkpoints/DRAGON-Plus/checkpoint_best.ckpt
- SPLADE++: https://huggingface.co/naver/splade-cocondenser-ensembledistil

Rerankers

- MonoT5: https://huggingface.co/castorini/monot5-3b-msmarco-10K
- deberta-10K-rank_net: https://drive.google.com/file/d/1-KEpJ2KnJCqiJof4zNEA4m78tnwgxKhb/view?usp=share_link

InPars Models:

- TREC-COVID: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-trec_covid
- NFCorpus: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-nfcorpus
- FiQA-2018: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-fiqa
- ArguAna: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-arguana
- Tóuche-2020: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-touche
- Quora: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-quora
- SCIDOCS: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-scidocs
- SciFact: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-scifact
- NQ: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-nq
- HotPotQA: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-hotpotqa
- DBPedia: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-dbpedia
- FEVER: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-fever

- Climate-FEVER: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-climate_fever
- Robust04: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-robust04
- Signal-1M: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-signal
- BioASQ: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-bioasq
- TREC-NEWS: https://huggingface.co/zeta-alpha-ai/monot5-3b-inpars-v2-trecnews

Bibliography

[1] Anthropic. Introducing claude. 2023. URL: https://www.anthropic.com/index/introducing-claude.

[2] J. A. Baktash and M. Dawodi. Gpt-4: a review on advancements and opportunities in natural language processing, 2023. arXiv: 2305.03195 [cs.CL].

[3] L. Bonifacio, H. Abonizio, M. Fadaee, and R. 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, pages 2387–2392, Madrid, Spain. Association for Computing Machinery, 2022. ISBN: 9781450387323. DOI: 10.1145/3477495.3531863. URL: https://doi.org/10.1145/3477495.3531863.

[4] S. Bruch, X. Wang, M. Bendersky, and M. Najork. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, pages 75–78, Santa Clara, CA, USA. Association for Computing Machinery, 2019. ISBN: 9781450368810. DOI: 10.1145/3341981.3344221. URL: https://doi.org/10.1145/3341981.3344221.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML

'05, pages 89–96, Bonn, Germany. Association for Computing Machinery, 2005. ISBN: 1595931805. DOI: 10.1145/1102351.1102363. URL: https://doi.org/10.1145/1102351.1102363.

[6] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: an open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[7] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: scaling language modeling with pathways, 2022. arXiv: 2204.02311 [cs.CL].

[8] G. G. Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010.

[9] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou,

Q. V. Le, and J. Wei. Scaling instruction-finetuned language models, 2022. arXiv: 2210.11416 [cs.LG].

[10] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos. Overview of the trec 2020 deep learning track, 2021. arXiv: 2102.07662 [cs.IR].

[11] J. Devlin, M.-W. Chang, K. Lee, and K. 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. Association for Computational Linguistics, June 2019. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[12] Y. Fan, X. Xie, Y. Cai, J. Chen, X. Ma, X. Li, R. Zhang, J. Guo, et al. Pre-training methods in information retrieval. *Foundations and Trends® in Information Retrieval*, 16(3):178–317, 2022.

[13] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. From distillation to hard negative sampling: making sparse neural ir models more effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2353–2359, 2022.

[14] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. Splade v2: sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*, 2021. arXiv: 2109.10086 [cs.IR].

[15] T. Formal, B. Piwowarski, and S. Clinchant. Splade: sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 2288–2292, Virtual Event, Canada. Association for Computing Machinery, 2021. ISBN: 9781450380379.

DOI: 10.1145/3404835.3463098. URL: https://doi.org/10.1145/3404835.3463098.

[16] F. Gilardi, M. Alizadeh, and M. Kubli. Chatgpt outperforms crowd-workers for text-annotation tasks, 2023. arXiv: 2303.15056 [cs.CL].

[17] A. Gudibande, E. Wallace, C. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine, and D. Song. The false promise of imitating proprietary llms, 2023. arXiv: 2305.15717 [cs.CL].

[18] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, October 2016. DOI: 10.1145/2983323.2983769. URL: https://doi.org/10.1145%2F2983323.2983769.

[19] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6):102067, 2020.

[20] P. He, J. Gao, and W. Chen. Debertav3: improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

[21] V. Jeronymo, L. Bonifacio, H. Abonizio, M. Fadaee, R. Lotufo, J. Zavrel, and R. Nogueira. InPars-v2: large language models as efficient dataset generators for information retrieval, 2023. DOI: 10.48550/ARXIV.2301.01820. URL: https://arxiv.org/abs/2301.01820.

[22] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. DOI: 10.1162/tacl_a_00276. URL: https://aclanthology.org/Q19-1026.

[23] M. Lee. Gelu activation function in deep learning: a comprehensive mathematical analysis and performance, 2023. arXiv: 2305.12073 [cs.LG].

[24] H. Li. Learning to rank for information retrieval and natural language processing, second edition. *Synthesis Lectures on Human Language Technologies*, 7:1–123, January 2015. DOI: 10.2200/S00607ED2V01Y201410HLT026.

[25] J. Lin, R. Nogueira, and A. Yates. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature, 2022.

[26] S.-C. Lin, A. Asai, M. Li, B. Oguz, J. Lin, Y. Mehdad, W.-t. Yih, and X. Chen. How to train your dragon: diverse augmentation towards generalizable dense retrieval. *arXiv preprint arXiv:2302.07452*, 2023.

[27] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009. ISSN: 1554-0669. DOI: 10.1561/1500000016. URL: https://doi.org/10.1561/1500000016.

[28] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[29] X. Ma, X. Zhang, R. Pradeep, and J. Lin. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*, 2023.

[30] B. Mitra and N. Craswell. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*, 2017.

[31] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: a human-generated MAchine reading COmprehension dataset, 2017. URL: https://openreview.net/forum?id=Hk1iOLcle.

[32] R. Nogueira and K. Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[33] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics, November 2020. DOI: 10.18653/v1/2020.findings-emnlp.63. URL: https://aclanthology.org/2020.findings-emnlp.63.

[34] R. Nogueira, J. Lin, and A. Epistemic. From doc2query to doctttttquery. *Online preprint*, 6:2, 2019.

[35] R. Nogueira, W. Yang, K. Cho, and J. Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

[36] OpenAI. Gpt-4 technical report, 2023. arXiv: 2303.08774 [cs.CL].

[37] OpenAI. Introdocing chatgpt. 2022. URL: https://openai.com/blog/chatgpt.

[38] B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4, 2023. arXiv: 2304.03277 [cs.CL].

[39] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, et al. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*, 2023.

[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[41] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[42] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[43] G. M. Rosa, L. Bonifacio, V. Jeronymo, H. Abonizio, M. Fadaee, R. Lotufo, and R. Nogueira. No parameter left behind: how distillation and model size affect zero-shot retrieval. *arXiv preprint arXiv:2206.02873*, 2022.

[44] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing &amp Management*, 24(5):513–523, January 1988. DOI: 10.1016/0306-4573(88)90021-0.

[45] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. arXiv: 1910.01108 [cs.CL].

[46] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, and Z. Ren. Is chatgpt good at search? investigating large language models as re-ranking agent. *arXiv preprint arXiv:2304.09542*, 2023.

[47] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: an instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[48] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, S. Shakeri, D. Bahri, T. Schuster, H. S. Zheng, D. Zhou, N. Houlsby, and D. Metzler. Ul2: unifying language learning paradigms, 2023. arXiv: 2205.05131 [cs.CL].

[49] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. 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.

[50] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E.

Grave, and G. Lample. Llama: open and efficient foundation language models, 2023. arXiv: 2302.13971 [cs.CL].

[51] A. VanBuskirk. Gpt-3.5 turbo vs gpt-4: what's the difference?, March 2023. URL: https://blog.wordbot.io/ai-artificial-intelligence/gpt-3-5-turbo-vs-gpt-4-whats-the-difference/.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. arXiv: 1706.03762 [cs.CL].

[53] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: aligning language model with self generated instructions, 2022. arXiv: 2212.10560 [cs.CL].

[54] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: a dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[55] C. Zhai et al. Statistical language models for information retrieval a critical review. *Foundations and Trends® in Information Retrieval*, 2(3):137–213, 2008.

[56] W. X. Zhao, J. Liu, R. Ren, and J.-R. Wen. Dense text retrieval based on pretrained language models: a survey. *arXiv preprint arXiv:2211.14876*, 2022.

[57] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, Z. Dou, and J.-R. Wen. Large language models for information retrieval: a survey. *arXiv preprint arXiv:2308.07107*, 2023.

[58] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky. Rankt5: fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313, 2023.