FOLLOWIR: Evaluating and Teaching Information Retrieval Models to Follow Instructions

Orion Weller [ι] **Benjamin Chang** [ι] **Sean MacAvaney** [λ] **Kyle Lo** [α]
Arman Cohan [γ α] **Benjamin Van Durme** [ι] **Dawn Lawrie** [ι] **Luca Soldaini** [α]

[ι] Johns Hopkins University [α] Allen Institute for AI
[λ] University of Glasgow [γ] Yale University
oweller@cs.jhu.edu

Abstract

Modern Language Models (LMs) are capable of following long and complex instructions that enable a large and diverse set of user requests. While Information Retrieval (IR) models use these LMs as the backbone of their architectures, virtually none of them allow users to provide detailed instructions alongside queries, thus limiting their ability to satisfy complex information needs. In this work, we study the use of instructions in IR systems. We build FOLLOWIR, a rigorous instruction evaluation benchmark for following real-world instructions in IR. FOLLOWIR repurposes detailed instructions—also known as *narratives*—developed for professional assessors to evaluate retrieval systems. In particular, we build our benchmark from three collections curated for shared tasks at the Text REtrieval Conference (TREC). Through this process, we can measure how well IR models follow instructions, through a new pairwise evaluation framework. Our results indicate that existing retrieval models fail to correctly use instructions, using them for basic keywords and struggling to understand long-form information. However, we show that it is possible for IR models to learn to follow complex instructions: our new FOLLOWIR-7B model has significant improvements after fine-tuning on our training set.[1]

1 Introduction

Modern language models (LMs) are extensively tuned to be able to follow user instructions faithfully (Chung et al., 2022; Ouyang et al., 2022a; Rafailov et al., 2023; Wang et al., 2023b; Ivison et al., 2023) and safely (Bai et al., 2022; Bianchi et al., 2024). Through these capabilities, LMs are able to successfully tackle a broad range of tasks (Chiang et al., 2024; Liang et al., 2023; Yang et al., 2023; Jimenez et al., 2024; Zeng et al., 2023), even when not explicitly fine-tuned for them.

In contrast to the broader LM community, information retrieval (IR) practitioners and researchers have yet to fully exploit instruction-tuned models. Thanks to their ability to effectively estimate semantic similarity between query and documents, LMs have been adopted as the main backbone of neural retrieval architectures (Karpukhin et al., 2020; Khattab and Zaharia, 2020; Reimers and Gurevych, 2019). However, the vast majority of these systems are fine-tuned to operate exclusively as text spans similarity estimators (Khattab and Zaharia, 2020; Izacard et al., 2021; Nogueira and Cho, 2019; Pradeep et al., 2023; Ma et al., 2023). Moving past these *ad-hoc* search systems to retrieve *with instructions* would enable support for complex information needs. For example, imagine a researcher seeking to identify papers that must contain numerous qualities to be relevant (from a given venue, using a given class of methods, etc.) while also making sure to avoid conditions that would make it not-relevant (using negative sentiment, using datasets from certain domains, etc.).

Recent work has started to move towards search with instructions, but this topic is still understudied with only a handful of papers (Su et al., 2022; Asai et al., 2022; Muennighoff et al., 2024). In particular, we find their use of instructions to be narrow: instructions are typically short (fewer than 10 words) and repetitive (only one instruction per dataset *e.g.*, Su et al. (2022); Asai et al. (2022); Li and Li (2023); Xiao et al. (2023)). Further, these works lack evaluation datasets that explicitly measure instruction following—instead focusing on standard *ad-hoc* retrieval benchmarks.

To address these gaps we introduce FOLLOWIR, which consists of (1) a benchmark that explicitly measures the instruction following ability of retrieval models, and (2) training data that includes diverse and realistic instructions. Our key intuition is to leverage instructions developed for *professional annotators* of IR systems in order to study

[1] Links to the code, data, and models are available at https://github.com/orionw/FollowIR

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 11926–11942

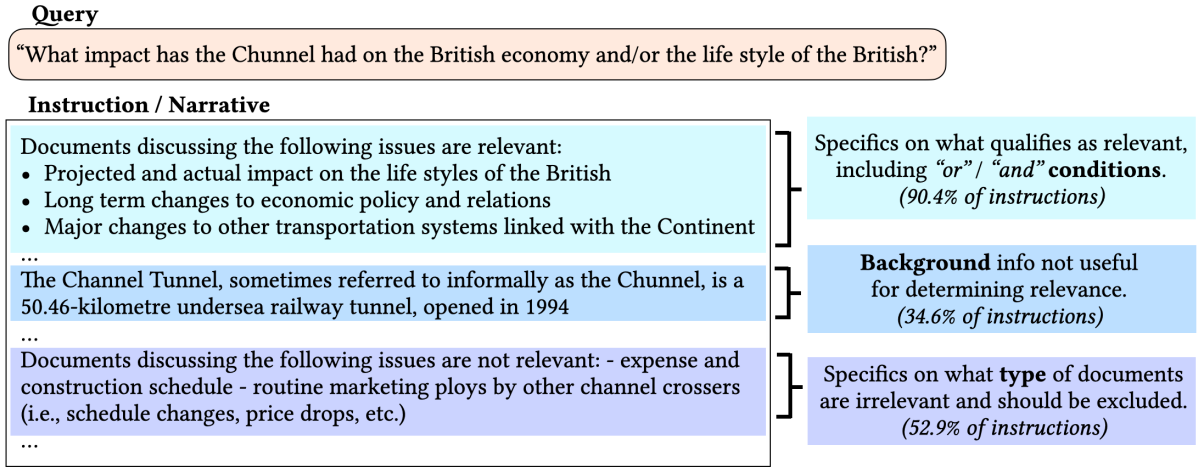

Figure 1: How do standard retrieval queries differ from instructions (or *narratives*)? Instructions contain more specific details about what is relevant, include less directly-relevant background info, and often have directives about what documents are *not* relevant, using negation. %'s are how often features appear in the TREC instructions.

the capabilities of instruction-following IR models. These instructions are used by annotators to judge document relevance for a given query. Fortunately, the IR field is rich with such data, as these instructions—also known as *narratives*—are created for all queries in any well-constructed IR dataset. In particular, we use narratives developed for shared tasks at the Text REtrieval Conference. These instructions are thorough and complex, including minute details about what makes a document relevant vs not-relevant. Thus if annotators can use these TREC instructions to annotate document relevance, so should instruction-following retrieval models (example query and instruction pairs are shown in Figures 1 and 2).

We use three deeply-judged TREC collections as the basis of our evaluation set: TREC Robust 2004 (Voorhees, 2005), TREC Common Core 2017 (Allan et al., 2017), and TREC News 2021 (Soboroff et al., 2020). These collections have been thoroughly annotated in order to evaluate recall in retrieval, with hundreds to thousands of documents judged as relevant or not-relevant. We take the instructions given to the professional annotators and alter them slightly, manually re-annotating the relevant documents. We then have paired instructions, which can be used to test how models react to changed instructions: we measure if models update their relevant docs to match the altered instructions.

As there are no existing methods to compare pairwise queries in IR, we develop a new evaluation framework to do so, measuring rank-wise score changes (which we call p-MRR) of documents given a pair of different instructions with the same query. Results on FOLLOWIR indicate that current models generally fail to follow instructions in retrieval unless they are 3B+ parameters or have not been trained for retrieval. Our analysis shows that these failures are due to two phenomena: (1) models are not used to long instructions, and (2) models use the instructions to do keyword search.

To further progress in building retrieval models that can understand instructions, we build a training set of real-world human-used instructions and fine-tune a model on them (FOLLOWIR-7B). Our results show marked improvement on FOLLOWIR for both standard IR metrics and for p-MRR, indicating a starting point for future progress.

In summary, we contribute the following: (1) a benchmark for evaluating instruction following in retrieval (FOLLOWIR) consisting of human annotations on top of three already highly-judged corpora, (2) analysis of why current models fail to understand instructions, and (3) training data for teaching retrieval models to follow instructions along with a new open-sourced IR model, FOLLOWIR-7B, that can handle long instructions in IR.

2 Related Work

TREC Conferences The United States National Institute of Science and Technology (NIST) created the TREC organization in 1993. Each year TREC sponsors many *tracks*, or shared tasks, on a given dataset. These tracks range from a variety of topics: anywhere from standard ad-hoc retrieval on news (Soboroff et al., 2018; Soboroff, 2021) to more complex domains such as legal retrieval (Oard et al.,

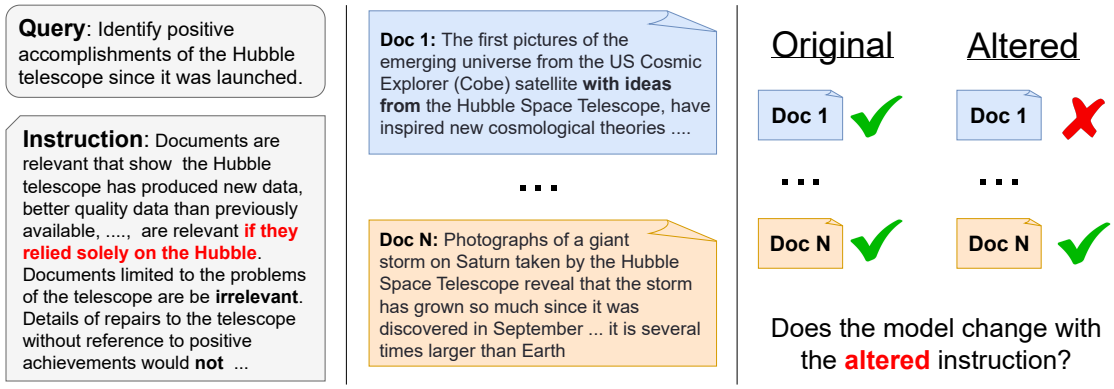

Figure 2: A visual depiction of the pairwise evaluation framework: models are evaluated on the query with the original instruction, and then on the query with the altered instruction. If the model correctly understands the instructions, it will change which documents are relevant w.r.t. the alteration (right). Note that the real-world instructions (left) given to TREC annotators includes fine-grained details about the relevance as well as negation.

2008), or retrieval-augmented generation/report-generation (Lawrie et al., 2024).

As part of this process, NIST sponsors annotations for these collections. Typically, this is done by *pooling* a set of results (*runs*) from a variety of retrieval models and then annotating them in rank order until funding runs out. To help facilitate annotation, track organizers provide a *narrative* (or instruction) for each query that will be given to the annotators—however, IR models are only ever given the query. As evaluating total recall would require annotating every document in the collection for every query (which is not feasible for collections with millions of documents), recall error is tested using post-hoc sampling and annotation. Although not every query and document pair can be evaluated, recall for queries is very high. We build off the rigorous evaluation done at TREC by starting with several of their collections.

Instructions for LMs Instruction-following LMs have been popularized by models such as InstructGPT (Ouyang et al., 2022a), FLAN (Wei et al., 2022), and T0 (Sanh et al., 2022). They have become a large area of interest for the natural language processing community (Touvron et al., 2023a; Jiang et al., 2023; Groeneveld et al., 2024; Black et al., 2022). There has been much work in evaluating if they can generalize to new instructions (Wang et al., 2022c; Ouyang et al., 2022b), if we can train them to follow instructions without human-annotated data (Wang et al., 2022b; Qin et al., 2023), and applying them to various domains (Zhao et al., 2021; Singhal et al., 2023; Shaghaghian et al., 2020). As the IR community uses LMs in their pipelines, we seek to broaden the scope of IR to include instructions, aligning it with the broader NLP community.

Instructions for Retrieval Using instructions in retrieval is a nascent area of exploration. Su et al. (2022) and Asai et al. (2022) were two of the earliest works that trained a retrieval model to use instructions along with the query. However, these instructions are typically very short, such as "Retrieve a Wikipedia paragraph that answers this question." Recent work incorporates instructions in smaller models (Xiao et al., 2023; Chen et al., 2023, 2024) as well as others which use Llama (Touvron et al., 2023a; Weller et al., 2023) or Mistral (Jiang et al., 2023) as the backbone of a larger retrieval model that can use instructions: GritLM (Muennighoff et al., 2024) trains Mistral to do both generation and embedding, while Wang et al. (2023a) uses Mistral for embeddings only.

Despite this flurry of activity, these efforts do not have an explicit instruction-related retrieval benchmark to evaluate on. Instead, they evaluate on standard retrieval benchmark suites such as MTEB (Muennighoff et al., 2022) and BEIR (Thakur et al., 2021) which do not contain instructions. Thus, these newer instruction retrieval models hand-write a few instructions, where typically each instruction is applied to an entire dataset, irrespective of the query. This makes these instructions generic: focused only on the task format, format of the "document" (paragraph, sentence, etc.), and the broad domain. Note that because of this, no current instructions contain any extra background information or negation (Weller et al., 2024) which are commonly found in real-world instructions (see Figure 1 for an example of these differences).

In work concurrent to ours, Oh et al. (2024) also propose a dataset to evaluate instructions in retrieval models. Their dataset uses the MS MARCO collection (Nguyen et al., 2016), and differs in sev-

eral crucial aspects: it only has one relevant document per query (*e.g.*, sparsely judged), is GPT-4 generated and validated, focuses on the background of the user ("I am a school teaching looking for ..."), and evaluates using the lowest score over N instructions for the same query (measuring robustness). In contrast, we use highly-judged corpora to ensure we can measure recall, use professionally generated instructions, have human-validated relevance judgements, propose a new paired evaluation protocol, and provide a training dataset and model for teaching instruction-following.

3 Building FOLLOWIR

We derive FOLLOWIR from three TREC collections: TREC News 2021 (derived from the Washington Post v4 corpus; Soboroff et al., 2020), TREC Robust 2004 (from news articles in Disks 4 and 5 collections; Voorhees, 2005), and TREC Common Core 2017 (from the New York Times Annotated corpus; Allan et al., 2017). Each of these was professionally assessed to include hundreds of annotations per query (see Table 1), with 50-180 relevant documents per query on average (and many more not-relevant annotations).

Each of these TREC tracks includes instructions for the professional annotators that we now also give to the models. Although using these alone can provide some indication of how well models can follow instructions, it doesn't explicitly test their instruction following ability. To more carefully isolate this in our benchmark, we test whether models can respond to small changes in the instruction.

To accomplish this, we ask two expert annotators to modify the TREC instructions. However, doing this in a naive way would require re-annotating all the document judgements, a non-trivial task requiring immense annotation efforts.[2] Instead, we task the annotators with making instructions *more specific* by including additional constraints that narrow the relevance definition. These transformations cause some previously relevant documents to become non-relevant without introducing any new relevant documents from the pool. Therefore, only those documents that were deemed relevant by the original TREC assessors need to be re-annotated. This makes the annotation tractable, with only dozens of documents to re-annotate per query instead of a collection of thousands.

We annotate a subset of the original TREC queries due to cost and overlap: we sample 50 queries from TREC Robust 2004 that do not overlap with TREC Common Core (as Common Core used 50 queries from Robust04 on a new collection), and 30 queries from TREC News 2021. Table 1 shows dataset statistics of judged documents and the final benchmark size. Annotators were asked to change the instructions so that the number of relevant documents was cut roughly in half, thus including a sizeable number of changed relevance judgements. We note that although the number of queries seems small by NLP standards, 30-50 queries is both effective (Webber et al., 2008) and standard in IR due to the expense of careful annotation over many documents per query (see §A).

Due to differences in retriever quality, if we evaluate by searching over the full collection, each model will retrieve a different number of relevant documents. However, because we evaluate instruction following based on changing the document relevance, models that do poorly in the initial retrieval will have fewer documents which change relevance in the instruction-following evaluation. To rectify this, we instead turn to a reranking task where we include all relevant documents, and use a pool of five models[3] to select the top non-relevant documents. To be able to freely distribute the data due to fair-use laws, we chunk the documents into 400-word passages with 200-word overlap and select the highest scoring passages using MaxP (Dai and Callan, 2019). This enables us to distribute our data, which we do by extending the MTEB evaluation framework (Muennighoff et al., 2022).

3.1 Evaluation Metrics for FOLLOWIR

Our benchmark provides two ways of measuring instruction following: (1) standard retrieval metrics when using the instructions with the queries and (2) pairwise evaluation of instruction following.

For (1), we use typical IR evaluation metrics but use the instruction along with the query: these metrics are mean average precision (MAP) for Core17/Robust04 and normalized discounted cumulative gain at 5 (nDCG@5) for News21.[4] For (2) we use our novel pairwise evaluation metric

[2] NIST's budget is $1–2 million USD/year: trec.nist.gov/pubs/2010.economic.impact.pdf

[3] We use BM25, BGE-base, E5-base-v2, TART-Contriever, and INSTRUCTOR-xl.

[4] We use nDCG@5 for News21 as that was the official metric used in that TREC track.

Dataset	# Q	$\lvert Q\rvert$	$\lvert I\rvert$	Rel. D/Q	# Q	$\lvert I\rvert$	Rel. D/Q
TREC News '21 (Soboroff et al., 2020)	50	15.3	40.1	50.1	32	46.9	19.2
TREC Core '17 (Allan et al., 2017)	50	16.6	44.0	180.0	20	53.5	32.7
TREC Robust '04 (Voorhees, 2005)	249	11.9	68.2	69.9	52	75.2	19.8

Table 1: FOLLOWIR evaluation set statistics before (left) and after (right) annotation. We use a subset of the queries in three popular TREC tracks for variety in queries and documents. $|Q|$ is the word length of the queries and $|I|$ is the word length of the instructions. Rel. D/Q indicates the number of relevant annotated documents in the collection, excluding irrelevant annotations. As designed, there are less relevantly-judged documents in the FOLLOWIR portion (as the annotations change the relevance of documents on purpose for evaluation).

that measures the delta in scores when following the modified instructions instead of the original.[5]

Our new pairwise evaluation metric, p-MRR, measures rank-wise changes between queries. In developing this metric we had the following desiderata: it should compare the results of the original instruction to those of the new instruction, it should have a standardized range from worst possible change in instruction-following score (*i.e.*, -1) to best possible instruction-following score (*i.e.*, 1) with an option for no change when using different instructions (*i.e.*, 0), and finally should take into account the document rank so that changes from rank 1 to rank 2 are more prominent than changes from rank 99 to 100. Given the above qualifications, we use the following equation applied to each *changed* relevance document per query (where RR is reciprocal rank, R_{og} is the rank of the doc when using the original instruction and R_{new} is the new rank):

$$p\text{-MRR} = \begin{cases} \frac{RR_{og}}{RR_{new}} - 1 & \text{if } R_{og} > R_{new} \\ 1 - \frac{RR_{new}}{RR_{og}} & \text{otherwise} \end{cases} \quad (1)$$

For the final score, we average first within a given query and then over all queries in the corpora—*i.e.*, macro-averaging across queries, to handle the different number of relevant documents per query.

3.2 Why do we need a new metric?

The original TREC datasets do not need instructions to be correctly solved. The nDCG metric only shows their ability to retrieve on the original task (with the added instruction). However, models could simply ignore the instruction and get high nDCG scores, thus, they don't evaluate instruction-following. Hence why we propose a new metric disentangled from model's keyword-search ability.

[5]Note that we do not show standard retrieval results on the modified instruction's relevant document set, as standard retrieval scores cannot be directly compared across different query relevance annotations (*qrels*).

4 Evaluating Instruction Following

In this section we describe the models we evaluate, their results on FOLLOWIR, and ablations performed to better understand current models.

4.1 Evaluation Settings

We evaluate a wide variety of IR models (trained with and without instructions), including neural models ranging from 100 million to 7 billion parameters. We evaluate on the original TREC instructions in the FOLLOWIR benchmark and then on the new instructions, showing both standard IR metrics and the new pairwise metric p-MRR. We group models into four categories:

No Instructions in Training These retrieval models did not see instructions in training and typically aren't given them: Contriever (Izacard et al., 2021), E5 (Wang et al., 2022a), MonoBERT (Nogueira et al., 2019), MonoT5 (Nogueira et al., 2020), and BM25 (Robertson et al., 1995).

Instructions in IR Training Most retrieval models using instructions received roughly one instruction per retrieval dataset, which generally defined the domain (*e.g.*, "Financial"), document size (sentence, passage, etc.), and task format. This includes INSTRUCTOR models (Su et al., 2022), the bi-encoder TART model trained from Contriever (Asai et al., 2022), the reranker TART trained from FLAN-T5 (Chung et al., 2022), E5 Mistral-Instruct (Wang et al., 2023a), and GritLM (Muennighoff et al., 2024). We also include BGE models (Xiao et al., 2023) in this category, although they are trained with only one instruction total for each broad task (retrieval, clustering, etc.).

API Models We use three of the best performing API embedding models: Cohere's v3 English, Google's Gecko (Lee et al., 2024) and OpenAI's Text-Embedding-v3-Large. It is mostly unknown what these models' training procedures were—including if they were trained on instructions or not—thus we place them in a distinct category.

	Model	Robust04 MAP	Robust04 p-MRR	News21 nDCG	News21 p-MRR	Core17 MAP	Core17 p-MRR	Average Score	Average p-MRR
No-Instruction IR	E5-base-v2	13.4	-6.7	20.9	-2.0	14.0	-2.9	16.1	-3.9
	Contriever	19.7	-6.1	22.9	-2.8	15.3	-2.5	19.3	-3.8
	MonoBERT	21.0	-9.4	25.1	-0.8	18.4	-0.2	21.5	-3.5
	BM25	12.1	-3.1	19.3	-2.1	8.1	-1.1	13.2	-2.1
	MonoT5-base	15.7	-6.2	11.0	+5.0	12.2	-4.1	13.0	-1.8
	E5-large-v2	17.4	-4.2	24.3	+0.9	17.0	+0.1	19.6	-1.1
	MonoT5-3B	27.3	+4.0	16.5	+1.8	18.2	+1.8	20.7	+2.5
Instruction-IR	TART-Contriever	14.3	-9.0	21.8	-3.0	13.3	-3.0	16.5	-5.0
	INSTRUCTOR-base	17.2	-10.4	22.1	-1.8	15.5	-1.1	18.3	-4.4
	E5-mistral	23.1	-9.6	27.8	-0.9	18.3	+0.1	23.1	-3.5
	BGE-base	16.8	-6.5	20.0	-0.1	14.6	-2.7	17.1	-3.1
	INSTRUCTOR-xl	19.7	-8.1	26.1	-0.9	16.8	+0.7	20.9	-2.8
	BGE-large	17.5	-7.8	22.3	+0.6	15.0	+0.1	18.3	-2.4
	GritLM-7B	28.6	-1.7	24.4	-1.0	20.8	+2.6	24.6	-0.0
	TART-FLAN-T5-xl	24.6	-0.7	12.8	+2.0	17.0	+2.8	18.1	+1.4
APIs	OpenAI v3 Large	27.2	-5.8	27.2	-2.0	21.6	-0.2	25.3	-2.7
	Cohere v3 English	22.3	-3.6	28.3	+0.2	20.6	+2.8	23.7	-0.2
	Google Gecko	23.3	-2.4	29.5	+3.9	23.2	+5.4	25.3	+2.3
Instruct LMs	FLAN-T5-base	6.4	+5.3	6.1	-0.1	6.5	-3.3	6.3	+0.6
	Llama-2-7B-chat	6.3	+2.0	1.7	+0.2	5.4	+2.8	4.5	+1.7
	FLAN-T5-large	14.7	+3.9	8.0	+8.9	11.4	+1.3	11.4	+4.7
	GritLM-Reranker	9.7	+6.1	10.2	+3.4	9.8	+8.6	9.9	+6.0
	Mistral-7B-instruct	23.2	+12.6	27.2	+4.8	19.7	+13.0	23.4	+10.1
	FollowIR-7B	24.8	+13.7	29.6	+6.3	20.0	+16.5	24.8	+12.2

Table 2: Evaluating instruction-following on FOLLOWIR. Introduced in this work, p-MRR is a pairwise evaluation metric measuring instruction following when instructions change, ranging from -100 to 100 (higher is better). Generally only models with over 3B parameters or instruction-tuned LMs that haven't been trained on retrieval tasks show success at following retrieval instruction.

However, we note that Google's model did explicitly train with instructions, as mentioned in their technical report.

Instruction-Tuned LMs We also evaluate several instruction-tuned LMs to be used as rerankers, including FLAN-T5 (Chung et al., 2022), Llama v2 (Touvron et al., 2023b), and Mistral-Instruct-v0.2 (Jiang et al., 2023). We evaluate these models in the same fashion as MonoT5 rerankers, comparing the true and false tokens. Note that these models were not trained on any retrieval-specific data.

4.2 FOLLOWIR Results

Table 2 shows the main results, with the standard IR score shown (either MAP or nDCG@5) as well as the pairwise evaluation metric, p-MRR.

No-Instruction IR Models We see that the no-instruction models range widely in standard IR metrics (in terms of nDCG@5 and MAP) but generally have negative scores for p-MRR (up to -3.9). The only non-instruction model to score positively on average is MonoT5-3B (+2.5 p-MRR).

Instruction IR Models We again see that these models have generally negative scores, with the exception being GritLM (with scores averaging roughly zero) and TART-FLAN-T5-xl which has slightly positive scores for two of the three datasets (with an average of +1.4 p-MRR).

API Models We see that the API models perform strongly in terms of standard IR metrics, with OpenAI's and Google's models performing the highest overall. However, Cohere's and OpenAI's models perform poorly at instruction-following with negative scores (-0.2 and -2.7 on average, respectively) whereas Google Gecko has positive scores (+2.3) likely from its dataset of instructions.

Instruct-Tuned LMs In contrast to the previous results, all instruction-tuned LMs show positive results for instruction following, although they have the widest range of performance using standard IR metrics (ranging from very poor scores to some of the higher scores). We see that the best performing model in this category is FOLLOWIR–7B, which we describe in more detail in Section 5.

Overall We see that the only models that show positive results at following instructions are either IR models with over 3B parameters or those that have been explicitly trained to follow instructions

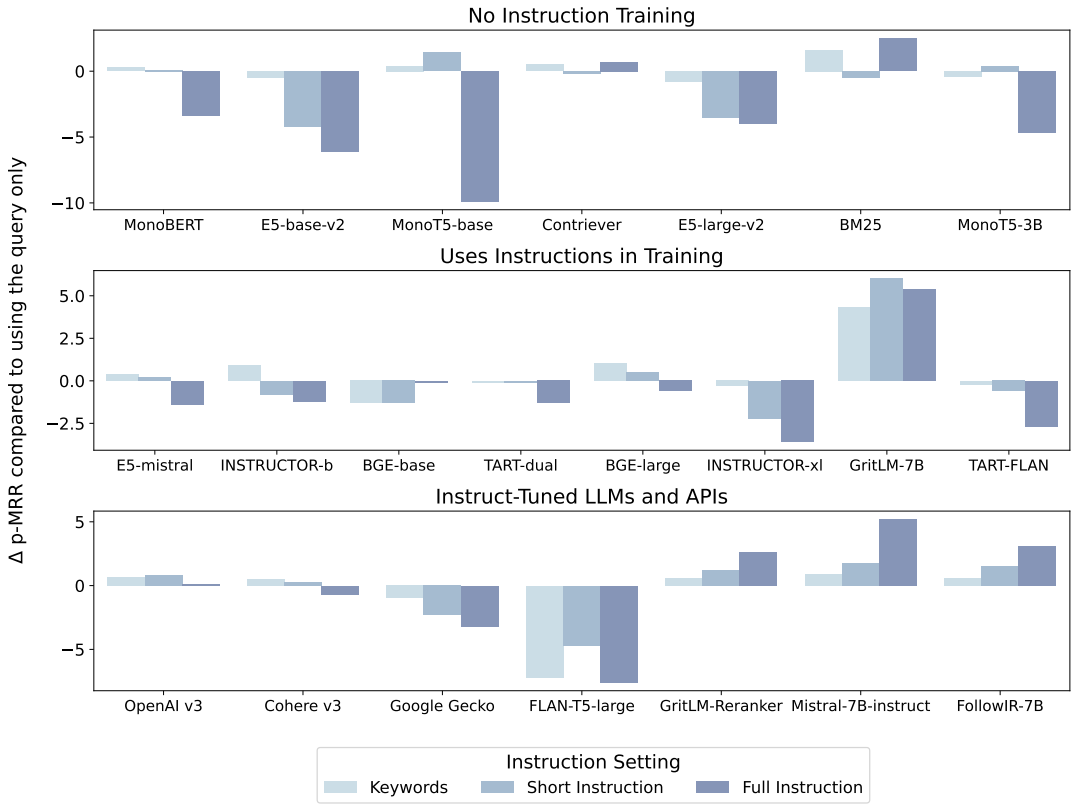

Figure 3: Score difference between using no instructions to using instructions formatted as keywords, short text, or the full text. While models that can correctly use instructions see gains with the additional information, most other models see decreasing performance as instruction length increases.

(*e.g.* FLAN-T5), without any retrieval-specific supervision. This aligns with work in the natural language processing community which has shown that the instruction-following ability improves with scale (Brown et al., 2020) and supervised instruction-tuning (Longpre et al., 2023).

4.3 Analysis

Why do so many models fail to correctly follow instructions when they do well on typical IR metrics such as nDCG and MAP? We answer this question by ablating several components that may impact results: (1) whether IR models are not used to text that cannot be used for simple keyword search (*i.e.* instructions) and (2) whether they are unused to the length of the longer instructions (as current retrievers have been trained on shorter input).

To test these, we compare the original query-only result to those where we additionally give the model either the full instruction, a shorter instruction, or keywords from the instruction. We gather these short instructions and keywords by prompting GPT-4-Turbo-1106 to generate them from the original full instruction (for TREC data) or otherwise use the original short instructions given by the authors of the model (for BEIR data). For the full prompt text, please see Appendix H.

We show results for these ablations in Table 3, where positive scores indicate that adding information improves the model while negative scores indicate a drop in performance. We see a consistent trend where models that did poorly on longer instructions perform better on keywords and shorter instructions than with the full instruction. However, models that are able to follow instructions generally see better results with the additional information.

These results show that models are (1) using the instruction text as keywords (as performance is higher when using only keywords) and (2) are not able to utilize the extra information in the instructions (as they generally decrease in performance with this additional information).

We also confirm that these results hold on datasets outside of TREC collections and show results on three BEIR datasets: SciFact, NFCorpus, and FiQA. We show in Table 3 the original score (using the short instructions from their papers) and the change in score when using just keywords from the instruction (again extracted from GPT-4). We show results only for models which performed poorly for instruction-following. We

	Model	SciFact OG	SciFact Δ w/Key.	NFCorpus OG	NFCorpus Δ w/Key.	FiQA OG	FiQA Δ w/Key.
No-Instruction	BM25	67.9	-1.7	32.2	-5.1	23.6	-1.6
	E5-base-v2	71.9	-2.7	35.4	-2.5	39.9	-0.4
	Contriever	64.9	+0.4	31.7	+0.0	24.5	-3.2
	MonoT5-base	73.1	-0.6	35.6	-0.9	41.2	-0.3
Uses Instruction	TART-Contriever	67.6	-0.3	33.4	-5.3	31.8	-0.4
	INSTRUCTOR-base	57.8	+1.0	31.6	-0.4	39.2	-0.1
	BGE-base	73.2	-0.5	35.5	+0.0	40.8	-2.3
	TART-FLAN-xl	74.2	+1.6	33.9	+0.4	39.6	-0.3
	INSTRUCTOR-xl	62.4	+0.2	36.0	-0.6	46.9	+0.8
	E5-Mistral	77.1	-5.1	38.8	+0.3	56.7	-6.5

Table 3: Ablation on BEIR benchmarks for models that do poorly with longer instructions, comparing their original short instructions vs domain keywords extracted from those instructions (see Appendix G for a list). OG stands for original input. If models had learned to use the instructions correctly we would see a divergence between the behavior of instruct and non-instruct models, however, we see comparable performance between the added keywords vs the full instruction (± one point).

Model	Robustness@10
BM25	26.9
TART-Contriever	47.5
RepLLaMa	52.6
E5-Mistral	55.4
Mistral-7B-instruct	35.3
FollowIR-7B	71.5

Table 4: Performance on the InstructIR benchmark using their "Robustness@10" scores, e.g. the min nDCG@10 score across 10 instructions. Upper portion is bi-encoders while lower is rerankers.

see that the scores for keywords vs the short instruction are generally similar, with most models seeing a change of around ± 1 point, except for the strongest of the non-instruction-following models, E5-Mistral, seeing a larger drop on some datasets.

Overall We find overall (on both TREC and BEIR datasets) that models use instructions for keyword matching and are unused to longer instructions that may contain less relevant words.

5 Teaching Instruction Following

Is it possible to improve model performance in following instructions? We show that fine-tuning on a training set of longer instructions can provide a method for doing so. We start by gathering a training set to teach models. We collect all TREC narratives (*i.e.*, instructions) from tasks not in FOLLOWIR, consisting of 1836 pairs of queries and narratives. However, we note that this does not provide any positive or negative documents for fine-tuning.

In order to obtain documents for training, we prompt GPT-3.5-Turbo-1106 to generate relevant and not-relevant documents, generating roughly two relevant and non-relevant instances per query. The prompts for this experiment can be found in Appendix H.

However, these synthetic documents are noisy and contains errors w.r.t. the labels—to remedy this, we perform a round of filtering and use the best performing open-source model from Table 2 (Mistral-7B-Instruct-v0.2) to score each of the generated documents according to the instruction. We then filter the documents according to whether Mistral correctly predicts the generated label, and finally balance the relevant and non-relevant samples, choosing only one relevant and non-relevant document per query. Our total is ∼1800 training instances on ∼1200 unique query/instructions pairs.

We then train our instruction-following model, FOLLOWIR-7B, by fine-tuning Mistral-7B-Instruct-v0.2 on our data using the Llama-Factory framework (Hiyouga, 2023) with LoRA (Hu et al., 2021). Full training hyperparameter details are found in Appendix C.

When we evaluate this model on FOLLOWIR (Table 2), we find that the scores consistently improve. Compared to the original Mistral-7B-Instruct-v0.2, our model improves on both standard IR metrics (+6.0% relative improvement) and on instruction following (+20.8% relative). We also show that this improvement holds on the concurrent InstructIR dataset (Table 4), where FollowIR-7B scores double the base Mistral-7B scores (71.5 Robustness@10 vs 35.3) and is the top scoring model overall. Thus, we can see that it is possible to train IR models to be better instruction followers.[6]

[6]See §B for a Llama-3-8B base version.

6 Conclusion

Despite the use of LMs as the backbone of neural retrieval models, most existing IR models do not take instructions that define document relevance. Further, there is no existing resource that measures how well retrieval models can follow instructions. We build a new benchmark that explicitly measures the instruction following ability of retrieval models and find that nearly all retrieval models do not follow instructions, with the exception of larger models (3B+ parameters) or instruction-tuned LMs that typically are not used for retrieval. However, we show that it is possible to improve their instruction following ability, and build and release a training corpus for teaching retrieval models to follow instructions. Our new model, FOLLOWIR-7B, shows improvement on both standard retrieval metrics as well as in instruction following, which we hope will inspire future work on the topic.

7 Limitations

Reranking vs Full Retrieval As our setup for evaluating instruction following requires evaluating the documents which changed relevance, we cannot use the full collection for retrieval (as each retriever finds different relevant documents by design). Further, due to licensing restrictions, we cannot distribute the full corpora from the TREC tracks—thus we distribute passages due to fair use laws. However, we show full corpus retrieval results for a subset of models in Appendix F and note similar trends in terms of the lack of instruction following.

Possible Errors Our work is built on the TREC document collections and judgements, as well as new annotation efforts. We do not check for potential errors in the TREC annotations, and our newly gathered annotations may have small errors. Despite these caveats, we see that our dataset still provides a useful evaluation setup for measuring instruction following.

8 Acknowledgments

OW is supported by a NSF GRFP fellowship.

References

AI@Meta. 2024. Llama 3 model card.

James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel, and Ellen M Voorhees. 2017. Trec 2017 common core track overview. In *TREC*.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2022. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. 2024. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *Preprint*, arXiv:2309.07875.

Sid Black, Stella Biderman, Eric Hallahan, Quentin G. Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Martin Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Benqi Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model. *ArXiv*, abs/2204.06745.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yinqiong Cai, Jiafeng Guo, Yixing Fan, Qingyao Ai, Ruqing Zhang, and Xueqi Cheng. 2022. Hard negatives or false negatives: Correcting pooling bias in training neural ranking models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 118–127.

Haonan Chen, Zhicheng Dou, Kelong Mao, Jiongnan Liu, and Ziliang Zhao. 2024. Generalizing conversational dense retrieval via llm-cognition data augmentation. *arXiv preprint arXiv:2402.07092*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 985–988.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, A. Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hanna Hajishirzi. 2024. Olmo: Accelerating the science of language models.

Hiyouga. 2023. Llama factory. `https://github.com/hiyouga/LLaMA-Factory`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *Preprint*, arXiv:2311.10702.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Dawn Lawrie, Sean MacAvaney, James Mayfield, Paul McNamee, Douglas W. Oard, Luca Soldaini, and Eugene Yang. 2024. Overview of the TREC 2023 NeuCLIR track.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

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 Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023.

Holistic evaluation of language models. *Preprint*, arXiv:2211.09110.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval. *arXiv preprint arXiv:2310.08319*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Ansong Ni, Matt Gardner, and Pradeep Dasigi. 2021. Mitigating false-negative contexts in multi-document question answering with retrieval marginalization. *arXiv preprint arXiv:2103.12235*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Douglas W Oard, Björn Hedin, Stephen Tomlinson, and Jason R Baron. 2008. Overview of the trec 2008 legal track. In *TREC*, pages 500–277.

Hanseok Oh, Hyunji Lee, Seonghyeon Ye, Haebin Shin, Hansol Jang, Changwook Jun, and Minjoon Seo. 2024. Instructir: A benchmark for instruction following of information retrieval models. *arXiv preprint arXiv:2402.14334*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022a. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022b. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Piotr Padlewski, Max Bain, Matthew Henderson, Zhongkai Zhu, Nishant Relan, Hai Pham, Donovan Ong, Kaloyan Aleksiev, Aitor Ormazabal, Samuel Phua, et al. 2024. Vibe-eval: A hard evaluation suite for measuring progress of multimodal language models. *arXiv preprint arXiv:2405.02287*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Shohreh Shaghaghian, Luna Yue Feng, Borna Jafarpour, and Nicolai Pogrebnyakov. 2020. Customizing contextualized language models for legal document reviews. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2139–2148. IEEE.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Ian Soboroff. 2021. Overview of trec 2021. In *30th Text REtrieval Conference. Gaithersburg, Maryland*.

Ian Soboroff, Shudong Huang, and Donna Harman. 2018. Trec 2018 news track overview. In *TREC*, volume 409, page 410.

Ian Soboroff, Shudong Huang, and Donna Harman. 2020. Trec 2020 news track overview. In *TREC*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Ellen M Voorhees. 2005. The trec robust retrieval track. In *ACM SIGIR Forum*, volume 39, pages 11–20. ACM New York, NY, USA.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022a. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023a. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023b. How far can camels go? exploring the state of instruction tuning on open resources. *Preprint*, arXiv:2306.04751.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022b. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022c. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

William Webber, Alistair Moffat, and Justin Zobel. 2008. Statistical power in retrieval experimentation. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 571–580.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Orion Weller, Dawn J Lawrie, and Benjamin Van Durme. 2024. Nevir: Negation in neural information retrieval. *Conference of the European Chapter of the Association for Computational Linguistics*.

Orion Weller, Kyle Lo, David Wadden, Dawn Lawrie, Benjamin Van Durme, Arman Cohan, and Luca Soldaini. 2023. When do generative query and document expansions fail? a comprehensive study across methods, retrievers, and datasets. *arXiv preprint arXiv:2309.08541*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. In *Advances in Neural Information Processing Systems*, volume 36, pages 71995–72007. Curran Associates, Inc.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *Preprint*, arXiv:2310.07641.

Ming Zhao, Peter Anderson, Vihan Jain, Su Wang, Alexander Ku, Jason Baldridge, and Eugene Ie. 2021. On the evaluation of vision-and-language navigation instructions. *arXiv preprint arXiv:2101.10504*.

A FAQ

Why is the dataset so small / only around 100 instances? In NLP, 100 instances would be a small dataset (although not unheard of, see HumanEval (Chen et al., 2021), VibeEval (Padlewski et al., 2024), etc. which are commonly used for evaluating LLMs and are well-respected).

However, in IR, as each query requires annotating hundreds of documents for relevance, the number of queries is smaller but the number of annotations is similar. Thus, for our 102 query set, there are roughly 102 queries * (50 relevant and 150 non-relevant) documents for a total of around 20k annotations total – which is similar to those in NLP datasets. Thus we can see that annotations for IR require 200x the cost of standard NLP benchmarks per instance.

The alternative is to gather less annotations per query (such as NQ or MSMarco which only have one relevant document per query but many more queries) but those have been shown to overwhelmingly contain documents that are relevant but not marked as relevant, making them low quality for evaluation (Ni et al., 2021; Cai et al., 2022). As such the IR community develops more thoroughly judged versions of them with a smaller number of queries (such as the Deep Learning Tracks 2019-2022 that build off of MSMarco, cited above, or those that we build off of).

Can LMs handle these type of long instructions? Yes! As the length of the query plus instruction is roughly 80 words on average and the documents are around 400 words, most (if not all) LMs can handle 480 words in their context length. This especially holds true for modern LMs with > 1024 token context lengths. Overall, this is really not long context for LMs.

Not all retrieval tasks will have instructions, what can we do then? We agree that this is the case, in fact, we could only find this one source of real-world instructions for retrieval! We believe this is partially due to the lack of systems that can handle them - why would a user give a retrieval system an instruction if it wouldn't use it? However, as shown by TREC we can see that there are use-cases that people want to be able to do with these instructions. Our dataset provides the first step into building systems that can handle them by providing datasets for training and testing. However, systems should be able to handle both the no-instruction case and the instruction case – as lots of datasets exist for evaluating with no-instructions, ours provides ways to test for the instruction case.

B Llama-3-8B version of FollowIR

We also train a Llama-3-8B (AI@Meta, 2024) version of FollowIR. However, as found by many others in the community, Llama3 is worse than Mistral for retrieval (BehnamGhader et al., 2024). We find that the base model has 0.03 p-MRR and after fine-tuning on FollowIR-train it goes to 5.1 p-MRR. For nDCG, FollowIR-Llama3-8B scores 15.4 whereas FollowIR-Mistral has 24.8.

C Hyperparameters for Fine-Tuning Mistral

We used the following hyperparameters for training Mistral: batch size of 32, cosine scheduler, 3e-5 learning rate, max length of 2048, lora rank 8 and alpha 16, bfloat16, and trained for 8 epochs. We used "q_proj,v_proj,o_proj,k_proj" for the LoRA tuning and trained from mistralai/Mistral-7B-Instruct-v0.2.

D Hyperparameters for Inference

We use default parameters for inference, taken from the original code of the authors of the papers we use (from their MTEB evaluations).

E Compute Used

We used a A100 80GB for the experiments. Training took roughly 6 hours while inference took between 3-12 hours according to model size.

F Full Retrieval Results

In Table 5 we show results for models searching on the full collections of the TREC tasks included in FOLLOWIR. Note that because each model retrieves different relevant documents, the instruction-following evaluation has a different set of instances that each model is evaluated on (as it can only be evaluated on documents it retrieved that then become not-relevant).

	Model	Robust04 (mAP) OG	Δ	News21 (nDCG@5) OG	Δ	Core17 (mAP) OG	Δ
No Instruct	BM25	21.4	-1.2	30.1	+5.3	16.8	-0.2
	E5-base-v2	22.7	-7.0	33.6	+1.8	19.7	-3.0
	Contriever	19.2	-7.7	22.5	+9.0	22.6	-7.6
Uses Instruct	TART-Contriever	25.5	-10.1	40.0	-5.0	22.6	-7.6
	BGE-base	23.6	-3.1	36.5	-7.8	23.0	-2.1
	INSTRUCTOR-base	22.5	-2.2	33.3	-2.8	20.0	-0.2
	INSTRUCTOR-XL	30.4	-3.1	38.1	-0.1	29.9	-2.8

Table 5: FOLLOWIR scores on the full retrieval collection (thus rerankers are not included). As the base score is different, there are different numbers of relevant documents they are being evaluated on for p-MRR. Thus, we only report the original (no-instruction) score and the delta when using the TREC instructions. We note that it shows similar results to the main text – retrieval models are not effectively using instructions and see performance degradations with longer text.

G Keywords used for BEIR experiments

GPT-4-Turbo-1106 extracted the following keywords (Table 6) from the instructions these models used, which generated the results in Table 3.

H Prompts Used

We use these prompts for generating the short instructions, the keywords, and the synthetic documents. The examples used in the prompt for the "Full Instructions to Short Instructions" prompt were partially created by the authors, as only the short instructions were provided by TART/INSTRUCTOR.

Model	Dataset	Keywords
BM25/Contriever/E5/MonoT5	FiQA	Finance web
BM25/Contriever/E5/MonoT5	SciFact	science paper verify
BM25/Contriever/E5/MonoT5	NFCorpus	medicine relevant
TART-dual	FiQA	financial web
TART-dual	SciFact	scientific paper verify
TART-dual	NFCorpus	scientific paper paragraph
INSTRUCTOR-base	FiQA	financial supporting:
INSTRUCTOR-base	SciFact	scientific supporting passage:
INSTRUCTOR-base	NFCorpus	medicine relevant
BGE-base	FiQA	relevant passages:
BGE-base	SciFact	relevant passages:
BGE-base	NFCorpus	relevant passages:
INSTRUCTOR-xl	FiQA	finance supporting:
INSTRUCTOR-xl	SciFact	scientific supporting passages:
INSTRUCTOR-xl	NFCorpus	nutrition facts public medical:
E5-Mistral	FiQA	financial replies
E5-Mistral	SciFact	scientific
E5-Mistral	NFCorpus	retrieve relevant
TART-T5-FLAN-xl	FiQA	financial web
TART-T5-FLAN-xl	SciFact	scientific paper verify
TART-T5-FLAN-xl	NFCorpus	Scientific paper paragraph

Table 6: Keywords used for the BEIR keyword analysis. Note that non-instruction models received the keywords used in INSTRUCTOR-base and TART-dual (as shown in the table).

Synthetic Document Creation

I need you to annotate some data for my business and it is super important that you follow instructions precisely or you will be fired.

Given a Google search query and instructions regarding what makes a document relevant, I need you to write two documents: one that would be relevant and one that would not.

Search: TITLE_HERE
Instructions: NARRATIVE_HERE

I need some different options to choose from, so give me three **different** options for both a relevant document and an irrelevant document. They should be **long** paragraph-sized documents (∼300 words each), one on each line. If there is no negation in the instructions, your irrelevant document should be slightly off topic:

Short Instructions to Keywords

I have instructions that are specific to a style of retrieval, but I want you to instead just focus on the relevant keywords that are in these instructions. Your job is to return a list of these keywords that are relevant in the query. There are probably one or two relevant keywords to extract only.

Examples
Example 1:
Instruction: Help me to find a highly related PubMed paper to answer this question.
Keywords: ["PubMed"]

Example 2:
Instruction: I want to find an answer for this Trivia question. Can you find some paragraphs that provide evidence from Wikipedia?
Keywords: ["Trivia", "Wikipedia"]

Example 3:
Instruction: Check if a Quora question is duplicated with this question.
Keywords: ["Quora", "duplicated"]

Example 4:
Instruction: I want to find a related question asked in StackExchange. Can you find one for me?
Keywords: ["related", "StackExchange"]

Your turn
Instruction: FILL_TEXT_HERE
Keywords (either one or two keywords, that are not "documents", "questions", "answer", or "articles"):

Full Instructions to Short Instructions

I have instructions that are specific to a question, but I need your help abstracting them to a general task format that I can give to someone else. I need you to turn them into an abstract command that just describe the general abstract task instead (e.g., where the data is from, what the type of document looks like). It is crucial that you read and follow these instructions, you will get a large bonus if you are successful ($200).

The abstract command should only mention the **task format**. Do **not** refer to any entities or specific text in the original instruction. Your response should be around 10 words. The command should be as if you were speaking to another human.

Examples
Example 1:
Original Instruction: A relevant document would provide information about the whole blood-base perfusate and whether or not it provides superior preservation of myocardial function during ex vivo heart perfusion. This may include research experiments, commentary, or survey/review papers. Information about whole blood-base perfusate alone is not relevant, unless it also mentions it's effect on myocardial function during ex vivo heart perfusion.
Abstract Command: Help me to find a highly related PubMed paper to answer this question.

Example 2:
Original Instruction: A relevant document will contain information that about the right of way in international waters that can be used to determine who should have the right of way in a given situation. For example, it should contain instances about who is at fault in an accident, if it depends on the size of the boat, or details about how this differs according to nationality. Especially relevant are documents describing who is at fault in a crash situation.
Abstract Command: Retrieve a Wikipedia paragraph that answers this question.

Example 3:
Original Instruction: A relevant instance will be a question that is semantically equivalent to the query given. For example, it may contain different lexical words or be a paraphrase of the other, but the underlying meaning will be the same. If the instance is not semantically the same as the query, it is irrelevant.
Abstract Command: Check if a Quora question is duplicated with this question.

Example 4:
Original Instruction: A relevant document would include details about the timing of medicare and what age patients can start using it for healthcare. It may include information about laws, insurance, or other details that describe the age the medicare begins. Less relevant are documents talking about potential laws or facts about medicare that do not answer the question of what age medicare begins. Just the mention of an age and when to start medicare would be relevant.
Abstract Command: I want to know the answer to the question. Can you find good evidence on the web?

Now you can easily see that the abstract command is vague and describes only a short command about how to get the information you need. Follow this exactly—do not reference specifics (like in the above, "international waters" and "medicare" are not included in the abstract command). You should instead keep the abstract command vague and well, abstract about the task only. Use the word "question".

Your turn
Original Instruction: FILL_TEXT_HERE
Abstract Command (remember to use the word "question"):