RAG（Retrieval-Augmented Generation）评测面

来自： AiGC面试宝典

宁静致远

2024年01月28日 10:12

- RAG（Retrieval-Augmented Generation）评测面
  - 一、为什么需要 对 RAG 进行评测?
  - 二、如何合成 RAG 测试集?
  - 三、RAG 有哪些评估方法?
    - 3.1 独立评估
      - 3.1.1 介绍一下 独立评估?
      - 3.1.2 介绍一下 独立评估 模块?
    - 3.2 端到端评估
      - 3.2.1 介绍一下 端到端评估
      - 3.2.2 介绍一下 端到端评估 模块?
  - 四、RAG 有哪些关键指标和能力?
  - 五、RAG 有哪些评估框架?
    - 4.1 RAGAS
    - 4.2 ARES
  - 致谢

一、为什么需要 对 RAG 进行评测?

在探索和优化 RAG（检索增强生成器）的过程中，如何有效评估其性能已经成为关键问题。

二、如何合成 RAG 测试集?

假设你已经成功构建了一个RAG 系统，并且现在想要评估它的性能。为了这个目的，你需要一个评估数据集，该数据集包含以下列：

- question（问题）：想要评估的RAG的问题
- ground_truths（真实答案）：问题的真实答案
- answer（答案）：RAG 预测的答案
- contexts（上下文）：RAG 用于生成答案的相关信息列表

前两列代表真实数据，最后两列代表 RAG 预测数据。

要创建这样的数据集，我们首先需要生成问题和答案的元组。

接下来，在RAG上运行这些问题以获得预测结果。

- 生成问题和基准答案（实践中可能会出现偏差）

要生成（问题、答案）元组，我们首先需要准备 RAG 数据，我们将其拆分为块，并将其嵌入向量数据库中。 完成这些步骤后，我们会指示 LLM 从指定主题中生成 num_questions 个问题，从而得

到问题和答案元组。
为了从给定的上下文中生成问题和答案，我们需要按照以下步骤操作：

1. 选择一个随机块并将其作为根上下文
2. 从向量数据库中检索 K 个相似的上下文
3. 将根上下文和其 K 个相邻上下文的文本连接起来以构建一个更大的上下文
4. 使用这个大的上下文和 num_questions 在以下的提示模板中生成问题和答案

```
"""\\
Your task is to formulate exactly {num_questions} questions from given context and
provide the answer to each one.

End each question with a '?' character and then in a newline write the answer to
that question using only
the context provided.
Separate each question/answer pair by "XXX"
Each question must start with "question:".
Each answer must start with "answer:".

The question must satisfy the rules given below:
1.The question should make sense to humans even when read without the given
context.
2.The question should be fully answered from the given context.
3.The question should be framed from a part of context that contains important
information. It can also be from tables,code,etc.
4.The answer to the question should not contain any links.
5.The question should be of moderate difficulty.
6.The question must be reasonable and must be understood and responded by humans.
7.Do no use phrases like 'provided context',etc in the question
8.Avoid framing question using word "and" that can be decomposed into more than one
question.
9.The question should not contain more than 10 words, make of use of abbreviation
wherever possible.

context: {context}
"""



"""\\
您的任务是根据给定的上下文提出{num_questions}个问题，并给出每个问题的答案。

在每个问题的末尾加上"?
提供的上下文写出该问题的答案。
每个问题/答案之间用 "XXX "隔开。
每个问题必须以 "question: "开头。
每个答案必须以 "answer: "开头。

问题必须符合以下规则：
1.即使在没有给定上下文的情况下，问题也应该对人类有意义。
```

```
2. 问题应能根据给定的上下文给出完整的答案。
3. 问题应从包含重要信息的上下文中提取。也可以是表格、代码等。
4. 问题答案不应包含任何链接。
5. 问题难度应适中。
6. 问题必须合理，必须为人类所理解和回答。
7. 不要在问题中使用 "提供上下文 "等短语。
8. 避免在问题中使用 "和 "字，因为它可以分解成多个问题。
9. 问题不应超过 10 个单词，尽可能使用缩写。

语境：{上下文}
"""
```

5. 重复以上步骤 num_count 次,每次改变上下文并生成不同的问题。

基于上面的工作流程，下面是我生成问题和答案的结果示例。

```
|    | question                                          | ground_truths
|
|---:|:--------------------------------------------------|:----------------------
--------------------------|
|  8 | What is the difference between lists and tuples in | ['Lists are mutable and
cannot be used as          |
|    | Python?                                           | dictionary keys, while
tuples are immutable and    |
|    |                                                   | can be used as
dictionary keys if all elements are |
|    |                                                   | immutable.']
|
|  4 | What is the name of the Python variant optimized   | ['MicroPython and
CircuitPython']                 |
|    | for microcontrollers?                             |
|
| 13 | What is the name of the programming language that  | ['ABC programming
language']                      |
|    | Python was designed to replace?                   |
|
| 17 | How often do bugfix releases occur?               | ['Bugfix releases occur
about every 3 months.']    |
|  3 | What is the significance of Python's release       | ['Python 2.0 was
released in 2000, while Python    |
|    | history?                                          | 3.0, a major revision
with limited backward        |
|    |                                                   | compatibility, was
released in 2008.']            |
```

• 编码用例

首先构建一个向量存储，其中包含 RAG 使用的数据。

1. 我们从 Wikipedia 加载它

```
from langchain.document_loaders import WikipediaLoader
```

```
topic = "python programming"
wikipedia_loader = WikipediaLoader(
    query=topic,
    load_max_docs=1,
    doc_content_chars_max=100000,
)
docs = wikipedia_loader.load()
doc = docs[0]
```

2. 加载数据后，我们将其分成块。

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

CHUNK_SIZE = 512
CHUNK_OVERLAP = 128
splitter = RecursiveCharacterTextSplitter(
    chunk_size=CHUNK_SIZE,
    chunk_overlap=CHUNK_OVERLAP,
    separators=[". "],
)
splits = splitter.split_documents([doc])
```

3. 在 Pinecone 中创建一个索引。

```
import pinecone

pinecone.init(
    api_key=os.environ.get("PINECONE_API_KEY"),
    environment=os.environ.get("PINECONE_ENV"),
)
index_name = topic.replace(" ", "-")
pinecone.init(
    api_key=os.environ.get("PINECONE_API_KEY"),
    environment=os.environ.get("PINECONE_ENV"),
)
if index_name in pinecone.list_indexes():
    pinecone.delete_index(index_name)
pinecone.create_index(index_name, dimension=768)
```

4. 使用 LangChain 包装器来索引其中的分片嵌入。

```
from langchain.vectorstores import Pinecone

docsearch = Pinecone.from_documents(
    splits,
    embedding_model,
    index_name=index_name,
)
```

5. 生成合成数据集

我们使用 LLM、文档拆分、嵌入模型和 Pinecone 索引名称从TestsetGenerator 类初始化一个对象。

```
from langchain.embeddings import VertexAIEmbeddings
from langchain.llms import VertexAI
from testset_generator import TestsetGenerator

generator_llm = VertexAI(
    location="europe-west3",
    max_output_tokens=256,
    max_retries=20,
)
embedding_model = VertexAIEmbeddings()
testset_generator = TestsetGenerator(
    generator_llm=generator_llm,
    documents=splits,
    embedding_model=embedding_model,
    index_name=index_name,
    key="text",
)
```

1. 通过传递两个参数来调用generate方法

```
synthetic_dataset = testset_generator.generate(
    num_contexts=10,
    num_questions_per_context=2,
)
```

2. 生成问题与答案如下

```
|    | question                                          | ground_truths
|
|---:|:--------------------------------------------------|:----------------------
---------------------------|
|  8 | What is the difference between lists and tuples in | ['Lists are mutable and
cannot be used as          |
|    | Python?                                           | dictionary keys, while
tuples are immutable and    |
|    |                                                   | can be used as
dictionary keys if all elements are |
|    |                                                   | immutable.']
|
|  4 | What is the name of the Python variant optimized   | ['MicroPython and
CircuitPython']                 |
|    | for microcontrollers?                             |
|
| 13 | What is the name of the programming language that  | ['ABC programming
language']                      |
|    | Python was designed to replace?                   |
|
```

```
| 17 | How often do bugfix releases occur?                | ['Bugfix releases occur
about every 3 months.']    |
|  3 | What is the significance of Python's release       | ['Python 2.0 was
released in 2000, while Python    |
|    | history?                                           | 3.0, a major revision
with limited backward       |
|    |                                                    | compatibility, was
released in 2008.']            |
```

接下来使用 RAG 来预测每个问题的答案，并提供用于支撑响应的上下文列表。

1. 初始化 RAG

```
# 初始化RAG
from rag import RAGimport RAG

rag = RAG(
    index_name,
    "text-bison",
    embedding_model,
    "text",
)
```

2. 通过对每个问题调用 predict 方法来迭代合成数据集并收集预测

```
rag_answers = []
contexts = []

for i, row in synthetic_dataset.iterrows():
    question = row["question"]
    prediction = rag.predict(question)
    rag_answer = prediction["answer"]
    rag_answers.append(rag_answer)
    source_documents = prediction["source_documents"]
    contexts.append([s.page_content for s in source_documents])
synthetic_dataset_rag = synthetic_dataset.copy()
synthetic_dataset_rag["answer"] = rag_answers
```

3. 最终结果如下

```
|    | question
| ground_truths               | answer
| contexts                                          |_truths              |
answer
| contexts                                          |
|---:|:-------------------------------------------------------------------------
|:---------------------------|:-------------------------------------------------
-------------------------------------------------|:-----------------------
------------------------|
|  7 | What are the two types of classes that Python supported before version 3.0?
| ['old-style and new-style'] | Before version 3.0, Python had two kinds of classes
```

```
(both using the same syntax): old-style and new-style. | ['. New instances of
classes are constructed by    |
|    |
|                              |
| calling the class (for example, SpamClass() or     |
|    |
|                              |
| EggsClass()), and the classes are instances of the |
|    |
|                              |
| metaclass type (itself an instance of itself),     |
|    |
|                              |
| allowing metaprogramming and reflection.\\nBefore   |
|    |
|                              |
| version 3.0, Python had two kinds of classes (both |
|    |
|                              |
| using the same syntax): old-style and new-style,   |
|    |
|                              |
| current Python versions only support the semantics |
|    |
|                              |
| new style.\\nPython supports optio     .........   |
```

基于以上步骤，我们已经为评估 RAG 做好了准备，接下来我们讲解如何进行 RAG 评估。

三、RAG 有哪些评估方法?

主要有两种方法来评估 RAG 的有效性：独立评估和端到端评估。

3.1 独立评估

3.1.1 介绍一下 独立评估?

- 介绍：独立评估涉及对检索模块和生成模块（即阅读和合成信息）的评估。

3.1.2 介绍一下 独立评估 模块?

- 介绍：生成模块指的是将检索到的文档与查询相结合，形成增强或合成的输入。这与最终答案或响应的生成不同，后者通常采用端到端的评估方式。
- 评估指标：
- 1、 答案相关性（Answer Relevancy）

此指标的目标是**评估生成的答案与提供的问题提示之间的相关性**。答案如果缺乏完整性或者包含冗余信息，那么其得分将相对较低。这一指标通过**问题和答案的结合来进行计算**，评分的范围通常在0到1之间，其中高分代表更好的相关性。

示例

问题：健康饮食的主要特点是什么？

低相关性答案：健康饮食对整体健康非常重要。
高相关性答案：健康饮食应包括各种水果、蔬菜、全麦食品、瘦肉和乳制品，为优化健康提供必要的营养素。

- 2、 忠实度（Faithfulness）

这个评价标准旨在**检查生成的答案在给定上下文中的事实准确性。**评估的过程涉及到**答案内容与其检索到的上下文之间的比对。**这一指标也使用一个介于0到1之间的数值来表示，其中更高的数值意味着答案与上下文的一致性更高。

示例

问题：居里夫人的主要成就是什么？

背景：玛丽·居里（1867-1934年）是一位开创性的物理学家和化学家，她是第一位获得诺贝尔奖的女性，也是唯一一位在两个不同领域获得诺贝尔奖的女性。

高忠实度答案：玛丽·居里在物理和化学两个领域都获得了诺贝尔奖，使她成为第一位实现这一成就的女性。
低忠实度答案：玛丽·居里只在物理学领域获得了诺贝尔奖。

- 3、上下文精确度（Context Precision）

在这个指标中，我们**评估所有在给定上下文中与基准信息相关的条目是否被正确地排序。**理想情况下，所有相关的内容应该出现在排序的前部。这一评价标准同样使用0到1之间的得分值来表示，其中较高的得分反映了更高的精确度。

指标：**命中率 (Hit Rate)**、**平均排名倒数 (MRR)**、**归一化折扣累积增益 (NDCG)**、**精确度(Precision)**等。

- 4、答案正确性（Answer Correctness）

该指标主要用于**测量生成的答案与实际基准答案之间的匹配程度。**这一评估考虑了基准答案和生成答案的对比，其得分也通常在0到1之间，较高的得分表明生成答案与实际答案的一致性更高。

示例：

基本事实：埃菲尔铁塔于 1889 年在法国巴黎竣工。

答案正确率高：埃菲尔铁塔于 1889 年在法国巴黎竣工。
答案正确率低：埃菲尔铁塔于 1889 年竣工，矗立在英国伦敦。

3.2 端到端评估

3.2.1 介绍一下 端到端评估

- 介绍：对 RAG 模型对特定输入生成的最终响应进行评估，涉及模型生成的答案与输入查询的相关性和一致性。

3.2.2 介绍一下 端到端评估 模块?

1. 无标签的内容评估：
    a. 评价指标：答案的准确性、相关性和无害性
2. 有标签的内容评估：

a. 评价指标：准确率 (Accuracy) 和精确匹配 (EM)

四、RAG 有哪些关键指标和能力?

评估 RAG 在不同下游任务和不同检索器中的应用可能会得到不同的结果。然而，一些学术和工程实践已经开始关注 RAG 的通用评估指标和有效运用所需的能力。

- 关键指标：集中于三个关键指标：答案的准确性、答案的相关性和上下文的相关性。
- 关键能力：
    - RGB的研究分析了不同大语言模型在处理 RAG 所需的四项基本能力方面的表现，**包括抗噪声能力、拒绝无效回答能力、信息综合能力和反事实稳健性**，从而为检索增强型生成设立了标准。

五、RAG 有哪些评估框架?

在 RAG 评估框架领域，RAGAS 和 ARES 是较新的方法。

5.1 RAGAS

- 介绍：

RAGAS 是一个基于简单手写提示的评估框架，通过这些提示全自动地衡量答案的准确性、相关性和上下文相关性。

- 算法原理：

1. **答案忠实度评估**：利用大语言模型 (LLM) 分解答案为多个陈述，检验每个陈述与上下文的一致性。最终，根据支持的陈述数量与总陈述数量的比例，计算出一个“忠实度得分”。
2. **答案相关性评估**：使用大语言模型 (LLM) 创造可能的问题，并分析这些问题与原始问题的相似度。答案相关性得分是通过计算所有生成问题与原始问题相似度的平均值来得出的。
3. **上下文相关性评估**：运用大语言模型 (LLM) 筛选出直接与问题相关的句子，以这些句子占上下文总句子数量的比例来确定上下文相关性得分。

5.2 ARES

- 介绍：

ARES 的目标是自动化评价 RAG 系统在上下文相关性、答案忠实度和答案相关性三个方面的性能。ARES 减少了评估成本，通过使用少量的手动标注数据和合成数据，并应用预测驱动推理 (PDR) 提供统计置信区间，提高了评估的准确性。

- 算法原理：

1. **生成合成数据集**：ARES 首先使用语言模型从目标语料库中的文档生成合成问题和答案，创建正负两种样本。
2. **训练大语言模型 (LLM) 裁判**：然后，ARES 对轻量级语言模型进行微调，利用合成数据集训练它们以评估上下文相关性、答案忠实度和答案相关性。
3. **基于置信区间对 RAG 系统排名**：最后，ARES 使用这些裁判模型为 RAG 系统打分，并结合手动标注的验证集，采用 PPI 方法生成置信区间，从而可靠地评估 RAG 系统的性能。