Instruction Distillation Makes Large Language Models Efficient Zero-shot Rankers

Weiwei Sun[1] **Zheng Chen**[1] **Xinyu Ma**[2] **Lingyong Yan**[2] **Shuaiqiang Wang**[2]
Pengjie Ren[1] **Zhumin Chen**[1] **Dawei Yin**[2] **Zhaochun Ren**[3]
[1]Shandong University, Qingdao, China [2]Baidu Inc., Beijing, China
[3]Leiden University, Leiden, The Netherlands
{sunnweiwei,xinyuma2016,lingyongy}@gmail.com
yindawei@acm.org, z.ren@liacs.leidenuniv.nl

Abstract

Recent studies have demonstrated the great potential of Large Language Models (LLMs) serving as zero-shot relevance rankers. The typical approach involves making comparisons between pairs or lists of documents. Although effective, these *listwise* and *pairwise* methods are not efficient and also heavily rely on intricate prompt engineering. To tackle this problem, we introduce a novel instruction distillation method. The key idea is to distill the pairwise ranking ability of open-sourced LLMs to a simpler but more efficient *pointwise* ranking. Specifically, given the same LLM, we first rank documents using the effective pairwise approach with complex instructions, and then distill the teacher predictions to the pointwise approach with simpler instructions. Evaluation results on the BEIR, TREC, and ReDial datasets demonstrate that instruction distillation can improve efficiency by 10 to 100× and also enhance the ranking performance of LLMs. Furthermore, our approach surpasses the performance of existing supervised methods like monoT5 and is on par with the state-of-the-art zero-shot methods. The code to reproduce our results is available at www.github.com/sunnweiwei/RankGPT/InstructDistill.

1 Introduction

Large Language Models (LLMs), such as ChatGPT and GPT-4, have achieved remarkable success in various Natural Language Processing (NLP) tasks (OpenAI, 2022; 2023). One notable capability of LLMs is their ability to solve tasks using carefully designed prompts or instructions (Microsoft, 2023). This has drawn much attention from the Information Retrieval (IR) community given its potential to significantly reduce the huge annotation costs (Shi et al., 2023; Sun et al., 2023).

Relevance ranking has been the most critical problem in IR, which aims at ranking a set of candidate items by their relevance given the query (Fan et al., 2021). Recently, there has been a series of works using large models as zero-shot rankers through pointwise, pairwise, and listwise ranking prompting, and these have achieved impressive results on IR benchmarks (Sun et al., 2023; Ma et al., 2023; Qin et al., 2023).

Employing LLMs for ranking tasks still faces several practical challenges, including application efficiency and output stability. On one hand, both listwise and pairwise ranking methods suffer from efficiency issues. For listwise ranking (Sun et al., 2023; Ma et al., 2023), the exponential time complexity of the Transformer with respect to input length renders it impractical for many industrial applications. Pairwise ranking requires pairing every document with every other, with the obvious drawback being its costly $O(n^2)$ calls to LLMs (Qin et al., 2023). On the other hand, while pointwise ranking is more efficient, it compromises on effectiveness (Liang et al., 2022). The pretraining objective of LLMs isn't inherently tailored for ranking tasks (i.e., generative language modeling vs. relevance ranking), meaning its prediction probability isn't calibrated to the relevance score (Zhao

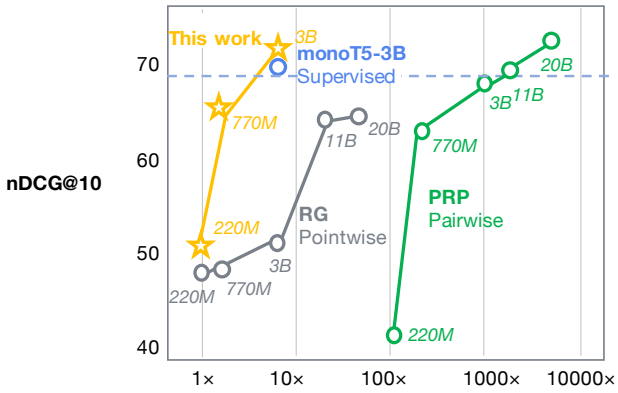

Figure 1: The average nDCG@10 of various LLMs-based re-ranking methods on TREC benchmarks. The horizontal axis represents the speed of each method relative to monoT5-Base (Nogueira et al., 2020), as measured by the average latency time per query. All methods are based on the T5 series foundation models. RG refers to the relevance generation method and PRP refers to the pairwise ranking method.

et al., 2021). Other challenges, such as unstable outputs, position bias, and repetitions from LLMs, become more pronounced in IR tasks, where deterministic output in terms of relevance is crucial (Sun et al., 2023).

To address these challenges, this paper introduces a novel *Instruction Distillation* method to enhance the efficiency and stability of LLMs in the ranking task. The key idea is to distill the predictions of pairwise ranking (PRP) with computationally demanding instruction (*teacher instruction*) to the efficient pointwise prompting method but with simpler instruction (*student instruction*). Through this distillation process, the task instructions used for ranking are substantially simplified, leading not only to increased efficiency but also to enhanced performance. In this work, we use open-sourced LLMs FLAN-T5 and our method is zero-shot text ranking since FLAN-T5 is not directly exposed to human-labeled data.

We empirically evaluate instruction distilled models against other baselines in Figure 1. These distilled student models are between 10 and 100× more efficient compared to their teacher models (i.e., PRP) while also yielding significant enhancements. Compared to vanilla pointwise ranking methods (Relevance Generation methods, RG), our distilled models show a 40% performance improvement in terms of nDCG@10. Remarkably, our distilled FLAN-T5-XL model even surpasses the SOTA supervised systems like monoT5-3B (Nogueira et al., 2020) in IR benchmarks. This is particularly notable as it achieves this without relying on any human relevance judgments. We also condu Further verification is conducted on various ranking tasks such as the BEIR benchmark and the conversational recommendation tasks present in the REDIAL benchmark.

In summary, this paper makes the following contributions:

- We propose Instruction Distillation, an unsupervised approach to specialize LLMs on IR tasks by distilling instructions.
- We show the instruction distilled LLM is both more efficient and effective compared to existing zero-shot LLMs with the same amount of parameters.
- We show how good performance of both passage ranking and movie recommendation tasks can be achieved by our method, which can also defeat the state-of-the-art supervised method.

2 Related Work

2.1 LLMs for Information Retrieval

Recently, we have found increasing applications of LLMs in information retrieval (Yu et al., 2023; Wu et al., 2023; Zhu et al., 2023). These methods can be broadly divided into two categories: synthetic data generation and relevance ranking.

Several approaches have been proposed to utilize LLMs to generate synthetic data for IR. For example, SGPT (Muennighoff, 2022) generates text embeddings using GPT for dense retrieval; and Gao et al. (2022); Wang et al. (2023a) proposes to generate pseudo-documents using LLMs and retrieve these pseudo-documents first using queries. Dai et al. (2023) proposes to generate pseudo-queries for few-shot learning.

In addition, LLMs have also been used for relevance ranking tasks. UPR (Sachan et al., 2022a) and SGPT-CE (Muennighoff, 2022) introduce instructional query generation methods, which rank documents based on the generation likelihood of query given the document. HELM (Liang et al., 2022) utilizes instructional relevance generation for ranking, prompting LLMs to generate relevance proxy tokens and rank documents based on the generation probability. RankGPT (Sun et al., 2023) proposes a zero-shot permutation generation method, which prompts LLMs to directly generation the ranking permutation and its performance surpasses supervised models when based on GPT4. Qin et al. (2023) proposes a pairwise ranking prompting method (PRP) based on open-sourced LLMs.

Though good results are achieved by the methods above, two challenges still remain: (1) Unstable output, sensitivity of input, repetition, and position bias could harm the performance severely. (2) Sophisticated instruction techniques and task designs are commonly adapted to achieve high performance at the cost of computational complexity. It would be hard for computationally costly methods to be applied to a practical scenario.

2.2 LLMs Distillation

Despite their impressive capabilities, LLMs such as GPT-4 often come with high costs and lack open-source availability. As a result, considerable research has explored various ways to distill the capabilities of LLMs into specialized, customized models. For instance, Fu et al. (2023) and Magister et al. (2022) have successfully distilled the reasoning ability of LLMs into smaller models. Self-instruct (Wang et al., 2023b; Taori et al., 2023) propose iterative approaches to distill GPT-3 using their outputs.

Additionally, Sachan et al. (2022b) and Shi et al. (2023) utilize the generation probability of LLMs to improve retrieval systems. Snell et al. (2022) introduces a similar context distillation method to simplify the overlong context when prompting LLMs on Text-to-SQL tasks. This paper presents the Instruction Distillation method, aiming at distilling the ability explored by sophisticated instructions into the model using more efficient instructions to enhance the model efficiency and output stability.

3 Method

In this section, we introduce the instruction distillation method in detail. This novel approach enhances both the effectiveness and efficiency of open-sourced LLMs during the inference stage by distilling the capabilities harnessed by complex instructions into a more efficient one. Thus, when deploying to real-world applications, our methodology is able to obtain good performance which necessitates only lower computation costs compared to others.

3.1 Task Formalization

The task of *relevance ranking* can be formally defined as follows: Given a query q and a set of candidate items $D = \{d_1, \ldots, d_n\}$, the objective is to determine the ranking of these

candidates, represented as $R = \{r_1, \ldots, r_n\}$. Here, $r_i \in \{1, 2, \ldots, n\}$ denotes the rank of candidate d_i. For instance, if $r_i = 3$, it denotes that d_i is ranked third among the n candidates.

A ranking model, denoted as $f(\cdot)$, assigns scores to the candidates based on their relevance to the query:

$$s_i = f(q, d_i) \tag{1}$$

Subsequently, the candidates are ranked according to these relevance scores: $r_i = \text{arg sort}_i(s_1, \ldots, s_n)$

3.2 Prompting LLMs for Ranking Tasks

Recent studies have explored the potential of using Large Language Models (LLMs) for the re-ranking task. Diverse prompting strategies have been explored. Based on the type of instruction employed, existing strategies can be categorized into three types: (1) pointwise ranking, (2) pairwise ranking, and (3) listwise ranking (Wu et al., 2023; Zhu et al., 2023).

Pointwise Ranking assigns an independent score to each item d_i, subsequently ranking the set D based on these scores. A prevalent pointwise prompting approach for LLMs is instructional relevance generation, which is exemplified in HELM (Liang et al., 2022). In this approach, LLMs are prompted to output either "Yes" or "No" to determine the relevance of the candidates to a given query. The generation probability is then converted to the relevance score:

$$s_i = \begin{cases} 1 + f(\text{Yes} \mid \mathcal{I}_{\text{RG}}(q, d_i)), & \text{if output Yes} \\ 1 - f(\text{No} \mid \mathcal{I}_{\text{RG}}(q, d_i)), & \text{if output No} \end{cases} \tag{2}$$

Here $f(\cdot)$ represents the large language model, and $\mathcal{I}_{\text{RG}}$ denotes the relevance generation instruction that converts the input q and d_i into the test-based prompt.

$$s_i = \frac{1}{|q|} \sum_t \log p(q_t \mid q_{<t}, p_i, \mathcal{I}_{\text{query}}) \tag{3}$$

Pairwise Ranking is employed by PRP (Qin et al., 2023). In this technique, both the query and a pair of candidate items serve as prompts, guiding the LLMs in ranking tasks. For every pair of items d_i and d_j, a specific pairwise comparison instruction, denoted by $\mathcal{I}_{\text{PRP}}$, is employed to instruct the LLMs, i.e., $f(\cdot)$, to determine which item is more relevant to the given query. This can be formalized as:

$$c_{i,j} = \begin{cases} 1, & \text{if } f(\mathcal{I}_{\text{PRP}}(q, d_i, d_j)) = i \\ 0, & \text{if } f(\mathcal{I}_{\text{PRP}}(q, d_i, d_j)) = j \\ 0.5, & \text{else} \end{cases} \tag{4}$$

Here, $c_{i,j}$ denotes the LLM's choice. Considering that LLMs may exhibit sensitivity to the order of text in the prompt, for every pair d_i and d_j, PRP consults the LLM twice, inverting their order between $\mathcal{I}_{\text{PRP}}(q, d_i, d_j)$ and $\mathcal{I}_{\text{PRP}}(q, d_j, d_i)$. Subsequently, to compute the relevance score of the i-th candidate d_i, PRP compares d_i against all other candidates in the set D:

$$s_i = \sum_{j \neq i} c_{i,j} + (1 - c_{j,i}) \tag{5}$$

The final relevance score aggregates all comparison results.

Listwise Ranking has been adopted by Sun et al. (2023); Ma et al. (2023). This approach involves feeding a set of items into the LLMs, where each item is identified by a unique identifier (e.g., [1], [2], etc.). The LLMs are then instructed to generate a permutation of these items, such as "[2] > [3] > [1] > ...":

$$\textbf{Perm} = f(\mathcal{I}_{\text{List}}(q, d_1, d_2, \ldots, d_n)) \tag{6}$$

This generated permutation **Perm** can be readily transformed into ranking results R, which bypasses the necessity to compute an explicit relevance score, s_i, for each candidate d_i. To ensure consistency in notation with scoring-based methodologies, the relevance score s_i is defined as the reciprocal of its rank: $s_i := \frac{1}{r_i}$.

Table 1: Computational complexity of different instruction methods. n is the number of items to be ranked. k is a constant related to the sliding window method.

Instruction	Complexity	Examples
Pointwise Ranking	$O(n)$	(Liang et al., 2022)
Pairwise Ranking	$O(n^2)$	(Qin et al., 2023)
Listwise Ranking	$O(k * n)$	(Sun et al., 2023)

3.3 Computational Complexity of Different Instructions.

Different ranking instructions offer various trade-offs in terms of efficiency and effectiveness. A summary of these instructions is listed in Table 1. Among these, the pointwise ranking is computationally the most efficient, having a complexity of $O(N)$. Nevertheless, this approach requires the model to yield a calibrated pointwise score, a feat which is notably challenging.

In contrast, the pairwise ranking paradigm resolves the calibration issue by engaging in one-to-one pairwise comparisons. This solution, however, elevates the computational complexity to $O(N^2)$. To tackle this, Qin et al. (2023) propose two methods to curtail the pairwise ranking's complexity: sorting and the sliding window technique. While promising, these methods are still in their nascent stages, proving challenging to stabilize and parallelize.

On another note, listwise ranking demonstrates good performance when tested on commercial and also proprietary LLMs, such as GPT-4. However, it performs poorly on smaller, open-source models. A possible reason could be the inferior comprehension of instructions in these open-source counterparts.

In summary, each ranking method comes with its set of pros and cons: the pointwise approach is efficient but may not be highly effective; the pairwise method is effective but computationally demanding; and the listwise method is most effective but limited to closed-source LLMs like GPT-4. These insights set the stage for our novel solution – the *instruction distillation* strategy., which we will introduce in the next section.

3.4 Instruction Distillation

The key idea of Instruction Distillatio is to distill the ability obtained from the complex but effective instruction technique (e.g., pairwise ranking instruction) into a model that is more efficient with the simple instruction technique (e.g., pointwise ranking instruction). We denote the sources of relevance scores or ranking results with superscripts $^{\mathrm{t}}$ and $^{\mathrm{s}}$ for teacher instruction and simplified student instruction, respectively. Our method unfolds in three stages: (1) Candidate generation, (2) Teacher inference, and (3) Student learning.

- **Candidate generation.** Suppose we have a dataset comprising a set of queries Q and a corresponding set of items $\mathcal{D}$. It is worth mentioning that none of the queries require a labeled item. For a query $q \in Q$, an unsupervised retriever (e.g., BM25) is employed to fetch n potentially relevant candidate samples $D = (d_1, d_2, \ldots, d_n)$ from the item set $\mathcal{D}$.
- **Teacher inference.** Then, LLMs with costly pairwise ranking are employed as the *teacher* models to re-rank the candidate set $D = (d_1, d_2, \ldots, d_n)$ corresponding to each query q. To adopt the pairwise method, the n items are juxtaposed in pairs, resulting in $n(n-1)$ ordered tuples (d_i, d_j) where $i \neq j$. The model then scores the relevance of d_i and d_j to the given query q using Eq. (5). Based on these scores, each document d_i is assigned a rank r_i^{t} for every query q.
- **Student learning.** In this phase, the pointwise ranking model serves as the *student*. To leverage the ranking lists r_i^{t} generated by the teacher, we employ the RankNet loss (Burges et al., 2005) to optimize the student model. RankNet is a pairwise loss

function that measures the accuracy of relative ordering between items:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{1}_{r_i^{\mathrm{t}} < r_j^{\mathrm{t}}} \log(1 + \exp(s_i^{\mathrm{s}} - s_j^{\mathrm{s}}))$$

Unlike other loss functions that utilize a sparse signal, the RankNet loss offers a richer transfer of ranking information from the teacher to the student.

After the instruction distillation process, the pointwise instruction technique is utilized during the inference stage. See Appendix A for more details about the prompts.

4 Experimental Setup

In order to comprehensively validate the effectiveness of the proposed method. We conduct experiments on a variety of IR tasks, including both the text-based *passage re-ranking* task and the item-based *conversational recommendation* task.

For passage re-ranking, the training data contain 10K queries sampled from the MS MARCO dataset (Campos et al., 2016). Each query is then paired with the top 10 documents retrieved by BM25. The trained models are evaluated on subtasks of TREC (Craswell et al., 2020) benchmarks and BEIR (Thakur et al., 2021) benchmarks. NDCG@1, 5, 10 are chosen as the metrics.

For conversational recommendation, we use the ReDial dataset (Li et al., 2018a), which is a movie recommendation task based on conversation logs between the user and the recommender. The trained models are then evaluated on the official test set. For this setting, Acc@1 is adopted as the metric.

4.1 Datasets

TREC (Campos et al., 2016) is a widely used benchmark dataset in IR research. We use the test sets of the 2019 and 2020 competitions. TREC-DL19 and TREC-DL20 are both derived from MS MARCO datasets with human-generated labels. Each query is paired with 100 retrieved documents retrieved by BM25. They share the same format. TREC-DL19 contains 43 test queries, and TREC-DL20 contains 54 test queries.

BEIR (Thakur et al., 2021) consists of diverse retrieval tasks and domains. We choose eight tasks in BEIR to evaluate the models: (1) Covid retrieves scientific articles for COVID-19 related questions. (2) NFCorpus is a bio-medical IR data. (3) Touche is a argument retrieval datasets. (4) DBPedia retrieves entities from DBpedia corpus. (5) SciFact retrieves evidence for claims verification. (6) Signal retrieves relevant tweets for a given news title. (7) News retrieves relevant news articles for news headlines. (8) Robust04 evaluates poorly performing topics. The evaluation results are averaged over the eight datasets.

Redial (Recommendation Dialogues) (Li et al., 2018b) is an annotated conversational movie recommendation dataset, where users recommend movies to each other.

4.2 Baselines

To compare our methods with existing unsupervised and supervised methods, we choose widely applied methods as below:

- **BM25** is an unsupervised, based on weighted term frequency. It is one of most the commonly adopted retrieval methods.
- **RankGPT** (Sun et al., 2023) is a listwise permutation generation approach based on `gpt-3.5-turbo` and `gpt-4`.
- **Relevance Gerneration** (Sachan et al., 2022a) is a pointwise ranking method based on FLAN-T5.

Table 2: Results on TREC-DL19 and TREC-DL20 by re-ranking top-100 passages retrieved by BM25. Sec/Q indicates the average time in seconds to the re-rank 100 passages for a query. Best performing unsupervised and overall system(s) are marked bold.

Method	LLM	Sec/Q	**DL19** nDCG@1/5/10	**DL20** nDCG@1/5/10
BM25	–	–	54.26 / 52.78 / 50.58	57.72 / 50.67 / 47.96
Supervised LLMs Methods				
monoT5	T5-Base	0.12	79.84 / 73.77 / 71.48	77.47 / 69.40 / 66.99
monoT5	T5-XL	1.30	79.07 / 73.74 / 71.83	80.25 / 72.32 / 68.89
Cohere Rerank	`english-v2.0`	–	77.13 / 76.17 / 73.22	79.32 / 71.00 / 67.08
Unsupervised LLMs Methods				
RankGPT	`gpt-3.5-turbo`	–	82.17 / 71.15 / 65.80	79.32 / 66.76 / 62.91
RankGPT	`gpt-4`	–	82.56 / 79.16 / 75.59	78.40 / 74.11 / 70.56
Relevance Generation	FLAN-T5-Base	0.12	58.13 / 48.52 / 47.43	55.25 / 50.35 / 48.32
PRP (Allpair)	FLAN-T5-Base	21.51	51.16 / 53.44 / 51.45	53.40 / 48.61 / 48.36
Instruction Distillation	FLAN-T5-Base	0.12	59.69 / 60.21 / 57.30	63.27 / 55.50 / 53.09
Relevance Generation	FLAN-T5-Large	1.10	43.41 / 47.65 / 48.41	40.43 / 45.19 / 46.67
PRP (Allpair)	FLAN-T5-Large	49.19	74.03 / 69.00 / 66.58	68.21 / 64.63 / 61.51
Instruction Distillation	FLAN-T5-Large	1.10	74.33 / 74.18 / 69.81	72.84 / 65.59 / 62.80
Relevance Generation	FLAN-T5-XL	1.30	50.00 / 54.33 / 52.85	45.37 / 48.56 / 49.07
PRP (Allpair)	FLAN-T5-XL	112.12	77.91 / 73.46 / 70.58	76.85 / 69.58 / 67.21
Instruction Distillation	FLAN-T5-XL	1.30	79.85 / 75.15 / 71.92	81.17 / 72.08 / 69.29

- **PRP** (Qin et al., 2023) is a pairwise ranking ranking method based on FLAN-T5.
- **MonoT5** (Sachan et al., 2022b) is pointwise ranking method based on T5 models and is supervised trained on MS MARCO.
- **Cohere Rerank** is a commercial text ranking system developed by Cohere[1].

4.3 Implementation Details

Passage Re-Ranking Task. Following Sun et al. (2023), we sample 10K queries from the MS MARCO training set. Utilizing the BM25 as the candidate generator, we retrieve 10 passages for each query. Our BM25 implementation is derived from BM25Okapi as presented in RankBM25 (Trotman et al., 2014). Prior to retrieval, we ensure that stopwords are eliminated. In implementing the pairwise prompting strategy, each query's 10 passages are juxtaposed in pairs, leading to a generation of 90 ordered passage pairs. The teacher models are instructed to determine which document is more relevance to the query, and subsequently produce the ranking results. The results oare then serve as the pseudo labels for pointwise instruction distillation. To harness the full potential of the ranking outcomes, we employ RankNet (Burges et al., 2005).

Conversational Recommendation Task. For this task, we use the dialogue history as the query, the descriptions of movies as documents, and use BM25 to fetch the top-5 movies into the candidate pool. Furthermore, following Hou et al. (2023), an additional 4 popular movies are incorporated into the candidate pool[2]. This is done to simulate the inherent feature of popularity bias in recommendations (Chen et al., 2023).

Training Details. Throughout the training phase, we employ the AdamW optimizer with a consistent learning rate of $3e-5$. We constrain the maximum input length to 512 tokens. The

[1] `https://cohere.com/rerank`

[2] The criterion for determining a movie's popularity is based on its frequency of mentions throughout the training dataset. Movies cited more than 200 times are classified as popular. The likelihood of selecting a popular movie is proportional to its representation in the overall popularity.

Table 3: Results (nDCG@10) on BEIR.

Method	LLM	BEIR
BM25	–	43.42
monoT5	T5-Base	49.07
monoT5	T5-XL	51.36
Cohere Rerank	`english-v2.0`	49.45
RankGPT	`gpt-3.5-turbo`	49.37
RankGPT	`gpt-4`	53.68
Instruction Distillation	FLAN-T5-XL	51.15
Instruction Distillation	FLAN-T5-Large	48.32
Instruction Distillation	FLAN-T5-Base	36.41

training environment is 4 * A800-80G, with a batch size fixed at 32. We train the model up to 3 epochs. Our experiments are based on the FLAN-T5 family (Chung et al., 2022), a suite of models which has been fine-tuned for various NLP tasks. Our experiments specifically leverage models such as FLAN-T5-XL (3B), FLAN-T5-Large (770M), and FLAN-T5-Base (220M).

The prompts used can be seen in Appendix A.

5 Experimental Results

5.1 Results on Passage Re-Ranking Tasks

The experimental results on TREC and BEIR datasets are presented in Table 2 and Table 3 respectively. Based on these results, we draw the following observations:

Firstly, when compared with previous unsupervised LLM prompting strategies, our instruction-distilled models' inference speed aligns with that of the *Relevance Generation* method, and it is notably over 100$\times$ faster than the PRP method. Moreover, the performance of our approach using FLAN-T5-XL and FLAN-T5-Large surpasses both the Relevance Generation and PRP methods with the same LLMs.

Secondly, the instruction-distilled models yield results akin to their supervised counterparts but with reduced annotation requirements. Specifically, our instruction-distilled FLAN-T5-XL model achieves nDCG@10 of 71.92 and 69.29 on TREC-DL19 and TREC-DL20, respectively, either matches or surpasses the performance of the supervised monoT5 of equivalent parameter size.

Lastly, the instruction-distilled models always perform superior to their teachers. For example, the distilled models of all different model sizes perform better than their PRP teachers. This can be attributed to the fact that unspecialized teacher models might produce unstable outputs. After distillation on task-related data, student models are able to strictly follow the given instructions, generating more reliable outputs. This specialization phase significantly enhances both the efficiency and performance of all involved models.

Similar findings can be observed on the BEIR dataset.

5.2 Results on Conversational Recommendation Tasks

Understanding user preferences from dialogue history presents a greater challenge than merely ranking relevance based on a specified query. Despite this, our method demonstrates noteworthy results, which are summarized in Table 4.

Firstly, our method achieves the best results among all the unsupervised methods. Specifically, our distillation technique outperforms other methods across all scales in terms of Acc@1 metrics. The FLAN-T5-XL distilled model achieves a peak value of 24.93% on Acc@1, outperforming all other unsupervised models.

Table 4: Results (Acc) on REDIAL.

Method	LLM	Sec/Q	Acc
Random	–	–	10.77
Popularity	–	–	7.69
BM25	–	–	8.62
Unsupervised LLMs Methods			
Listwise Ranking	T5-XL	0.02	16.92
Pairwise Ranking	T5-XL	7.90	20.00
Pointwise Ranking	T5-XL	1.44	12.00
Instruction Distillation	T5-XL	1.44	24.93
Listwise Ranking	T5-Large	0.01	13.85
Pairwise Ranking	T5-Large	3.06	16.62
Pointwise Ranking	T5-Large	0.49	8.00
Instruction Distillation	T5-Large	0.49	19.71
Listwise Ranking	T5-Base	0.01	1.54
Pairwise Ranking	T5-Base	1.00	11.69
Pointwise Ranking	T5-Base	0.18	10.77
Instruction Distillation	T5-Base	0.18	15.07

Secondly, when compared with the teacher model, the student model exhibits either comparable or superior performance. The teacher model, employing FLAN-T5-XL with PRP techniques, posts an Acc@1 of 20%. In contrast, the distilled model with equivalent parameter size achieves an impressive 24.93% in terms of Acc@1. Meanwhile, the Large model, with less than a third of the teacher model's parameters, records a close Acc@1 score of 19.71%.

Lastly, there is a notable improvement in the performance metrics of all the distilled models after instruction distillation. For instance, the FLAN-T5-XL model, when used with the pointwise prompt, only marginally surpasses the random recommendation. However, after the proposed instruction distillation process, its Acc@1 nearly doubles. A similar improvement is observed for FLAN-T5-Large, with its Acc@1 soaring from 8% to 19.71%. Even though the increase might not seem substantial due to the model's capacity, it represents a growth of over 5%.

5.3 Analytical Experiments

To gain deeper insights into the impact of model size and training signal, we carried out an analytical experiment. The results are depicted in Figure 2. Several key observations can be made from these results: (1) Instruction distillation models, represented by the yellow line in the figure, outperform the state-of-the-art supervised system, monoT5 (or SFT (500K), illustrated by the blue line), when the model size surpasses 3B. Moreover, our approach consistently exceeds the performance of earlier zero-shot LLM methods, namely RG and PRP, across all scales. (2) Distilling from larger models can enhance the performance of their smaller counterparts. As evidenced by our results labeled "Ours (XL)" in Figure 2 – which captures the process of distilling the predictions from FLAN-T5-XL to smaller models – it becomes clear that instruction distillation from larger models invariably boosts the capabilities of smaller ones. (3) Given the same training data size, our approach, which distilling from FLAN-T5-XL (referred to as "Ours (XL)" in Figure 2) and is unsupervised, significantly outperforms its supervised counterpart (referred to as "SFT (10k)" in Figure 2). This finding shows the promising potential of leveraging LLMs as data labelers in ranking tasks.

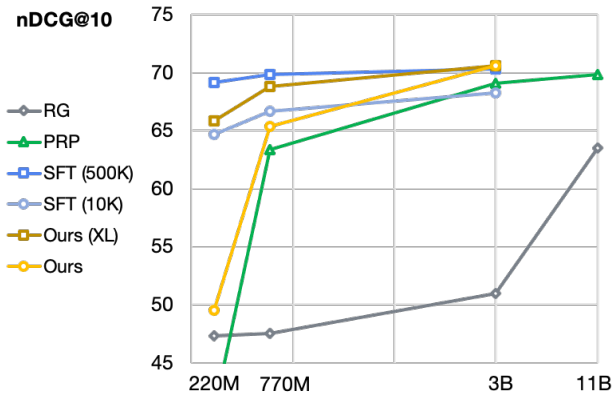

Figure 2: Compare the proposed method with baselines in terms of model size. We can see that our methods (denoted by yellow line) outperform supervised finetuning (SFT) methods when the number of parameters exceeds 3B.

6 Conclusion

This paper proposes instruction distillation, an unsupervised method that distills LLMs' ability uncovered by complex instructions into the same model but with simpler instructions. This method significantly improves LLMs' efficiency and stability which is very friendly for industrial application deployment. Our experimental results on passage ranking and conversational recommendation verify the effectiveness of the proposed method. With our method, the efficiency of the models is improved significantly. A $10 - 100\times$ increase in efficiency can be observed when compared to a comparable unsupervised method.

References

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *ICML*.

Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 41(3):1–39.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2020 deep learning track. *ArXiv*, abs/2102.07662.

Zhuyun Dai, Vincent Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. In *ICLR*.

Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, and Jiafeng Guo. 2021. Pre-training methods in information retrieval. *ArXiv*, abs/2111.13853.

Yao Fu, Hao-Chun Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *ArXiv*, abs/2301.12726.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *ArXiv*, abs/2212.10496.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *ArXiv*, abs/2305.08845.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018a. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)*.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Christopher Joseph Pal. 2018b. Towards deep conversational recommendations. *ArXiv*, abs/1812.07617.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher R'e, Diana Acosta-Navas, Drew A. Hudson, E. Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan S. Kim, Neel Guha, Niladri S. Chatterji, O. Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas F. Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models. *ArXiv*, abs/2211.09110.

Xueguang Ma, Xinyu Crystina Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *ArXiv*, abs/2305.02156.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *ArXiv*, abs/2212.08410.

Microsoft. 2023. Confirmed: the new bing runs on openai's gpt-4. `https://blogs.bing.com/search/march_2023/Confirmed-the-new-Bing-runs-on-OpenAI%E2%80%99s-GPT-4`.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *ArXiv*, abs/2202.08904.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of EMNLP*.

OpenAI. 2022. Introducing chatgpt. `https://openai.com/blog/chatgpt`.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large language models are effective text rankers with pairwise ranking prompting. *ArXiv*, abs/2306.17563.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joëlle Pineau, and Luke Zettlemoyer. 2022a. Improving passage retrieval with zero-shot question generation. In *EMNLP*.

Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joëlle Pineau, and Manzil Zaheer. 2022b. Questions are all you need to train a dense passage retriever. *ArXiv*, abs/2206.10658.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *ArXiv*, abs/2301.12652.

Charles Burton Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *ArXiv*, abs/2209.15189.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP 2023*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Nandan Thakur, Nils Reimers, Andreas Ruckl'e, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS*, volume abs/2104.08663.

Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, pages 58–65.

Liang Wang, Nan Yang, and Furu Wei. 2023a. Query2doc: Query expansion with large language models. *ArXiv*, abs/2303.07678.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language model with self generated instructions. In *ACL*.

Likang Wu, Zhilan Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A survey on large language models for recommendation. *ArXiv*, abs/2305.19860.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *ICLR*.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji rong Wen. 2023. Large language models for information retrieval: A survey.

A Prompts

A.1 Passage Ranking

Pointwise Ranking Prompt

Question: Given a query "`{{query}}`", Is the following passage relevant to the query?

Passage : `{{passage}}`

If it is relevant answer Yes, else answer No.

Answer:

Pairwise Ranking Prompt

Question: Given a query "`{{query}}`", which of the following two passages is more relevant to the query?

passage A: `{{passage_A}}`

passage B: `{{passage_B}}`

Output the identifier of the more relevant passage. The answer must be passage A or passage B.

Answer:

A.2 Conversational Recommendation

Pointwise Ranking Prompt

Question: Given the conversation history between the recommender and the user:

`{{query}}`

Based on the user's preference, is the following movie suitable to the user?

Movie: `{{movie}}`

The answer must be Y or N. Give the answer after Answer: .

Pairwise Ranking Prompt

Question: Given the conversation history between the recommender and the user:

`{{query}}`

Based on the user's preference, which of the following two movies is more suitable to the user?

Movie A: `{{movie_A}}`

Movie B: `{{movie_B}}`

The answer must be A or B. Give the answer after the Answer: .

Listwise Ranking Prompt

Question: Given the conversation history between the recommender and the user:

`{{query}}`

Based on the user's preference, which of the following movies is the most suitable for the user?

[1]: `{{movie_1}}`

[2]: `{{movie_2}}`

...

Answer the question with the number of the movie. The answer will include one and only one number. Give the answer after Answer: .