MicroSoft-GraphRAG

有关**MicroSoft-GraphRAG**相关的整理。后续再写paper的解读。

一. RAG与GraphRAG

从RAG过渡到GraphRAG，以及图谱之前的应用与关联。

1.1 什么是RAG

RAG（Retrieval-Augmented Generation）是一种结合了信息检索和生成式语言模型的技术，主要用于增强语言模型处理复杂问题的能力。这种方法通过以下步骤工作：

1. **检索阶段**：当用户提出一个问题时，系统会首先从一个大型的文档库中检索出与问题最相关的信息片段。
2. **生成阶段**：然后，系统使用这些检索到的信息作为上下文，引导语言模型生成更准确、更丰富的回答。

RAG的关键优势在于它结合了检索的精确性和生成模型的灵活性，能够提供比单独使用生成模型或检索系统更加详细和准确的答案。这种方法在问答系统、摘要生成、聊天机器人等领域具有广泛的应用前景。

1.2 GraphRAG

GraphRAG是微软研究院开发的一个工具，它结合了图形表示和检索增强生成（RAG）技术，用于提高问题回答的质量和结构。这个工具使用大型语言模型（LLM）从文本数据中自动提取出知识图谱，然后利用这个图谱在问答系统中提供支持，特别是针对全局问题的解答，即那些需要考虑整个数据集的问题。

GraphRAG的工作流程如下：

1. **数据提取**：从大量未结构化文本中提取信息，构建出一个包含实体和关系的知识图谱。
2. **层次化处理**：通过识别知识图谱中的密集连接的节点群体（社区），在多个层次上对数据进行分区，以便生成层次化的数据总结。
3. **回答生成**：在用户提问时，利用已构建的知识图谱检索相关信息，然后生成回答。

GraphRAG的特点包括高效的信息检索、能够处理全局性问题和通过层次化的数据总结来提供更深入的见解。这些特性使得GraphRAG在需要深入理解和处理复杂数据集的应用场景中尤为有用。

1.3 GraphRAG与Graph+LLM

GraphRAG 与传统的知识图谱和大型语言模型（LLM）的组合在多个方面有所不同，尤其在如何利用这些技术来增强信息检索和生成回答的方法上。以下是一些主要的区别：

1. **图形结构的深度利用**：
   - **传统方法**：通常，知识图谱与LLM结合的方法可能更多地侧重于使用图谱作为事实的存储库，LLM则用于生成语言输出。在这种情况下，知识图谱通常用来直接提供答案或辅助LLM理解和生成相关内容。
   - **GraphRAG**：GraphRAG不仅利用知识图谱作为信息源，而且通过构建一个多层次的图形结构来优化信息的检索和处理流程。它通过识别数据中的“社区”或密集连接的节点群体，按层次组织信息，从而在回答问题时可以更加精确地定位到相关信息。
2. **层次化的数据处理**：
   - **传统方法**：虽然一些知识图谱系统可能支持层次化数据的表示，但它们通常不会自动从大规模文本数据中生成这种结构化的层次化表示。
   - **GraphRAG**：它自动从文本数据中生成层次化的图谱，并利用这些层次来改善问题的理解和回答的生成。这种层次化结构使得GraphRAG能够有效地处理和回答全局性问题，即那些需要考虑整个数据集来回答的问题。
3. **应用和性能优化**：
   - **传统方法**：知识图谱和LLM的传统组合可能需要手动优化和调整来适应特定的应用场景。
   - **GraphRAG**：GraphRAG通过提供一个端到端的解决方案，包括数据提取、图谱生成和查询处理，使得整个系统的部署和使用变得更加简单和高效。此外，它还通过Azure托管的API体验来简化部署和管理过程 。

总的来说，GraphRAG通过其独特的结构化方法和技术优化，提供了一种更为高效和精确的方式来处理和生成基于复杂数据集的查询回答。

二. Demo运行

GraphRAG的运行（Deepseek+GLM）。

首先安装GraphRAG：

```
$ conda acitvate GraphRAG
$ conda activate GraphRAG
$ pip install graphrag pandas
```

创建workspace：

```
$ mkdir -p /gr_test/input
$ python -m graphrag.index --init --root ./gr_test
```

数据准备：

如果是结构化数据，微软官方的支持不是很好，效果不如直接直接LLM去做推理，可以csv->json->txt。这样方便LLM去理解。纯文本的做一些统计的功能效果不太行。

```python
import pandas as pd

# 确保正确设置了路径和文件名
csv_file_path = '1.csv'
output_json_path = '1.json'

# 读取CSV文件，指定UTF-8编码
df = pd.read_csv(csv_file_path, encoding='utf-8')

# 转换为JSON格式，确保中文不被转义
json_data = df.to_json(orient='records', lines=True, force_ascii=False)

# 打印JSON数据
print(json_data)

# 将JSON数据保存到文件，同时保持UTF-8编码
with open(output_json_path, 'w', encoding='utf-8') as f:
    f.write(json_data)
```

将转换好的数据移动到input文件夹。

设置setting.yaml（Deepseek+GLM可以无脑复制）：

```yaml

encoding_model: cl100k_base
skip_workflows: []
llm:
  api_key: ${GRAPHRAG_API_KEY}
  type: openai_chat # or azure_openai_chat
  model: deepseek-chat
  model_supports_json: False # recommended if this is available for your model.
  max_tokens: 4000
  request_timeout: 180.0
  api_base: https://api.deepseek.com/v1
  # api_version: 2024-02-15-preview
  # organization: <organization_id>
  # deployment_name: <azure_model_deployment_name>
  tokens_per_minute: 50000 # set a leaky bucket throttle
```

```yaml
  requests_per_minute: 5000 # set a leaky bucket throttle
  max_retries: 20
  max_retry_wait: 10.0
  sleep_on_rate_limit_recommendation: true # whether to sleep when azure suggests wait-times
  concurrent_requests: 10 # the number of parallel inflight requests that may be made

parallelization:
  stagger: 0.3
  num_threads: 50 # the number of threads to use for parallel processing

async_mode: threaded #threaded # or asyncio

embeddings:
  ## parallelization: override the global parallelization settings for embeddings
  async_mode: threaded # or asyncio
  llm:
    api_key: ${GRAPHRAG_API_KEY_1}
    type: openai_embedding # or azure_openai_embedding
    model: embedding-2
    api_base: https://open.bigmodel.cn/api/paas/v4
    # api_version: 2024-02-15-preview
    # organization: <organization_id>
    # deployment_name: <azure_model_deployment_name>
    # tokens_per_minute: 150 # set a leaky bucket throttle
    # requests_per_minute: 10 # set a leaky bucket throttle
    # max_retries: 10
    # max_retry_wait: 10.0
    sleep_on_rate_limit_recommendation: true # whether to sleep when azure suggests wait-times
    concurrent_requests: 10 # the number of parallel inflight requests that may be made
    batch_size: 16 # the number of documents to send in a single request
    batch_max_tokens: 8191 # the maximum number of tokens to send in a single request
    # target: required # or optional

chunks:
  size: 300
  overlap: 100
  group_by_columns: [id] # by default, we don't allow chunks to cross documents

input:
  type: file # or blob
  file_type: text # or csv
```

```yaml
  base_dir: "input"
  file_encoding: utf-8
  file_pattern: ".*\\.txt$"

cache:
  type: file # or blob
  base_dir: "cache"
  # connection_string: <azure_blob_storage_connection_string>
  # container_name: <azure_blob_storage_container_name>

storage:
  type: file # or blob
  base_dir: "output/${timestamp}/artifacts"
  # connection_string: <azure_blob_storage_connection_string>
  # container_name: <azure_blob_storage_container_name>

reporting:
  type: file # or console, blob
  base_dir: "output/${timestamp}/reports"
  # connection_string: <azure_blob_storage_connection_string>
  # container_name: <azure_blob_storage_container_name>

entity_extraction:
  ## llm: override the global llm settings for this task
  ## parallelization: override the global parallelization settings for this task
  ## async_mode: override the global async_mode settings for this task
  prompt: "prompts/entity_extraction.txt"
  entity_types: [organization,person,geo,event]
  max_gleanings: 0

summarize_descriptions:
  ## llm: override the global llm settings for this task
  ## parallelization: override the global parallelization settings for this task
  ## async_mode: override the global async_mode settings for this task
  prompt: "prompts/summarize_descriptions.txt"
  max_length: 500

claim_extraction:
  ## llm: override the global llm settings for this task
  ## parallelization: override the global parallelization settings for this task
  ## async_mode: override the global async_mode settings for this task
  # enabled: true
  prompt: "prompts/claim_extraction.txt"
```

```yaml
  description: "Any claims or facts that could be relevant to information discovery."
  max_gleanings: 0

community_report:
  ## llm: override the global llm settings for this task
  ## parallelization: override the global parallelization settings for this task
  ## async_mode: override the global async_mode settings for this task
  prompt: "prompts/community_report.txt"
  max_length: 2000
  max_input_length: 8000

cluster_graph:
  max_cluster_size: 10

embed_graph:
  enabled: false # if true, will generate node2vec embeddings for nodes
  # num_walks: 10
  # walk_length: 40
  # window_size: 2
  # iterations: 3
  # random_seed: 597832

umap:
  enabled: false # if true, will generate UMAP embeddings for nodes

snapshots:
  graphml: false
  raw_entities: false
  top_level_nodes: false

local_search:
  # text_unit_prop: 0.5
  # community_prop: 0.1
  # conversation_history_max_turns: 5
  # top_k_mapped_entities: 10
  # top_k_relationships: 10
  # max_tokens: 12000

global_search:
  # max_tokens: 12000
  # data_max_tokens: 12000
  # map_max_tokens: 1000
  # reduce_max_tokens: 2000
  # concurrency: 32
```

添加自己的.env

```
GRAPHRAG_API_KEY= Deepseek的api key
GRAPHRAG_API_KEY_1= GLM embedding的api key
```

运行GraphRAG：

```
$ python -m graphrag.index --root ./gr_test
```

测试：

测试分全局和局部查找。

```
$ python -m graphrag.query --root ./gr_test --method global "问题"
$ python -m graphrag.query --root ./gr_test --method local "问题"
```

效果如下：

```
SUCCESS: Local Search Response: 翼状胬肉是一种常见的眼科疾病，主要影响中老年人群，尤其是那些长期暴露在阳光和风沙环境中的人。河北医科大学第二医院作为治疗翼状胬肉的核心机构，提供了包括翼状胬肉切除和结膜瓣移植术在内的多种治疗方案。这些治疗对改善患者的生活质量有显著影响。

在河北医科大学第二医院的治疗社区中，翼状胬肉的治疗案例显示了医院对不同年龄和性别的患者的关注和治疗能力。例如，老年女性患者在该社区中占有重要地位，她们患有多种眼科疾病并接受了相应的治疗，包括翼状胬肉的手术治疗。这些治疗案例不仅展示了医院的专业技术，也反映了翼状胬肉在老年女性中的普遍性。

此外，翼状胬肉的治疗还包括药物治疗，如使用抗炎药物来减轻症状和促进愈合。河北医科大学第二医院在治疗翼状胬肉时广泛使用药物治疗，显示了医院在药物治疗方面的专业性。

综上所述，翼状胬肉主要影响中老年人群，尤其是那些长期暴露在恶劣环境中的人。河北医科大学第二医院提供了全面的治疗方案，包括手术和药物治疗，以帮助患者恢复视力和改善生活质量。
```

三. 后续展望

GraphRAG在多模态大模型、AGI的展望。

优点

1. **结构化的信息检索**：GraphRAG通过构建知识图谱，提供比传统RAG方法更加结构化的信息检索和回答生成能力。
2. **处理全局问题的能力**：GraphRAG能有效处理全局性问题，这种问题需要考虑数据集中的所有文本，而不仅仅是与查询语义相似的文本块。
3. **层次化数据总结**：GraphRAG能识别出数据中的密集连接节点群体，并在多个层次上对数据进行分区，从而生成层次化的数据总结。

缺点

1. **索引成本**：构建知识图谱的索引过程可能涉及较高的成本和资源消耗。
2. **初始设置复杂**：GraphRAG的初始设置和调整可能需要专业知识和技术，尤其是在定制提示和实体类型枚举等方面。

后续还需要继续完善和调研，但是总体来说是可以用的。