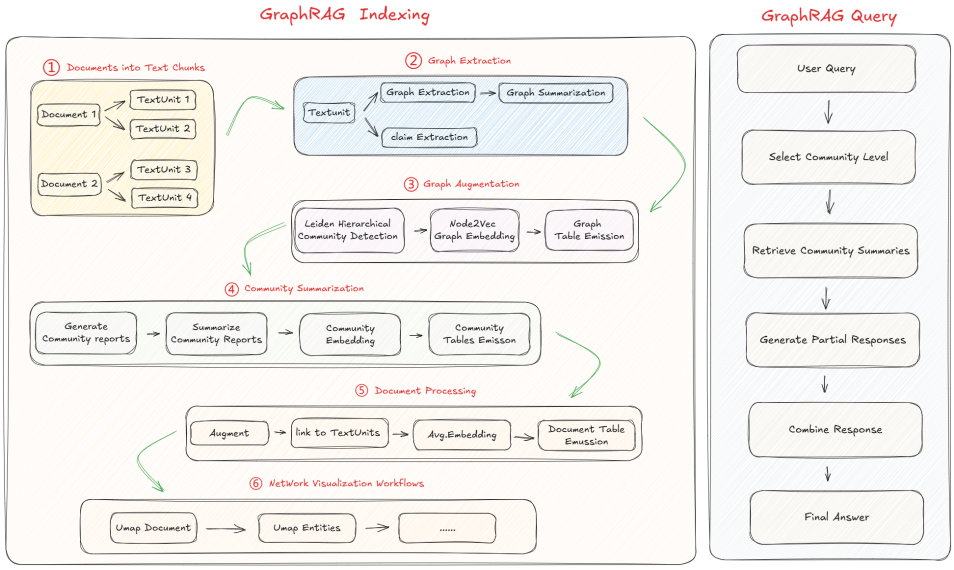
Deepseek企业级Agent项目开发实战

Part 4. Microsoft GraphRAG Query构建流程详解

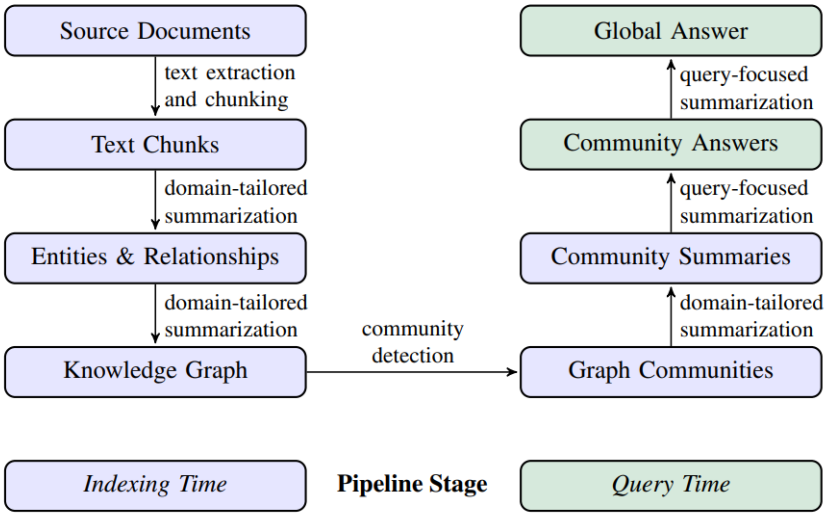
在完成了`Microsoft GraphRAG`的索引构建及自定义接入图数据库`Neo4j`构建完整的知识图谱后，我们在上一小节课程中已经初步实践了可以通过`Cypher`语句来查询结构化数据中的信息。当然，传统的`Cypher`查询方式，需要我们具备一定的图数据库知识，对非技术人员来说，使用门槛较高。 而`Microsoft GraphRAG`则提供了一种更为直观、易用的查询方式，我们只需要输入自然语言查询，即可获得结构化的查询结果。



这就需要我们了解`Microsoft GraphRAG`使用的第二阶段，即查询（Query）阶段。

索引阶段我们利用大语言模型结合提示工程，从非结构化文本（`.txt`、`.csv`）中提取出实体（Entities）与关系（Relationships），构建出了基础的`Knowledge Graph`，并且通过建立层次化的`community`结构，`community`以及`community_report`的丰富语义，相较于传统基于`Cypher`的查询方式可以提供更多灵活性的`Query`操作，`Microsoft GraphRAG`在项目开源之初是提供了`local`和`global`两种查询方式，分别对应了`local search`和`global search`，而后在不断的迭代更新过程中，除了优化了`local search`和`global search`的效果，还新增了`DRIFT Search`和`Multi Index Search`作为扩展优化的可选项，以进一步丰富`Query`操作的多样性。

如下图所以，原图来源于`Microsoft GraphRAG`的官方论文：https://arxiv.org/pdf/2404.16130



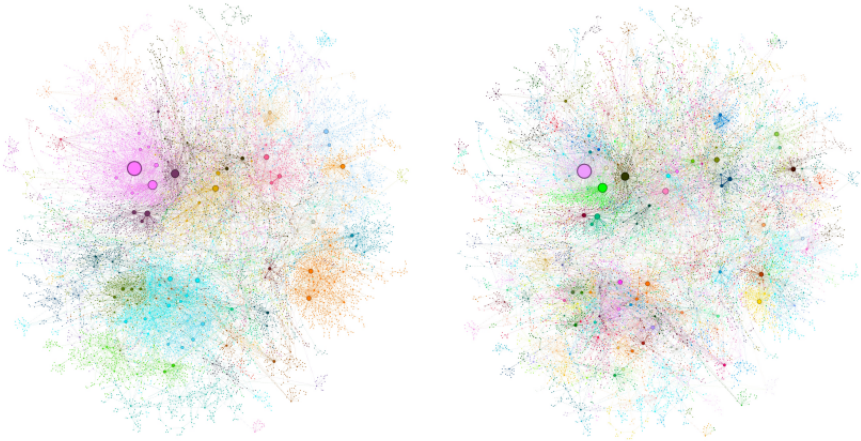
`Microsoft GraphRAG` 在查询阶段构建的流程，相较于构建索引阶段会更为直观，核心的具体步骤包括：

1. 接收用户的查询请求。
2. 根据查询所需的详细程度，选择合适的社区级别进行分析。
3. 在选定的社区级别进行信息检索。
4. 依据社区摘要生成初步的响应。
5. 将多个相关社区的初步响应进行整合，形成一个全面的最终答案。

通过学习`Microsoft GraphRAG`索引构建的源码大家应该已经能够清晰的知道，`Indexing` 过程中并不是在创建完第一层社区后就停止了，而是是分层的。也就是说，当创建第一层社区（即基础社区）后，会将这些社区视为节点，进一步构建更高层级的社区。这种方法就实现在知识图谱中可以以不同的粒度级别上组织和表示数据。比如第一层社区可以包含具体的实体或数据，而更高层级的社区则可以聚合这些基础社区，形成更广泛的概览。

因此最核心的`Local Search`和`Global Search`的实现，就是源于不同的粒度级别而构建出来用于处理不同类型问题的`Pipeline`，其中：

- `Local Search`是基于实体的检索。
- `Global Search`则是基于社区的检索。



(a) Root communities at level 0　　(b) Sub-communities at level 1

因此接下来，我们就分别从源码层面，来详细介绍`Local Search`和`Global Search`的实现原理，并实际操作不同检索方式的查询操作。

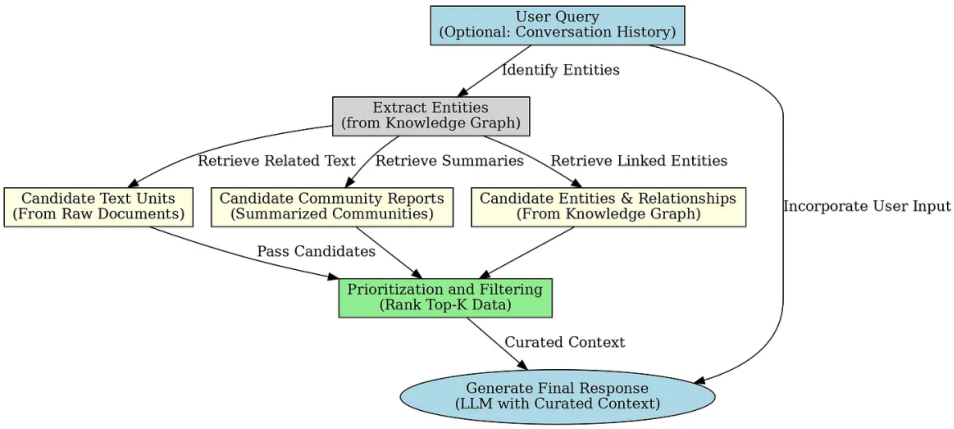
首先来介绍`Local Search`， 即本地检索。

1. Local Search 实现原理与源码解析

`Local Search`即本地检索，是基于实体的检索。本地搜索从相关实体开始，使用知识图谱来查找最相关的信息。例如，给定查询中的实体，使用的是连接节点的信息，通过辨识与查询相关的实体与关系，检索特定文本片段、摘要和关联性资料。

所以`Local Search`本质上是基于实体的推理。特别适合回答“who”、“what”、“when” 类型的问题。

在`Microsoft GraphRAG`源码中实现的内部原理如下图所示：



接下来，我们就进入源码，逐层的解析`Local Search`的实现过程。

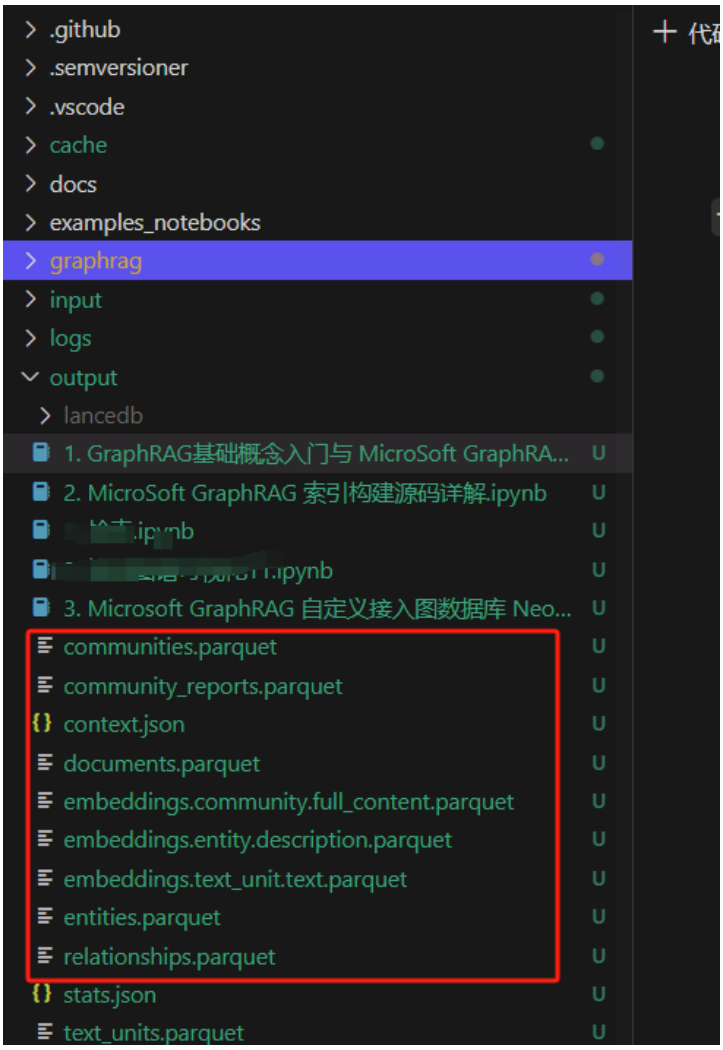
1.1 构建索引

首先，我们还是在当前的运行环境下先构建索引。（如果已经构建过索引，则可以跳过此步骤）这里我们使用的还是单`txt`文件即`technology_companies.txt`文件，切分的`chunk_size`为`300`，`chunk_overlap`为`100`。在源码环境下，执行构建索引的命令为：

```
poetry run poe index --root ./
```

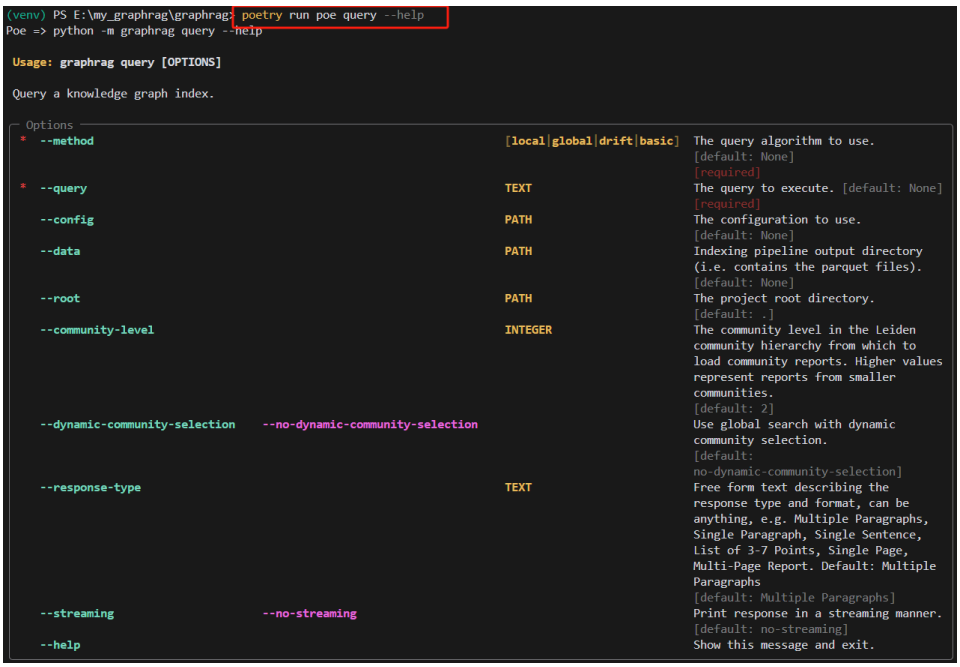
```
(venv) PS E:\my_graphrag\graphrag> poetry run poe index --root ./
Poe => python -m graphrag index --root ./

Logging enabled at E:\my_graphrag\graphrag\logs\indexing-engine.log
🚀 LLM Config Params Validated
🚀 Embedding LLM Config Params Validated
load result:
                                                text  ...             creation_date
0  在过去的几十年中，全球科技行业经历了翻天覆地的变化。以硅谷为中心的创新生态系统催生了许多世界...  ...  2025-03-12 15:10:35 +0800

[1 rows x 4 columns]
Running standard indexing.
🚀 create_base_text_units
                                     id  ... n_tokens
0  b514214bf611fd3c88afe3af8d4f74d976a7335123c5a...  ...      300
1  ec2f0866b1c262786fee51f64816f505a516f5165df594...  ...      300
2  a3a9c831917df38c452ddf4082b515c688b45d483ad467...  ...      300
3  be36cdf44648d79b188ec090f3ef69a54efdab948d433a...  ...      300
4  5fefb1a54a8af9e38a5e7e30e7ae383af9abcf8141e8c2...  ...      300
5  e0ae7d724be95809a79b8c1d01b3f67abaeb9e8b51c073...  ...      300
6  052bfb9e5086f8505a9a5c3a5891214ad0bc8d4ea93c54...  ...      300
7  b0cd465703709aaea7bed9bf4bb5217cdc90104020d164...  ...      300
8  8b587b1473a78a4d0fe3c7fc864f4a3e4c2c4c1aabc6d9...  ...      300
9  ae28d95d3f6de5e5af8ae61f4c7fb20d28afa2dd1d8aec...  ...      104
```

构建索引完成后，在`output`目录下依次生成`document.parent`、`text_units.parent`、`entities.parent`、`relationships.parent`、`communities.parent`以及`community_report.parent`主要的索引，在检索过程中，会加载这些索引文件中的相关数据内容。



1.2 Local Search 源码执行方法

与`grahrang index`命令类似，`Microsoft GraphRAG`的检索工作流可以通过`graphrag query`命令支持，在源码环境下其命令格式如何使用，我们可以通过`graphrag query --help`命令来查看，如下图所示：



各个字段参数的含义如下表所示：

graphrag query 命令参数说明

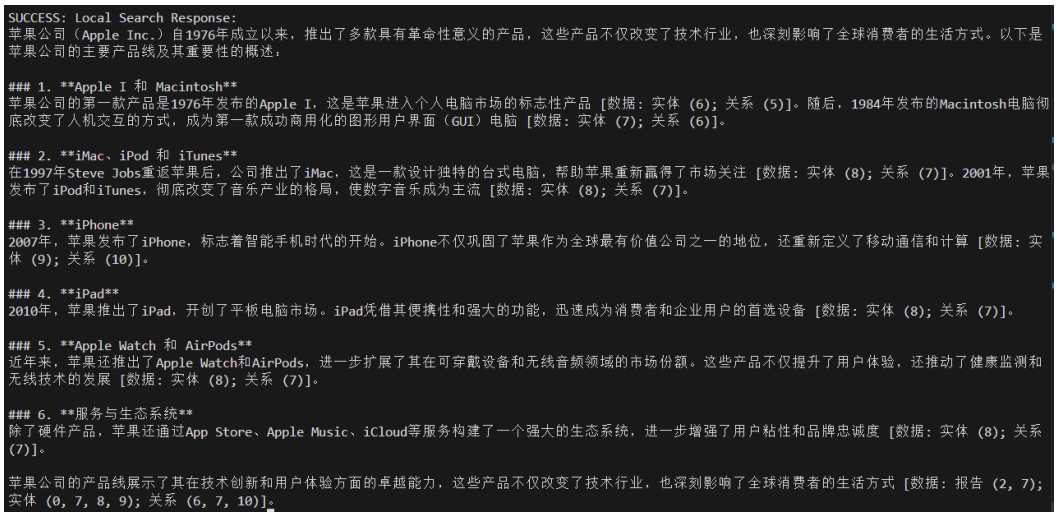
参数名称	类型	描述	默认值	是否必需
`--method`	Tpye	可以选择local、global、drift或basic算法。	None	是
`--query`	TEXT	要执行的查询，即提出的问题。	None	是
`--config`	PATH	要使用的配置文件路径。	None	否
`--data`	PATH	索引管道输出目录（即包含 parquet 文件的目录）。	None	否
`--root`	PATH	项目根目录的路径。	.	否
`--community-level`	INTEGER	从中加载社区报告的 Leiden 社区层级。较高的值表示来自较小社区的报告。	2	否
`--dynamic-community-selection`		使用动态社区选择的全局搜索。	no-dynamic-community-selection	否
`--response-type`	TEXT	描述响应类型和格式的自由文本，可以是任何内容，例如多个段落、单个段落、单句、3-7点列表、单页、多页报告。	Multiple Paragraphs	否
`--streaming`		以流式方式打印响应。	no-streaming	否
`--help`		显示帮助信息并退出。		否

其中，在执行查询时必须指定的参数是`--method`和`--query`，其他参数为可选参数。其中：

- `--method`参数可以选择`local`、`global`、`drift`或`basic`算法。(接下来我们会依次介绍这几种算法)
- `--query`参数是要执行的查询，即提出的问题。

了解到这里，就可以通过命令行快速启动问答检索了。这里我们先来看`local`本地搜索。输入如下命令：

```
poetry run poe query --root ./ --method local --query "苹果公司都有哪些产品？"
```



最终会显示`SUCCESS: Local Search Response`成功提示，并会显示最终的问答结果。整个过程使用起来非常简单，但是简单并不意味着可以直接使用，大多数情况下基于通用流程的问答检索，并不能满足实际业务需求。比如检索的效果不准确，效率不高，检索结果不全面等，因此，我们需要进一步掌握`Microsoft GraphRAG`的检索原理，并根据实际业务需求，进行针对性的优化和调整。

所以接下来，我们就深入源码底层，详细的给大家介绍一下`Local Search`的完整实现过程，并对其中的一些优化策略进行详细的介绍。