Unsupervised Cross-Task Generalization via Retrieval Augmentation

Bill Yuchen Lin[†] **Kangmin Tan**[†] **Chris Miller**[†] **Beiwen Tian**[‡] **Xiang Ren**[†]
[†] University of Southern California [‡] Tsinghua University
{yuchen.lin,kangmint,millercs,xiangren}@usc.edu

Abstract

Humans can perform unseen tasks by recalling relevant skills that are acquired previously and then generalizing them to the target tasks, even if there is no supervision at all. In this paper, we aim to improve such cross-task generalization ability of massive multi-task language models such as T0 (Sanh et al., 2021) in an unsupervised setting. We propose a retrieval-augmentation method named `ReCross` that takes a few *unlabelled* examples as queries to retrieve a small subset of upstream data and uses them to update the multi-task model for better generalization. Our empirical results show that the proposed `ReCross` consistently outperforms non-retrieval baselines by a significant margin. [1]

1 Introduction

Cross-task generalization is the ability of a multi-task model to work on a new task that is unseen during its training stage (Ye et al., 2021; Mishra et al., 2021). It is important, yet challenging, to teach machines this ability that we humans naturally have. Recent studies show tremendous promise in cross-task generalization ability in natural language processing (NLP) via training massive multi-task language models (Ye et al., 2021; Sanh et al., 2021; Wei et al., 2021). The general recipe of such multitask training is to fine-tune a text-to-text language model (LMs) such as T5 (Raffel et al., 2020) on a multitask mixture of diverse NLP datasets that are converted to seq2seq formats. We use the term *upstream learning* to refer to this multi-task training stage. In the generalization stage, we are given a target task that is unseen during upstream learning, and we want the upstream model can also work well on this task via reusing the skills acquired previously.

Particularly, in the CrossFit framework (Ye et al., 2021), cross-task generalization requires a small number of labeled instances of the target task for fine-tuning. It is because the templates of CrossFit use the task names as the hard prefixes. For example, an instance of a sentiment analysis task can be "*amazon_review: best cast iron skillet you will ever buy.*" → "positive." Therefore, it is necessary to fine-tune the upstream model with a few examples that have the target task names as prefixes (i.e., few-shot learning), but this largely limits the application scenarios of these multi-task NLP models in practice. In this paper, we instead focus on *unsupervised* cross-task generalization, where we do not have any *labeled* data of an unseen task (i.e., zero-shot learning). We formulate this setting in Sec. 2.

To this end, FLAN (Wei et al., 2021) and T0 (Sanh et al., 2021) use natural language (NL) instructions as prompts to format the data. For example, the T0 models (Sanh et al., 2021) use data with PromptSource (Bach et al., 2022) templates, where the same example of the sentiment analysis task becomes "*Is this review positive or negative? Review: Best cast iron skillet you will ever buy.*" → "positive". Using NL templates to create data mixture is promising for unsupervised generalization, and both methods indeed show decent zero-shot generalization ability. Why can such models generalize to unseen tasks without supervision? We believe it is because these NL instructions

[1] Our code, models, and appendix will be publicly available at https://inklab.usc.edu/ReCross/.

Preprint. Under review.

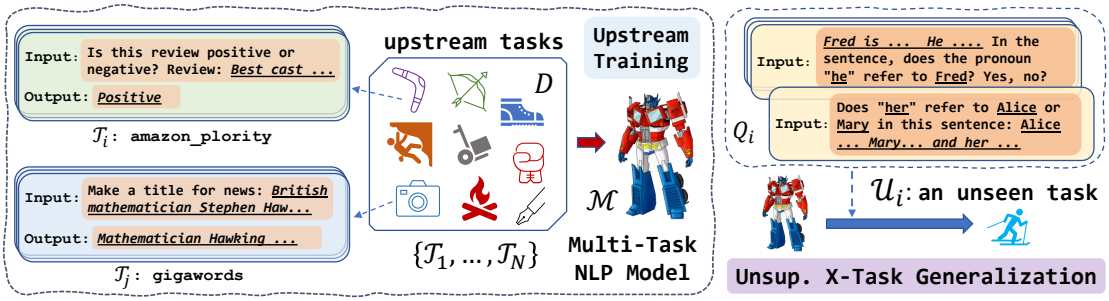

Figure 1: **The unsupervised cross-task generalization problem.** In the upstream training stage, we train a massive multi-task NLP model, $\mathcal{M}$, with a diverse collection of upstream tasks (e.g., $\mathcal{T}_i$ and $\mathcal{T}_j$). In the generalization stage, given an unseen task $\mathcal{U}_i$ with a few unlabeled examples Q_i, we want to update the upstream model such that it can generalize to the target task.

are semantically meaningful, and thus learning with them implicitly helps upstream models *recall acquired skills that are relevant to the target task*.

Inspired by this idea, we propose to further improve the cross-task generalization ability of T0-like models via *retrieval augmentation*. In Section 3, we present a retrieval-augmentation framework, ReCross, for unsupervised cross-task generalization. Specifically, we pre-compute a dense index by encoding all upstream data as vectors. Given a set of unlabeled examples, we first use them to retrieve an initial list of upstream data by using encoded queries to efficiently search over the dense index. Then, we apply the reranking module for carefully analyzing the utility of each candidate example. To get such a reranker, we learn a cross-encoder model with distant supervision mined by our proposed algorithm. Finally, we take top-ranking retrieved data to fine-tune the upstream model for a few steps and use this updated model for inference on the target task. Our intuition is that cross-task generalization can benefit if the upstream model re-learns a small subset of upstream data that share skills that are required by the target task.

To more efficiently evaluate generalization methods without losing the generality, we train a variant of T0-like models, named BART0, which has comparable performance with T0-3B yet is 8x smaller. Our extensive experiments show that the proposed ReCross outperforms the baseline methods by a large margin. For example, ReCross improves the non-retrieval methods by 4 points (in exact-match) on the overall performance of 10 target tasks and similarly on BigBench tasks. We also analyze the distribution of the retrieved data to better understand the behavior of retrieval-augmentation methods and find that ReCross has a very different distribution compared to semantic retrieval baselines.

2 Problem Formulation

We here formulate the problem of *unsupervised* cross-task generalization of a massive multi-task language model and point out its challenges. Figure 1 illustrates the problem of unsupervised cross-task generalization using the notations introduced in the following sub-sections.

Massive Multi-Task Language Models. We assume there are N different **upstream tasks**, dubbed as $\{\mathcal{T}_1, \ldots, \mathcal{T}_N\}$. We use D to denote the collection of all labeled data (i.e., the **upstream data**) for these upstream tasks, which are used for training a massive multi-task model $\mathcal{M}$. The datasets of these upstream tasks are all converted to a shared *text-to-text* format using natural-language instruction templates such as PromptSource (Bach et al., 2022) to reformat data of different NLP tasks. This pipeline has become a common approach, adopted by several recent massive multi-task models for NLP, such as T0 (Sanh et al., 2021), FLAN (Wei et al., 2021), and CrossFit (Ye et al., 2021), which are good at cross-task generalization towards unseen tasks using limited data.

Unsupervised Cross-Task Generalization. Let us consider a collection of M unseen tasks that are not included in the above upstream tasks, and we denote them with $\{\mathcal{U}_1, \ldots, \mathcal{U}_M\}$. For each unseen task $\mathcal{U}_i$, we have only a small number of ***unlabeled*** examples, and our objective is to use these unlabeled data to improve the general performance of upstream model $\mathcal{M}$ for the task $\mathcal{U}_i$. Specifically,

we consider two sets of examples for the task $\mathcal{U}_i$: 1) a small unlabeled *query set* Q_i and 2) a large labeled *test set* E_i, where $|Q_i| \ll |E_i|$. An unsupervised cross-task generalization method f thus needs to enhance $\mathcal{M}$ (by only using the inputs from Q_i) so that the updated model $\mathcal{M}_i$ can perform better on E_i. To get a comprehensive assessment, we evaluate the overall performance of multiple unseen tasks. Please find more evaluation details in Section 4.1.

Challenges. Unlike the formulation of few-shot learning for cross-task generalization presented in the CrossFit framework (Ye et al., 2021), this zero-shot setup is more challenging due to the lack of human-annotated supervision on the query set. To enhance these upstream multi-task LMs for a new task U_i, one may want to use the query examples Q_i to re-weight the upstream data $\mathcal{D}$ and then re-train a new model dedicated for the target task U_i. However, such re-training methods are hardly feasible in practice because they require too much time and computation, especially when we have many target tasks (i.e., M is a large number) to generalize to. We discuss more in Sec. 5.

3 ReCross: Retrieval Augmentation for Cross-Task Generalization

We introduce a simple and general retrieval-augmentation framework for unsupervised cross-task generalization, named `ReCross`. It consists of two components: a dense retriever and a re-ranking module. Figure 2 illustrates the overview of the proposed framework.

3.1 Overview

Recall that our goal is to improve the upstream model $\mathcal{M}$ for an unseen task $\mathcal{U}$ by using its unlabelled query examples Q. Our motivation is thus to retrieve a subset of upstream data $R \subset D$ and use R to fine-tune $\mathcal{M}$ for learning a dedicated model $\mathcal{M}'$ that can do better on the task $\mathcal{U}$. The retrieved examples R should be highly related to the query examples Q that reveal the skills required by the task $\mathcal{U}$. We believe re-learning these examples R can help us better generalize $\mathcal{M}$ to the unseen task $\mathcal{U}$, even though they have already been used for upstream training. To efficiently retrieve such data, we present a two-stage retrieval-augmentation framework, namely `ReCross`.

In the first stage, we build a dense index of all upstream examples, where each upstream example in D is encoded as a dense vector. We then build query vectors by encoding query examples Q for quickly retrieving the most relevant examples via *maximum inner product search* (MIPS) over the dense index of upstream data. In the second stage, we re-rank these initial retrieved examples with a more computationally expensive module that can further improve the quality of the final retrieved data. We refer the version without the reranking stage as ReCross†.

3.2 Dense Retriever

Efficient indexing and searching. Because the size of the upstream data D can be extremely large, we must ensure that the retrieval step is efficient. Therefore, we pre-compute a dense index by encoding each example $(x, y) \in D$ as a d-dimensional vector $\mathbf{x} \in \mathbb{R}^d$, so that we can build the matrix $\mathbf{D} \in \mathbb{R}^{|D| \times d}$ as the index of all upstream examples. Given a query set Q, we then encode each query example to get a set of query vectors for retrieval augmentation on the fly. Finally, we use these vectors to search for the closest upstream examples over the index $\mathbf{D}$ via MIPS. The MIPS operation produces a set of retrieved data that can be used as the initially retrieved data which can be further re-ranked. We use FAISS (Johnson et al., 2019) to build and query these dense indices efficiently.

Example encoding. The example encoder is a key piece of the above dense-retrieval pipeline. We use the encoder to embed both upstream examples in D and the query examples in Q. As we seek to use MIPS to enable efficient retrieval, a simple and effective strategy is to use the same encoder for both D and Q such that a simple L2 distance metric can work effectively. We propose to re-use the upstream model $\mathcal{M}$ for encoding the examples. Without loss of generality, let us assume $\mathcal{M}$ to be a text-to-text Transformer that has multiple layers for both encoder and decoder, such as BART (Lewis et al., 2020a) and T5 (Raffel et al., 2020). We encode an example by first collecting the hidden representation of each token at the last encoder layer, and then performing a mean-pooling operation to get a single dense vector to represent this example.

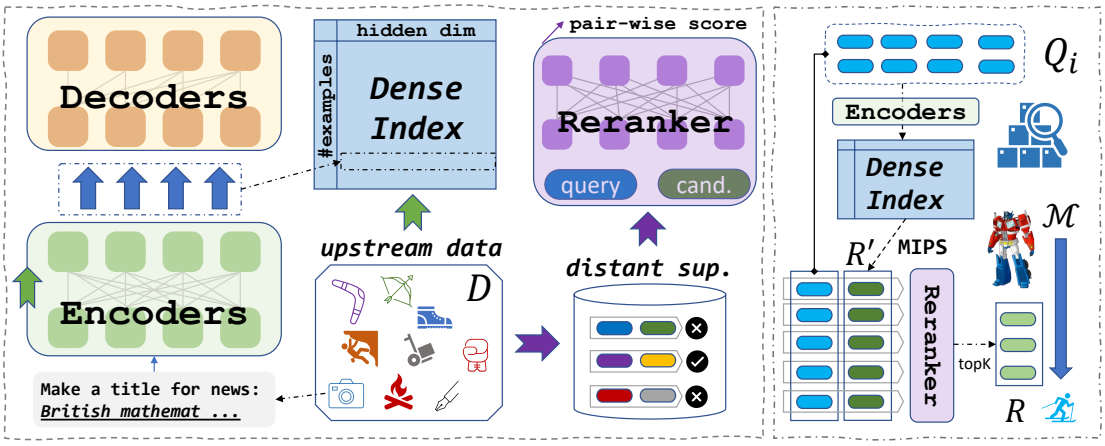

Figure 2: **We propose the ReCross, a retrieval-augmentation method for unsupervised cross-task generalization.** We reuse the encoder layers of the upstream model (green) to build a dense index, which consists of vectors of the upstream examples D. We also propose an algorithm to generate distant supervision for training a reranker, which takes a pair of examples as input and outputs a score. During the evaluation, we encode query examples Q_i for querying the index and get initial ranking results R' and then pair them with the query examples again for reranking. Finally, we take the top-K results (i.e., R) for generalizing the upstream model $\mathcal{M}$ to the unseen task $\mathcal{U}_i$.

Retrieval aggregation. Note that our target size of retrieved data is $|R|$ and we have $|Q|$ query examples. To retrieve $|R|$ examples, we search for the top-K examples for each query example, where $K = \lceil \frac{|R|}{|Q|} \rceil$, and then take the $|R|$ of them when $K|Q| > |R|$. Our results have shown that this method is more effective than other strategies, such as combining the distance scores generated for each query example. Note that by retrieving the top-K examples, we may repeat examples that are close to multiple query vectors. This effect is desirable because it allows us to naturally focus more on the especially relevant upstream examples in re-learning.

3.3 Reranking Module

The above dense retrieval module allows us to efficiently retrieve a subset of possibly-relevant examples from the upstream data D. This efficiency is with the cost of not permitting expensive processing of all possible query/candidate pairs, the number of which would be $|Q| * |D|$ and thus too huge to compute in practice. To further improve the retrieval process, we present a re-ranking module based on a cross-encoder, which is designed to score a query-and-candidate pair of examples. This enables us to leverage both efficient retrieval from the extensive upstream data and more computationally expensive scoring to produce the final retrieved results. We leave the details of creating the supervision for training such a reranker in Sec. 3.4. This is also shown in Figure 2.

Encoding query-candidate pairs. The cross-encoder architecture has been widely used in sentence-pair classification tasks such as natural language inference and paraphrase detection. We here want to use a cross-encoder to encode the *concatenation* of a query example and a candidate example. Specifically, we fine-tune a RoBERTa-base (Liu et al., 2019) model to classify whether an example pair is a positive or negative match. The confidence of classifying such an example-pair to be positive can thus be used as the score of the candidate upstream example for this query example. On top of this, we can then develop a reranking module for further improving retrieval performance.

Scoring paired data. To rerank the initially retrieved data from the dense retriever, we apply the cross-encoder over all pairs of query examples Q and candidate retrieved examples R, thus producing the scores of all $|Q| * |R|$ query-candidate pairs. For each candidate example, we use the average of all scores involving it as its new score. Finally, we take the top-K examples based on this new ranking of candidate examples in R' as the final retrieved data R. Note that we refer to the ratio between R' and R to be the *upsampling ratio* μ.

3.4 Creating Distant Supervision for Reranking

How do we train such a reranking module? We need some training data. Note that at this stage we know nothing about the unseen tasks, and only have access to the upstream data D. To learn such a reranker from D, we propose creating a distant supervision dataset based on the dense retriever, which is shown in Alg. 1. We define a data point of such distant supervision as a tuple $Z = (Z_q, Z_p, Z_n)$: 1) Z_q is a set of query examples of a particular task $\mathcal{T}_q$; 2) Z_p is the set of positive examples from other tasks; 3) Z_n is the set of negative examples from other tasks.

Algorithm 1: Distant Supervision Creation

Input: $\mathcal{M}$; D; $\mathcal{T}_q$
Output: $Z = (Z_q, Z_p, Z_n)$

$D_{\mathcal{T}_q} \leftarrow \{x \in D | x \text{ is an example of } \mathcal{T}_q\}$
$Z_q \leftarrow \text{Sample}(D_{\mathcal{T}_q})$; $H_q \leftarrow \text{Sample}(D_{\mathcal{T}_q})$
$R_Z \leftarrow \text{DenseRetrieve}(Z_q, D)$
/* Delete retrieved examples from the same task as queries. */
$R_Z \leftarrow R_Z.\text{discard}(D_{\mathcal{T}_q})$
foreach *round* **do**
  $R_Z.\text{shuffle}()$
  /* Split retrieved examples into n groups */
  $\{G_1, ..., G_n\} \leftarrow R_Z.\text{split}()$
  foreach G_i *in* $\{G_1, ..., G_n\}$ **do**
    $\mathcal{M}' \leftarrow \mathcal{M}.\text{copy}()$
    $\mathcal{M}'.\text{fine\_tune}(G_i)$
    $\ell \leftarrow \mathcal{M}'.\text{calc\_loss}(H_q)$
    foreach $x \in G_i$ **do**
      $scores[x].\text{append}(\ell)$ /* Score each example in the group w/ the loss. */
/* Use mean group score as score for single examples */
foreach $x \in R_Z$ **do**
  $score[x] \leftarrow \text{mean}(scores[x])$
/* Sort R_Z by $score$ in increasing order. */
$R_Z.\text{sort}(\text{key: score, order: increasing})$
$Z_p \leftarrow$ First W items of R_Z
$Z_n \leftarrow$ Last W items of R_Z

We expect that Z_p is more suitable than Z_n as the retrieved data if Z_q would be a query set. To this end, we first randomly sample an upstream task $\mathcal{T}_q$ and use a small subset of its training data as the Z_q. Here, we also sample a larger held-out set H_q examples of task $\mathcal{T}_q$ to facilitate

Then, we apply the dense retriever using Z_q as the query examples and get the initially retrieved data R_Z. This R_Z is thus the candidate pool where we create Z_p and Z_n. That is, $Z_p \subset R_Z$ and $Z_n \subset R_Z$. We discard examples that are from the $\mathcal{T}_q$, so that the generated tuples are closer to the scenarios where we use the reranker on the query sets of unseen tasks.

Our criteria to select Z_p and Z_n from R_Z is motivated by the hypothesis that a more suitable set of retrieved examples should improve the performance $\mathcal{M}$ on $\mathcal{T}_i$ after fine-tuning with it. Therefore, we iteratively sample a small subset from R_Z, then fine-tune $\mathcal{M}$ with it, and finally use the fine-tuned model to evaluate on Z'_q. The performance of such a temporarily fine-tuned model can be seen as a score telling us how well this subset can generalize to the unseen task $\mathcal{T}_q$. Through multiple rounds of such sample-train-test procedures, we can thus score each example in R_Z by taking the average of all test results where it is involved. With such a new ranking of examples in R_Z, we take the best W examples as Z_p and the worst W as Z_n.

With such distant supervision, we then can create pair of query-positive instances and query-negative instances via pairing Z_q with Z_p and Z_n respectively. Now we can fine-tune a RoBERTa-base model by concatenating each pair and learning a binary-classification objective. The output logits of this trained model will be used for the reranking procedure as shown in Sec. 3.3.

3.5 Fine-Tuning with Retrieved Data

When we have the final retrieved data R_i for a certain query set Q_i, we can now enhance the upstream model $\mathcal{M}$ for the unseen task $\mathcal{U}_i$. To this end, we use a smaller learning rate to continually fine-tune $\mathcal{M}$ on the R_i for a small number of steps. We find that the learning rate has to be very small so that this step can be seen as a natural continuation of the finished upstream training and avoid over-fitting on the retrieved data. We acknowledge that there could be more effective methods to reuse the query examples Q as guidance for fine-tuning, and we leave this as future work.

4 Evaluation

In this section, we first introduce the experimental setups, including the task distribution, upstream learning details, and the configurations of the main experiments. Then, we present the main experimental results and finally reveal several findings with our analysis.

4.1 Evaluating Cross-Task Generalization

We follow Sanh et al. (2021) to use the templates from PromptSource (Bach et al., 2022) for converting data of different types of NLP tasks to text-to-text formats. In total, we have 36 upstream tasks and 10 target unseen tasks for our main experiments. The upstream tasks are the same as the ones that the T0 models used for upstream learning. We follow the evaluation protocol proposed by Sanh et al. (2021) and select the target tasks that are significantly different from the upstream tasks. Besides, we also include 5 additional tasks from the BIG-bench project (BIG-bench collaboration, 2021), which are even more out-of-distribution and thus helpful for further analyzing the generalization ability.

Metric. When we apply the natural-language templates for the test examples, we only keep the templates that can be evaluated with an exact match (e.g., classification, question answering, etc.) so that it is possible to use an exact match for evaluating all tasks. To allow smoother grading, our metric also counts the cases when outputs and truths are sub-strings of each other, which we call **SoftEM**.

4.2 BART0: Upstream Learning with a Smaller LM

The T0(++) models are all very huge, and the smallest version, T0-3B (3 billion parameters), is still too large to be effectively fine-tuned on popular affordable GPUs. To improve the efficiency and keep the generality of our study on the retrieval augmentation methods, we propose to train a more parameter-efficient alternative. Therefore, we fine-tune a BART-large (Lewis et al., 2020a) (0.4 billion parameters) following the recipe of training T0. Specifically, we sample 50k examples at most from each upstream task to build a large upstream dataset consisting of 1.7 million examples (i.e., $|D| = 1.7$m), and then we fine-tune a BART-large with 22k steps with this upstream dataset. Finally, we use the fine-tuned checkpoint as our upstream model $\mathcal{M}$ and call it **BART0**. Surprisingly, we find that BART0 and T0-3B have comparable zero-shot performance, even though T0-3B is about 8x larger than BART0. More implementation details about this are shown in the appendix.

4.3 Setup and Configurations

In our main experiments, we use $|Q_i| = 16$ query examples for each unseen task $\mathcal{U}_i$ and retrieve $|R_i| = 512$ examples for augmenting the BART0 to achieve better zero-shot performance on the target tasks. In the fine-tuning stage, we use a learning rate of 1e-6 and a batch size of 4 to continually fine-tune all layers of BART0 for 2 epochs. As for re-ranking, we set the upsampling ratio $\mu = 2$, meaning that we will first retrieve 1024 examples for reranking and use the top 512 as the final retrieved data. Note that we also show the performance with other configurations in Table 3.

We average the scores of all target tasks to show the general zero-shot performance. For each task $\mathcal{U}_i$, we use five different query sets, $\{Q_i^{(1)}, \ldots, Q_i^{(5)}\}$, to conduct five individual rounds of retrieval, thus resulting in five average scores for all tasks. To get a comprehensive assessment, we report the mean, std, median, min, and max of these five overall scores in the lower part of Table 1.

4.4 Experimental Results

BART0 vs T0-3B. As we mentioned before, we find that BART0 is comparable with the much larger T0-3B in terms of their zero-shot performance on our unseen tasks (41.33 vs 40.38). As we use BART0 as our base model for testing different retrieval-augmentation methods, its overall performance *40.38* is what we want retrieval-augmentation methods to beat. Note that when using BART0 and T0-3B for non-retrieval zero-shot inference, they do not use any information from the query sets, so their mean, median, min, and max are always the same.

Random Retrieval. The *Random* column shows the results when we randomly sample R_i from the upstream data D without using any information from Q_i. From the all@mean and all@median, we can see that such random retrieval does not produce better overall performance, which is expected. However, in a few target tasks, such a random-retrieval baseline can indeed outperform the base model, e.g., anli-r3 (30.50→35.34), cb (39.64→47.07), and winogrande (51.10→52.68). This suggests that it is promising to study better retrieval methods to get consistent improvement in overall performance.

Target Task	T0-3B	**BART0**	Random	SBERT	ReCross†	**ReCross**	Δ
anli_r3	26.00	30.50	$35.34_{\pm 1.52}$	$32.64_{\pm 2.53}$	$36.70_{\pm 0.53}$	$35.76_{\pm 0.90}$	5.26
h-swag	34.40	39.40	$33.84_{\pm 5.59}$	$30.92_{\pm 7.82}$	$44.36_{\pm 3.07}$	$47.28_{\pm 2.95}$	7.88
cb	53.93	39.64	$47.07_{\pm 1.25}$	$48.00_{\pm 3.28}$	$44.50_{\pm 4.20}$	$44.79_{\pm 3.36}$	5.15
wic	45.70	46.70	$41.04_{\pm 2.18}$	$46.78_{\pm 2.22}$	$49.90_{\pm 0.50}$	$50.58_{\pm 0.24}$	3.88
wsc	50.00	57.88	$52.50_{\pm 2.29}$	$52.69_{\pm 6.13}$	$59.27_{\pm 1.96}$	$61.46_{\pm 1.47}$	3.58
winogrande	47.60	51.10	$52.68_{\pm 0.83}$	$52.18_{\pm 3.20}$	$54.60_{\pm 1.35}$	$55.46_{\pm 0.88}$	4.36
arc-chan.	41.30	35.70	$33.28_{\pm 1.50}$	$37.90_{\pm 1.22}$	$37.78_{\pm 0.73}$	$38.44_{\pm 0.99}$	2.74
obqa	38.50	34.40	$28.72_{\pm 2.46}$	$33.28_{\pm 1.24}$	$36.98_{\pm 1.55}$	$39.58_{\pm 2.80}$	5.18
piqa	45.30	36.10	$37.00_{\pm 2.71}$	$38.54_{\pm 2.17}$	$41.34_{\pm 1.75}$	$41.42_{\pm 1.02}$	5.32
squadv2	30.60	32.40	$29.86_{\pm 5.46}$	$29.46_{\pm 0.84}$	$30.26_{\pm 1.54}$	$30.58_{\pm 1.61}$	-1.82
All@mean	41.33	40.38	$39.13_{\pm 2.06}$	$40.24_{\pm 1.61}$	$43.57_{\pm 0.68}$	$44.53_{\pm 0.42}$	4.15
@median	41.33	40.38	39.93	40.91	43.43	44.31	3.93
@min	41.33	40.38	35.66	38.28	42.65	44.16	3.77
@max	41.33	40.38	40.59	41.76	44.51	45.07	4.69

Table 1: **The main experimental results (%) for unsupervised cross-task generalization.** Each result in the upper section is the average (and the std) performance of using 5 different query sets for a task. The lower section of this table reports the mean, max, min, and median of the overall performance (i.e., the average performance on all tasks) of these five rounds.

SBERT and ReCross†. We here use SentenceBERT (Reimers and Gurevych, 2019) (SBERT)[2] as a strong baseline method to create a dense index of the upstream data, compared with our proposed indexing method (ReCross†) — using the encoder of the upstream model (i.e., BART0) to produce embeddings of upstream data. We can see that (ReCross†) always outperforms the other methods. Even its minimum performance in the five rounds (*42.65*) is better than the maximum of the SBERT (*41.76*). Besides, the standard deviation also becomes much smaller (1.61→ 0.68), which means that improvement by the ReCross† is more consistent under different query sets.

The SBERT indexing relies mainly on the *semantic similarities* between a query example and the upstream data. Instead, our proposed ReCross† uses the hidden representations inside the upstream model $\mathcal{M}$ for representing examples. We believe using such an indexing method can better help us find examples that share *similar reasoning skills* acquired by the upstream model.

ReCross = ReCross† + Reranking. The full version of our ReCross with reranking can indeed further improve the performance very much on multiple dimensions. Both all@mean and median are improved by 1 point from the ReCross†, and the std is also reduced from 0.68 to 0.42. The last column (Δ) in Table 1 shows its improvement compared to the base model BART0, and we can see that ReCross consistently outperforms non-retrieval methods (e.g., BART0) by a significant gap.

To explore the potential benefits of retrieval-augmentation methods such as our ReCross, we also conduct the same experiments on five tasks selected from the BIG-Bench project. The results are shown in Table 2, where we can see that ReCross still outperforms the non-retrieval methods. An interesting case is the movie_dialog task, where the prompt in the template requires a model to output "same" or "different." However, both T0-3B and BART0 fail to follow the prompt instruction, and can only output "yes/no." Only via retrieval-augmentation, we can see the performance improvement on this task.

Task	T0-3B	BART0	ReCross
hindu_knowledge	24.75	23.48	$24.87_{\pm 0.27}$
known_unknowns	47.83	43.48	$47.17_{\pm 1.65}$
logic_grid_puzzle	23.60	20.70	$17.12_{\pm 6.29}$
strategyqa	47.70	48.30	$49.76_{\pm 0.80}$
movie_dialog	0.00	4.40	$37.22_{\pm 13.26}$
All@Mean	28.78	28.07	$35.23_{\pm 2.85}$

Table 2: **Results on a subset of BigBench tasks.**

4.5 Analysis & More Findings.

[2]We use the `all-distilroberta-v1` checkpoint for its great performance and relatively smaller size.

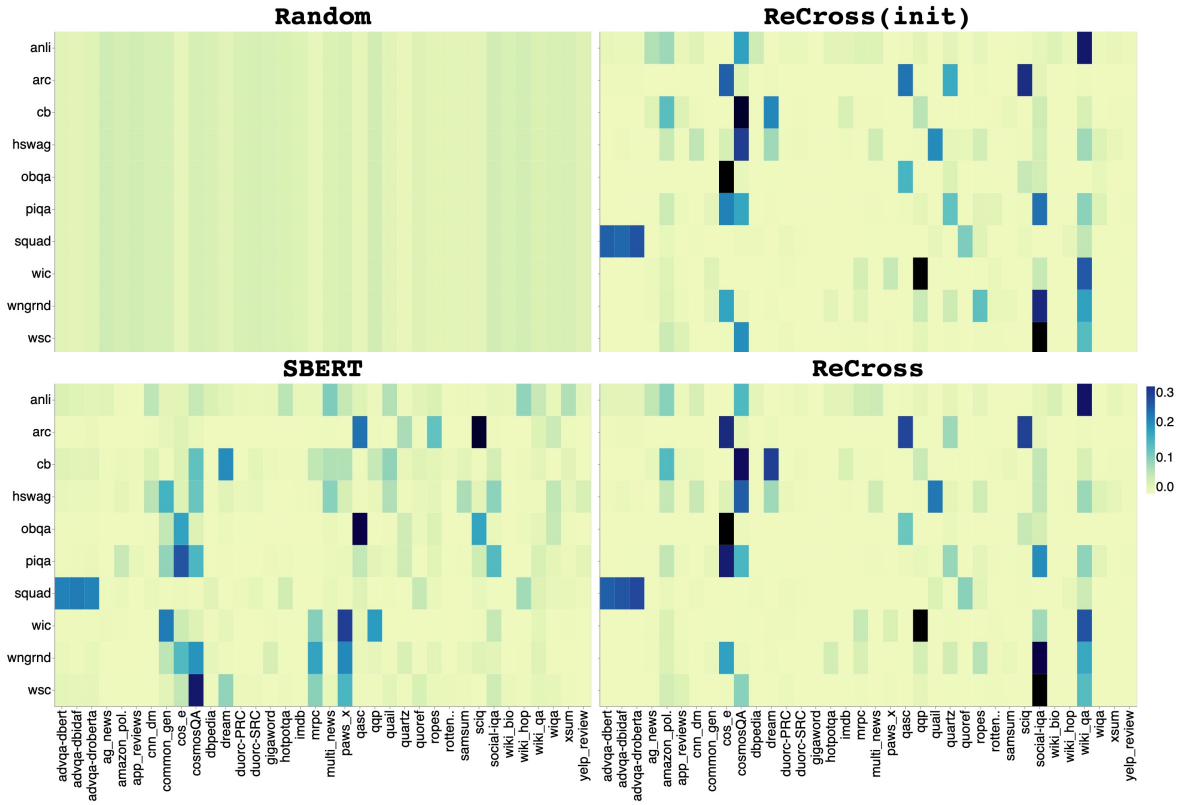

Figure 3: **The mapping between unseen tasks (as rows) and upstream tasks (as columns).** The darker upstream tasks take more percentage in retrieved data. For example, in ReCross's retrieved results towards the task of `WIC`, the examples from `QQP` are the most, which take about 30%.

Setup\All@	Mean	std.	Min	Max	Median
Main Exp.	44.53	0.42	44.16	45.07	44.31
$\|Q\|$=1	43.20	0.83	42.58	44.58	42.88
$\|Q\|$=8	43.67	0.90	42.09	44.32	43.90
$\|Q\|$=32	42.52	1.17	40.52	43.40	42.96
$\|R\|$=256	40.80	0.83	39.45	41.68	40.96
$\|R\|$=1024	44.02	1.43	42.26	45.35	44.59
μ=3	43.92	0.58	43.08	44.57	43.89
μ=4	43.91	0.99	42.76	45.10	44.26

Table 3: **The ablation study of ReCross.**

More configurations. We have used a particular configuration in our main experiments that are in Table 1, which is $|Q|$=16, $|R|$=512, and $|u|$=2. In Table 3, we explore more configurations as ablation studies. The "Main Exp." row refers to the results shown in Table 1, and the configurations of other rows are only changed with one factor at a time. Even using a single query example, ReCross is better than BART0. However, when increasing the query size to 32, we find that the performance starts to decrease, meaning that there could be an optimal query size for a certain $|R|$=512. We find that increasing $|R|$ is generally beneficial, while the all@mean decreases when $|R|$ is changed from 512 to 1024, although the max and the median slightly increased. Finally, we see that increasing μ increases the std. and does not improve the overall performance.

Retrieved data distribution. Figure 3 presents the difference between the four methods in terms of their retrieved data. We draw the distribution of the retrieved data among different upstream tasks for each unseen task individually. From the heatmap, we can see that ReCross tends to have more dominant retrieved tasks (i.e., darker cells), while SBERT's results are more sparse. They both can identify that `squad` is most similar to the three `adversarial_qa` tasks. Their behaviors are very different too. Taking the unseen task `winogrande` (wngrnd) as an example, we can see that the SBERT retrieves from multiple upstream tasks such as `paws-x` and `cosmosQA` , but the ReCross mainly retrieves from `social-iqa`, `wiki-qa`, and `cos-e`. The experimental results in Table 1 show that ReCross produces a better performance than SBERT (i.e., 55.46 vs 52.18), while it is not

clear how we can predict such task correlation in advance. This suggests that we should explore more about the utility of instances and tasks in future work.

5 Related Work

Multi-task training of NLP models. Text-to-text Transformer language models such as T5 and BART enable us to train a multi-task NLP model with a more straightforward recipe: mixing the data of multiple tasks into a unified seq2seq format, and then fine-tuning text-to-text LMs for implicit multi-task learning. UnifiedQA (Khashabi et al., 2020) is among the first works in this direction. It combines the data of many different question answering (QA) tasks with a unified data format and fine-tunes T5 for learning a powerful QA model. Although this method shows great generalization performance within the general QA tasks, it can hardly generalize to other NLP tasks.

To further explore the limit of such a multi-task training recipe, Ye et al. (2021) proposed CrossFit, a framework that trains a massive multi-task NLP model on a repository of 160 distinct tasks. Similarly, FLAN (Wei et al., 2021) and T0 (Sanh et al., 2021) follow the same recipe for training, while they both propose to use prompts to format the data. These prompts, or instructions, are a set of templates that explain the goal of a task in natural language. In this paper, we use our BART0 for all experiments, which shows comparable performance with T0-3B, yet with a much smaller size.

Cross-task generalization. A significant goal of developing these massive NLP models is to achieve better cross-task generalization ability – i.e., the ability to rapidly generalize to an unseen task. Although the CrossFit models show decent generalization ability on unseen tasks, they have to assume there are a few labeled examples (i.e., few-shot learning). Using natural-language instructions as prompts, both FLAN and T0 show that it is promising to perform *zero-shot* cross-task generalization. In this work, we also focus on such an unsupervised setting, where we only require a few unlabelled examples. Besides, Mishra et al. (2021) show that the instructions used in crowd-sourcing can be helpful in cross-task generalization too.

Retrieval augmentation. We aim to tackle the unsupervised cross-task generalization problem by retrieving useful examples from the upstream data and re-learning them. Our pipeline is inspired by open-ended QA methods such as DPR (Karpukhin et al., 2020), DrFact (Lin et al., 2021), and RAG (Lewis et al., 2020b). Retrieval augmentation also shows great performance in pre-training LMs (Guu et al., 2020). Unlike these works, we do not aim to retrieve useful information from a knowledge corpus (e.g., Wikipedia). Instead, we want to retrieve examples (i.e., input-output pairs) from the upstream data that are of multiple NLP tasks. Also, prior retrieval augmentation methods aim to use the retrieved info to address a particular *instance* (e.g., an open-domain question), but our ReCross method is designed to improve the generalization for a new *task* (e.g., coreference resolution). In particular, both our reranking module and the one in DrFact are cross-encoders and they both enhance the final results very much. However, unlike these QA challenges, our setting does not have any truth "answers" for a given query, so we develop a dedicated method to collect distant supervision via iteratively sampling and testing (see Alg. 1 in Sec. 3.3).

6 Conclusion & Future Directions

We demonstrate that retrieval augmentation can largely improve the cross-task generalization ability to multitask LMs in unsupervised settings. Our proposed method, ReCross, is a straightforward yet effective retrieval method that combines both efficient dense retrieval and effective pair-wise reranking. Our empirical results show that it significantly outperforms both non-retrieval methods and other baseline methods. We perform ablation studies showing the impact of changing query sizes, retrieval sizes, upsampling ratios, etc. We also find the distribution of retrieved data for analyzing the behavior differences between ReCross and others. We believe that our paper will spur further research on retrieval-augmentation methods for cross-task generalization. Interesting future directions include: 1) improve the re-learning stage by including more information from query examples, 2) extend the distant supervision mining process as a self-training procedure, 3) rigorously analyze the correlation between upstream data and target tasks, etc.

References

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts.

BIG-bench collaboration. 2021. Beyond the imitation game: Measuring and extrapolating the capabilities of language models. *In preparation*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

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

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Bill Yuchen Lin, Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Xiang Ren, and William Cohen. 2021. Differentiable open-ended commonsense reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4611–4625, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, 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, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv preprint*, abs/2109.01652.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.