Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
检索增强生成用于知识密集型自然语言处理任务

Patrick Lewis[†‡] , Ethan Perez[⋆] , Aleksandra Piktus[†] , Fabio Petroni[†] , Vladimir Karpukhin[†] , Naman Goyal [†] , Heinrich Küttler [†] , Mike Lewis [†] , Wen-tau Yih [†] , Tim Rocktäschel [†‡] , Sebastian Riedel [†‡] , Douwe Kiela [† †]
Facebook AI Research; [‡] University College London; [⋆] New York University;
plewis@fb.com

Abstract

Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable access mechanism to explicit nonparametric memory can overcome this issue, but have so far been only investigated for extractive downstream tasks. We explore a general-purpose fine-tuning recipe for retrieval-augmented generation (RAG) - models which combine pre-trained parametric and non-parametric memory for language generation. We introduce RAG models where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We compare two RAG formulations, one which conditions on the same retrieved passages across the whole generated sequence, and another which can use different passages per token. We fine-tune and evaluate our models on a wide range of knowledge-intensive NLP tasks and set the state of the art on three open domain QA tasks, outperforming parametric seq2seq models and task-specific retrieve-and-extract architectures. For language generation tasks, we find that RAG models generate more specific, diverse and factual language than a state-of-the-art parametric-only seq2seq baseline.

大型预训练语言模型已被证明在其参数中存储了事实知识, 并在下游自然语言处理任务上进行微调时取得了最先进的结果。然而, 它们访问和精确操作知识的能力仍然有限, 因此在知识密集型任务上, 它们的性能落后于特定任务的架构。此外, 为它们的决策提供来源和更新其世界知识仍然是开放的研究问题。具有可微分访问机制的预训练模型可以克服这个问题, 但迄今为止仅在抽取式下游任务中进行了研究。我们探索了一种通用的微调方法, 用于检索增强生成 (RAG) 模型, 这些模型结合了预训练的参量和非参量记忆用于语言生成。我们引入了 RAG 模型, 其中参量记忆是一个预训练的序列到序列模型, 非参量记忆是维基百科的密集向量索引, 通过预训练的神经检索器访问。我们比较了两种 RAG 公式, 一种是在整个生成序列中使用相同的检索段落, 另一种是每个标记可以使用不同的段落。我们在广泛的知识密集型自然语言处理任务上微调和评估了我们的模型, 并在三个开放领域问答任务上设定了最先进的结果, 优于参量序列到序列模型和特定任务的检索和提取架构。对于语言生成任务, 我们发现 RAG 模型生成的文本比最先进的仅参量序列到序列基线更具体、多样和事实准确。

1 Introduction

Pre-trained neural language models have been shown to learn a substantial amount of in-depth knowledge from data [47]. They can do so without any access to an external memory, as a parameterized implicit knowledge base [51, 52]. While this development is exciting, such models do have down-sides: They cannot easily expand or revise their memory, can't straightforwardly provide insight into their predictions, and may produce "hallucinations" [38]. Hybrid models that combine parametric memory with non-parametric (i.e., retrieval-based) memories [20, 26, 48] can address some of these issues because knowledge can be directly revised and expanded, and accessed knowledge can be inspected and interpreted.

预训练的神经语言模型已被证明能够从数据中学习大量深入的知识 [47]。它们可以在不访问外部存储器的情况下做到这一点, 作为一个参数化的隐式知识库 [51, 52]。虽然这一发展令人兴奋, 但这类模型确实存在一些缺点: 它们无法轻松扩展或修改其记忆, 无法直接提供对其预测的洞察, 并且可能会产生" 幻觉" [38]。结合参数化记忆与非参数化 (即基于检索的) 记忆的混合模型 [20, 26, 48] 可以解决其中一些问题, 因为知识可以直接修改和扩展, 并且可以检查和解释所访问的知识。

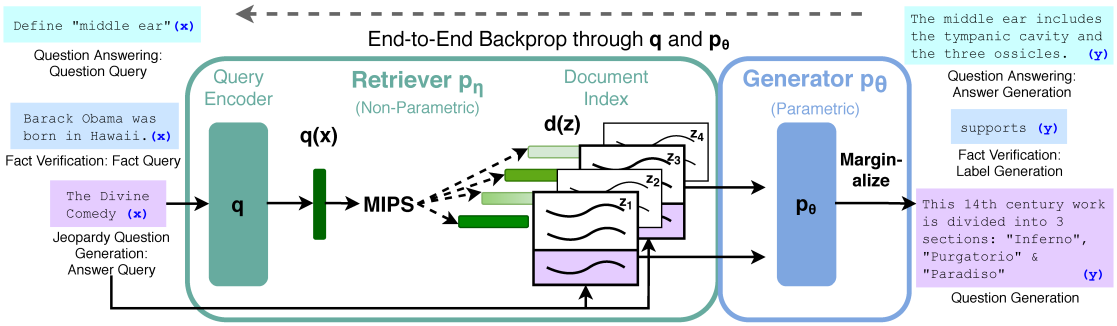

Figure 1: Overview of our approach. We combine a pre-trained retriever (Query Encoder + Document Index) with a pre-trained seq2seq model (Generator) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.
我们方法的概述。我们将预训练的检索器 (查询编码器 + 文档索引) 与预训练的 seq2seq 模型 (生成器) 结合, 并进行端到端的微调。对于查询 x , 我们使用最大内积搜索 (MIPS) 来找到前 K 个文档 z_i 。对于最终预测 y , 我们将 z 视为潜在变量, 并在给定不同文档的情况下对 seq2seq 预测进行边缘化。

REALM [20] and ORQA [31], two recently introduced models that combine masked language models [8] with a differentiable retriever, have shown promising results, but have only explored open-domain extractive question answering. Here, we bring hybrid parametric and non-parametric memory to the "workhorse of NLP, " i.e. sequence-to-sequence (seq2seq) models.

REALM [20] 和 ORQA [31] 是最近引入的两个模型, 它们将掩码语言模型 [8] 与可微分检索器结合, 显示出有希望的结果, 但仅探索了开放域抽取式问答。在这里, 我们将混合参数化和非参数化记忆引入到"NLP 的主力" 中, 即序列到序列 (seq2seq) 模型。

We endow pre-trained, parametric-memory generation models with a non-parametric memory through a general-purpose fine-tuning approach which we refer to as retrieval-augmented generation (RAG). We build RAG models where the parametric memory is a pre-trained seq2seq transformer, and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We combine these components in a probabilistic model trained end-to-end (Fig. 1). The retriever (Dense Passage Retriever [26], henceforth DPR) provides latent documents conditioned on the input, and the seq2seq model (BART [32]) then conditions on these latent documents together with the input to generate the output. We marginalize the latent documents with a top-K approximation, either on a per-output basis (assuming the same document is responsible for all tokens) or a per-token basis (where different documents are responsible for different tokens). Like T5 [51] or BART, RAG can be fine-tuned on any seq2seq task, whereby both the generator and retriever are jointly learned.

我们通过一种通用的微调方法, 为预训练的、参数化记忆生成模型赋予非参数化记忆, 我们称之为检索增强生成 (RAG)。我们构建了 RAG 模型, 其中参数化记忆是预训练的 seq2seq transformer, 非参数化记忆是维基百科的密集向量索引, 通过预训练的神经检索器访问。我们将这些组件结合在一个端到端训练的概率模型中 (图 1)。检索器 (密集段落检索器 [26], 以下简称 DPR) 提供基于输入的潜在文档, 然后 seq2seq 模型 (BART [32]) 基于这些潜在文档和输入生成输出。我们使用前 K 近似对潜在文档进行边缘化, 可以基于每个输出 (假设同一文档负责所有标记) 或基于每个标记 (不同文档负责不同标记)。与 T5 [51] 或 BART 类似, RAG 可以在任何 seq2seq 任务上进行微调, 生成器和检索器共同学习。

There has been extensive previous work proposing architectures to enrich systems with non-parametric memory which are trained from scratch for specific tasks, e.g. memory networks [64, 55], stack-augmented networks [25] and memory layers [30]. In contrast, we explore a setting where both parametric and non-parametric memory components are pre-trained and pre-loaded with extensive knowledge. Crucially, by using pre-trained access mechanisms, the ability to access knowledge is present without additional training.

之前有大量工作提出了通过非参数化记忆丰富系统的架构, 这些架构是为特定任务从头开始训练的, 例如记忆网络 [64, 55]、堆栈增强网络 [25] 和记忆层 [30]。相比之下, 我们探索了一种设置, 其中参数化和非参数化记忆组件都是预训练的, 并预加载了大量知识。关键的是, 通过使用预训练的访问机制, 无需额外训练即可具备访问知识的能力。

Our results highlight the benefits of combining parametric and non-parametric memory with generation for knowledge-intensive tasks-tasks that humans could not reasonably be expected to perform without access to an external knowledge source. Our RAG models achieve state-of-the-art results on open Natural Questions [29], WebQuestions [3] and CuratedTree [2] and strongly outperform recent approaches that use specialised pre-training objectives on TriviaQA [24]. Despite these being extractive tasks, we find that unconstrained generation outperforms previous extractive approaches. For knowledge-intensive generation, we experiment with MS-MARCO [1] and Jeopardy question generation, and we find that our models generate responses that are more factual, specific, and diverse than a BART baseline. For FEVER [56] fact verification, we achieve results within 4.3% of state-of-the-art pipeline models which use strong retrieval supervision. Finally, we demonstrate that the non-parametric memory can be replaced to update the models' knowledge as the world changes. [1]

我们的结果突出了将参数化和非参数化记忆与生成结合用于知识密集型任务的优势——这些任务是人类在没有外部知识源的情况下无法合理完成的。我们的 RAG 模型在开放自然问题 [29]、WebQuestions [3] 和 CuratedTree [2] 上取得了最先进的结果，并且在 TriviaQA [24] 上显著优于最近使用专门预训练目标的方法。尽管这些是抽取式任务，但我们发现无约束生成优于之前的抽取式方法。对于知识密集型生成，我们在 MS-MARCO [1] 和 Jeopardy 问题生成上进行了实验，发现我们的模型生成的响应比 BART 基线更具事实性、具体性和多样性。对于 FEVER [56] 事实验证，我们取得了与使用强检索监督的最先进管道模型相差 4.3% 以内的结果。最后，我们展示了非参数化记忆可以随着世界的变化而替换，以更新模型的知识。

2 Methods

We explore RAG models, which use the input sequence x to retrieve text documents z and use them as additional context when generating the target sequence y . As shown in Figure 1, our models leverage two components: (i) a retriever $p_\eta(z \mid x)$ with parameters η that returns (top-K truncated) distributions over text passages given a query x and (ii) a generator $p_\theta(y_i \mid x, z, y_{1:i-1})$ parametrized by θ that generates a current token based on a context of the previous $i-1$ tokens $y_{1:i-1}$, the original input x and a retrieved passage z .

我们探索了 RAG 模型，该模型使用输入序列 x 来检索文本文档 z ，并在生成目标序列 y 时将其作为额外的上下文。如图 1 所示，我们的模型利用了两个组件:(i) 一个检索器 $p_\eta(z \mid x)$ ，其参数为 η ，在给定查询 x 时返回文本段落的 (top-K 截断) 分布；(ii) 一个生成器 $p_\theta(y_i \mid x, z, y_{1:i-1})$ ，其参数为 θ ，基于前 $i-1$ 个标记 $y_{1:i-1}$ 的上下文、原始输入 x 和检索到的段落 z 生成当前标记。

To train the retriever and generator end-to-end, we treat the retrieved document as a latent variable. We propose two models that marginalize over the latent documents in different ways to produce a distribution over generated text.

为了端到端地训练检索器和生成器，我们将检索到的文档视为潜在变量。我们提出了两种模型，它们以不同的方式对潜在文档进行边缘化，以生成文本的分布。

In one approach, RAG-Sequence, the model uses the same document to predict each target token. The second approach, RAG-Token, can predict each target token based on a different document. In the following, we formally introduce both models and then describe the p_η and p_θ components, as well as the training and decoding procedure.

在一种方法中，RAG-Sequence 模型使用相同的文档来预测每个目标标记。第二种方法，RAG-Token，可以基于不同的文档预测每个目标标记。在下文中，我们正式介绍这两种模型，然后描述 p_η 和 p_θ 组件，以及训练和解码过程。

[1] Code to run experiments with RAG has been open-sourced as part of the HuggingFace Transformers Library [66] and can be found at https://github.com/huggingface/transformers/blob/master/ examples/rag/ An interactive demo of RAG models can be found at https://huggingface.co/rag/

2.1 Models

RAG-Sequence Model The RAG-Sequence model uses the same retrieved document to generate the complete sequence. Technically, it treats the retrieved document as a single latent variable that is marginalized to get the seq2seq probability $p(y \mid x)$ via a top-K approximation. Concretely, the top K documents are retrieved using the retriever, and the generator produces the output sequence probability for each document, which are then marginalized.

RAG-Sequence 模型 RAG-Sequence 模型使用相同的检索文档生成完整序列。从技术上讲, 它将检索到的文档视为单个潜在变量, 通过 top-K 近似进行边缘化以获得 seq2seq 概率 $p(y \mid x)$ 。具体来说, 使用检索器检索 top K 文档, 生成器为每个文档生成输出序列概率, 然后进行边缘化。

$$p_{\text{RAG-Sequence}}(y \mid x) \approx \sum_{z \in \text{top-}k(p(\cdot \mid x))} p_\eta(z \mid x)\, p_\theta(y \mid x, z) = \sum_{z \in \text{top-}k(p(\cdot \mid x))} p_\eta(y_i \mid x, z, y_{1:i-1})$$

RAG-Token Model In the RAG-Token model we can draw a different latent document for each target token and marginalize accordingly. This allows the generator to choose content from several documents when producing an answer. Concretely, the top K documents are retrieved using the retriever, and then the generator produces a distribution for the next output token for each document, before marginalizing, and repeating the process with the following output token, Formally, we define:

RAG-Token 模型在 RAG-Token 模型中, 我们可以为每个目标标记绘制不同的潜在文档并进行相应的边缘化。这使得生成器在生成答案时可以从多个文档中选择内容。具体来说, 使用检索器检索前 K 个文档, 然后生成器为每个文档生成下一个输出标记的分布, 再进行边缘化, 并对下一个输出标记重复此过程。形式上, 我们定义:

$$p_{\text{RAG-Token}}(y \mid x) \approx \prod_{i}^{N} \sum_{z \in \text{top-}k(p(\cdot \mid x))} p_\eta(z \mid x)\, p_\theta(y_i \mid x, z_i, y_{1:i-1})$$

Finally, we note that RAG can be used for sequence classification tasks by considering the target class as a target sequence of length one, in which case RAG-Sequence and RAG-Token are equivalent.

最后, 我们注意到 RAG 可以用于序列分类任务, 通过将目标类视为长度为 1 的目标序列, 在这种情况下, RAG-Sequence 和 RAG-Token 是等价的。

2.2 Retriever **检索器**: DPR

The retrieval component $p_\eta(z \mid x)$ is based on DPR [26]. DPR follows a bi-encoder architecture:

检索组件 $p_\eta(z \mid x)$ 基于 DPR [26]。DPR 遵循双编码器架构:

$$p_\eta(z \mid x) \propto \exp\left(\mathrm{d}(z)^\top \mathrm{q}(x)\right) \quad \mathrm{d}(z) = \mathrm{BERT}_d(z), \ \mathrm{q}(x) = \mathrm{BERT}_q(x)$$

where $\mathrm{d}(z)$ is a dense representation of a document produced by a $\mathrm{BERT}_{\text{BASE}}$ document encoder [8], and $\mathrm{q}(x)$ a query representation produced by a query encoder, also based on $\mathrm{BERT}_{\text{BASE}}$.

其中 $\mathrm{d}(z)$ 是由 $\mathrm{BERT}_{\text{BASE}}$ 文档编码器 [8] 生成的文档的密集表示, $\mathrm{q}(x)$ 是由查询编码器生成的查询表示, 同样基于 $\mathrm{BERT}_{\text{BASE}}$ 。

Calculating top-k $(p_\eta(\cdot \mid x))$, the list of k documents z with highest prior probability $p_\eta(z \mid x)$, is a Maximum Inner Product Search (MIPS) problem, which can be approximately solved in sub-linear time [23]. We use a pre-trained bi-encoder from DPR to initialize our retriever and to build the document index. This retriever was trained to retrieve documents which contain answers to TriviaQA [24] questions and Natural Questions [29]. We refer to the document index as the non-parametric memory.

计算前 k 个 $(p_\eta(\cdot \mid x))$, 即具有最高先验概率 $p_\eta(z \mid x)$ 的 k 文档 z 列表, 是一个最大内积搜索 (MIPS) 问题, 可以在亚线性时间内近似解决 [23]。我们使用 DPR 的预训练双编码器来初始化我们的检索器并构建文档索引。该检索器经过训练, 用于检索包含 TriviaQA [24] 问题和自然问题 [29] 答案的文档。我们将文档索引称为非参数记忆。

2.3 Generator **生成器**: BART

The generator component $p_\theta(y_i \mid x, z, y_{1:i-1})$ could be modelled using any encoder-decoder. We use BART-large [32], a pre-trained seq2seq transformer [58] with 400M parameters. To combine the input x with the retrieved content z when generating from BART, we simply concatenate them. BART was pre-trained using a denoising objective and a variety of different noising functions. It has obtained state-of-the-art results on a diverse set of generation tasks and outperforms comparably-sized T5 models [32]. We refer to the BART generator parameters θ as the parametric memory henceforth.

生成器组件 $p_\theta(y_i \mid x, z, y_{1:i-1})$ 可以使用任何编码器-解码器进行建模。我们使用 BART-large [32], 这是一个预训练的 seq2seq transformer [58], 具有 4 亿个参数。为了在从 BART 生成时将输入 x 与检索到的内容 z 结合, 我们简单地将它们连接起来。BART 是使用去噪目标和各种不同的噪声函数进行预训练的。它在各种生成任务上取得了最先进的结果, 并且优于同等规模的 T5 模型 [32]。我们此后将 BART 生成器参数 θ 称为参数记忆。

2.4 Training

We jointly train the retriever and generator components without any direct supervision on what document should be retrieved. Given a fine-tuning training corpus of input/output pairs (x_j, y_j) , we minimize the negative marginal log-likelihood of each target, $\sum_j -\log p(y_j \mid x_j)$ using stochastic gradient descent with Adam [28]. Updating the document encoder BERT_d during training is costly as it requires the document index to be periodically updated as REALM does during pre-training [20]. We do not find this step necessary for strong performance, and keep the document encoder (and index) fixed, only fine-tuning the query encoder BERT_q and the BART generator.

我们联合训练检索器和生成器组件, 而不对应该检索哪些文档进行任何直接监督。给定一个输入/输出对的微调训练语料库 (x_j, y_j) , 我们使用 Adam [28] 的随机梯度下降法最小化每个目标的负边际对数似然 $\sum_j -\log p(y_j \mid x_j)$ 。在训练期间更新文档编码器 BERT_d 是昂贵的, 因为它需要像 REALM 在预训练期间那样定期更新文档索引 [20]。我们发现这一步对于强性能并不是必要的, 因此保持文档编码器 (和索引) 固定, 仅微调查询编码器 BERT_q 和 BART 生成器。

2.5 Decoding **解码**

At test time, RAG-Sequence and RAG-Token require different ways to approximate $\arg\max_y p(y \mid x)$.

在测试时, RAG-Sequence 和 RAG-Token 需要不同的方法来近似 $\arg\max_y p(y \mid x)$ 。

RAG-Token The RAG-Token model can be seen as a standard, autoregressive seq2seq generator with transition probability: $p'_\theta(y_i \mid x, y_{1:i-1}) = \sum_{z \in \text{top-}k(p(\cdot \mid x))} p_\eta(z_i \mid x)\, p_\theta(y_i \mid x, z_i, y_{1:i-1})$

RAG-Token 模型可以看作是一个标准的自回归 seq2seq 生成器, 其转移概率为: $p'_\theta(y_i \mid x, y_{1:i-1}) = \sum_{z \in \text{top-}k(p(\cdot \mid x))} p_\eta(z_i \mid x)\, p_\theta(y_i \mid x, z_i, y_{1:i-1})$

To decode, we can plug $p'_\theta(y_i \mid x, y_{1:i-1})$ into a standard beam decoder.

为了解码, 我们可以将 $p'_\theta(y_i \mid x, y_{1:i-1})$ 插入到标准束解码器中。

RAG-Sequence For RAG-Sequence, the likelihood $p(y \mid x)$ does not break into a conventional per-token likelihood, hence we cannot solve it with a single beam search. Instead, we run beam search for each document z , scoring each hypothesis using $p_\theta(y_i \mid x, z, y_{1:i-1})$. This yields a set of hypotheses Y , some of which may not have appeared in the beams of all documents.

对于 RAG-Sequence, 似然 $p(y \mid x)$ 不会分解为传统的每标记似然, 因此我们无法通过单一的束搜索来解决它。相反, 我们对每个文档 z 运行束搜索, 使用 $p_\theta(y_i \mid x, z, y_{1:i-1})$ 对每个假设进行评分。这产生了一组假设 Y , 其中一些可能没有出现在所有文档的束中。

To estimate the probability of an hypothesis y we run an additional forward pass for each document z for which y does not appear in the beam, multiply generator probability with $p_\eta(z \mid x)$ and then sum the probabilities across beams for the marginals. We refer to this decoding procedure as "Thorough Decoding." For longer output sequences, $|Y|$ can become large, requiring many forward passes. For more efficient decoding, we can make a further approximation that $p_\theta(y \mid x, z_i) \approx 0$ where y was not generated during beam search from x, z_i . This avoids the need to run additional forward passes once the candidate set Y has been generated. We refer to this decoding procedure as "Fast Decoding."

为了估计假设 y 的概率, 我们对每个文档 z 运行额外的前向传递, 其中 y 没有出现在束中, 将生成器概率与 $p_\eta(z \mid x)$ 相乘, 然后对束中的概率求和以获得边际。我们将此解码过程称为" 彻底解码"。对于较长的输出序列, $|Y|$ 可能变得很大, 需要许多前向传递。为了更高效地解码, 我们可以进一步近似 $p_\theta(y \mid x, z_i) \approx 0$, 其中 y 在 x, z_i 的束搜索期间未生成。这避免了在生成候选集 Y 后运行额外的前向传递的需要。我们将此解码过程称为" 快速解码"。

3 Experiments

We experiment with RAG in a wide range of knowledge-intensive tasks. For all experiments, we use a single Wikipedia dump for our non-parametric knowledge source. Following Lee et al. [31] and Karpukhin et al. [26], we use the December 2018 dump. Each Wikipedia article is split into disjoint 100-word chunks, to make a total of 21M documents. We use the document encoder to compute an embedding for each document, and build a single MIPS index using FAISS [23] with a Hierarchical Navigable Small World approximation for fast retrieval [37]. During training, we retrieve the top k documents for each query. We consider $k \in \{5, 10\}$ for training and set k for test time using dev data. We now discuss experimental details for each task.

我们在各种知识密集型任务中实验了 RAG。对于所有实验, 我们使用单一的维基百科转储作为我们的非参数知识源。遵循 Lee 等人 [31] 和 Karpukhin 等人 [26] 的做法, 我们使用 2018 年 12 月的转储。每篇维基百科文章被分割成不重叠的 100 字块, 总共生成 21M 个文档。我们使用文档编码器计算每个文档的嵌入, 并使用 FAISS [23] 构建一个单一的 MIPS 索引, 采用分层可导航小世界近似以实现快速检索 [37]。在训练期间, 我们为每个查询检索前 k 个文档。我们考虑 $k \in \{5, 10\}$ 进行训练, 并使用开发数据设置 k 进行测试。我们现在讨论每个任务的实验细节。

3.1 Open-domain Question Answering **开放域问答**

Open-domain question answering (QA) is an important real-world application and common testbed for knowledge-intensive tasks [20]. We treat questions and answers as input-output text pairs(x, y) and train RAG by directly minimizing the negative log-likelihood of answers. We compare RAG to the popular extractive QA paradigm [5, 7, 31, 26], where answers are extracted spans from retrieved documents, relying primarily on non-parametric knowledge. We also compare to "Closed-Book QA" approaches [52], which, like RAG, generate answers, but which do not exploit retrieval, instead relying purely on parametric knowledge. We consider four popular open-domain QA datasets: Natural Questions (NQ) [29], TriviaQA (TQA) [24]. WebQuestions (WQ) [3] and CuratedTrec (CT) [2]. As CT and WQ are small, we follow DPR [26] by initializing CT and WQ models with our NQ RAG model. We use the same train/dev/test splits as prior work [31, 26] and report Exact Match (EM) scores. For TQA, to compare with T5 [52], we also evaluate on the TQA Wiki test set.

开放域问答 (QA) 是一个重要的现实世界应用和知识密集型任务的常见测试平台 [20]。我们将问题和答案视为输入-输出文本对 (x, y), 并通过直接最小化答案的负对数似然来训练 RAG。我们将 RAG 与流行的抽取式 QA 范式 [5, 7, 31, 26] 进行比较, 其中答案是从检索到的文档中提取的片段, 主要依赖于非参数知识。我们还与" 闭卷 QA" 方法 [52] 进行比较, 这些方法与 RAG 一样生成答案, 但不利用检索, 而是完全依赖于参数知识。我们考虑了四个流行的开放域 QA 数据集: 自然问题 (NQ)[29]、TriviaQA(TQA)[24]、WebQuestions(WQ)[3] 和 CuratedTrec(CT)[2]。由于 CT 和 WQ 较小, 我们遵循 DPR [26] 的做法, 使用我们的 NQ RAG 模型初始化 CT 和 WQ 模型。我们使用与先前工作 [31, 26] 相同的训练/开发/测试分割, 并报告精确匹配 (EM) 分数。对于 TQA, 为了与 T5 [52] 进行比较, 我们还在 TQA Wiki 测试集上进行了评估。

3.2 Abstractive Question Answering **抽象问答**

RAG models can go beyond simple extractive QA and answer questions with free-form, abstractive text generation. To test RAG's natural language generation (NLG) in a knowledge-intensive setting, we use the MSMARCO NLG task v2.1 [43]. The task consists of questions, ten gold passages retrieved from a search engine for each question, and a full sentence answer annotated from the retrieved passages. We do not use the supplied passages, only the questions and answers, to treat MSMARCO as an open-domain abstractive QA task. MSMARCO has some questions that cannot be answered in a way that matches the reference answer without access to the gold passages, such as "What is the weather in Volcano, CA?" so performance will be lower without using gold passages. We also note that some MSMARCO questions cannot be answered using Wikipedia alone. Here, RAG can rely on parametric knowledge to generate reasonable responses.

RAG 模型可以超越简单的抽取式问答, 并通过自由形式的抽象文本生成来回答问题。为了在知识密集型环境中测试 RAG 的自然语言生成 (NLG) 能力, 我们使用了 MSMARCO NLG 任务 v2.1 [43]。该任务包括问题、每个问题从搜索引擎检索到的十个黄金段落, 以及从检索到的段落中标注的完整句子答案。我们不使用提供的段落, 仅使用问题和答案, 将 MSMARCO 视为开放领域的抽象问答任务。MSMARCO 中有一些问题无法在不访问黄金段落的情况下以匹配参考答案的方式回答, 例如" 加利福尼亚州火山的天气如何? ", 因此在不使用黄金段落的情况下, 性能会较低。我们还注意到, 一些 MSMARCO 问题无法仅使用维基百科来回答。在这里, RAG 可以依赖参数化知识生成合理的回答。

3.3 Jeopardy Question Generation **危险问题生成**

To evaluate RAG's generation abilities in a non-QA setting, we study open-domain question generation. Rather than use questions from standard open-domain QA tasks, which typically consist of short, simple questions, we propose the more demanding task of generating Jeopardy questions. Jeopardy is an unusual format that consists of trying to guess an entity from a fact about that entity. For example, "The World Cup" is the answer to the question "In 1986 Mexico scored as the first country to host this international sports competition twice." As Jeopardy questions are precise, factual statements, generating Jeopardy questions conditioned on their answer entities constitutes a challenging knowledge-intensive generation task.

为了在非问答环境中评估 RAG 的生成能力, 我们研究了开放领域问题生成。与使用标准开放领域问答任务中的问题不同, 这些任务通常由简短、简单的问题组成, 我们提出了更具挑战性的生成危险问题的任务。危险问题是一种不寻常的格式, 它要求从关于某个实体的事实中猜测该实体。例如, " 世界杯" 是问题"1986 年, 墨西哥成为第一个两次主办这一国际体育赛事的国家" 的答案。由于危险问题是精确的事实陈述, 生成以答案实体为条件的危险问题构成了一个具有挑战性的知识密集型生成任务。

We use the splits from SearchQA [10], with 100K train, 14K dev, and 27K test examples. As this is a new task, we train a BART model for comparison. Following [67], we evaluate using the SQuAD-tuned Q-BLEU-1 metric [42]. Q-BLEU is a variant of BLEU with a higher weight for matching entities and has higher correlation with human judgment for question generation than standard metrics. We also perform two human evaluations, one to assess generation factuality, and one for specificity. We define factuality as whether a statement can be corroborated by trusted external sources, and specificity as high mutual dependence between the input and output [33]. We follow best practice and use pairwise comparative evaluation [34]. Evaluators are shown an answer and two generated questions, one from BART and one from RAG. They are then asked to pick one of four options—quuestion A is better, question B is better, both are good, or neither is good.

我们使用 SearchQA [10] 的分割, 包含 10 万训练样本、1.4 万开发样本和 2.7 万测试样本。由于这是一个新任务, 我们训练了一个 BART 模型进行比较。根据 [67], 我们使用 SQuAD 调整的 Q-BLEU-1 指标 [42] 进行评估。Q-BLEU 是 BLEU 的一个变体, 对匹配实体赋予更高的权重, 并且在问题生成方面与人类判断的相关性高于标准指标。我们还进行了两项人工评估, 一项用于评估生成的事实性, 另一项用于评估特异性。我们将事实性定义为陈述是否可以通过可信的外部来源得到证实, 特异性定义为输入和输出之间的高度相互依赖性 [33]。我们遵循最佳实践, 使用成对比较评估 [34]。评估者会看到一个答案和两个生成的问题, 一个来自 BART, 一个来自 RAG。然后他们被要求从四个选项中选择一个——问题 A 更好, 问题 B 更好, 两者都好, 或者两者都不好。

3.4 Fact Verification **事实验证**

FEVER [56] requires classifying whether a natural language claim is supported or refuted by Wikipedia, or whether there is not enough information to decide. The task requires retrieving evidence from Wikipedia relating to the claim and then reasoning over this evidence to classify whether the claim is true, false, or unverifiable from Wikipedia alone. FEVER is a retrieval problem coupled with an challenging entailment reasoning task. It also provides an appropriate testbed for exploring the RAG models' ability to handle classification rather than generation. We map FEVER class labels (supports, refutes, or not enough info) to single output tokens and directly train with claim-class pairs. Crucially, unlike most other approaches to FEVER, we do not use supervision on retrieved evidence. In many real-world applications, retrieval supervision signals aren't available, and models that do not require such supervision will be applicable to a wider range of tasks. We explore two variants: the standard 3-way classification task (supports/refutes/not enough info) and the 2-way (supports/refutes) task studied in Thorne and Vlachos [57]. In both cases we report label accuracy.

FEVER [56] 要求分类自然语言声明是否被维基百科支持或反驳, 或者是否没有足够的信息来决定。该任务需要从维基百科中检索与声明相关的证据, 然后对这些证据进行推理, 以分类声明是真实的、虚假的, 还是仅凭维基百科无法验证的。FEVER 是一个检索问题, 结合了一个具有挑战性的蕴含推理任务。它还为探索 RAG 模型处理分类而非生成的能力提供了一个合适的测试平台。我们将 FEVER 类别标签 (支持、反驳或信息不足) 映射到单个输出标记, 并直接使用声明-类别对进行训练。关键的是, 与大多数其他 FEVER 方法不同, 我们不使用检索证据的监督。在许多实际应用中, 检索监督信号不可用, 而不需要这种监督的模型将适用于更广泛的任务。我们探索了两个变体: 标准的三分类任务 (支持/反驳/信息不足) 和 Thorne 和 Vlachos [57] 中研究的二分类任务 (支持/反驳)。在这两种情况下, 我们都报告标签准确性。

4 Results **结果**

4.1 Open-domain Question Answering **开放域问答**

Table 1: Open-Domain QA Test Scores. For TQA, left column uses the standard test set for Open- Domain QA, right column uses the TQA-Wiki test set. See Appendix for further details.
开放域问答测试分数。对于 TQA, 左列使用开放域问答的标准测试集, 右列使用 TQA-Wiki 测试集。更多详情请参见附录 D。

	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [?]	34.5	- /50.1	37.4	-
	T5-11B+SSM[?]	36.6	- /60.5	44.7	-
Open Book	REALM [?]	40.4	- / -	40.7	46.8
	DPR [?]	41.5	57.9/ -	41.1	50.6
	RAG-Tok.	44.1	55.2/66.1	45.5	50.0
	RAG-Seq.	44.5	56.8/68.0	45.2	52.2

Table 2: Generation and classification Test Scores. MS-MARCO SotA is [?], FEVER-3 is [?] and FEVER-2 is [?] *Uses gold context/evidence. Best model without gold access underlined.
生成和分类测试分数。MS-MARCO SotA 是 [?], FEVER-3 是 [?], FEVER-2 是 [?] * 使用黄金背景/证据。没有黄金访问权限的最佳模型标下划线。

Model	Jeopardy B-1	Jeopardy QB-1	MSMARCO R-L	MSMARCO B-1	FVR3 Label Acc.	FVR2 Label Acc.
SotA	-	-	49.8*	49.9*	76.8	92.2*
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok.	17.3	22.2	40.1	41.5	72.5	<u>89.5</u>
RAG-Seq.	14.7	21.4	<u>40.8</u>	<u>44.2</u>		

Table 1 shows results for RAG along with state-of-the-art models. On all four open-domain QA tasks, RAG sets a new state of the art (only on the T5-comparable split for TQA). RAG combines the generation flexibility of the "closed-book" (parametric only) approaches and the performance of "open-book" retrieval-based approaches. Unlike REALM and T5+SSM, RAG enjoys strong results without expensive, specialized "salient span masking" pre-training [20]. It is worth noting that RAG's retriever is initialized using DPR's retriever, which uses retrieval supervision on Natural Questions and TriviaQA. RAG compares favourably to the DPR QA system, which uses a BERT-based "cross-encoder" to re-rank documents, along with an extractive reader. RAG demonstrates that neither a re-ranker nor extractive reader is necessary for state-of-the-art performance.

表 1 显示了 RAG 以及最先进模型的结果。在所有四个开放域问答任务中, RAG 都设定了新的最先进水平 (仅在 T5 可比较的 TQA 分割上)。RAG 结合了" 闭卷"(仅参数化) 方法的生成灵活性和基于检索的" 开卷" 方法的性能。与 REALM 和 T5+SSM 不同, RAG 在没有昂贵、专门的" 显著跨度掩码" 预训练的情况下取得了强劲的结果 [20]。值得注意的是, RAG 的检索器是使用 DPR 的检索器初始化的, 该检索器在 Natural Questions 和 TriviaQA 上使用检索监督。RAG 与 DPR 问答系统相比表现良好, 后者使用基于 BERT 的" 交叉编码器" 对文档进行重新排序, 并使用提取式阅读器。RAG 表明, 重新排序器或提取式阅读器对于最先进的性能都不是必需的。

There are several advantages to generating answers even when it is possible to extract them. Documents with clues about the answer but do not contain the answer verbatim can still contribute towards a correct answer being generated, which is not possible with standard extractive approaches, leading to more effective marginalization over documents. Furthermore, RAG can generate correct answers even when the correct answer is not in any retrieved document, achieving 11.8% accuracy in such cases for NQ, where an extractive model would score 0%.

即使在可以提取答案的情况下, 生成答案也有几个优点。包含答案线索但不包含逐字答案的文档仍然可以有助于生成正确答案, 这是标准提取方法无法实现的, 从而在文档上实现更有效的边缘化。此外, 即使正确答案不在任何检索到的文档中, RAG 也能生成正确答案, 在 NQ 中, 这种情况下的准确率为 11.8%, 而提取模型的准确率为 0%。

4.2 Abstractive Question Answering **抽象问答**

As shown in Table 2, RAG-Sequence outperforms BART on Open MS-MARCO NLG by 2.6 Bleu points and 2.6 Rouge-L points. RAG approaches state-of-the-art model performance, which is impressive given that (i) those models access gold passages with specific information required to generate the reference answer, (ii) many questions are unanswerable without the gold passages, and (iii) not all questions are answerable from Wikipedia alone. Table 3 shows some generated answers from our models. Qualitatively, we find that RAG models hallucinate less and generate factually correct text more often than BART. Later, we also show that RAG generations are more diverse than BART generations (see §4.5).

如表 2 所示, RAG-Sequence 在 Open MS-MARCO NLG 上比 BART 高出 2.6 个 Bleu 点和 2.6 个 Rouge-L 点。RAG 方法接近最先进的模型性能, 这令人印象深刻, 因为 (i) 这些模型访问了生成参考答案所需的特定信息的黄金段落, (ii) 许多问题在没有黄金段落的情况下是无法回答的, (iii) 并非所有问题都可以仅从维基百科中回答。表 3 展示了我们模型生成的一些答案。从质量上看, 我们发现 RAG 模型比 BART 更少产生幻觉, 并且更频繁地生成事实正确的文本。稍后, 我们还展示了 RAG 生成的内容比 BART 生成的内容更加多样化 (见 §4.5)。

4.3 Jeopardy Question Generation **危险问题生成**

Table 2 shows that RAG-Token performs better than RAG-Sequence on Jeopardy question generation, with both models outperforming BART on Q-BLEU-1. Table 4 shows human evaluation results, over 452 pairs of generations from BART and RAG-Token. Evaluators indicated that BART was more factual than RAG in only 7.1% of cases, while RAG was more factual in 42.7% of cases, and both RAG and BART were factual in a further 17% of cases, clearly demonstrating the effectiveness of RAG on the task over a state-of-the-art generation model. Evaluators also find RAG generations to be more specific by a large margin. Table 3 shows typical generations from each model.

表 2 显示, RAG-Token 在危险问题生成上表现优于 RAG-Sequence, 两种模型在 Q-BLEU-1 上都优于 BART。表 4 展示了人类评估结果, 超过 452 对 BART 和 RAG-Token 生成的样本。评估者指出, BART 仅在 7.1% 的情况下比 RAG 更符合事实, 而 RAG 在 42.7% 的情况下更符合事实, 两者在 17% 的情况下都符合事实, 这清楚地展示了 RAG 在该任务上相对于最先进生成模型的有效性。评估者还发现 RAG 生成的内容在特异性上大幅领先。表 3 展示了每个模型的典型生成内容。

Jeopardy questions often contain two separate pieces of information, and RAG-Token may perform best because it can generate responses that combine content from several documents. Figure 2 shows an example. When generating “Sun”, the posterior is high for document 2 which mentions “The Sun Also Rises”. Similarly, document 1 dominates the posterior when “A Farewell to Arms” is generated. Intriguingly, after the first token of each book is generated, the document posterior flattens. This observation suggests that the generator can complete the titles without depending on specific documents. In other words, the model’s parametric knowledge is sufficient to complete the titles.

危险问题通常包含两个独立的信息片段, RAG-Token 可能表现最佳, 因为它可以生成结合多个文档内容的响应。图 2 展示了一个例子。当生成” 太阳” 时, 提到” 太阳照常升起” 的文档 2 的后验概率较高。同样, 当生成” 永别了, 武器” 时, 文档 1 的后验概率占主导地位。有趣的是, 在生成每本书的第一个词后, 文档的后验概率趋于平缓。这一观察表明生成器可以在不依赖特定文档的情况下完成标题。

Table 3: Examples from generation tasks. RAG models generate more specific and factually accurate responses. '?' indicates factually incorrect responses, * indicates partially correct responses.
生成任务中的示例。RAG 模型生成更具体且事实准确的响应。"? " 表示事实不正确的响应, "*" 表示部分正确的响应。

Task	Input	Model	Generation
MS-MARCO	define middle ear	BART	?The middle ear is the part of the ear between the middle ear and the nose.
		RAG-T	The middle ear is the portion of the ear internal to the eardrum.
		RAG-S	The middle ear includes the tympanic cavity and the three ossicles.
	what currency needed in scotland	BART	The currency needed in Scotland is Pound sterling.
		RAG-T	Pound is the currency needed in Scotland.
		RAG-S	The currency needed in Scotland is the pound sterling.
Jeopardy Question Gener-ation	Washington	BART	?This state has the largest number of counties in the U.S.
		RAG-T	It's the only U.S. state named for a U.S. president
		RAG-S	It's the state where you'll find Mount Rainier National Park
	The Divine Comedy	BART	*This epic poem by Dante is divided into 3 parts: the Inferno, the Purgatorio & the Purgatorio
		RAG-T	Dante's "Inferno" is the first part of this epic poem
		RAG-S	This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso"

We find evidence for this hypothesis by feeding the BART-only baseline with the partial decoding "The Sun. BART completes the generation "The Sun Also Rises" is a novel by this author of "The Sun Also Rises" indicating the title "The Sun Also Rises" is stored in BART's parameters. Similarly, BART will complete the partial decoding "The Sun Also Rises" is a novel by this author of "A with "The Sun Also Rises" is a novel by this author of "A Farewell to Arms". This example shows how parametric and non-parametric memories work together—the non-parametric component helps to guide the generation, drawing out specific knowledge stored in the parametric memory.

换句话说, 模型的参数知识足以完成标题。我们通过向仅使用 BART 的基线模型提供部分解码" 太阳。BART 完成了生成" 太阳照常升起" 是这位作者的小说" 太阳照常升起" 表明标题" 太阳照常升起" 存储在 BART 的参数中。同样, BART 将完成部分解码" 太阳照常升起" 是这位作者的小说"A" 与" 太阳照常升起" 是这位作者的小说" 永别了, 武器"。这个例子展示了参数和非参数记忆如何协同工作——非参数组件帮助引导生成, 提取存储在参数记忆中的特定知识。

Table 4: Human assessments for the Jeopardy Question Generation Task.
Jeopardy 问题生成任务的人工评估。

	Factuality	Specificity
BART better	7.1%	16.8%
RAG better	42.7%	37.4%
Both good	11.7%	11.8%
Both poor	17.7%	6.9%
No majority	20.8%	20.1%

Table 5: Ratio of distinct to total tri-grams for generation tasks.

	MSMARCO	Jeopardy QGen
Gold	89.6%	90.0%
BART	70.7%	32.4%
RAG-Token	77.8%	46.8%
RAG-Seq.	83.5%	53.8%

4.4 Fact Verification 事实验证

Table 2 shows our results on FEVER. For 3-way classification, RAG scores are within 4.3% of state-of-the-art models, which are complex pipeline systems with domain-specific architectures and substantial engineering, trained using intermediate retrieval supervision, which RAG does not require.

表 2 展示了我们在 FEVER 上的结果。对于三分类任务, RAG 的得分与最先进的模型相差在 4.3% 以内, 这些模型是具有特定领域架构和大量工程的复杂管道系统, 使用中间检索监督进行训练, 而 RAG 不需要这些。

Document 1: his works are considered classics of American literature ... His wartime experiences formed the basis for his novel "A Farewell to Arms" (1929) ... Document 2: ... artists of the 1920s "Lost Generation" expatriate community. His debut novel, "The Sun Also Rises", was published in 1926.

文档 1: 他的作品被认为是美国文学的经典之作……他的战时经历构成了他的小说《永别了, 武器》(1929 年) 的基础……文档 2:……1920 年代" 迷惘的一代" 流亡社区的艺术家。他的处女作《太阳照常升起》于 1926 年出版。

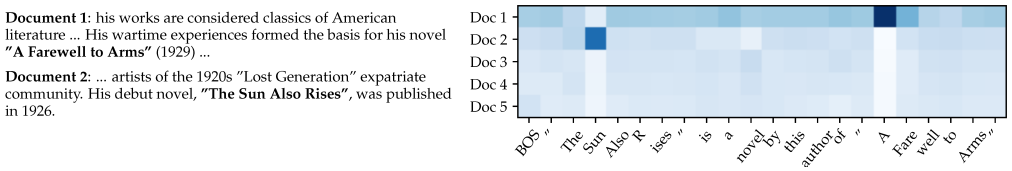

Figure 2: RAG-Token document posterior $p(_i|, y_i, y_{-i})$ for each generated token for input "Hemingway" for Jeopardy generation with 5 retrieved documents. The posterior for document 1 is high when generating "A Farewell to Arms" and for document 2 when generating "The Sun Also Rises".
RAG-Token 文档后验 $p(_i|, y_i, y_{-i})$ 为每个生成的标记, 输入"Hemingway" 用于 Jeopardy 生成, 检索了 5 个文档。生成" 永别了, 武器" 时文档 1 的后验较高, 生成" 太阳照常升起" 时文档 2 的后验较高。

For 2-way classification, we compare against Thorne and Vlachos [57], who train RoBERTa [35] to classify the claim as true or false given the gold evidence sentence. RAG achieves an accuracy within 2.7% of this model, despite being supplied with only the claim and retrieving its own evidence. We also analyze whether documents retrieved by RAG correspond to documents annotated as gold evidence in FEVER. We calculate the overlap in article titles between the top k documents retrieved by RAG and gold evidence annotations. We find that the top retrieved document is from a gold article in 71% of cases, and a gold article is present in the top 10 retrieved articles in 90% of cases.

对于二分类任务, 我们与 Thorne 和 Vlachos [57] 进行了比较, 他们训练 RoBERTa [35] 根据给定的黄金证据句子将声明分类为真或假。尽管 RAG 仅提供了声明并检索了自己的证据, 但其准确率仍在该模型的 2.7% 以内。我们还分析了 RAG 检索到的文档是否与 FEVER 中标注为黄金证据的文档相对应。我们计算了 RAG 检索到的前 k 个文档与黄金证据标注之间的文章标题重叠率。我们发现, 在 71% 的情况下, 检索到的前一个文档来自黄金文章, 而在 90% 的情况下, 黄金文章出现在检索到的前 10 篇文章中。

4.5 Additional Results **额外结果**

Generation Diversity Section 4.3 shows that RAG models are more factual and specific than BART for Jeopardy question generation. Following recent work on diversity-promoting decoding [33, 59, 39], we also investigate generation diversity by calculating the ratio of distinct ngrams to total ngrams generated by different models. Table 5 shows that RAG-Sequence's generations are more diverse than RAG-Token's, and both are significantly more diverse than BART without needing any diversity-promoting decoding.

生成多样性第 4.3 节显示, RAG 模型在 Jeopardy 问题生成方面比 BART 更具事实性和特异性。根据最近关于促进多样性解码的研究 [33, 59, 39], 我们还通过计算不同模型生成的不同 ngram 与总 ngram 的比率来研究生成多样性。表 5 显示, RAG-Sequence 的生成比 RAG-Token 的生成更具多样性, 并且两者都比 BART 显著更具多样性, 而无需任何促进多样性解码。

Table 6: Ablations on the dev set. As FEVER is a classification task, both RAG models are equivalent.
在开发集上进行消融。由于 FEVER 是一项分类任务, 因此两种 RAG 模型都是等效的。

Model	NQ	TQA	WQ	CT	Jeopardy-QGen		MSMarco		FVR-3	FVR-2
		Exact Match			B-1	QB-1	R-L	B-1	Label Accuracy	
RAG-Token-BM25	29.7	41.5	32.1	33.1	17.5	22.3	55.5	48.4	75.1	91.6
RAG-Sequence-BM25	31.8	44.1	36.6	33.8	11.1	19.5	56.5	46.9		
RAG-Token-Frozen	37.8	50.1	37.1	51.1	16.7	21.7	55.9	49.4	72.9	89.4
RAG-Sequence-Frozen	41.2	52.1	41.8	52.6	11.8	19.6	56.7	47.3		
RAG-Token	43.5	54.8	46.5	51.9	17.9	22.6	56.2	49.4	74.5	90.6
RAG-Sequence	44.0	55.8	44.9	53.4	15.3	21.5	57.2	47.5		

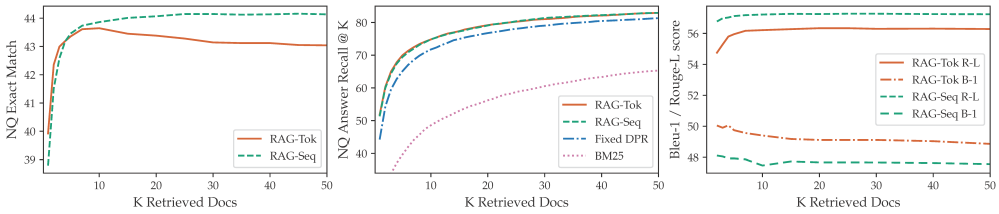

Figure 3: Left: NQ performance as more documents are retrieved. Center: Retrieval recall performance in NQ. Right: MS-MARCO Bleu-1 and Rouge-L as more documents are retrieved.
左: 随着检索更多文档, NQ 性能的变化。中:NQ 中的检索召回性能。右: 随着检索更多文档, MS-MARCO 的 Bleu-1 和 Rouge-L 的变化。

Retrieval Ablations A key feature of RAG is learning to retrieve relevant information for the task. To assess the effectiveness of the retrieval mechanism, we run ablations where we freeze the retriever during training. As shown in Table 6, learned retrieval improves results for all tasks. We compare RAG's dense retriever to a word overlap-based BM25 retriever [53]. Here, we replace RAG's retriever with a fixed BM25 system, and use BM25 retrieval scores as logits when calculating $p(z \mid x)$. Table 6 show the results. For FEVER, BM25 performs best, perhaps since FEVER claims are heavily entity-centric and thus well-suited for word overlap-based retrieval. Differentiable retrieval improves results on all other tasks, especially for Open-Domain QA, where it is crucial.

检索消融 RAG 的一个关键特征是学习为任务检索相关信息。为了评估检索机制的有效性, 我们进行了消融实验, 在训练期间冻结检索器。如表 6 所示, 学习到的检索提高了所有任务的结果。我们将 RAG 的密集检索器与基于词重叠的 BM25 检索器 [53] 进行了比较。在这里, 我们用固定的 BM25 系统替换 RAG 的检索器, 并在计算 $p(z \mid x)$ 时使用 BM25 检索分数作为 logits。表 6 显示了结果。对于 FEVER, BM25 表现最佳, 可能是因为 FEVER 声明高度以实体为中心, 因此非常适合基于词重叠的检索。可微检索提高了所有其他任务的结果, 尤其是对于开放域问答任务, 这一点至关重要。

Index hot-swapping An advantage of non-parametric memory models like RAG is that knowledge can be easily updated at test time. Parametric-only models like T5 or BART need further training to update their behavior as the world changes. To demonstrate, we build an index using the DrQA [5] Wikipedia dump from December 2016 and compare outputs from RAG using this index to the newer index from our main results (December 2018). We prepare a list of 82 world leaders who had changed between these dates and use a template "Who is {position}?" (e.g. "Who is the President of Peru?") to query our NQ RAG model with each index. RAG answers 70% correctly using the 2016 index for 2016 world leaders and 68% using the 2018 index for 2018 world leaders. Accuracy with mismatched indices is low (12% with the 2018 index and 2016 leaders, 4% with the 2016 index and 2018 leaders). This shows we can update RAG's world knowledge by simply replacing its non-parametric memory.

索引热交换像 RAG 这样的非参数记忆模型的一个优势是, 可以在测试时轻松更新知识。像 T5 或 BART 这样的纯参数模型需要进一步训练以随着世界的变化更新其行为。为了证明这一点, 我们使用 2016 年 12 月的 DrQA [5] 维基百科转储构建了一个索引, 并将使用此索引的 RAG 输出与主要结果中的较新索引 (2018 年 12 月) 进行了比较。我们准备了一份在这两个日期之间发生变化的 82 位世界领导人名单, 并使用模板" 谁是 {职位}? "(例如" 谁是秘鲁总统? ") 来查询我们的 NQ RAG 模型, 每个索引都使用。RAG 使用 2016 年索引对 2016 年世界领导人的回答正确率为 70%, 使用 2018 年索引对 2018 年世界领导人的回答正确率为 68%。使用不匹配索引的准确率较低 (使用 2018 年索引和 2016 年领导人为 12%, 使用 2016 年索引和 2018 年领导人为 4%)。这表明我们可以通过简单地替换 RAG 的非参数记忆来更新其世界知识。

Effect of Retrieving more documents Models are trained with either 5 or 10 retrieved latent documents, and we do not observe significant differences in performance between them. We have the flexibility to adjust the number of retrieved documents at test time, which can affect performance and runtime. Figure 3 (left) shows that retrieving more documents at test time monotonically improves Open-domain QA results for RAG-Sequence, but performance peaks for RAG-Token at 10 retrieved documents. Figure 3 (right) shows that retrieving more documents leads to higher Rouge-L for RAG-Token at the expense of Bleu-1, but the effect is less pronounced for RAG-Sequence.

检索更多文档的效果模型使用 5 或 10 个检索到的潜在文档进行训练, 我们没有观察到它们之间的性能有显著差异。我们可以在测试时灵活调整检索到的文档数量, 这可能会影响性能和运行时间。图 3(左) 显示, 在测试时检索更多文档会单调地提高 RAG-Sequence 的开放域问答结果, 但 RAG-Token 在检索 10 个文档时性能达到峰值。图 3(右) 显示, 检索更多文档会提高 RAG-Token 的 Rouge-L, 但以牺牲 Bleu-1 为代价, 而 RAG-Sequence 的效果则不那么明显。

5 Related Work

Single-Task Retrieval Prior work has shown that retrieval improves performance across a variety of NLP tasks when considered in isolation. Such tasks include open-domain question answering [5, 29], fact checking [56], fact completion [48], long-form question answering [12], Wikipedia article generation [36], dialogue [41, 65, 9, 13], translation [17], and language modeling [19, 27]. Our work unifies previous successes in incorporating retrieval into individual tasks, showing that a single retrieval-based architecture is capable of achieving strong performance across several tasks.

单任务检索先前的工作表明, 当单独考虑时, 检索可以提高各种 NLP 任务的性能。这些任务包括开放域问答 [5, 29]、事实核查 [56]、事实补全 [48]、长形式问答 [12]、维基百科文章生成 [36]、对话 [41, 65, 9, 13]、翻译 [17] 和语言建模 [19, 27]。我们的工作统一了先前在将检索纳入单个任务中的成功经验, 表明基于检索的单一架构能够在多个任务中实现强大的性能。

General-Purpose Architectures for NLP Prior work on general-purpose architectures for NLP tasks has shown great success without the use of retrieval. A single, pre-trained language model has been shown to achieve strong performance on various classification tasks in the GLUE benchmarks [60, 61] after fine-tuning [49, 8]. GPT-2 [50] later showed that a single, left-to-right, pre-trained language model could achieve strong performance across both discriminative and generative tasks. For further improvement, BART [32] and T5 [51, 52] propose a single, pre-trained encoder-decoder model that leverages bi-directional attention to achieve stronger performance on discriminative and generative tasks. Our work aims to expand the space of possible tasks with a single, unified architecture, by learning a retrieval module to augment pre-trained, generative language models.

通用 NLP 架构先前关于 NLP 任务的通用架构的工作在不使用检索的情况下取得了巨大成功。单一的预训练语言模型在 GLUE 基准测试中的各种分类任务上经过微调后表现出色 [60, 61][49, 8]。GPT-2[50] 后来表明, 单一的从左到右的预训练语言模型可以在判别性和生成性任务中实现强大的性能。为了进一步改进, BART[32] 和 T5[51, 52] 提出了一个单一的预训练编码器-解码器模型, 利用双向注意力在判别性和生成性任务中实现更强的性能。我们的工作旨在通过学习检索模块来增强预训练的生成性语言模型, 从而扩展单一统一架构可能执行的任务范围。

Learned Retrieval There is significant work on learning to retrieve documents in information retrieval, more recently with pre-trained, neural language models [44, 26] similar to ours. Some work optimizes the retrieval module to aid in a specific, downstream task such as question answering, using search [46], reinforcement learning [6, 63, 62], or a latent variable approach [31, 20] as in our work. These successes leverage different retrieval-based architectures and optimization techniques to achieve strong performance on a single task, while we show that a single retrieval-based architecture can be fine-tuned for strong performance on a variety of tasks.

学习检索在信息检索中, 学习检索文档的工作很多, 最近使用与我们类似的预训练神经语言模型 [44, 26]。一些工作优化检索模块以帮助特定的下游任务, 如问答, 使用搜索 [46]、强化学习 [6, 63, 62] 或潜在变量方法 [31, 20](如我们的工作)。这些成功利用不同的基于检索的架构和优化技术在单一任务上实现强大的性能, 而我们展示了基于检索的单一架构可以通过微调在多种任务上实现强大的性能。

Memory-based Architectures Our document index can be seen as a large external memory for neural networks to attend to, analogous to memory networks [64, 55]. Concurrent work [14] learns to retrieve a trained embedding for each entity in the input, rather than to retrieve raw text as in our work. Other work improves the ability of dialog models to generate factual text by attending over fact embeddings [9, 13] or, closer to our work, over retrieved text directly [15]. A key feature of our memory is that it is comprised of raw text rather distributed representations, which makes the memory both (i) human-readable, lending a form of interpretability to our model, and (ii) human-writable, enabling us to dynamically update the model's memory by editing the document index.

基于记忆的架构我们的文档索引可以被视为神经网络关注的大型外部记忆, 类似于记忆网络 [64, 55]。同时的工作 [14] 学习为输入中的每个实体检索训练好的嵌入, 而不是像我们的工作那样检索原始文本。其他工作通过关注事实嵌入 [9, 13] 或更接近我们的工作, 直接检索文本 [15], 提高了对话模型生成事实文本的能力。我们记忆的一个关键特征是它由原始文本而非分布式表示组成, 这使得记忆既 (i) 可读, 为我们的模型提供了一种可解释性, 又 (ii) 可写, 使我们能够通过编辑文档索引动态更新模型的记忆。

Retrieve-and-Edit approaches Our method shares some similarities with retrieve-and-edit style approaches, where a similar training input-output pair is retrieved for a given input, and then edited to provide a final output. These approaches have proved successful in a number of domains including Machine Translation [18, 22] and Semantic Parsing [21]. Our approach does have several differences, including less of emphasis on lightly editing a retrieved item, but on aggregating content from several pieces of retrieved content, as well as learning latent retrieval, and retrieving evidence documents rather than related training pairs. This said, RAG techniques may work well in these settings, and could represent promising future work.

检索与编辑方法我们的方法与检索与编辑风格的方法有一些相似之处, 其中为给定输入检索类似的训练输入-输出对, 然后进行编辑以提供最终输出。这些方法在包括机器翻译 [18, 22] 和语义解析 [21] 在内的多个领域已被证明是成功的。我们的方法确实有几个不同之处, 包括较少强调对检索项进行轻微编辑, 而是从多个检索内容中聚合内容, 以及学习潜在检索, 并检索证据文档而非相关的训练对。尽管如此, RAG 技术在这些环境中可能表现良好, 并可能代表有前途的未来工作。

6 Discussion

In this work, we presented hybrid generation models with access to parametric and non-parametric memory. We showed that our RAG models obtain state of the art results on open-domain QA. We found that people prefer RAG's generation over purely parametric BART, finding RAG more factual and specific. We conducted an thorough investigation of the learned retrieval component, validating its effectiveness, and we illustrated how the retrieval index can be hot-swapped to update the model without requiring any retraining. In future work, it may be fruitful to investigate if the two components can be jointly pre-trained from scratch, either with a denoising objective similar to BART or some another objective. Our work opens up new research directions on how parametric and non-parametric memories interact and how to most effectively combine them, showing promise in being applied to a wide variety of NLP tasks.

在这项工作中, 我们提出了可以访问参数化和非参数化记忆的混合生成模型。我们展示了我们的 RAG 模型在开放域问答上取得了最先进的结果。我们发现人们更喜欢 RAG 的生成, 而不是纯参数化的 BART, 认为 RAG 更事实和具体。我们对学习的检索组件进行了彻底的调查, 验证了其有效性, 并说明了如何热交换检索索引以更新模型而无需任何重新训练。在未来的工作中, 研究这两个组件是否可以联合从头开始预训练, 无论是使用类似于 BART 的去噪目标还是其他目标, 可能会富有成果。我们的工作开辟了关于参数化和非参数化记忆如何相互作用以及如何最有效地结合它们的新研究方向, 展示了在广泛 NLP 任务中应用的前景。

Broader Impact 更广泛的影响

This work offers several positive societal benefits over previous work: the fact that it is more strongly grounded in real factual knowledge (in this case Wikipedia) makes it "hallucinate" less with generations that are more factual, and offers more control and interpretability. RAG could be employed in a wide variety of scenarios with direct benefit to society, for example by endowing it with a medical index and asking it open-domain questions on that topic, or by helping people be more effective at their jobs.

这项工作比之前的工作提供了几个积极的社会效益: 它更强烈地基于真实的事实知识 (在这种情况下是维基百科), 使其生成的" 幻觉" 更少, 生成的内容更事实, 并提供了更多的控制和可解释性。RAG 可以在多种场景中应用, 直接造福社会, 例如通过赋予它医学索引并询问其开放域问题, 或帮助人们更有效地完成工作。

With these advantages also come potential downsides: Wikipedia, or any potential external knowledge source, will probably never be entirely factual and completely devoid of bias. Since RAG can be employed as a language model, similar concerns as for GPT-2 [50] are valid here, although arguably to a lesser extent, including that it might be used to generate abuse, faked or misleading content in the news or on social media; to impersonate others; or to automate the production of spam/phishing content [54]. Advanced language models may also lead to the automation of various jobs in the coming decades [16]. In order to mitigate these risks, AI systems could be employed to fight against misleading content and automated spam/phishing.

这些优势也伴随着潜在的缺点: 维基百科或任何潜在的外部知识源可能永远不会完全真实且完全没有偏见。由于 RAG 可以用作语言模型, 类似于 GPT-2 [50] 的担忧在这里也是有效的, 尽管可以说程度较轻, 包括它可能被用来生成滥用、伪造或误导性的新闻或社交媒体内容; 冒充他人; 或自动化生产垃圾邮件/钓鱼内容 [54]。高级语言模型也可能在未来几十年内导致各种工作的自动化 [16]。为了减轻这些风险, 可以部署 AI 系统来对抗误导性内容和自动化垃圾邮件/钓鱼。

Acknowledgments

The authors would like to thank the reviewers for their thoughtful and constructive feedback on this paper, as well as HuggingFace for their help in open-sourcing code to run RAG models. The authors would also like to thank Kyunghyun Cho and Sewon Min for productive discussions and advice.

作者感谢审稿人对本文的深思熟虑和建设性反馈, 以及 HuggingFace 在开源运行 RAG 模型代码方面的帮助。作者还要感谢 Kyunghyun Cho 和 Sewon Min 的富有成效的讨论和建议。

Funding Disclosure

EP thanks supports from the NSF Graduate Research Fellowship. PL is supported by the FAIR PhD program. This work was funded by Facebook.

EP 感谢 NSF 研究生研究奖学金的支持。PL 由 FAIR 博士项目支持。这项工作由 Facebook 资助。

Appendices for RAG for Knowledge-Intensive NLP Tasks
知识密集型 NLP 任务的检索增强生成: 附录

A Implementation Details 实现细节

For Open-domain QA we report test numbers using 15 retrieved documents for RAG-Token models. For RAG-Sequence models, we report test results using 50 retrieved documents, and we use the Thorough Decoding approach since answers are generally short. We use greedy decoding for QA as we did not find beam search improved results. For Open-MSMarco and Jeopardy question generation, we report test numbers using ten retrieved documents for both RAG-Token and RAG-Sequence, and we also train a BART-large model as a baseline. We use a beam size of four, and use the Fast Decoding approach for RAG-Sequence models, as Thorough Decoding did not improve performance.

对于开放域问答, 我们使用 15 个检索文档报告 RAG-Token 模型的测试数据。对于 RAG-Sequence 模型, 我们使用 50 个检索文档报告测试结果, 并且由于答案通常较短, 我们使用 Thorough Decoding 方法。我们使用贪婪解码进行问答, 因为我们没有发现束搜索能改善结果。对于 Open-MSMarco 和 Jeopardy 问题生成, 我们使用 10 个检索文档报告 RAG-Token 和 RAG-Sequence 的测试数据, 并且我们还训练了一个 BART-large 模型作为基线。我们使用束大小为 4, 并对 RAG-Sequence 模型使用 Fast Decoding 方法, 因为 Thorough Decoding 并未提高性能。

B Human Evaluation 人工评估

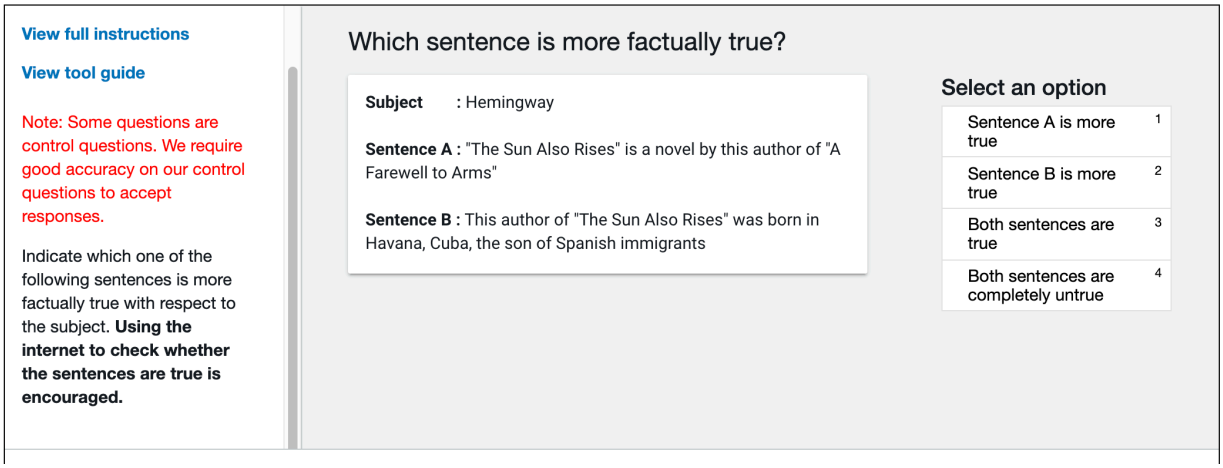

Figure 4: Annotation interface for human evaluation of factuality. A pop-out for detailed instructions and a worked example appear when clicking "view tool guide".
用于事实性人工评估的注释界面。点击" 查看工具指南" 时, 会弹出详细说明和工作示例。

Figure 4 shows the user interface for human evaluation. To avoid any biases for screen position, which model corresponded to sentence A and sentence B was randomly selected for each example. Annotators were encouraged to research the topic using the internet, and were given detailed instructions and worked examples in a full instructions tab. We included some gold sentences in order to assess the accuracy of the annotators. Two annotators did not perform well on these examples and their annotations were removed from the results.

图 4 显示了用于人工评估的用户界面。为了避免屏幕位置的偏见, 每个示例中句子 A 和句子 B 对应的模型是随机选择的。鼓励注释者使用互联网研究主题, 并在完整说明标签中提供了详细说明和工作示例。我们包含了一些黄金句子以评估注释者的准确性。两名注释者在这些示例上表现不佳, 他们的注释已从结果中移除。

C Training setup Details **训练设置详情**

We train all RAG models and BART baselines using Fairseq [45] [2] We train with mixed precision floating point arithmetic [40], distributing training across 8, 32GB NVIDIA V100 GPUs, though training and inference can be run on one GPU. We find that doing Maximum Inner Product Search with FAISS is sufficiently fast on CPU, so we store document index vectors on CPU, requiring $\sim$ 100 GB of CPU memory for all of Wikipedia. After submission, We have ported our code to HuggingFace Transformers [66] [3] , which achieves equivalent performance to the previous version but is a cleaner and easier to use implementation. This version is also open-sourced. We also compress the document index using FAISS' s compression tools, reducing the CPU memory requirement to 36GB. Scripts to run experiments with RAG can be found at https://github.com/huggingface/transformers/ blob/master/examples/rag/README. md and an interactive demo of a RAG model can be found at https://huggingface.co/rag/

我们使用 Fairseq [45] [2] 训练所有 RAG 模型和 BART 基线。我们使用混合精度浮点运算 [40] 进行训练, 将训练分布在 8 个 32GB NVIDIA V100 GPU 上, 尽管训练和推理可以在一个 GPU 上运行。我们发现使用 FAISS 进行最大内积搜索在 CPU 上足够快, 因此我们将文档索引向量存储在 CPU 上, 需要 $\sim$ 100 GB 的 CPU 内存来存储整个维基百科。提交后, 我们将代码移植到 HuggingFace Transformers [66][3] , 该版本实现了与之前版本相同的性能, 但实现更简洁且更易于使用。此版本也已开源。我们还使用 FAISS 的压缩工具压缩文档索引, 将 CPU 内存需求减少到 36GB。运行 RAG 实验的脚本可以在 https://github.com/huggingface/transformers/blob/master/examples/rag/README.md 找到, RAG 模型的交互式演示可以在 https://huggingface.co/rag/ 找到。

D Further Details on Open-Domain QA **开放域问答的进一步详情**

For open-domain QA, multiple answer annotations are often available for a given question. These answer annotations are exploited by extractive models during training as typically all the answer annotations are used to find matches within documents when preparing training data. For RAG, we also make use of multiple annotation examples for Natural Questions and WebQuestions by training the model with each(q, a)pair separately, leading to a small increase in accuracy. For TriviaQA, there are often many valid answers to a given question, some of which are not suitable training targets, such as emoji or spelling variants. For TriviaQA, we filter out answer candidates if they do not occur in top 1000 documents for the query.

对于开放域问答, 通常有多个答案注释可用于给定问题。这些答案注释在训练期间被提取模型利用, 通常在准备训练数据时使用所有答案注释来在文档中查找匹配项。对于 RAG, 我们还通过分别训练每个 (q, a) 对来利用 Natural Questions 和 WebQuestions 的多个注释示例, 从而略微提高了准确性。对于 TriviaQA, 通常有许多有效答案可用于给定问题, 其中一些不适合作为训练目标, 例如表情符号或拼写变体。对于 TriviaQA, 如果答案候选不在查询的前 1000 个文档中出现, 我们会将其过滤掉。

CuratedTrec preprocessing The answers for CuratedTrec are given in the form of regular expressions, which has been suggested as a reason why it is unsuitable for answer-generation models [20]. To overcome this, we use a pre-processing step where we first retrieve the top 1000 documents for each query, and use the answer that most frequently matches the regex pattern as the supervision target. If no matches are found, we resort to a simple heuristic: generate all possible permutations for each regex, replacing non-deterministic symbols in the regex nested tree structure with a whitespace.

CuratedTrec 预处理 CuratedTrec 的答案以正则表达式的形式给出, 这被认为是其不适合答案生成模型的原因 [20]。为了克服这一点, 我们使用了一个预处理步骤, 首先检索每个查询的前 1000 个文档, 并使用最频繁匹配正则表达式模式的答案作为监督目标。如果未找到匹配项, 我们采用简单的启发式方法: 为每个正则表达式生成所有可能的排列, 将正则表达式嵌套树结构中的非确定性符号替换为空格。

[2] https://github.com/pytorch/fairseq
[3] https://github.com/huggingface/transformers

TriviaQA Evaluation setups The open-domain QA community customarily uses public development datasets as test datasets, as test data for QA datasets is often restricted and dedicated to reading comprehension purposes. We report our results using the datasets splits used in DPR [26], which are consistent with common practice in Open-domain QA. For TriviaQA, this test dataset is the public TriviaQA Web Development split. Roberts et al. [52] used the TriviaQA official Wikipedia test set instead. Févry et al. [14] follow this convention in order to compare with Roberts et al. [52] (See appendix of [14]). We report results on both test sets to enable fair comparison to both approaches. We find that our performance is much higher using the official Wiki test set, rather than the more conventional open-domain test set, which we attribute to the official Wiki test set questions being simpler to answer from Wikipedia.

TriviaQA 评估设置开放域问答社区通常使用公共开发数据集作为测试数据集, 因为问答数据集的测试数据通常受到限制并专门用于阅读理解目的。我们使用 DPR [26] 中使用的数据集分割报告我们的结果, 这与开放域问答中的常见做法一致。对于 TriviaQA, 这个测试数据集是公共的 TriviaQA Web 开发分割。Roberts 等人 [52] 使用了 TriviaQA 官方的维基百科测试集。Févry 等人 [14] 遵循这一惯例, 以便与 Roberts 等人 [52] 进行比较 (参见 [14] 的附录)。我们报告了两个测试集的结果, 以便与这两种方法进行公平比较。我们发现, 使用官方的维基测试集时, 我们的性能要高得多, 而不是使用更传统的开放域测试集, 我们将此归因于官方的维基测试集问题更容易从维基百科中回答。

E Further Details on FEVER 的进一步细节

For FEVER classification, we follow the practice from [32], and first re-generate the claim, and then classify using the representation of the final hidden state, before finally marginalizing across documents to obtain the class probabilities. The FEVER task traditionally has two sub-tasks. The first is to classify the claim as either "Supported", "Refuted" or "Not Enough Info", which is the task we explore in the main paper. FEVER's other sub-task involves extracting sentences from Wikipedia as evidence supporting the classification prediction. As FEVER uses a different Wikipedia dump to us, directly tackling this task is not straightforward. We hope to address this in future work.

对于 FEVER 分类, 我们遵循 [32] 的做法, 首先重新生成声明, 然后使用最终隐藏状态的表示进行分类, 最后在文档之间进行边缘化以获得类别概率。FEVER 任务传统上有两个子任务。第一个是将声明分类为" 支持"、" 反驳" 或" 信息不足", 这是我们在主论文中探讨的任务。FEVER 的另一个子任务涉及从维基百科中提取句子作为支持分类预测的证据。由于 FEVER 使用了与我们不同的维基百科转储, 直接解决这个任务并不容易。我们希望在未来工作中解决这个问题。

F Null Document Probabilities 空文档概率

We experimented with adding "Null document" mechanism to RAG, similar to REALM [20] in order to model cases where no useful information could be retrieved for a given input. Here, if k documents were retrieved, we would additionally "retrieve" an empty document and predict a logit for the null document, before marginalizing over $k + 1$ predictions. We explored modelling this null document logit by learning (i) a document embedding for the null document, (ii) a static learnt bias term, or (iii) a neural network to predict the logit. We did not find that these improved performance, so in the interests of simplicity, we omit them. For Open MS-MARCO, where useful retrieved documents cannot always be retrieved, we observe that the model learns to always retrieve a particular set of documents for questions that are less likely to benefit from retrieval, suggesting that null document mechanisms may not be necessary for RAG.

我们尝试在 RAG 中添加" 空文档" 机制, 类似于 REALM [20], 以模拟无法为给定输入检索到有用信息的情况。在这里, 如果检索到 k 个文档, 我们还会" 检索" 一个空文档, 并在对 $k + 1$ 个预测进行边缘化之前预测空文档的对数概率。我们通过以下方式探索了建模这个空文档对数概率:(i) 学习空文档的文档嵌入, (ii) 静态学习偏置项, 或 (iii) 使用神经网络预测对数概率。我们发现这些方法并没有提高性能, 因此为了简化, 我们省略了它们。对于 Open MS-MARCO, 由于有用的检索文档并不总是能够检索到, 我们观察到模型学会了总是为不太可能从检索中受益的问题检索一组特定的文档, 这表明空文档机制可能对 RAG 来说是不必要的。

G Parameters 参数

Our RAG models contain the trainable parameters for the BERT-base query and document encoder of DPR, with 110M parameters each (although we do not train the document encoder ourselves) and 406M trainable parameters from BART-large, 406M parameters, making a total of 626M trainable parameters. The best performing "closed-book" (parametric only) open-domain QA model is T5-11B with 11 Billion trainable parameters. The T5 model with the closest number of parameters to our models is T5-large (770M parameters), which achieves a score of 28.9 EM on Natural Questions [52], substantially below the 44.5 that RAG-Sequence achieves, indicating that hybrid parametric/nonparametric models require far fewer trainable parameters for strong open-domain QA performance. The non-parametric memory index does not consist of trainable parameters, but does consists of 21M 728 dimensional vectors, consisting of 15.3B values. These can be easily be stored at 8-bit floating point precision to manage memory and disk footprints.

我们的 RAG 模型包含 DPR 的 BERT-base 查询和文档编码器的可训练参数, 每个编码器有 1.1 亿个参数 (尽管我们不自己训练文档编码器), 以及来自 BART-large 的 4.06 亿个可训练参数, 总共 6.26 亿个可训练参数。表现最好的" 闭卷"(仅参数化) 开放域问答模型是 T5-11B, 具有 110 亿个可训练参数。与我们模型参数数量最接近的 T5 模型是 T5-large(7.7 亿个参数), 在 Natural Questions [52] 上得分为 28.9 EM, 远低于 RAG-Sequence 的 44.5 分, 这表明混合参数化/非参数化模型在强开放域问答性能上需要的可训练参数要少得多。非参数化记忆索引不包含可训练参数, 但包含 2100 万个 728 维向量, 共计 153 亿个值。这些可以轻松地以 8 位浮点精度存储, 以管理内存和磁盘占用。

H Retrieval Collapse 检索崩溃

In preliminary experiments, we observed that for some tasks such as story generation [11], the retrieval component would "collapse" and learn to retrieve the same documents regardless of the input. In these cases, once retrieval had collapsed, the generator would learn to ignore the documents, and the RAG model would perform equivalently to BART. The collapse could be due to a less-explicit requirement for factual knowledge in some tasks, or the longer target sequences, which could result in less informative gradients for the retriever. Perez et al. [46] also found spurious retrieval results when optimizing a retrieval component in order to improve performance on downstream tasks.

在初步实验中, 我们观察到, 对于某些任务, 如故事生成 [11], 检索组件会" 崩溃", 并学会检索相同的文档, 而不管输入是什么。在这些情况下, 一旦检索崩溃, 生成器就会学会忽略这些文档, RAG 模型的表现将与 BART 相当。崩溃可能是由于某些任务对事实知识的要求不够明确, 或者目标序列较长, 这可能导致检索器的梯度信息较少。Perez 等人 [46] 在优化检索组件以提高下游任务性能时, 也发现了虚假的检索结果。

I Number of instances per dataset 每个数据集的实例数量

The number of training, development and test datapoints in each of our datasets is shown in Table 7.

我们每个数据集的训练、开发和测试数据点的数量如表 7 所示.

Table 7: Number of instances in the datasets used. *A hidden subset of this data is used for evaluation

Task	Train	Development	Test
Natural Questions	79169	8758	3611
TriviaQA	78786	8838	11314
WebQuestions	3418	362	2033
CuratedTrec	635	134	635
Jeopardy Question Generation	97392	13714	26849
MS-MARCO	153726	12468	101093*
FEVER-3-way	145450	10000	10000
FEVER-2-way	96966	6666	6666