**Active Retrieval Augmented Generation**

**主动检索增强生成**

**摘要**

尽管大模型有生成和理解语言的强大能力，他们有产生幻觉和事实上不准确输出的倾向。传统方法采用retrieve-and-generate 的范式进行检索增强，这种方式一般仅检索一次。本文提出了一种名为Forward-Looking Active Retrieval augmented generation (FLARE)的主动检索的方法，可以在生成过程中主动决定何时检索以及检索的内容。作者在 4 个长篇知识密集型生成任务/数据集上全面测试了 FLARE 和基线。 FLARE 在所有任务上都取得了卓越或有竞争力的性能，证明了该方法的有效性。

**1介绍**

生成式语言模型已成为许多自然语言处理 （NLP） 系统的基础组件，具有非凡的理解和语言生成能力。 尽管 LM 已经记住了在训练期间观察到的一些世界知识，他们仍然倾向于产生幻觉并创造虚构的内容。为了解决幻觉问题，一个有前途的方向是通过检索来增强生成，这涉及使用非参数检索组件来增强参数 LM，这些组件可以从外部知识资源（如文档语料库）中查找相关信息

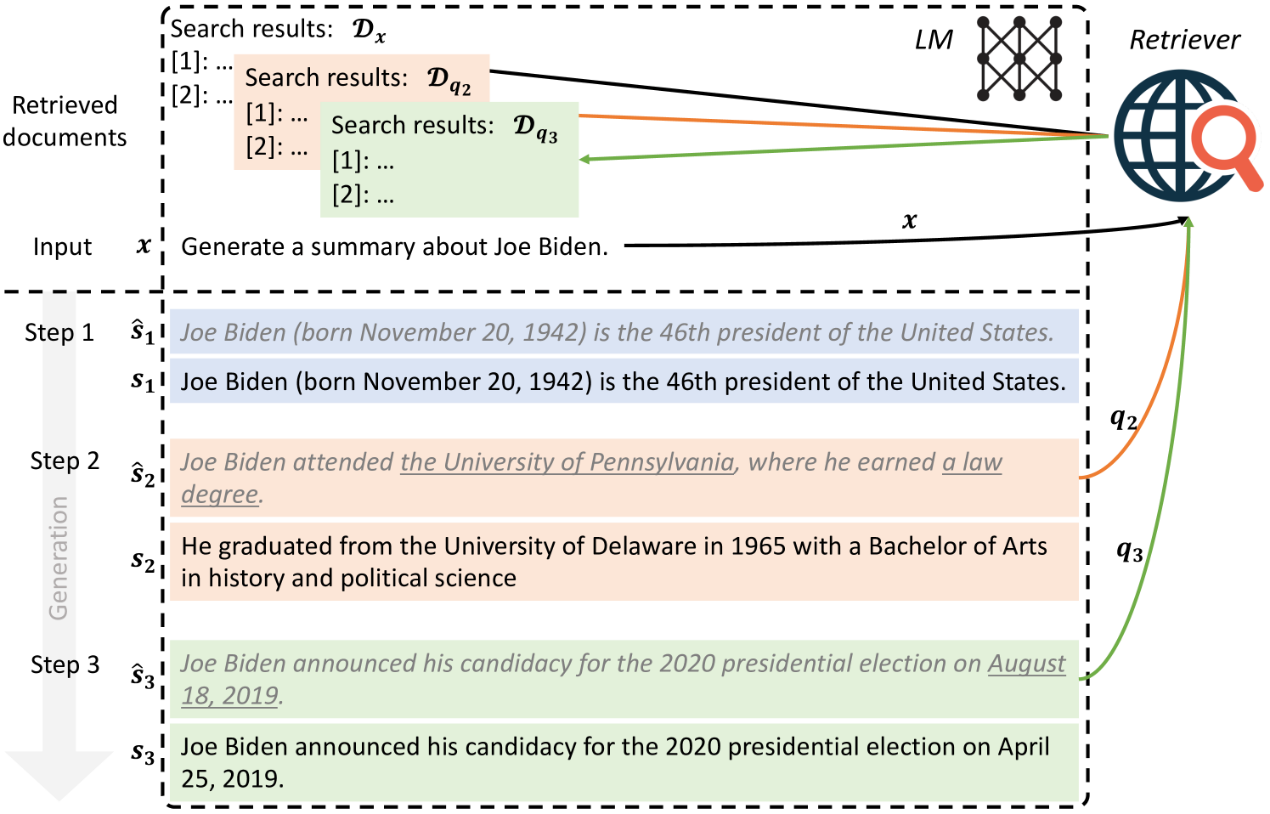


图1

检索增强的 LM 通常使用检索和生成设置，它们根据用户的输入（例如，问答中的问题）检索文档，然后根据检索到的文档生成完整的答案条件. 这些单次检索增强的 LM 已被发现优于纯参数 LM，特别是对于短格式知识密集型生成任务，其中信息需求在用户输入中是明确的，并且仅根据输入检索一次相关知识就足够了。

近年来，越来越强大的大型 LM 在涉及生成长格式输出（例如长格式 QA）、开放域总结和CoT推理等更复杂任务中表现出了能力。与短格式生成相比，长格式生成提供了复杂的信息需求，而这些需求并不总是仅从输入中显而易见。与人类在创建论文、论文或书籍等内容时逐渐收集信息的方式类似，使用 LM 的长篇生成需要在整个生成过程中收集多个知识。例如，在开放域摘要中，目标是通过从开放网络检索参考文献来生成有关特定主题的摘要。 基于主题名称（例如，Joe Biden）的初始检索可能无法涵盖所有方面和细节。 因此，在生成过程中根据需要检索额外信息至关重要，例如在生成某个方面（例如，乔·拜登的教育历史）或特定细节（例如，乔·拜登何时宣布参选 2020 年总统竞选）。

先前已经有了构建在整个生成过程中多次检索的系统。 这些尝试包括被动地利用过去的上下文（例如，以前的句子或标记）以固定间隔（例如，每个句子或每个几个标记）检索附加信息的这可能无法准确反映 LM 打算在未来生成的内容或在不适当的时间点检索的内容。 多跳 QA 中的一些工作通过将完整问题分解为子问题来解决多个信息需求，每个子问题都用于检索额外信息

作者考虑了一种新的范式，即主动检索增强生成。其关于何时检索的假设是，LM 只有在缺乏所需的知识时才应该检索信息，以避免在被动检索增强 LM 中发生的不必要或不适当的检索. 鉴于观察到大型 LM 往往经过良好校准，低概率/置信度通常表明缺乏知识，我们采用主动检索策略，仅在 LM 生成低概率代币时进行检索。在决定检索什么时，我们认为重要的是要考虑LM在未来打算产生什么，因为主动检索的目标是帮助之后的生成。 因此，作者建议通过生成一个临时的下一句来预测未来，将其用作检索相关文档的查询，然后在检索到的文档上重新生成下一句条件。 结合这两个方面，作者提出了**FLARE**。FLARE迭代生成一个临时的下一个句子，如果它包含低概率标记，则将其用作检索相关文档的查询，并重新生成下一个句子，直到到达末尾。

FLARE 适用于推理时的任何现有 LM，无需额外训练。 考虑到 GPT-3.5 取得的令人印象深刻的性能在各种任务中，我们检查了我们在 text-davinci-003 上的方法的有效性。 我们在 4 个不同的任务/数据集上评估 FLARE，这些任务/数据集涉及生成长输出，包括多跳 QA （2WikiMultihopQA）、常识推理 （StrategyQA）、长格式 QA （ASQA） 和开放域摘要。在所有任务中，与单次和多次检索基线相比，FLARE 实现了卓越或有竞争力的性能，证明了其方法的有效性和可推广性。

**2检索增强生成**

**2.1符号和定义**

在检索增强型 LM 中，LM 通常与可以检索文档列表的检索器配对

*D* ***q*** = ret(***q***)对于查询𝒒; LM利用条件𝒙和检索到的文件𝒟𝒒生成答案。由于作者专注于检查确定何时检索以及检索内容的各种方法，因此作者遵循现有方法在用户输入之前预置检索到的文档，以帮助将来生成基线和其方法公平比较。

2．2主动检索增强生成

为了帮助长格式生成检索，作者提出了主动检索增强生成。 它是一个通用框架，通过生成过程主动决定何时以及检索什么，从而导致检索和生成的交错。 正式地，在步骤中***t***(***t***≥1)，检索查询𝒒t是根据用户输入制定的𝒙和以前生成的输出***y****<t* = [***y***0*, ...,* ***y****t-*1]:

***q****t* = qry(***x****,* ***y****<t*)*,*

其中qry(⋅)是查询公式函数。 在生成开始时（t=1），上一代为空（𝒚<1=∅），用户输入用作初始查询 （𝒒1=𝒙). 给定检索到的文档𝒟𝒒，LM 会不断生成答案，直到触发下一次检索或到达终点：

***y****t* = LM([*D****q****t,* ***x****,* ***y****<t*])

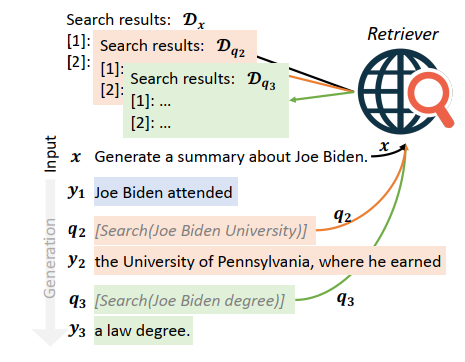
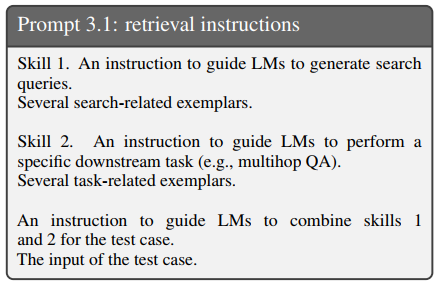
哪里𝒚t表示当前步骤中生成的token，LM 的输入是检索到的文档，用户输入和上一次的生成结果 在每个步骤中，都会丢弃以前检索到的文档，并且仅使用从当前步骤中检索到的文档来调节下一代，以防止达到 LM 的输入长度限制。

**3 FLARE：前瞻性主动检索增强生成**

作者认为：（1）LM应该只在没有必要知识的情况下检索信息，以避免不必要或不适当的检索，（2）检索查询应反映后代的意图。 因此，作者提出了两种前瞻性的主动检索增强生成（FLARE）方法来实现主动检索增强生成框架。第一种方法提示 LM 在必要时生成检索查询，同时使用检索鼓励指令生成答案指导. 第二种方法直接使用 LM 的生成作为搜索查询，它迭代生成下一个句子以深入了解未来的主题，如果存在不确定的标记，则检索相关文档以重新生成下一个句子。

3.1带有检索指令的FLARE

表达检索信息需求的一种直接方法是在需要其他信息时生成“[Search（query）]”，具体来说，对于下游任务，将与搜索相关的指令和示例放在开头作为技能 1，然后是下游任务的指令和示例作为技能 2。 给定一个测试用例，要求 LM 将技能 1 和 2 结合起来，在执行任务时生成搜索查询。



如上右图所示当 LM 生成“[Search（query）]”（以*灰色斜体*显示）时，停止生成并使用查询词来检索相关文档，这些文档在用户输入之前附加以帮助将来生成，直到生成下一个搜索查询或到达末尾。

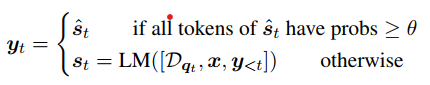
3.2直接 FLARE

由于作者无法微调黑盒 LM，因此作者发现了上一种框架的生成可能不可靠。 因此，我们提出了一种更直接的前瞻性主动检索方法，即使用下一句话来决定何时检索以及检索什么。

3.2.1基于置信度的主动检索

如笔记开始的图1所示，作者使用句子作为迭代的基础，是因为它们作为语义单位的重要性，既不像短语和段落那样太短也太长。 然而，值得注意的是，作者的方法也可以使用短语、段落或固定大小的窗口作为基础。

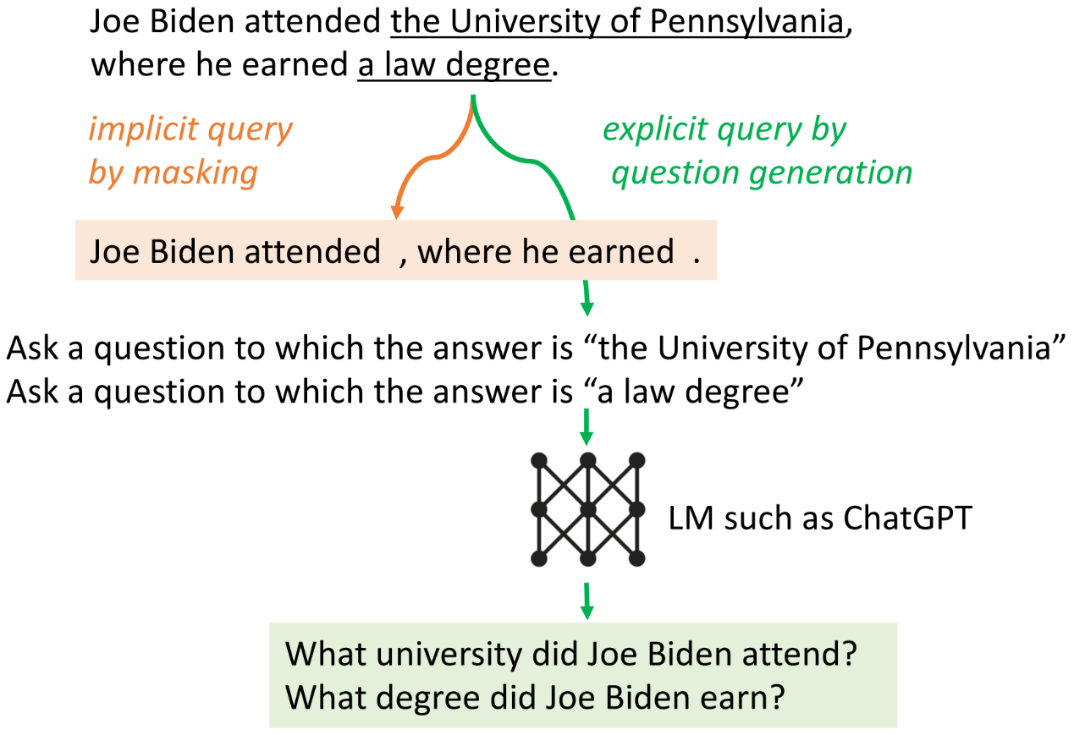
由于 LM 往往经过良好校准，因此低概率/置信度通常表明缺乏知识，如果有任何token的置信度低于阈值，将会主动触发检索，如下面的分段函数所示



3.2.2基于置信度的查询公式

执行检索的另一种方法是直接使用下一句话作为查询。 这与使用LM生成的假设性的标题或段落作为检索查询的现有方法具有相似处。作者将这些技术转移并推广到长格式生成场景中，在这些场景中，主动信息访问是必不可少的。

作者发现检索下一句比上一句要好得多， 但是，它存在使其中包含的错误永久化的风险。 例如，如果 LM 生成句子“Joe Biden 曾就读于宾夕法尼亚大学”而不是他就读于特拉华大学的正确事实，则使用这个错误的句子作为查询可能会提示检索器检索不相关的信息，这可能会误导后代。作者提出了两种简单的方法来克服这个问题，如下图所示。

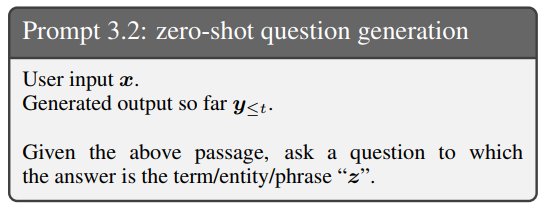


1）将掩码句子作为隐式查询

第一种方法屏蔽了低置信度的token，较高的阈值将会导致更激进的屏蔽。 这消除了句子中的潜在干扰，以提高检索的准确性。

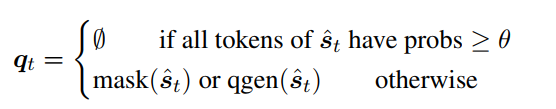
2）将问题生成为显式查询

这种方法生成针对低置信度跨度的显式问题. 例如，如果LM不确定“宾夕法尼亚大学”，那么像“乔·拜登就读于哪所大学”这样的问题可以帮助检索相关信息。因为手动将后续问题插入到下游任务示例中需要特定于任务的注释工作，所以作者开发了一种通用方法，可以在没有额外注释的情况下为低置信度跨度生成问题如下图所示：



作者使用每个生成的问题进行检索，并将返回的文档交错排列到一个排名列表中，以帮助未来的生成。

上面两种方法可以总结为如下的分段函数



3.3实施细节

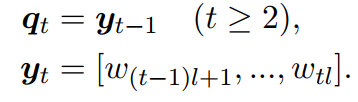
作者使用 GPT-3.5 LMs text-davinci-003 通过迭代查询其 API 来验证其方法。由于作者关注检索和生成的集成，因此作者使用现成的检索器，这些检索器将查询作为输入并返回相关文档的列表。 对于主要依赖维基百科知识的数据集，作者使用 Wikipedia dump并使用 BM25作为检索器。 对于依赖于开放网络知识的数据集，作者使用 Bing 搜索引擎作为我们的检索器。作者将多个检索到的文档根据其排名进行线性化，然后添加到用户输入的开头：

**4多时间检索基线**

现有的被动多时间检索增也可以使用我们的框架。接下来，将根据检索时间和内容正式介绍三个基线类别。这些基线并不是相应论文的精确复制品，因为许多设计选择与以前的工作不同，这使得直接比较变得不可能。作者排除了不相关的设计，并确保使用相同的设置来实现它们，唯一的变化是何时以及检索什么。

窗口

该方法触发检索每l个token，其中l表示窗口大小。从上一个窗口生成的token会被用作查询：



语句

方法触发了检索每个句子，并使用前一个句子作为查询，并且IRCoT属于 这个类别。

问题分解

该方法手动注释特定于任务的示例，以指导 LM 在生成输出时生成分解的子问题。

上述三种方法有明显的缺点（1）固定间隔方法使用以前生成的token作为查询，可能无法反映LM打算在未来生成的内容。（2） 以固定的时间间隔检索信息可能效率低下，因为它可能发生在不适当的时间点。（3）问题分解方法需要针对特定任务的提示工程，这限制了它们在新任务中的泛化性。

**5实验装置**

作者使用少量上下文学习评估 FLARE 对 4 种不同知识密集型任务的有效性。为了确保公平的比较，作者将使用相同的设置将 FLARE 的结果与基线进行比较，即相同的上下文示例、提示格式、检索器和文档语料库。

5.1 **Multihop QA**

作者使用2WikiMultihopQA其中包含来自维基百科文章的2-hop复杂问题，其需要撰写、比较或推理。作者使用正则表达式来提取最终的答案并且使用exact match，token-levelF1 ，precision和recall将其参考答案相比。

5.2 **Commonsense reasoning**

常识推理要求系统利用世界和常识知识来生成答案。作者使用StrategyQA作为测试平台，它是crowdsourced yes/no 问题的集合。作者提取是/否答案，并使用精确匹配将其与正确答案进行匹配。

5.3 **Long-form QA**

长篇 QA 旨在为寻求复杂信息的问题提供全面的答案。作者使用ASQA作为其测试平台，输入是具有多种解释的模棱两可的问题，输出是涵盖所有内容的综合答案。为了回答模棱两可的问题，系统必须首先确定可能的解释，然后为每个解释提供答案。 作者发现，在许多情况下，即使是人类也很难确定原始问题的哪个方面是模棱两可的。 因此，作者创建了另一个设置，在该设置中，我们提供了一个简短而通用的提示，以指导 LM 在生成解释和相应答案时保持正轨。 如“这个问题在所指的具体地点或地点方面是模棱两可的”。对于原始设置 （ASQA） 和带提示的设置 （ASQA-hint），作者手动注释 8 个示例，在维基百科语料库上使用 BM25，以及 3 个检索到的文档来运行实验。

5.4 **Open-domain summarization**

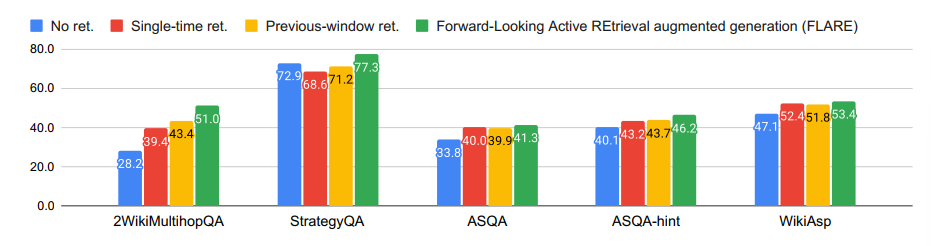
开放域摘要的目标是通过从开放网络收集信息来生成有关特定主题的全面摘要。作者使用 WikiAsp作为其测试平台，旨在生成有关维基百科中 20 个域的实体的基于方面的摘要。最初的 WikiAsp 数据集是为多文档摘要而设计的，并提供了对系统的引用列表。 作者通过删除相关的引用并从开放网络收集信息，将其转换为开放域设置。

**6实验结果**

作者首先报告了 4 个任务/数据集的总体结果，并将 FLARE 的性能与介绍的所有基线进行比较。 然后，作者进行消融实验，以研究其方法的各种设计选择的功效。

6.1与基线的比较

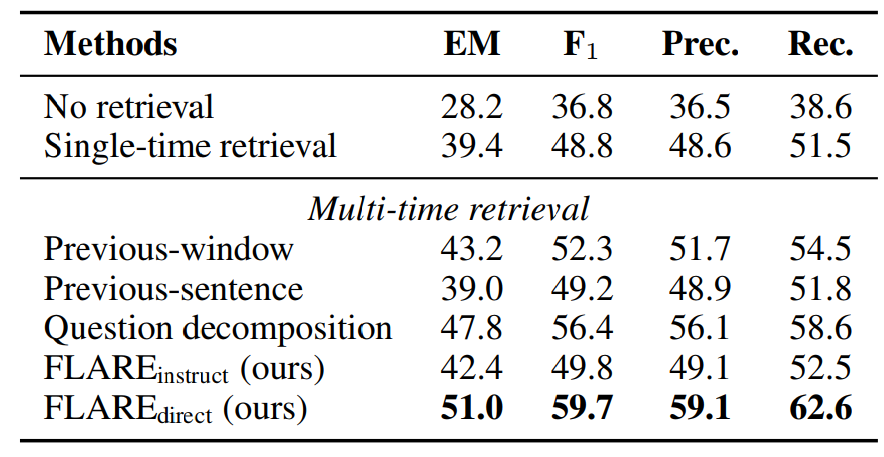
总体结果



上图报告了所有任务/数据集的 FLARE 和基线的整体性能。 FLARE 在所有任务/数据集上都优于所有基线，这表明 FLARE 是一种通用方法，可以在整个生成过程中有效地检索其他信息。

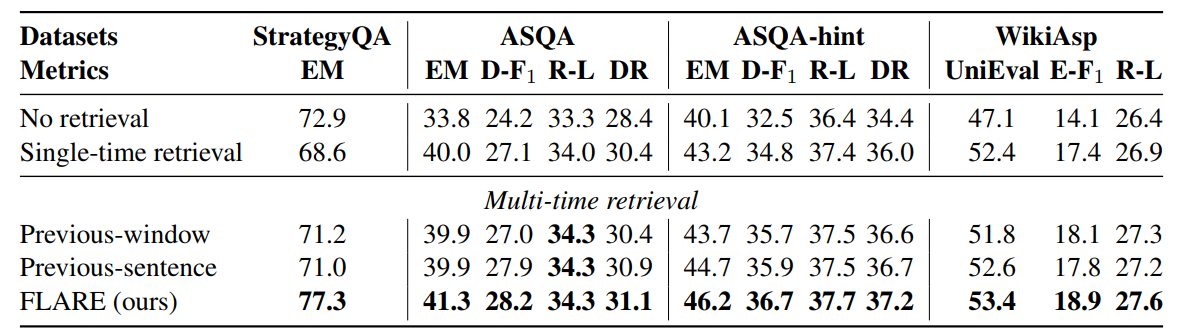
在各种任务和数据集中，多跳QA的改进最为显著。 这主要是由于该任务的明确定义和通过 2-hop理过程生成最终答案的具体目标，这使得 LM 更容易生成主题输出。 相比之下，ASQA 和 WikiAsp 的定义不那么明确，而且更开放，这增加了生成和评估的难度。 ASQA-hint 的改进比 ASQA 的改进更大，因为在许多情况下，即使对人类来说，识别模棱两可的方面也具有挑战性，而提供通用提示有助于 LM 保持主题

与基线进行全面比较。



上表报告了2WikiMultihopQA中讨论的所有基线的性能。 FLARE 的性能大大优于所有基线，这证实了前瞻性主动检索非常有效。 大多数多时间检索增强方法优于单次检索，但余量不同。 使用前一句检索的改进相对较小，我们假设这主要是因为前一句经常描述的实体或关系与2WikiMultihopQA中的下一句不同。 而上一个窗口方法可能会使用句子的前半部分作为查询来检索可能有助于生成后半部分的信息。 在所有基线中，问题分解方法实现最佳性能。 这并不奇怪，因为用分解的子问题手动注释的上下文示例指导 LM 生成符合后代主题/意图的合适子问题。 FLARE 优于此基准，表明手动示例注释对于有效的未来感知检索不是必需的。 FLARE 之间的差距指导问题分解量很大，这表明教 LM 使用任务泛型检索指令和示例生成搜索查询具有挑战性。

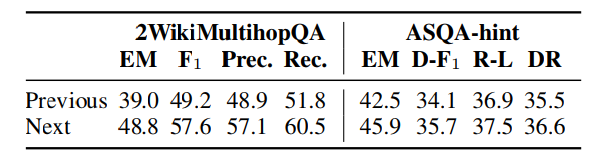
其他数据集的结果如下表

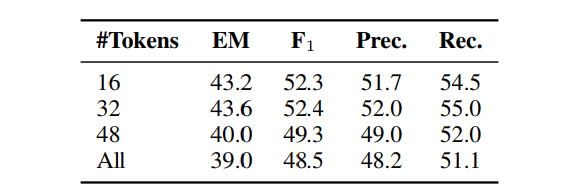


6.2消融研究

前瞻性检索的重要性。

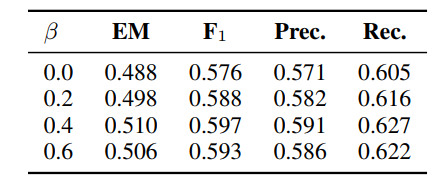
作者首先验证了其假设，即前瞻性检索确实比基于过去上下文的检索更强大。在 2WikiMultihopQA 和 ASQA-hint 数据集上运行消融实验，比较使用前一句和下一句的检索，确保两种方法除了用于检索的查询外相同。 具体来说，这两种方法都检索每个句子，并直接使用完整的句子（不屏蔽或问题生成）进行检索。 如下图所示，在两个数据集上，使用下一句话进行检索显然比使用前一句话要好，这证实了作者的假设。





主动检索的重要性。

接下来，作者研究性能与主动检索阈值之间的关系。 为了将作者的方法从不检索任何东西改为检索每个句子，我们调整了置信阈值用于确定何时触发从0到1的检索。 我们计算每个阈值触发检索的步骤/句子的百分比，并根据检索的百分比显示性能。



如上图所示，在 2WikiMultihopQA 上，当检索百分比超过 60% 时，性能趋于稳定，这表明在 LM 有把握时不需要进行检索。 在 StrategyQA 上，当检索百分比超过 50% 时，性能会下降，这表明使用高置信度句子进行检索可能会引入噪音并阻碍原始生成过程。 根据任务/数据集的不同，我们发现平均触发 40%-60% 的句子检索通常会带来良好的性能。

不同查询表述方法的有效性

最后，作者研究了通过掩码形成隐式查询，并通过问题生成研究了显式查询公式。我们比较了不同屏蔽阈值下 FLARE 的性能. 直接检索完整句子比屏蔽低概率的token更糟糕，证实低置信度的错误token会分散检索者的注意力。

**7 结论**

为了通过检索增强来帮助长格式生成，作者提出了一个主动检索增强生成框架，该框架决定在生成过程中何时以及检索什么。 作者通过前瞻性主动检索来实现这个框架，如果即将到来的句子包含低置信度的标记，则迭代地使用即将到来的句子来检索相关信息，并重新生成下一个句子。 在4个任务/数据集上的实验结果验证了该方法的有效性。 未来的发展方向包括更好的主动检索替代方案，以及开发LM架构以实现高效的主动检索增强。