RA-DIT：检索增强的双重指令调整

# **太长不看版**

RA-DIT是一种创新的微调框架，它通过两个关键步骤为大型语言模型（LLM）增强检索能力：首先，对LLM进行微调以更好地整合检索信息；其次，调整检索器以返回LLM更倾向于选择的相关结果。在实验中，RA-DIT在多个知识密集型任务上展现了显著的性能提升。特别是在零次学习设置中，RA-DIT 65B模型相较于现有方法平均提升了8.9%，在少次学习设置中提升了1.4%。实验还涉及了对微调策略、检索器配置以及不同规模LLMs的深入分析，验证了RA-DIT在提升模型性能方面的有效性。这些实验结果支持了RA-DIT作为一种高效且灵活的方法，能够在不牺牲LLM原有知识的情况下，通过指令调整显著提升其性能。

# **摘要**

检索增强的语言模型（RALMs）通过访问外部数据存储中的长尾和最新知识来提高性能，但构建起来具有挑战性。

现有的方法要么需要对 LM 预训练进行高成本的检索特定修改，要么使用数据存储的事后集成，导致性能不佳。我们介绍了检索增强的双重指令调整（RA-DIT），这是一种轻量级微调方法，通过改装任何大型语言模型（LLM）为之提供检索能力，提出了第三种选择。

我们的方法在两个不同的微调步骤中操作：

（1）更新预训练的 LM，以更好地使用检索到的信息

（2）更新检索器，以返回 LM 更喜欢的（preferred by LM）更相关结果。

通过在需要知识利用和上下文意识的任务上进行微调，我们展示了每个阶段都能显著提高性能，两者结合使用可以带来额外的收益。我们最好的模型，RA-DIT 65B，在一系列知识密集型的零次和少次学习基准测试中实现了最先进的性能，在 0 次设置中平均比现有的上下文 RALM 方法提高了 8.9%，在 5 次设置中提高了 1.4%。

# 1 引言

LLMs当前缺陷（由于知识仅在模型参数中表示）：

1 难以捕捉长尾知识

2 需要大量资源来更新

RALM通过将 LLMs 与非参数信息检索集成来克服这些限制。通过明确将知识检索与骨干（backbone）语言模型分离，这些架构在知识密集型任务（如开放域问答和实时聊天交互）上表现出优越的性能。

现有的 RALM 开发工作主要集中在两个高层次挑战上：

1. 增强 LLM 对于检索到的知识的整合能力
2. 完善检索组件，以返回更相关的内容

检索能力也在模型训练过程的不同阶段引入。REALM和 RETRO选择端到端预训练，从一开始就整合检索组件。ATLAS基于 T5 语言模型，并持续预训练框架以处理无监督文本。REPLUG和上下文 RALM将现成的 LLMs 与通用检索器结合，展示了即使独立优化的 LLMs 和检索器也可以通过 LLMs 的新兴上下文学习能力有效地融合。

然而，这种架构的广泛预训练会产生高昂的计算成本，而现成的融合方法也有局限性，特别是当 LLMs 没有被训练来整合检索内容时。

在这项工作中，我们展示了单独的轻量级指令调整可以显著提高 RALMs 的性能，特别是在需要访问大型外部知识源的场景中。我们提出了检索增强的双重指令调整（RA-DIT），这是一种通过在一组任务上进行微调来为任何 LLM 改装上检索能力的的方法，这些任务旨在培养语言模型预测中的知识利用和上下文意识。我们使用预训练的 LLAMA和最先进的基于双编码器的密集检索器 DRAGON+初始化框架。我们根据语言模型提示检索相关文本块。每个检索到的块都附加到提示前面，并且从多个块并行计算的预测被集成以产生最终输出。

我们分两个独立的步骤进行指令调整。对于语言模型微调（LM-ft），我们采用受监督的微调目标，同时在每个微调提示中增加一个检索到的“背景”字段，该字段附加到指令前面。

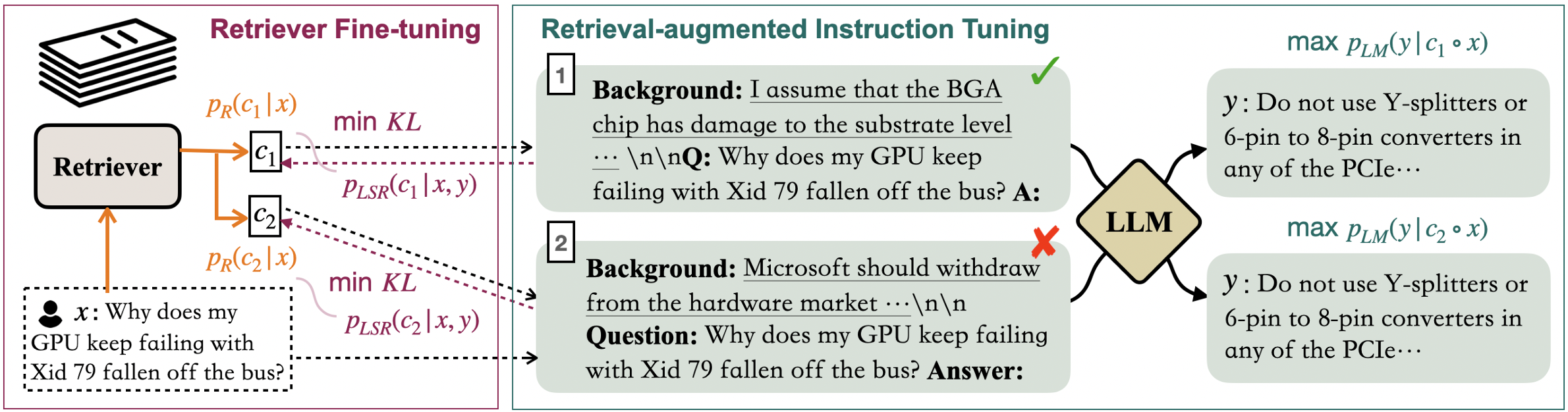


图 1：RA-DIT 方法分别微调 LLM 和检索器。

对于给定的示例，LM-ft 组件更新 LLM，以最大化在检索增强指令下正确答案的可能性（§2.3）；

R-ft 组件更新检索器，以最小化检索器得分分布与 LLM 偏好之间的 KL-散度（§2.4）。

我们还利用现有 NLP 任务的设计，并用这些任务的地面真实上下文填充这个字段，例如阅读理解和摘要任务。通过在微调期间整合背景文本，我们引导 LLM 最优地利用检索到的信息并忽略干扰内容。对于检索器微调（R-ft），我们使用一个广义的 LM-Supervised Retrieval训练目标，该目标在受监督任务和无监督文本完成的组合上计算。这样，我们使检索器能够产生更符合 LLM 偏好的上下文相关结果。

我们展示了每个微调步骤都能提供显著的性能提升，并且微调后的 LLM 和检索器可以结合使用以实现进一步的改进。我们最大的模型 RA-DIT 65B，在零次和少次设置的知识密集型基准测试中达到了最先进的性能，显著超越了未调整的上下文 RALM 方法，特别是在包括 MMLU（Hendrycks et al., 2021b）（+8.2% 零次；+0.7% 五次）和 Natural Questions（Kwiatkowski et al., 2019）（+22% 零次；+3.8% 五次）的数据集上。此外，RA-DIT 65B 还在 8 个知识密集型任务上大幅超越了 ATLAS 11B（在 64 次微调设置中平均提高了 7.2%）。

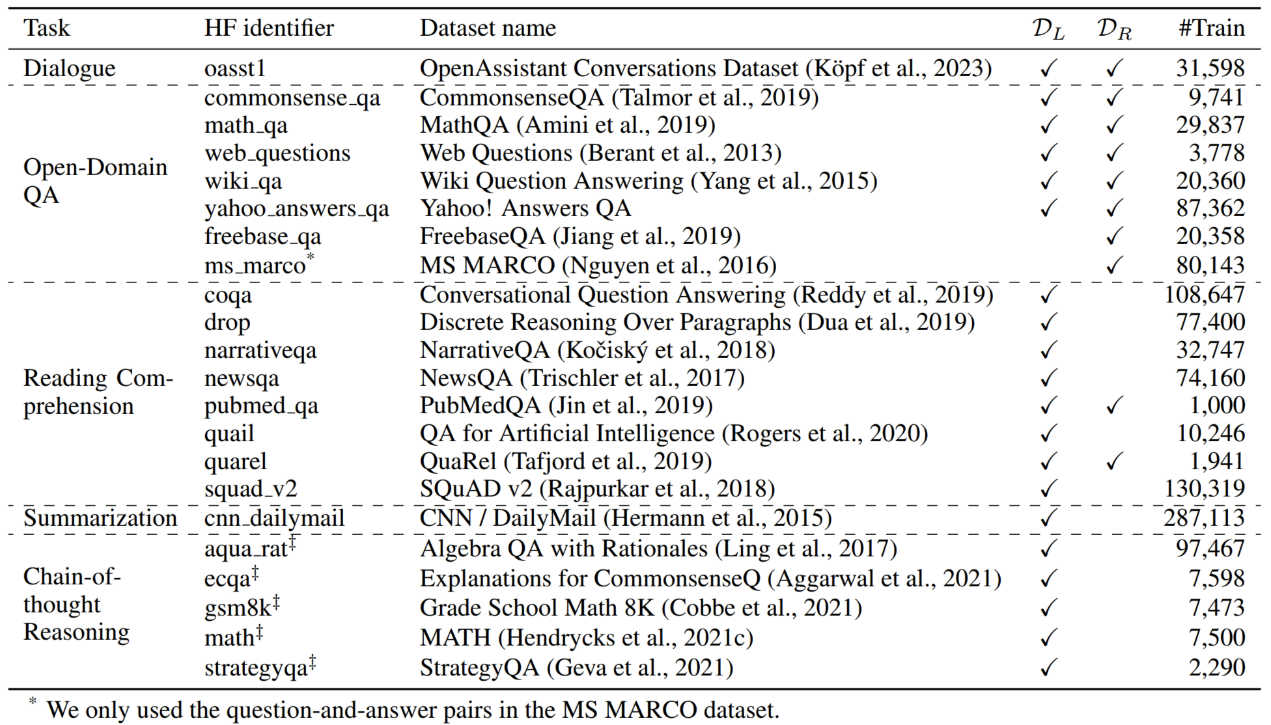
这表明，当独立优化并通过指令调整融合时，语言模型和检索器可以有效地与经过广泛连续预训练的 RALM 竞争。我们进一步进行了全面的模型分析，展示了我们的方法在不同大小的 LLMs 上的有效性，以及评估了不同的微调策略和检索器配置的影响。

# 2 方法

## 2.1 架构

**语言模型** 我们专注于检索增强预训练的自动回归语言模型。特别地，我们使用了LLAMA，这是一个开源的语言模型系列。语言模型系列。

表 1：我们的结构调整数据集。



**检索器** 我们采用基于双编码器的检索器架构，因为这种架构易于微调，而且在推理阶段效率很高。给定语料库 C 和查询 q，文档编码器将每个文本块 c∈C 映射到嵌入 Ed(c)，查询编码器将 q 映射到嵌入 Eq(q)。根据查询-文档嵌入相似度（通常通过点积计算），检索出 q 的前 k 个相关文本块：

我们使用了DARGON+（采用对比学习目标和大规模数据增强技术训练出的最先进的双编码器模型）初始化了检索器。

**并行的上下文检索增强** 对于一个给出的语言模型提示x，我们检索前k个相关的文本块，。为了不超出上下文窗口大小的限制，每个检索到的语块都会被分次添加到提示语中，而来自多个增强提示语的语言模型预测则会被并行计算。 最终的输出概率是每个增强提示的概率与语块相关性得分的混合物

代表顺序连接，则为检索器得分在前k个相关的块中重正化。

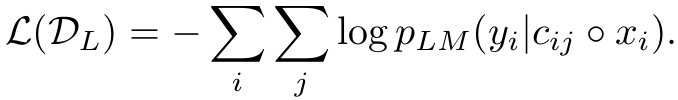
## 2.2 微调数据集

我们选择了一组旨在提升语言模型有效利用知识并提高生成预测时上下文意识的任务来进行微调。如表1所示，我们的语言模型微调数据集（DL）由20个数据集组成，涵盖5个不同类别：对话、开放域问答、阅读理解、摘要和思维链推理。

## 2.3 检索增强语言模型微调

为了提高语言模型（LM）利用检索到的信息的能力，我们对选定的数据集DL进行了上下文检索增强的微调。具体来说，我们将每个微调序列分为指令部分（x）和输出部分（y）。对于数据集DL中的每个例子（），我们基于检索出最相关的文本块 ⊂ C。在推理时，我们会将每个检索到的块添加到指令前，形成多个增强后的提示，然后并行计算这些增强提示的LM预测。最终的输出概率是每个增强提示的概率，根据块的相关性得分加权混合得到。

我们采用下一个标记预测目标对LM进行微调，并最小化每个实例输出部分的损失：

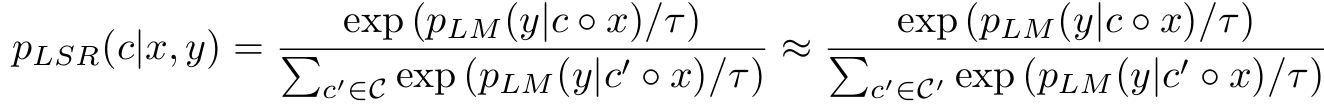


在微调过程中整合上下文检索增强有两个主要好处。首先，它使大型语言模型（LLM）能够更好地利用相关的背景知识来做出预测。其次，即使是最先进的检索器也可能出错并返回不准确的结果。通过训练LM在给出错误检索块时仍能做出正确预测，我们使LM能够在这种情况下忽略误导性的检索内容，并依赖其参数知识。这种微调策略的有效性在第5.1节中得到了实证验证。

## 2.4 检索器微调

除了对语言模型进行检索增强的微调外，我们还对检索器进行了微调，以使其输出更好地与语言模型对齐。具体来说，我们采用了一种通用的LM-Supervised Retrieval训练目标，该目标利用语言模型本身为检索器微调提供监督。

对于检索器微调数据集DR中的训练样本（x, y），我们定义检索块c的LSR分数如下：



其中τ是一个温度超参数，C′⊂ C表示x的前k个检索到的块。LSR分数越高，表示c在提高语言模型预测正确答案的可能性方面越有效。LSR训练的目标是让检索器为那些能够提高LLM生成正确答案可能性的块分配更高的分数。为了实现这一目标，我们最小化pLSR和检索器分数pR（定义在方程2中）之间的KL散度：



在实践中，我们只更新检索器的查询编码器，因为同时微调两个编码器会损害性能（§5.1）。虽然先前的工作仅依赖于未标记文本进行LSR训练，但我们展示了LSR可以泛化以包含多任务指令数据（在§2.2中介绍，称为MTI数据）。MTI数据为检索器提供了直接的监督，以返回在各种下游任务中增强语言模型的相关信息。如§5.1所示，结合这两种类型的数据可以产生最佳结果，并且优于单独使用任一数据源。

# 3 实验设置

## 3.1 检索器

我们使用DRAGON+初始化我们框架中的检索器，并使用它来研究各种检索器配置。为了构建检索语料库，我们将Izacard等人发布的2021年12月20日的Wikipedia数据转储中的文本块（37M）与2017-2020年CommonCrawl转储中的额外文本块（362M）结合起来。我们在附录A中详细介绍了语料库的预处理和索引。

我们的最终检索数据存储，将这两个数据源结合起来，包含399M个文本块，每个文本块的最大长度为200个单词。在§5.3中，我们对使用我们检索索引的各种子集以及不同Wikipedia快照的检索语料库的影响进行了分析。我们使用手动构造的模板（表10和12）获取用于我们微调和评估任务的检索查询。

3.2 基线

我们专注于将我们的方法与基础LLAMA模型和REPLUG进行比较，REPLUG是一种将现成的LLMs和检索器集成的最新方法，在零次和上下文少次学习设置中。我们使用LLAMA和DRAGON+实例化REPLUG。此外，我们还在64次微调设置中将RA-DIT与ATLAS进行比较（§4）。

3.3 评估

我们主要在不包含在我们的微调数据集中的知识密集型任务上进行评估，包括MMLU、Natural Questions、TriviaQA以及KILT基准测试的任务子集。我们使用六个KILT任务的开发分割（不包括ELI5）来确定微调超参数（附录B）。

这使我们能够为四个评估任务报告真正的少次评估结果。对于其余任务，我们假设可以访问领域内开发数据来报告少次结果。我们从KILT任务的官方训练分割中随机选择少次示例，除了FEV、NQ和TQA，我们使用Izacard等人发布的64次示例。

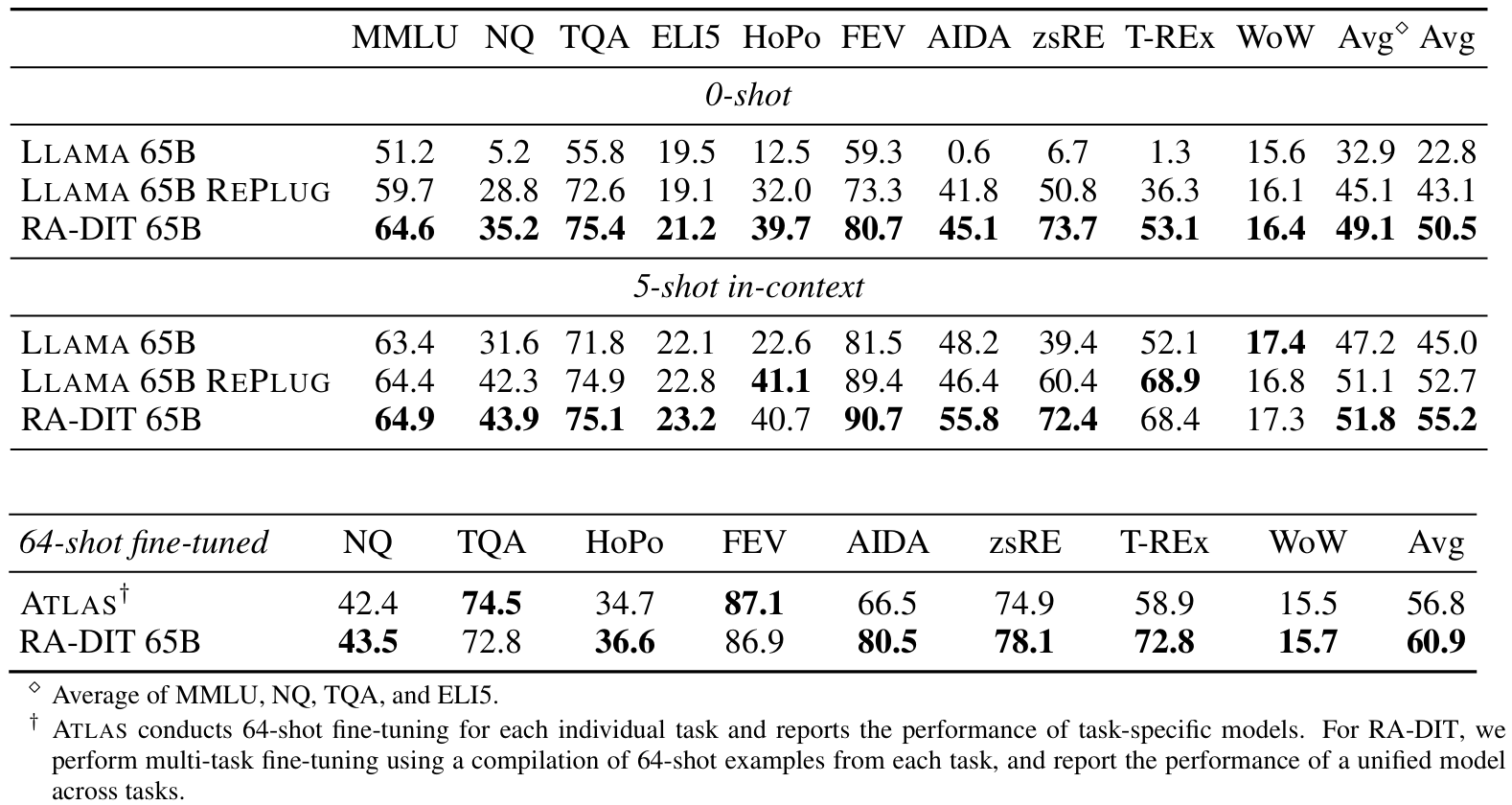
对于这三个数据集，我们还确保5次示例是64次示例的子集。在我们的检索增强模型中，我们为上下文少次示例使用最相关的块。此外，我们还评估模型在常识推理任务上的表现，以评估检索增强指令微调对LLM的参数知识和推理能力的影响。在这里，我们报告整个开发集的结果。我们的评估数据集的详细信息，包括评估指标、模板和使用的评分函数，可以在附录D中找到。

# 4 主要结果

**知识密集型任务** 我们在表2中报告了主要结果。特别是，RA-DIT与LLAMA以及REPLUG在0次和5次设置中进行比较。我们首先观察到REPLUG比基础LLAMA 65B表现得好得多，证实了RALMs在知识密集型任务上的优势。

此外，RA-DIT显著优于REPLUG（在MMLU、NQ、TQA和ELI5上平均0次设置提高8.9%，在5次设置中提高1.4%），并在大多数数据集上取得了最佳性能。这支持了我们的主张，即结合现成的LLMs和检索器是次优的，我们的双重指令微调方法是为LLMs提供检索能力的有效的改装方式。

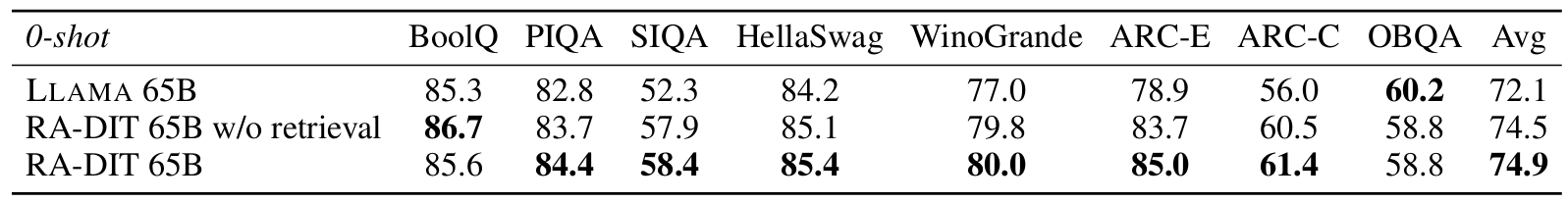
表二：主要结果：在知识密集型任务上的表现



我们还与ATLAS进行了比较，ATLAS是一种基于编码器-解码器的最新RALM，它联合预训练语言模型和检索器。在这里，我们采用与Izacard等人类似的64次设置，但有所不同。虽然ATLAS对每个单独任务进行64次微调，并报告特定任务模型的性能，但我们使用来自所有任务组合的64次示例连续微调RA-DIT检查点，并报告跨任务的统一模型的性能。如表2所示，尽管只使用了一个模型，RA-DIT的平均性能比ATLAS高出4.1个百分点，在8个数据集中有6个表现更好。

**常识推理** 我们在一组常识推理任务上对RA-DIT 65B进行了基准测试。我们还评估了不使用检索的性能，以检查检索增强指令微调对LM的参数知识和推理能力的影响。如表3所示，RA-DIT在8个评估数据集中的7个上展示了改进。没有使用检索的模型变体与完整模型表现相近，表明这些任务主要依赖于语言模型的参数知识和推理能力。此外，这一观察结果证实了应用于LM的微调方法并未损害其固有知识。如附录F所讨论的，保留LM的参数知识和能力至关重要，因为当检索器出错时，它可以作为后备。

表三：没有检索增强时在常识推理任务上的表现



# 5 分析

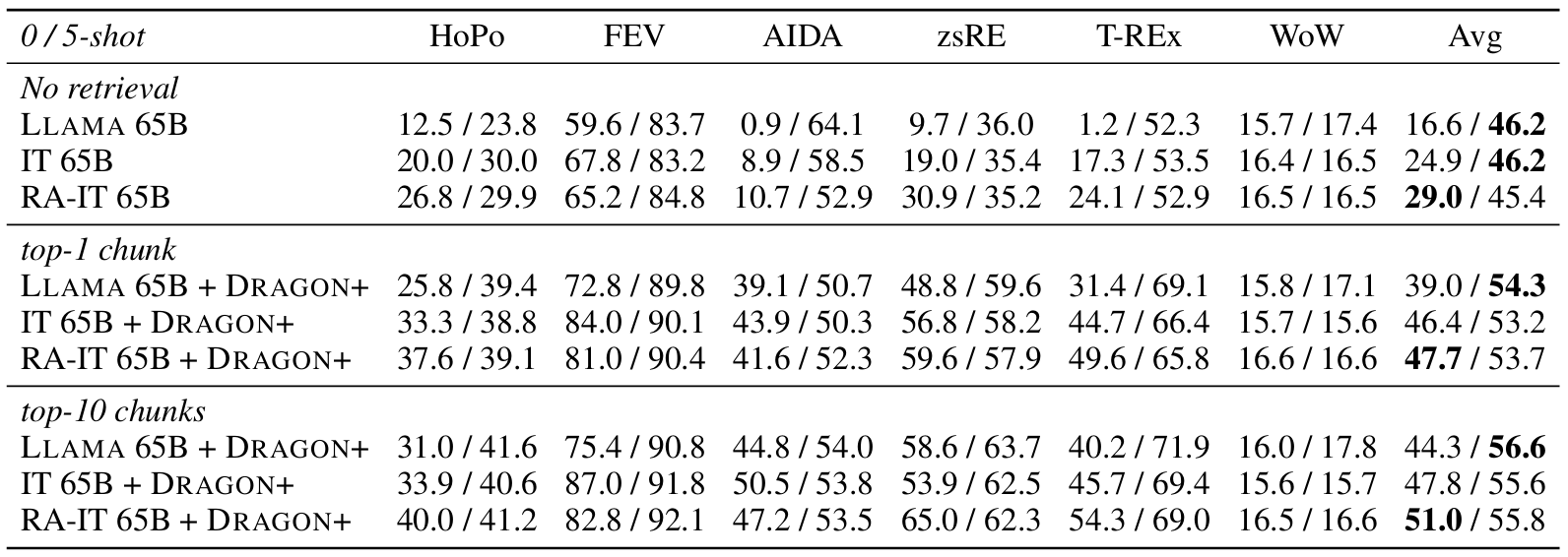
在这个部分，我们展示了一系列关于各种各样建模决策的分析。

## 5.1 微调策略

**语言模型微调**：我们比较了带有检索增强的LLAMA指令微调（RA-IT 65B）与基础语言模型，以及常规指令微调的LLAMA（IT 65B）在相同任务集上的表现。我们评估了所有模型在上下文检索增强下的性能，使用DRAGON+检索器，并调整检索到的块数为0、1或10。

如表4所示，虽然两种指令微调方法都显著提高了零次性能，但在5次设置中，除了HotpotQA之外，它们对大多数任务的模型性能提供了边际改进甚至有所损害。

表四：语言模型微调策略的消融



当应用上下文检索增强时，所有模型在两种设置下都显示出显著的增益，即使仅限于前1个检索块。在零次设置中，使用前10个检索块时，RA-IT 65B模型的性能显著优于IT 65B模型（51.0%对比47.7%）。在这种设置下，我们观察到，检索增强的指令微调显著提高了LLM整合检索文本块信息的能力。模型能够更有信心地从相关块中提取正确答案，同时在出现不相关内容块时有效地依赖其参数知识进行预测（附录F）。在附录E.1中，我们还讨论了将RA-IT模型应用于较小的LLAMA模型（7B和13B）时的性能，表明在这些情况下它提供了更大的性能提升。

**检索器微调：**在表5中，我们研究了不同的检索器微调策略。正如在§2.4中提到的，我们探索了两种类型的检索器微调数据，多任务指令（MTI）数据和语料库数据。

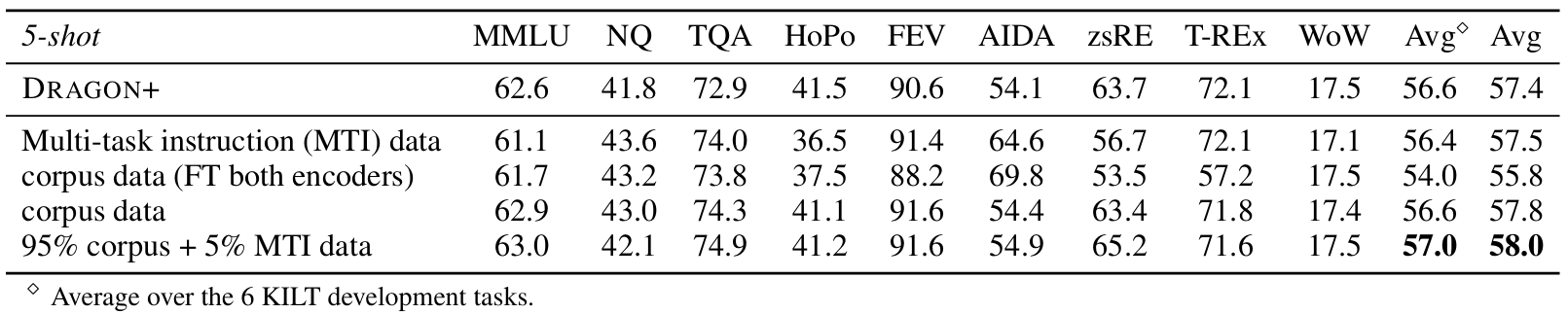
我们观察到，仅使用语料库数据对检索器进行微调，与基础DRAGON+模型相比，平均提高了0.4个百分点，而仅使用MTI数据进行微调，提高了较小的0.1个百分点。

虽然仅使用MTI数据在某些数据集上（如NQ）表现良好（可能是由于其与MTI数据的相似性），但使用语料库数据进行微调似乎具有更好的泛化能力，并导致整体性能更强。

此外，我们还尝试了同时使用MTI和语料库数据进行微调。表5显示，使用“95%语料库数据+5% MTI数据”进行微调，在所有模型中实现了最佳准确性，平均比非微调基线提高了0.6个百分点。

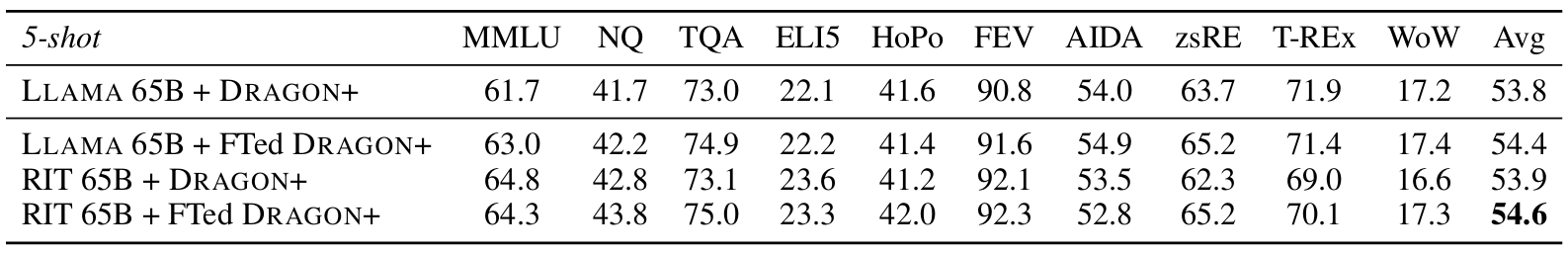
最后，我们还比较了同时微调查询和文档编码器与仅微调查询编码器（同时冻结文档编码器）的效果。表5显示，使用语料库数据进行实验时，冻结文档编码器产生了显著更好的性能。因此，我们在这项工作中只对查询编码器进行了微调。

表五：检索微调策略的消融



## 5.2 双重指令微调消融

表6：在我们的RA-DIT方法中，我们隔离了语言模型微调（LM-ft）和检索器微调（R-ft）的影响，并展示了各自的益处。我们报告了5次开发集性能。



在我们的 RA-DIT 方法中，我们将语言模型微调的影响与寻回器微调的影响分离开来，并说明了每种微调的益处。方法中语言模型微调与检索器微调的影响，并说明两者各自的优势。

根据表6，LM-ft和R-ft单独使用时都有益处，并且优于使用LLAMA 65B和DRAGON+检索器的REPLUG。

另一方面，当我们在RA-DIT方法中结合LM-ft和R-ft时，可以获得最大的增益，平均比REPLUG基线高出0.8个百分点。

在我们的初步实验中，我们还尝试了迭代双重指令微调，即使用RA-IT LM的LSR分数微调检索器，或者使用微调后的检索器返回的段落进行RA-IT步骤，进行了一到两次这样的迭代，但没有观察到进一步的增益。我们将探索多步RA-DIT的探索留作未来工作。

## 5.3 检索器设置

在本节中，我们研究了我们框架中不同检索器选择的影响。我们使用LLAMA 65B作为语言模型，并将其与不同的检索器结合。

表7首先比较了DRAGON+与其他最先进的检索器，如Contriever。所有检索增强模型都在LLAMA基线上显著提高，DRAGON+显著优于Contriever和Contriever-MSMARCO。因此，我们在所有实验中采用DRAGON+作为基础检索器。

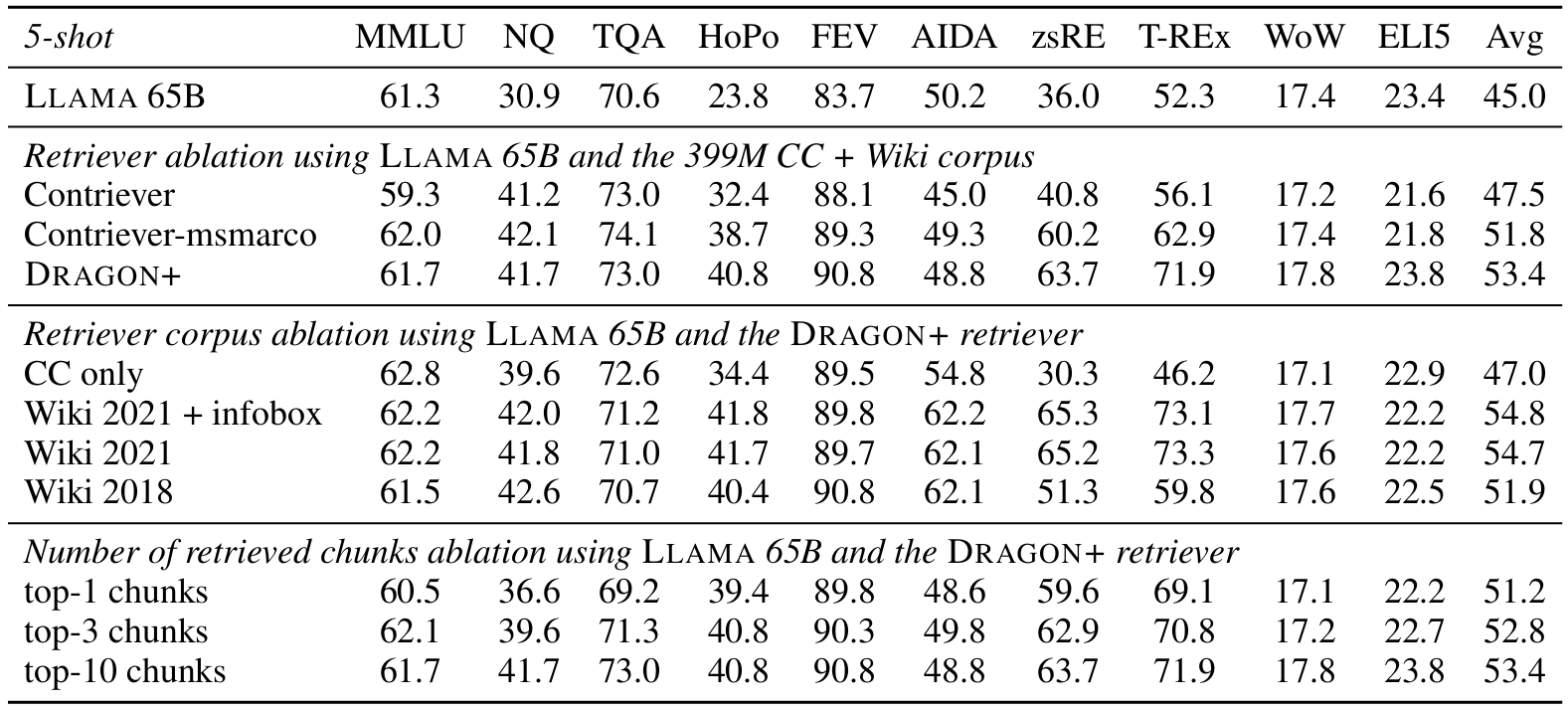
表7中间部分展示了检索语料库变化的影响。具体来说，我们考虑了我们399M检索语料库的几个子集，即仅CommonCrawl（362M）和仅Wikipedia（带和不带infoboxes）。

我们还与文献中常用的另一个Wikipedia快照进行了比较。我们观察到，仅从Wikipedia检索对一些KILT任务（如AIDA和zsRE）有益，因为Wikipedia是KILT任务的预期语料库。我们发现Wiki 2018对NQ更有效，因为该语料库更接近其数据收集日期，这与Izacard等人的观察结果相似。

这表明我们的检索增强LM忠实于提供的检索语料库，并且可以通过在测试时更新检索索引来提供最新信息。

最后，我们尝试了在生成过程中提供给LLAMA的不同数量的检索段落。表7显示，即使检索到前1个段落，也能显著提高LLAMA的平均性能，从45.0提高到51.2，并且随着使用的检索段落数量增加，性能继续提高。由于收益递减和推理成本，我们在实验中默认采用10个检索段落。

表七：检索器设置：我们报告了在 REPLUG 设置中使用 LLAMA 65B 和各种 检索器在 REPLUG 设置中的性能。



# 7 结论

在本文中，我们提出了RA-DIT，一种轻量级的检索增强双重指令微调框架，可以有效地为任何预训练的大型语言模型（LLM）提供检索能力。RA-DIT通过检索增强的指令微调更新LLM，使其更好地利用检索到的知识，并忽略不相关或分散注意力的信息。它还通过LLM提供的监督对检索器进行微调，以检索能够帮助LLM生成正确输出的文本。RA-DIT在零次和少次评估中取得了最先进的性能，超越了未调整的上下文检索增强语言模型（RALM）方法，如REPLUG，并有效地与需要广泛预训练的方法（如ATLAS）竞争。