Improving Multitask Retrieval by Promoting Task Specialization

Wenzheng Zhang[1]* **Chenyan Xiong**[2] **Karl Stratos**[1] **Arnold Overwijk**[2]
[1]Rutgers University, USA [2]Microsoft, USA
{wenzheng.zhang, karl.stratos}@rutgers.edu
{chenyan.xiong, arnold.overwijk}@microsoft.com

Abstract

In multitask retrieval, a single retriever is trained to retrieve relevant contexts for multiple tasks. Despite its practical appeal, naive multitask retrieval lags behind task-specific retrieval, in which a separate retriever is trained for each task. We show that it is possible to train a multitask retriever that outperforms task-specific retrievers by promoting task specialization. The main ingredients are: (1) a better choice of pretrained model—one that is explicitly optimized for multitasking—along with compatible prompting, and (2) a novel adaptive learning method that encourages each parameter to specialize in a particular task. The resulting multitask retriever is highly performant on the KILT benchmark. Upon analysis, we find that the model indeed learns parameters that are more task-specialized compared to naive multitasking without prompting or adaptive learning.[1]

1 Introduction

A standard approach to knowledge-intensive language tasks such as question answering (QA), entity disambigution, and fact verification is retrieval-based. Given an query, a retriever is used to efficiently search a large knowledge base (KB) to retrieve relevant ‘‘contexts’’, typically in the form of short paragraphs. How these contexts are used is task-specific (e.g., entity disambiguation takes the title of the article in which the top retrieved context is found; QA predicts an answer from the contexts through a reader model). In this paper, we focus on the retrieval step.

In particular, we focus on multitask retrieval. In this setting, there are $K > 1$ downstream tasks that benefit from retrieval from a shared KB. A single retriever is then tasked with performing retrieval for K tasks. Multitask retrieval contrasts with task-specific retrieval, in which a separate retriever is trained for each task, and has compelling advantages such as model simplicity (i.e., we can use the same model for all tasks rather than having to design potentially different models for different tasks) and memory efficiency at test time (K times smaller).

Despite the practical appeal, the performance of multitask retrieval has been underwhelming, severely limiting its real-world applicability. Specifically, previous work by Maillard et al. (2021) trains DPR (Karpukhin et al., 2020) on the union of all training datasets in the KILT benchmark (Petroni et al., 2021), but the model is outperformed by task-specific retrievers in 5 out of 8 tasks (page-level R-precision, validation split). In our experiments, we find that it is in fact outperformed in all tasks (often by substantial margins) when a stronger task-specific baseline is used. This result is surprising as well as disappointing given the usual benefits of multitask learning (e.g., data efficiency, reduced overfitting) when properly done.

We debunk the previous negative result by presenting a multitask retriever that outperforms task-specific retrievers. The main theme of our work is that it is beneficial to explicitly promote task specialization. A first important source of improvement is a better choice of pretrained model, one that is explicitly optimized for multitasking. Specifically, instead of the standard retrieval encoder BERT (Devlin et al., 2019), we use T5 (Raffel et al., 2019), which includes multitasking in its pretraining stage. Importantly, we use the same prompting as in pretraining (i.e., task indicator) to reduce the gap between pretraining and finetuning for multitask retrieval. A second source of improvement is a novel adaptive learning method in which we adatively upweight the task gradients by the parameter’s sensitivity to these tasks to encourage task specialization.

*Work done during an internship at Microsoft.

[1]Our code and model checkpoints are publicly available at https://github.com/WenzhengZhang/TACO.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 1201–1212, 2023. https://doi.org/10.1162/tacl_a_00597
Action Editor: Dani Yogatama. Submission batch: 1/2023; Revision batch: 5/2023; Published 9/2023.

The resulting multitask retriever is highly performant on the KILT benchmark. We achieve 73.74% average page-level R-precision on KILT validation data and 72.84% average page-level R-precision on KILT test data. Upon analysis, we find that the model indeed learns parameters that are more task-specialized compared to naive multitasking without prompting or adaptive learning.

2 Related Work

Maillard et al. (2021) propose multitask retrieval largely as an extension of DPR. Their best model is a BERT-based dual encoder trained on the union of 8 retrieval tasks. While it performs comparably with task-specific DPRs on some tasks, it generally lags behind. In this work, we use stronger task-specific retrievers based on T5 and ANCE (Xiong et al., 2021), all of which substantially outperform their multitask retriever. We argue that this negative result undermines the case for multitask retrieval and that it is crucial to demonstrate competitive performance. Our main contribution is producing this demonstration.

We emphasize that achieving competitive multitask retrieval in practice is a highly difficult empirical problem. One might think that it is simply an application of multitask learning, which has no shortage of sophisticated techniques. These techniques typically modify the gradients during training, such as gradient surgery (Yu et al., 2020), gradient vaccine (Wang et al., 2020), common gradient descent (Piratla et al., 2021), and GradNorm (Chen et al., 2018). We experiment with these techniques and find that they do not help, thus motivating us to develop one that works.

Our technical contribution is a new method for multitask learning based on the notion of task sensitivity. Given a loss function $J(\theta)$, the sensitivity of the i-th parameter to the loss at θ is defined as the absolute change in the loss when θ_i is set to zero, which can be approximated by a first-order Taylor approximation as

$$|J(\theta) - J(\theta_{-i})| \approx \left| \frac{\partial J(\theta)}{\partial \theta_i} \times \theta_i \right|$$

where θ_{-i} is equal to θ except that its i-th element is zero. This quantity has been used in the context of model pruning—as a way of identifying weakly sensitive weights (Molchanov et al., 2016, 2019; Michel et al., 2019; Liang et al., 2021) and updating them more aggresively (Liang et al., 2022). In contrast, we use the quantity to identify weights that are strongly sensitive to a particular task and increase their sensitivity even further, intuitively to achieve per-parameter task specialization. To our knowledge, we are the first to use parameter sensitivity for multitasking learning.

We briefly differentiate our work from other recent work on multitask retrieval. Chen et al. (2022) present CorpusBrain, an autoregressive multitask retriever trained in largely the same style as GENRE (De Cao et al., 2021) with excellent performance. Autoregressive retrieval has different pros and cons compared to dense retrieval which is our setting; it can be more memory and runtime efficient, but it does not ‘‘read’’ the description of the target and thus not suited for retrieval tasks that require involved reasoning over query-target pairs (e.g., zero-shot entity retrieval [Logeswaran et al., 2019]). Thus we consider the contribution of CorpusBrain to be at least partially orthogonal to ours. Nevertheless, we show that our model outperforms CorpusBrain in a similar training setting in experiments. Asai et al. (2022) propose instruction-based retrieval in which the retriever is given an intent as well as a query to find the intended target. While this is a form of multitask retrieval, the problem formulation is different and it is evaluated on its own dataset benchmark.

3 Method

We build on the well-established framework of dual encoder (Bromley et al., 1993; Huang et al., 2013; Gillick et al., 2019; Karpukhin et al., 2020, *inter alia*). Let $\mathcal{X}$ denote the set of all queries and $\mathcal{Y}$ the set of all targets (i.e., KB). First, we assume mappings $\text{text}_X : \mathcal{X} \to \mathcal{V}^+$ and $\text{text}_Y : \mathcal{Y} \to \mathcal{V}^+$ where $\mathcal{V}$ denotes the vocabulary to ‘‘verbalize’’ queries and targets. Second, we assume encoders $\text{enc}_X^\theta, \text{enc}_Y^\theta : \mathcal{V}^+ \to \mathbb{R}^d$ with parameters θ defining the relevance score function $\text{s}_\theta(x, y) = \left\langle \text{enc}_X^\theta(\text{text}_X(x)), \text{enc}_Y^\theta(\text{text}_Y(y)) \right\rangle$. Third, assuming iid samples $(x_1, y_1) \ldots (x_N, y_N) \sim$ **pop**, we learn the parameters by noise contrastive estimation (NCE):

$$\min_\theta \; -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\text{s}_\theta(x_i, y_i))}{\sum_{y \in \mathcal{Y}_i} \exp(\text{s}_\theta(x_i, y))}$$

where $\mathcal{Y}_i \subset \mathcal{Y}$ satisfying $y_i \in \mathcal{Y}_i$ is a set containing the gold and negative targets for the i-th

labeled example. We pre-encode every $y \in \mathcal{Y}$ to $v_y = \text{enc}_Y^\theta(\text{text}_Y(y))$ at test time and efficiently compute the highest scoring target $\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \left\langle \text{enc}_X^\theta(\text{text}_X(x)), v_y \right\rangle$ for any $x \in \mathcal{X}$ by maximum inner product search.

In multitask retrieval, there are K retrieval tasks, each with N_k training examples $(x_1^{(k)}, y_1^{(k)}) \ldots (x_{N_k}^{(k)}, y_{N_k}^{(k)}) \sim \textbf{pop}_k$ drawn iid from the k-th population distribution $\textbf{pop}_k$. We use the KILT benchmark, which includes $K = 8$ tasks addressing QA, entity linking, fact checking, slot filling, and dialogue.[2] The per-task loss is

$$J_k(\theta) = -\frac{1}{N_k} \sum_{i=1}^{N_k} \log \frac{\exp(\text{s}_\theta(x_i^{(k)}, y_i^{(k)}))}{\sum_{y \in \mathcal{Y}_i^{(k)}} \exp(\text{s}_\theta(x_i^{(k)}, y))}$$

defining the final loss

$$J(\theta) = \sum_{k=1}^{K} \frac{N_k}{N} \times J_k(\theta)$$

Previous work by Maillard et al. (2021) uses the following setting. The KB $\mathcal{Y}$ consists of 100-token disjoint Wikipedia passages. The text mappings $\text{text}_X, \text{text}_Y$ apply the BERT tokenizer to unmodified queries and passages. The encoders $\text{enc}_X^\theta, \text{enc}_Y^\theta$ are initialized with independent pretrained BERT-bases (uncased). The task-specific training datasets are downsampled to be of similar sizes. As in DPR, they train the model using hard negatives based on BM25, followed by one round of hard negative mining from the model (only on Natural Questions and TriviaQA in which verifying if a candidate negative is indeed incorrect is expedient).

We now describe the main sources of improvement that we achieve over the baseline multitask retriever: a better choice of the base model with appropriate prompting, and better optimization.

3.1 Base Model

We use a shared T5 to parameterize and initialize the query and passage encoder $\text{enc}^\theta = \text{enc}_X^\theta = \text{enc}_Y^\theta$. Specifically, we follow the ST5-EncDec architecture (Ni et al., 2021), which encodes any $z \in \mathcal{V}^+$ as

$$\text{enc}^\theta(z) = \text{T5.generate}(z,\ \text{length} = 1).\text{state}$$

(i.e., we run the T5-encoder on z, run the T5-decoder for 1 step from the special start symbol, and take the resulting hidden state prior to token prediction). In addition, we define the text mapping for queries $x \in \mathcal{X}$ in task k as

$$\text{text}_X(x) = \text{T5Tokenizer}(\pi_k \oplus \text{[SEP]} \oplus x)$$

where $\oplus$ is string concatenation, [SEP] is the special separation token, and π_k is a text prefix that indicates which task x is a query of. We use dataset names as prefixes (e.g., $\pi_1 =$ ''NQ''). The text mapping for passages $y \in \mathcal{Y}$ does not use prefixes, that is

$$\text{text}_Y(y) = \text{T5Tokenizer}(y)$$

This allows us to pre-encode passage embeddings at test time and retain the efficiency of the single-task dual encoder framework.

While simple, this choice is the most crucial component in our apporach to improving multitask retrieval. We take a model pretrained for multitasking and adopt the same prefix concatenation scheme for task adaptation, treating multitask retrieval as a continuation of the T5 training.

Interestingly, using task markers is reported to be not helpful in Maillard et al. (2021). This is likely because their base model, BERT, is not pretrained for multitasking. Another difference is that they use task markers to indicate the 5 task types (e.g., ''QA''), whereas we use fine-grained markers to indicate the 8 tasks (e.g., ''NQ''). While there are previous works that use T5 for dense retrieval (Ni et al., 2021), we are the first to exploit the multitasking component of T5 pretraining for multitask retrieval.

3.2 Adaptive Learning

For the k-th task, the linear approximation of $J_k(\theta)$ around $a \in \mathbb{R}^d$ is

$$J_k(\theta) \approx J_k(a) + \langle \nabla J_k(a), \theta - a \rangle$$

Let $\theta^{(t)}$ denote the parameter value at the t-th update in gradient-based training. For any $i = 1 \ldots d$, we define $\theta_{-i}^{(t)}$ to be equal to $\theta^{(t)}$ except

[2] We write ''task'' and ''dataset'' synonymously instead of distinguishing datasets from task types as done in some previous work. Thus KILT has 8 tasks and 5 task types.

that its i-th element is zero. The approximation of $J_k(\theta)$ around $a = \theta^{(t)}_{-i}$ at $\theta = \theta^{(t)}$ is

$$J_k(\theta^{(t)}) \approx J_k(\theta^{(t)}_{-i}) + \left\langle \nabla J_k(\theta^{(t)}_{-i}), \theta^{(t)} - \theta^{(t)}_{-i} \right\rangle$$
$$= J_k(\theta^{(t)}_{-i}) + \frac{\partial J_k(\theta^{(t)})}{\partial \theta_i} \times \theta^{(t)}_i$$

Rearranging and taking the absolute value, we have

$$\sigma^{(t)}_{i,k} = \left| \frac{\partial J_k(\theta^{(t)})}{\partial \theta_i} \times \theta^{(t)}_i \right| \approx \left| J_k(\theta^{(t)}) - J_k(\theta^{(t)}_{-i}) \right| \quad (1)$$

which is easily computable and can be viewed as measuring how sensitive the i-th parameter is with respect to the k-th task in the t-th iteration of training. We propose to use this quantity, previously used in the model pruning literature (Molchanov et al., 2016), to encourage task specialization during training. We define a conditional distribution over K tasks by

$$q(k|\theta^{(t)}, t, i) = \frac{\exp(\bar{\sigma}^{(t)}_{i,k}/\tau_t)}{\sum_{k=1}^{K} \exp(\bar{\sigma}^{(t)}_{i,k}/\tau_t)} \quad (2)$$

where $\tau_t > 0$ is a temperature and $\bar{\sigma}^{(t)}_{i,k}$ is an appropriately normalized and amortized estimation of $\sigma^{(t)}_{i,k}$ in Eq. (1) (see Section 3.2.1). Assuming training examples are sampled to roughly balance the size across tasks (i.e., $N_k \approx N_{k'}$), we take the following gradient step for the i-th parameter in the t-th iteration:

$$\theta^{(t+1)}_i = \theta^{(t)}_i - \eta \sum_{k=1}^{K} q(k|\theta^{(t)}, t, i) \times \frac{\partial J_k(\theta^{(t)})}{\partial \theta_i}$$

Note that this is a per-parameter adaptive learning. Each parameter $\theta_i \in \mathbb{R}$ maintains a distribution over K tasks and is updated more aggresively for tasks that θ_i is sensitive to.

3.2.1 Sensitivity Normalization

The d parameters $\theta^{(t)}$ can be of very different magnitudes. To reduce the parameter-wise variance in the sensitivity scores, for task k we divide the scores by the median of across all parameters with respect to task k:

$$\tilde{\sigma}^{(t)}_{i,k} = \frac{\sigma^{(t)}_{i,k}}{\text{median}_{j=1\ldots d}(\sigma^{(t)}_{j,k})}$$

We use the median instead of the mean to account for the long tail distribution of task-specific sensitivity scores. We also use momentum to amortize the scores: assuming some $\beta > 0$

$$\bar{\sigma}^{(t)}_{i,k} = (1-\beta)\bar{\sigma}^{(t-1)}_{i,k} + \beta\tilde{\sigma}^{(t)}_{i,k}$$

where $\bar{\sigma}^{(0)}_{i,k} = 0$. This is the final version of sensitivity that we use in Eq. (2). The algorithm in matrix form is given in Algorithm 1 (Appendix A).

4 Experiments

4.1 Setup

Datasets. We follow Maillard et al. (2021) and use eight tasks from KILT (Petroni et al., 2021) for training and evaluation. We randomly downsample the training data of the two largest datasets (T-REx and zsRE) to the same order of magnitude as the rest. All the datasets share the same knowledge base of 36 million disjoint 100-token Wikipedia passages preprocessed by Maillard et al. (2021). The data statistics and other data-related details can be found in Appendix B.

Evaluation. We use the page-level R-precision (the suggested main metric in KILT) to measure the retrieval performance. Page-level R-precision is the fraction of the R gold pages captured by the retriever in the top-R candidates. We map the retrieved passages to the their corresponding pages and use official KILT evaluation scripts to evaluate the page-level R-precision. We also report passage-level R-precision proposed by Maillard et al. (2021) on dev sets in Appendix E. We use TREC Eval[3] to evaluate the passage-level R-precision.

Model Details. We initialize our dual encoder with the official T5-base (Raffel et al., 2019) checkpoint. The query encoder and passage encoder share weights. Following the ANCE (Xiong et al., 2021) training paradigm, we first warmup our model for 20 epochs with BM25 hard negatives by naive multitask learning with task prefix. Then we train the model for 8 ANCE episodes with the model-mined hard negatives refreshed at the begining of each ANCE episode. We adopt naive multitask learning with task prefix for the first

[3] https://trec.nist.gov/trec_eval/.

	Fact Check.	Ent. L.	Slot Filling		Open Domain QA			Dial.	
Model	**FEV**	**AY2**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
Baselines.									
BM25*	50.13	3.47	58.60	66.43	25.83	43.95	29.44	27.50	38.17
$\text{BART}^{\dagger}_{mt}$	81.92	89.17	75.18	91.08	58.62	48.69	67.64	50.98	70.41
$\text{CorpusBrain}^{\dagger}_{mt}$	82.06	**90.84**	77.62	98.26	59.10	50.07	68.78	53.75	72.56
MT-DPR*	74.72	83.78	69.18	77.23	61.51	44.21	61.95	39.70	64.04
Task-specific DPR*	73.60	81.77	69.08	97.74	63.24	46.63	65.12	40.32	67.19
Task-specific BART†	80.03	87.98	74.46	93.91	50.96	39.21	66.13	50.75	67.93
Task-specific CorpusBrain†	81.77	90.36	76.90	98.49	57.67	**50.62**	69.25	53.60	72.33
Task-specific (ours)	74.28	85.28	77.18	**99.38**	**65.39**	46.79	69.08	53.63	71.38
Non-Comparable Models (For Reference).									
$\text{CorpusBrain}^{\dagger}_{mt+BLINK}$	85.03	92.86	80.22	98.49	64.61	52.23	71.71	59.72	75.61
GENRE†	84.68	92.75	79.68	94.84	64.26	51.82	71.11	56.32	74.43
TABi (Leszczynski et al., 2022)	85.8	–	82.0	95.2	62.4	52.7	71.5	51.8	–
TACO	**86.17**	84.64	**78.12**	97.91	61.86	50.61	**69.62**	**60.97**	**73.74**

Table 1: Page-level R-precision on KILT validation data. **Bold** indicates the best model and underline indicates the second. † and ∗ mark results from Chen et al. (2022) and Maillard et al. (2021), respectively. The non-comparable models are trained on additional data or use extra information. We list them only for reference, not for comparison. Task-specific models use a separate retriever for each task while all the other models use a single retriever across all the tasks.

7 ANCE episodes and apply the adaptive learning introduced in Section 3.2 for the last ANCE episode to improve the performance further. We use Adam (Kingma and Ba, 2015) with a linear learning rate decay schedule with warmup proportion 0.1 over 3 epochs for each ANCE iteration. We provide more details and hyperparameters in Appendix C.

4.2 Main Results

We refer to our model as **TACO**, which stands for **TA**sk spe**C**ialty **O**ptimization. Table 1 and Table 2 show our main results on the KILT validation data and test data respectively. Fewer comparable baselines are available for KILT test data than for KILT validation data.

Let avg val denote average validation page-level R-Precision. TACO achieves the best performance on 4 out of 8 tasks for both validation and test data. The performance is either the second best or close to the second best except AIDA, an entity linking dataset favoring autoregressive retrieval models over dense retrieval models (De Cao et al., 2021). TACO outperforms the previous multitask dense retrieval model MT-DPR (Maillard et al., 2021) significantly (+7.34% avg val). TACO also achieves better performance compared with current top performing multitask autoregressive retrieval models in comparable setting (finetuned purely on KILT). TACO outperforms BART_{mt} (+3.33% avg val) with smaller model size (T5-base vs Bart-large). Compared with BART_{mt}, CorpusBrain_{mt} employs additional pretraining and yields significant improvement over BART_{mt} (+2.15% avg val). TACO still outperforms CorpusBrain_{mt} (+1.18% avg val) with smaller model size and no additional pretraining. We also list various top performing multitask retrieval models for reference but not for comparison because they are not in comparable setting. Both GENRE and $\text{CorpusBrain}_{mt+BLINK}$ are finetuned on a large amount of additional training data besides KILT training data. Specifically, they also use BLINK training data (Wu et al., 2020) for finetuning, which contains 8.9M annotated wikipedia sentences. TABi (Leszczynski et al., 2022) uses extra type labels information and leverages knowledge graph that is very effective for retrieval. TACO even rivals these non-comparable models on all the tasks except AIDA.

	Fact Check.	Ent. L.	Slot Filling		Open Domain QA			Dial.	
Model	**FEV**	**AY2**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
Baselines.									
TF-IDF†	50.9	3.7	44.7	60.8	28.1	34.1	46.4	49.0	39.7
SEAL‡	81.4	–	62.1	91.6	**63.2**	**58.8**	68.4	57.5	–
MT-DPR*	74.5	26.5	69.5	80.9	59.4	42.9	61.5	41.1	57.0
MT-DPR$^{\ddagger}_{WEB}$	74.8	–	75.6	89.7	59.8	45.4	58.9	41.5	–
Task-specific (ours)	73.22	79.52	77.00	**99.15**	60.87	46.50	**69.12**	55.03	70.05
Non-Comparable Models (For Reference).									
CorpusBrain$^{\dagger}_{mt+BLINK}$	84.07	89.98	79.98	98.27	60.32	51.80	70.19	64.79	74.93
GENRE†	83.64	89.85	79.42	95.81	60.25	51.27	69.16	62.88	74.04
TABi (Leszczynski et al., 2022)	84.4	–	81.9	96.2	62.6	53.1	70.4	59.1	–
TACO	**84.07**	**80.64**	**77.22**	98.21	60.80	50.70	68.45	**62.64**	**72.84**

Table 2: Page-level R-precision on KILT test data. **Bold** indicates the best model and underline indicates the second. †, ∗, and ‡ mark results from Chen et al. (2022), Maillard et al. (2021), and Bevilacqua et al. (2022), respectively. The non-comparable models are trained on additional data or use extra information. We list them only for reference not for comparison.

	Fact Check.	Ent. L.	Slot Filling		Open Domain QA			Dial.	
Variants	**FEV**	**AY2**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
TACO	**86.17**	84.64	**78.12**	**97.91**	61.86	50.61	**69.62**	**60.97**	**73.74**
w/o task prefix	85.71	84.68	74.82	94.68	61.05	49.38	67.79	58.81	72.12
w/o adaptive	84.81	85.49	75.00	92.24	62.81	51.47	68.95	60.54	72.66
w/o task prefix w/o adaptive	84.03	85.62	70.96	86.04	62.46	49.78	66.04	59.95	70.61
task query encoder	82.71	**87.56**	72.72	85.15	**64.01**	49.74	69.12	55.93	70.87
task type marker	84.49	85.51	73.88	89.37	62.85	50.97	67.70	60.02	71.85
PCG (Yu et al., 2020)	84.97	85.26	74.90	91.43	62.67	51.47	68.54	60.48	72.47
CGD (Piratla et al., 2021)	82.25	80.39	71.62	83.40	62.67	49.66	66.73	59.33	69.51
GradNorm (Chen et al., 2018)	84.70	85.28	75.32	91.73	63.80	**51.97**	69.30	60.31	72.80

Table 3: Ablation study results on KILT validation data. We report page-level R-precision. **Bold** indicates the best variant. Each line makes a single or multiple changes from the TACO model. The performance of the recent general multitask algorithms, PCG (Yu et al., 2020), CGD (Piratla et al., 2021), and GradNorm (Chen et al., 2018), are obtained from our own implementation.

TACO is the only model that outperforms strong task-specific models noticeably. Our task-specific baseline is significantly stronger than the task-specific DPR, likely due to better training paradigm (ANCE) and better model (T5 vs BERT). Task-specific CorpusBrain is even stronger, especially for FEVER and AIDA. Only TACO and CorpusBrain$_{mt}$ outperform the strong task-specific models. TACO achieves a 2.36% improvement over its task-specific counterpart and a 1.41% improvement over the task-specific CorpusBrain, but CorpusBrain$_{mt}$ is only slightly better than its task-specific counterpart (+0.23% avg val).

4.3 Analysis

4.3.1 Ablation Study

Table 3 shows the results of ablation studies on KILT validation data.

Model Components. We first conduct experiments to understand the impact of individual components of our model. Removing task prefix

results in 1.62% R-precision decrease and disabling adaptive learning yields 1.08% R-precision decrease. Removing both task prefix and adaptive learning significantly degrades the performance (−3.13%). This demonstrates that both task prefix and adaptive learning contribute to the effectiveness of TACO.

Query Variants. We conduct experiments to investigate other query side variants besides task prefix. These variants are not trained with adaptive learning and only change the query input format or model. Leveraging task-specific query encoder yields slightly better performance (70.87% vs 70.61%), but is outperformed by task prefix significantly (70.87% vs 72.66%). The task type marker introduced in Maillard et al. (2021) is not helpful for BERT-based MT-DPR, but we find them effective for our T5-based model. This is likely because T5 is pretrained for multitasking. We conduct experiments to leverage their task type markers for our model. Using task type markers (i.e., 5 symbols indicating the 5 classes of task in KILT) leads to 1.24% R-precision improvement (71.85% vs 70.61%), but is less effective than our fine-grained dataset-level task prefix (71.85% vs 72.66%).

Mutltitask Learning Variants. We compare our adaptive learning method with recent general multitask learning algorithms with our own implementation. PCG (Yu et al., 2020) focuses on mitigating the conflict of gradients from different tasks. It performs on par with the ''w/o adaptive'' variant (72.47% vs 72.66%), but underperforms TACO which leverages our adaptive learning (72.47% vs 73.74%). This shows that the gradient conflict is not the main bottleneck in our multitask retrieval setting. CGD (Piratla et al., 2021) aims to improve multitask learning by encouraging update towards common directions of different tasks, which is opposite to our method that encourages task specialties. It performs much worse than TACO (69.51% vs 73.74% and lags behind the ''w/o adaptive'' variant significantly (69.51% vs 72.66%). This shows that we should encourage task specialty rather than emphasizing tasks shared part for multitask retrieval. GradNorm (Chen et al., 2018) tries to weight different tasks losses by using the average gradient norm. It performs slightly better than the naive ''w/o adaptive'' variant (72.47% vs 72.66%). Our adaptive learning method achieves descent improvement over GradNorm (73.74% vs 72.80%). Note that our adaptive update is more fine-grained and critically different because we adjust learning rates along both task dimension and parameter dimension compared with GradNorm that only do loss re-weighting.

Adaptive Learning. We consider variations of the main version of adaptive learning which is applied only in the last ANCE episode. Specifically, we investigate the impact of applying adaptive learning to the last four ANCE episodes using an exponential softmax temperature decay scheduler. This approach yields an average page-level R-precision of 73.47%. In comparison, when adaptive learning is applied only to the last ANCE episode, we achieve an average page-level R-precision of 73.74%. These results suggest that extending adaptive learning to more ANCE episodes does not yield improvement. Additionally, we examine the effectiveness of encouraging task specialization within adaptive learning. For this purpose, we focus on the second ANCE episode and experiment with positive softmax temperature (encouraging task specialty) and negative softmax temperature (discouraging task specialty). Encouraging task specialization results in an average page-level R-precision of 70.53%, while discouraging task specialization leads to an average page-level R-precision of 68.39%. In comparison, the performance of the standard multitask baseline at the second ANCE episode is 69.28%. These results highlight the benefits of encouraging task specialization and the detrimental effect of discouraging task specialization within adaptive learning. Normalizing task sensitivity using the median is preferred over using the mean or not applying any normalization, as different tasks exhibit variations in magnitude while sharing similar distribution shapes (see Figure 2).

4.3.2 Task Specialization

Figure 1 plots the histograms of task entropy for the learned parameters. The task entropy for each parameter is calculated with the distribution defined in Equation 2. We first group parameters into two special bins. The first is a ''Task Specific'' bin that includes parameters whose entropy is smaller than 0.3, which is the entropy of 95% probability on one task and the 5% uniformly

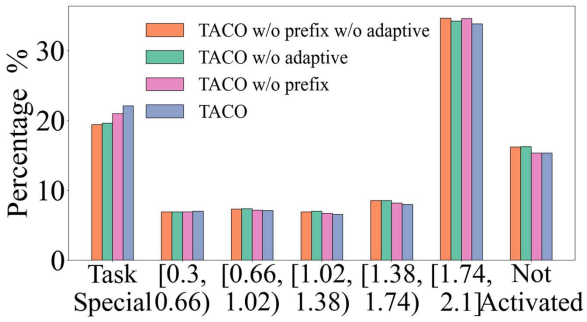

Figure 1: Task entropy histograms for model variants.

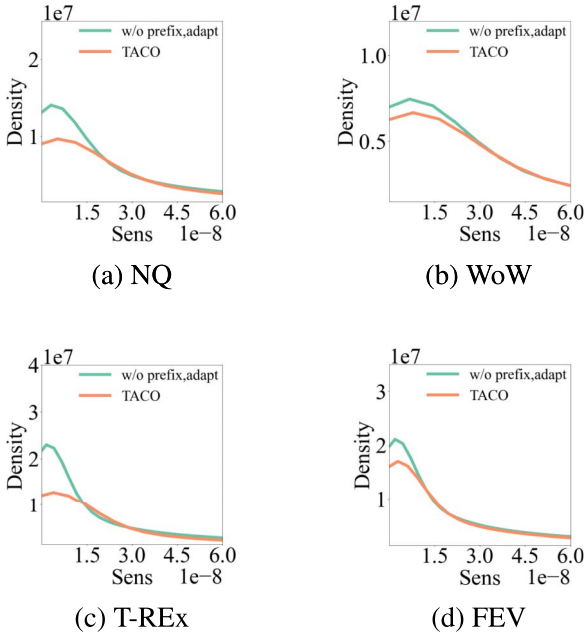

Figure 2: Task-specific sensitivity density distribution on the training data of four KILT tasks. The final models are used. The *x*-axis is sensitivity, and we drop outliers that are far from the median to ease visualization.

on the rest seven. The ''Not Activated'' bin includes parameters whose sensitivity w.r.t. all tasks is near zero ($< 1e-8$). TACO significantly improves the fraction of task specific parameters to 22%, in comparison with 19% in naive multitask model (w/o prefix w/o adaptive). It also reduces the fraction of not activated parameters, showing optimizing task specialty also better utilizes the model capacity.

Figure 2 plots the kernel density estimated distribution of task-specific sensitivity in TACO and the standard multitask model for four KILT tasks. We drop outliers that deviates significantly from the median to ease visualization. Notably, TACO exhibits a noticeable reduction in the peak on the low sensitivity side for each task compared to the standard multitasking model. This observation suggests that TACO activates a larger number of parameters and enhances their sensitivity towards individual tasks.

	MS	ZES	FEV	NQ	Avg
Task-specific	73.3	67.3	90.0	71.8	75.6
TACO	85.8	67.6	91.2	76.8	80.4
w/o adapt	85.9	67.5	91.3	76.6	80.3
w/o prefix, adapt	86.2	68.1	92.2	76.4	80.7
PCG	86.1	67.9	91.7	76.9	80.7
CGD	86.8	69.1	94.4	76.2	81.6
GradNorm	86.0	67.3	91.7	76.9	80.5

Table 4: Recall@100 on an additional benchmark containing MS-MARCO (MS), ZESHEL (ZES), FEVER (FEV), and Natural Questions (NQ).

4.3.3 Additional Benchmark

To test the performance of TACO in a different setup other than KILT, we constructed an additional benchmark containing MS-MARCO (Nguyen et al., 2016), ZESHEL (Logeswaran et al., 2019), a document-level version of FEVER from BEIR (Thakur et al., 2021), and Natural Questions from KILT. We chose this combination for a few reasons. First, we found that few public datasets outside KILT provide sufficiently large and high-quality training data other than MS-MARCO and ZESHEL. Second, each task now has its own KB to retrieve from, making this a rather different setup from KILT in which all tasks share one KB. We compare task-specific retrievers and multitask retrievers trained by TACO and other methods. Table 4 shows their recall at 100 on the validation split. We see that multitasking is clearly beneficial for this benchmark. The best performance is obtained by CGD and it is the only multitask optimization method that yields noticeable improvements over the standard multitask model. Given that CGD aims to improve multitask learning by encouraging update towards common directions of different tasks, we hypothesize that the need for task specialization is diminished here because the tasks are more similar in difficulty (e.g., in KILT, T-REx and zsRE are much easier than HotpotQA). This experiment sheds light on what multitask settings most benefit from task specialization.

5 Conclusions

Multitask retrieval has compelling practical advantages such as model simplicity and memory efficiency, but it lags behind task-specific retrieval in the existing literature. We have shown that it is possible to significantly improve the performance of multitask retrieval by promoting task specialization. The key steps are the use of a base model optimized for multitasking with appropriate prompting and a per-parameter adaptive learning technique that upweights the task gradients by the parameters' sensitivity to the task losses. We have achieved strong results on the KILT retrieval benchmark.

References

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*.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. *arXiv preprint arXiv:2204.10628*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a ''Siamese'' time delay neural network. *Advances in Neural Information Processing Systems*, 6. https://doi.org/10.1142/S0218001493000339

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. *arXiv preprint arXiv:2208.07652*. https://doi.org/10.1145/3511808.3557271

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186. https://doi.org/10.18653/v1/N19-1423

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/K19-1049

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2333–2338. https://doi.org/10.1145/2505515.2505665

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. https://doi.org/10.18653/v1/2020.emnlp-main.550

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Megan Leszczynski, Daniel Fu, Mayee Chen, and Christopher Ré. 2022. Tabi: Type-aware bi-encoders for open-domain entity retrieval. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2147–2166. https://doi.org/10.18653/v1/2022.findings-acl.169

Chen Liang, Haoming Jiang, Simiao Zuo, Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. 2022. No parameters left behind: Sensitivity guided adaptive learning rate for training large transformer models. In *International Conference on Learning Representations*.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. *arXiv preprint arXiv:2105.12002*. https://doi.org/10.18653/v1/2021.acl-long.510

Zhenghao Liu, Kaitao Zhang, Chenyan Xiong, Zhiyuan Liu, and Maosong Sun. 2021. Openmatch: An open source library for neu-ir research. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2531–2535. https://doi.org/10.1145/3404835.3462789

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460. https://doi.org/10.18653/v1/P19-1335

Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau Yih, Barlas Oguz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-task retrieval for knowledge-intensive tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1098–1111. https://doi.org/10.18653/v1/2021.acl-long.89

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272. https://doi.org/10.1109/CVPR.2019.01152

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

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. *Choice*, 2640:660.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. Kilt: A benchmark for knowledge intensive language tasks. In *NAACL-HLT*. https://doi.org/10.18653/v1/2021.naacl-main.200

Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. 2021. Focus on the common good: Group distributional robustness follows. *arXiv preprint arXiv:2110.02619*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

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*.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2020. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

A Algorithm in Matrix Form

Algorithm 1 is the matrix form of our adaptive learning algorithm.

B Data Details

See Table 5 for data statistics and some data-related hyperparameters. We randomly downsample T-REx and zsRE to bring them to the same order of magnitude as the others. We follow Raffel et al. (2019) and use temperature-scaled mixing sampling strategy to compute batch size for each task k: $B_k \propto (N_k / \sum_{k'=1}^{K} N_{k'})^{1/c}$ for some temperature c (we set it to 4 in our experiments). Here N_k is the dataset size of task k. Note that we compute task loss of each task batch independently instead of mixing all task batches for every optimization step. Each dataset needs to sample a different number of batches to cover every training sample in that dataset once. We set the maximum as the number of batches that every dataset needs to sample. We shuffle and cycle batch sampling iterators of datasets that finish iterating early. Batch size of each dataset computed by setting mixing temperature $c = 4$ and $\sum_{k'=1}^{K} N_{k'} = 120$ is in Table 5.

C Other Training Details

The data-related hyperparameters, such as maximum input query length and batch size, are listed in Table 5. The training hyperparameters are listed in Table 6. We use NCE loss with cross device in-batch negative mixed with hard negatives to compute each task loss. We sample two hard negatives for each query. We use a ‘‘burn in’’ period for the first 10% training steps with uniform learning rates for parameters to declare their tendency during adaptive learning. All of our experiments are run on a machine with 8 A100-80GB GPUS. Our implementations are built upon OpenMatch (Liu et al., 2021).

Dataset	#Train	B	L
Natural Questions	76k	16	32
TriviaQA	53k	14	32
HotpotQA	69k	15	32
Wizard of Wikipedia	80k	16	256
T-REx	95k	16	32
FEVER	71k	15	64
Zero Shot RE	100k	17	32
AIDA-YAGO 2	18k	11	128

Table 5: Data statistics and some data-related hyperparameters for our experiments. B denotes batch size. L denotes query maximum input length excluding the task prefix.

lr	warmup	#negs	epochs	τ	β	B_{total}
5e-6	0.1	2	3	2	0.999	120

Table 6: Training hyperparameters for training our TACO-DR model. We use Adam (Kingma and Ba, 2015) with learning rate $5e-6$. We use linear learning rate schedule with warmup ratio 0.1. Each query uses 2 hard negatives for training. Each ANCE episode trains for 3 epochs. Total batch size of all task batches is 120.

τ	0.1	1	2	5	10	100
Avg R-prec	72.45	73.23	73.74	73.72	73.48	72.85

Table 7: Average page-level R-precision w.r.t softmax temperature for our adaptive learning.

β	0	0.6	0.7	0.8	0.9	0.999
Avg R-prec	72.91	72.98	73.02	73.16	73.61	73.74

Table 8: Average page-level R-precision w.r.t momentum factor for our adaptive learning.

D Softmax Temperature and Momentum Ratio

Table 7 shows the impact of softmax temperature on validation R-precision for our adaptive learning. Table 8 shows the impact of momentum

Algorithm 1 Task sensitivity-guided adaptive learning

Require: Model parameter $\theta \in \mathbb{R}^d$; minibatches $\mathcal{B}$ where each batch $B \in \mathcal{B}$ is further divided by tasks $B = \{B_k\}_{k=1\ldots K}$; moving average rate $\beta \in [0, 1]$; temperature $\tau > 0$; learning rate $\eta > 0$

Ensure: median : $\mathbb{R}^{d\times K} \to \mathbb{R}^K$ is the column-wise median; softmax : $\mathbb{R}^{d\times K} \to \mathbb{R}^{d\times K}$ is the row-wise softmax; $\mathbf{1}_K$ is a vector of K ones; $\odot$ is the Hadamard product.

Initialize $I \leftarrow \mathbf{0} \in \mathbb{R}^{d\times K}$.
for each batch $B = \{B_k\}_{k=1\ldots K}$ in $\mathcal{B}$ **do**
    Compute the task-specific loss $J_k(\theta)$ on B_k for each $k = 1 \ldots K$.
    Compute the gradient matrix $G \in \mathbb{R}^{d\times K}$ with each column $G_k \leftarrow \nabla J_k(\theta)$.
    Compute the sensitivity matrix $I' \in \mathbb{R}^{d\times K}$ with each column $I'_k \leftarrow G_k \odot \theta$.
    Normalize the sensitivity scales across tasks $I' \leftarrow I'\text{diag}\left(\text{median}(I')\right)^{-1}$.
    Update the moving average $I \leftarrow \beta I + (1 - \beta)I'$.
    Update the parameter $\theta \leftarrow \theta - \eta(G \odot U)\mathbf{1}_K$ where $U \leftarrow \text{softmax}(I/\tau) \in \mathbb{R}^{d\times K}$.
end for

factor on validation R-precision for our adaptive learning.

E Passage-level Performance

Table 9 shows the passage-level R-precision on KILT validation data. We also list the passage-level performance from Maillard et al. (2021) for comparison.

	Fact Check.	Slot Filling		Open Domain QA			Dial.	
Model	**FEV**	**T-REx**	**zsRE**	**NQ**	**HoPo**	**TQA**	**WoW**	**Avg**
MT-DPR*	<u>46.96</u>	53.54	41.70	28.80	38.42	24.56	24.07	36.86
Task-specific DPR*	43.92	58.54	78.81	28.13	<u>43.47</u>	23.79	20.73	42.48
Task-specific (ours)	44.89	<u>72.09</u>	**84.47**	**33.14**	43.40	**29.57**	<u>27.64</u>	<u>47.89</u>
TACO	**60.76**	**72.57**	<u>82.80</u>	<u>31.16</u>	**46.72**	<u>28.32</u>	**33.24**	**50.80**

Table 9: Passage-level R-precision on KILT validation data. **Bold** indicates the best model and <u>underline</u> indicates the second. $*$ marks results from Maillard et al. (2021). Only page-level R-precision is defined for AIDA.