

# Deepseek企业级Agent项目开发实战

## Part 5: Microsoft GraphRAG 多源数据索引构建方案

### 关系型数据库与CSV文件格式优化策略

在前面的章节中我们依次介绍了 Microsoft GraphRAG 在构建索引和问答检索两个流程的实现细节，在讲解原理的过程中主要是借助相对简单的 `txt` 文件来进行流程的演示，目的是在于帮助大家以尽可能低的门槛先跑通流程，并在这个过程中熟练掌握 Microsoft GraphRAG 各个阶段的工作原理。

经历过第一个阶段的认知性学习后，我们从本节开始将进入第二个阶段，即实战性学习阶段。

实际的落地需求场景中，往往需要处理的是更加复杂的数据，`.txt` 文件是最基础，同时也并不是特别常用的数据格式，更加常见的数据类型是关系型数据库中的数据、`CSV`、`JSON`、`PDF`、`DOC` 等文件格式，每个文件格式都有其独特的数据结构和数据处理方式，比如：

- 关系型数据库中的数据有成百上千张表，数据量大且关联复杂；
- `PDF` 文件中既包含文本信息，又包含图片、表格、公式等复杂结构；

同时，Microsoft GraphRAG 默认支持的文件类型仅有 `.txt`、`.csv` 以及在刚刚发布的 `v2.0.1` 版本新增的 `.json` 文件格式，因此，如果要适配到实际的落地场景中，需要对以上几种类型做深度的集成及优化，才能更好的利用 Microsoft GraphRAG 的优秀特性。所以接下来的内容，我们就针对 Microsoft GraphRAG 最新发布的 `v2.0.1` 版本进行源码级别的二次开发，自定义接入 `.pdf`、`.docx` 等文件类型的支持，并结合智能客服场景的数据，展开详细的优化策略和思路讲解。

本节首先进行关系型数据库与 `.csv` 文件的优化策略及在 Microsoft GraphRAG 中集成后的使用方法。

## 1. 智能客服数据准备

智能客服领域自 2023 年起以大模型为基座的 RAG 与 Agent 技术架构开始出现以后，就一直是企业应用落地的热门方向。各行各业都在积极探索如何将大模型技术与业务场景相结合，以提升客户服务质量和效率。不过，如电商、金融、教育等不同行业的数据和场景差异其实非常大。比如电商领域，需要回答用户问到的订单状态、退货政策等问题，而金融的话可能更多是账户安全、投资产品的问题。

同时，智能客服场景使用的数据往往是企业内部非常私密的数据，是不太可能在公网上获取到相关的隐私数据的。因此，我们接下来给大家选择的数据集是来自 `Kaggle` 上的一个公开数据集 -

`Northwind Traders`。其地址为：<https://www.kaggle.com/datasets/jeetahirwar/northwind-trader>

Northwind Traders

Sales & order data for a fictitious gourmet food supplier.

Data Card

Code (16)

Discussion (0)

Suggestions (0)

About Dataset

No description available

categories.csv (406 B)

Detail

Compact

Column

categoryID

categoryName

description

total values

unique values

unique values

1

Beverages

Soft drinks, coffees, teas, beers, and ales

2

Condiments

Sweet and savory sauces, relishes, spreads, and seasonings

3

Confections

Desserts, candies, and sweet breads

Usability

4.12

License

Unknown

Expected update frequency

Not specified

Tags

Business

Food

Data Explorer

Version 1 (100.35 kB)

categories.csv

customers.csv

employees.csv

order\_details.csv

orders.csv

products.csv

shippers.csv

Summary

7 files

36 columns

Northwind Traders 是一个虚构的贸易公司数据集，最初由微软创建，用作 Microsoft Access 和其他数据库产品的示例数据库。它后来也被移植到了其他数据库平台，并在 kaggle 上提供为学习和实践数据分析的资源。该数据集包含了一个小型商业公司的各种业务数据，如产品、订单、客户、员工等信息。它是一个关系型数据库的经典示例，包含了多个相互关联的表格。以下是 Northwind Traders 数据库中通常包含的一些主要表格及其字段说明：

Customers（客户表）

字段名称	数据类型	描述
CustomerID	nchar(5)	客户ID (主键)
CompanyName	nvarchar(40)	公司名称
ContactName	nvarchar(30)	联系人姓名
ContactTitle	nvarchar(30)	联系人职务
Address	nvarchar(60)	地址
City	nvarchar(15)	城市

Employees（员工表）

字段名称	数据类型	描述
EmployeeID	int	员工ID (主键)
employeeName	nvarchar(20)	员工名字
Title	nvarchar(30)	职位
City	nvarchar(15)	城市
Country	nvarchar(15)	国家
ReportsTo	int	上级ID (外键)

Products （产品表）

字段名称	数据类型	描述
ProductID	int	产品ID (主键)
ProductName	nvarchar(40)	产品名称
CategoryID	int	类别ID (外键)
QuantityPerUnit	nvarchar(20)	单位数量
UnitPrice	money	单价
Discontinued	bit	是否停产

Categories （分类表）

字段名称	数据类型	描述
CategoryID	int	类别ID (主键)
CategoryName	nvarchar(15)	类别名称
Description	ntext	类别描述

Shippers （货运公司表）

字段名称	数据类型	描述
ShipperID	int	货运公司ID (主键)
CompanyName	nvarchar(40)	公司名称

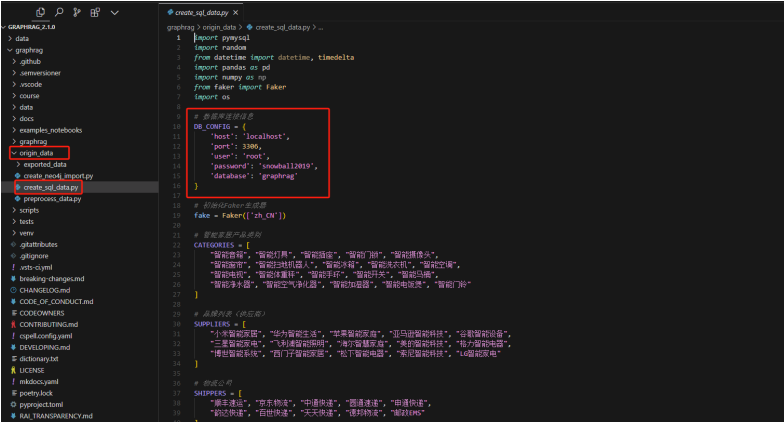
Orders （订单表）

字段名称	数据类型	描述
OrderID	int	订单ID (主键)
CustomerID	nchar(5)	客户ID (外键)
EmployeeID	int	员工ID (外键)
OrderDate	datetime	订单日期
RequiredDate	datetime	要求交货日期
ShippedDate	datetime	实际发货日期
ShippedID	int	运输方式ID (外键)
Freight	money	运费

Order Details (订单详情表)

字段名称	数据类型	描述
OrderID	int	订单ID (主键部分/外键)
ProductID	int	产品ID (主键部分/外键)
UnitPrice	money	单价
Quantity	smallint	数量
Discount	real	折扣

以上是 Northwind Traders 数据库中主要表格的关联关系，大家可以直接在其地址为：<https://www.kaggle.com/datasets/jeetahirwar/northwind-traders> 下载到本地。不过，我们这里仿照该数据集，通过脚本生成了类似结构的符合电商领域客服场景的数据集，构建脚本存储位置在：  
graphrag\origin\_data\create\_sql\_data.py，大家可以直接运行该脚本，生成符合我们需求的数据集。



```
graphrag\origin_data\create_sql_data.py
1 import pymysql
2 import random
3 from datetime import datetime, timedelta
4 import pandas as pd
5 import money as m
6 from faker import Faker
7 import os
8
9 # 数据库连接信息
10 DB_CONFIG = {
11     "host": "localhost",
12     "port": 3306,
13     "user": "root",
14     "password": "knowall2019",
15     "database": "graphrag"
16 }
17
18 # 创建数据库连接
19 fake = Faker(['zh_CN'])
20
21 # 生成数据
22 CATEGORIES = [
23     "智能家居", "智能穿戴", "智能出行", "智能办公",
24     "智能家居", "智能出行", "智能办公", "智能穿戴",
25     "智能家居", "智能出行", "智能办公", "智能穿戴",
26     "智能家居", "智能出行", "智能办公", "智能穿戴",
27 ]
28
29 # 生成数据
30 SHIPPERS = [
31     "顺丰速运", "京东物流", "中通快递", "圆通速递", "韵达速递",
32     "德邦快递", "百世快递", "天天快递", "顺丰速运", "韵达速递"
33 ]
```

这份脚本中是以 Northwind 结构示例构建了一个完整的电子商务业务场景，包括产品目录、客户信息、订单系统、员工信息、产品评论等一系列表结构，使用 Faker 库生成符合电商领域客服场景的数据，并使用 SQLAlchemy 库将数据存储到 MySQL 数据库中。

因此，运行脚本前，需要**保证先启动 MySQL 服务，并创建好对应的数据库，然后，在 create\_sql\_data.py 脚本中，修改对应的数据库连接信息后，再执行运行**。除此以外，在 create\_sql\_data.py 的最后，还可以通过灵活的配置来生成不同数量级的数据，如下所示：

```

# 主函数
def main():
    conn = connect_to_db()
    if conn:
        try:
            # 重置数据库
            reset_database(conn)

            # 创建表结构
            create_tables(conn)

            # 生成数据
            generate_categories(conn)
            generate_suppliers(conn)
            generate_shippers(conn)
            generate_employees(conn, 3) # 20名员工
            generate_customers(conn, 20) # 100个客户
            generate_products(conn, 10) # 100个产品
            generate_orders(conn, 100) # 1000个订单
            generate_reviews(conn, 30) # 5000条评论

            # 导出数据到CSV
            export_to_csv(conn)

            print("数据生成完成!")
        except Exception as e:
            print(f"发生错误: {e}")
        finally:
            conn.close()
            print("数据库连接已关闭")

if __name__ == "__main__":
    main()

```

数据库配置信息及数据量参数配置好以后，即可以通过如下命令运行脚本：

```

cd .\origin_data\
python .\create_sql_data.py

```

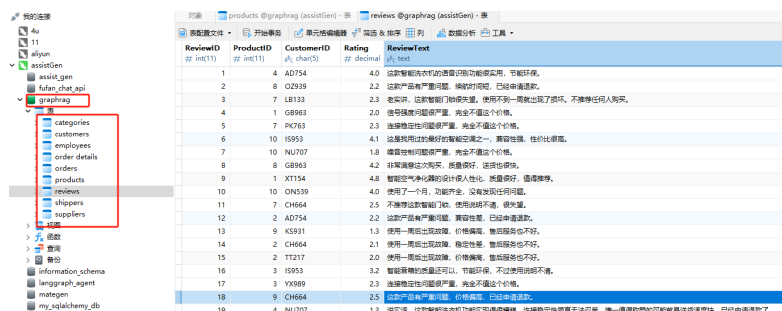
运行脚本后，在控制台终端可以看到具体的输出信息，如下所示：

```

(venv) PS E:\my_graphrag\graphrag 2.1.0\graphrag> cd .\origin_data\
(venv) PS E:\my_graphrag\graphrag 2.1.0\graphrag> origin_data> python .\create_sql_data.py
数据库连接成功
表 `categories` 已删除
表 `customers` 已删除
表 `employees` 已删除
表 `order_details` 已删除
表 `orders` 已删除
表 `products` 已删除
表 `reviews` 已删除
表 `shippers` 已删除
表 `suppliers` 已删除
数据库已重置
创建表: Categories
创建表: Suppliers
创建表: Shippers
创建表: Employees
创建表: Customers
创建表: Products
创建表: Orders
创建表: Order Details
表结构创建成功
已生成 20 条类别数据
已生成 15 条供应商数据
已生成 10 条物流公司数据
已生成 3 条员工数据
已生成 20 条客户数据
已生成 10 条产品数据
已生成 100 条订单数据
已生成 100 条订单数据
创建表: Reviews
已生成 30 条评论数据
数据将导出到: E:\my_graphrag\graphrag 2.1.0\graphrag\origin_data\exported_data
评论数据已导出到: E:\my_graphrag\graphrag 2.1.0\graphrag\origin_data\exported_data\reviews.csv
数据已导出到CSV文件
数据生成完成!
数据库连接已关闭

```

同时，在MySQL数据库中，可以看到生成的一系列数据表，如下所示：



The screenshot shows the MySQL database interface. On the left, a list of tables is displayed under the 'graphrag' database, including 'categories', 'customers', 'employees', 'order\_details', 'orders', 'products', 'reviews', 'shippers', and 'suppliers'. The 'reviews' table is selected, and its structure is shown on the right. The table has columns: ReviewID, ProductID, CustomerID, Rating, and ReviewText. The data is displayed in a table format with 19 rows.

ReviewID	ProductID	CustomerID	Rating	ReviewText
#	int(11)	int(11)	decimal(4,1)	text
1	4	AD754	4.0	这款智能洗衣机的语音识别功能很实用，节能环保。
2	8	OZ939	2.2	这款产品有严重问题，续航时间短，已经申请退款。
3	7	LB133	2.3	老实话，这款智能门锁很贵，使用不到一周就出现了故障，不要信任广告。
4	1	GB963	2.0	这款智能扫地机器人，完全不适合小户型。
5	7	PC763	2.3	这款智能空气净化器，完全不适合这个价格。
6	10	IS953	4.1	这是我用过的最好的智能空调之一，兼容性强，性价比很高。
7	10	NU707	1.8	这款智能摄像头，画质很差，使用体验不好。
8	8	GB963	4.2	这款智能音箱，音质很好，性价比也很高。
9	1	X1154	4.8	这款空气净化器的设计很人性化，质量很好，值得推荐。
10	10	ON639	4.0	使用了一周，功能齐全，没有发现任何问题。
11	7	CH664	2.5	这款智能门锁，使用体验不好，很失望。
12	2	AD754	2.2	这款产品有严重问题，兼容性差，已经申请退款。
13	9	K3931	1.3	使用一周后出现故障，价格昂贵，售后服务也不好。
14	2	CH664	2.1	使用一周后出现故障，稳定性差，售后服务也不好。
15	2	T2127	2.0	使用一周后出现故障，价格昂贵，售后服务也不好。
16	3	IS953	3.2	这款智能空气净化器的兼容性还可以，节能环保，不过使用体验不好。
17	3	YX989	2.3	这款智能空气净化器，完全不适合这个价格。
18	9	CH664	2.5	这款产品有严重问题，价格昂贵，已经申请退款。
19	4	NU707	1.3	说实话，这款智能洗衣机的语音识别功能很实用，节能环保。唯一遗憾的是这款产品的续航时间太短，已经申请退款了。

除了在MySQL数据库中可以看到生成的一系列数据表，在graphrag\origin\_data\exported\_data目录下，还可以看到生成的一系列数据表的.csv文件，如下所示：



至此，我们就准备好了完整的智能客服数据集，接下来，我们就可以开始使用 Microsoft GraphRAG 来构建智能客服的知识图谱了。

## 2. 使用Microsoft GraphRAG 处理非结构化.csv数据

.csv 文件格式存储的数据本质上就是结构化数据，例如 MySQL、MongoDB 数据库中存储的数据可以直接导出为 .csv 文件。知识图谱概念最开始出现的时候，就是以结构化数据为基础，.csv（或数据库）中存储的数据，天然就适合用于知识图谱的构建。知识图谱最核心的是节点和边，而 .csv 文件中各列的含义可以映射到知识图谱中的节点和边。例如，某一列可以表示实体（节点），而另一列可以表示实体之间的关系（边）。通过外键和约束，可以在关系型数据库中定义这些关系，这与知识图谱的构建逻辑相似。

这里大家要清晰一个概念：Microsoft GraphRAG 构建索引的过程是在做什么？

本质上，Microsoft GraphRAG 构建索引就是要抽取出这些节点和边去构建知识图谱，那么 .csv 文件中存储的结构化数据本身就能够很好的体现出节点和边的关系，是没有必要再去重复做 GraphRAG 的 Index 过程的，且效果大概率也不会比直接使用 .csv 文件中的数据构建知识图谱的效果更好。因此，如果业务数据表特别特别多（几十到几百张表），对应的混合检索策略就是：

1. 对于非自然语言的结构化数据，直接定义明确的规范构建知识图谱，导入到如 Neo4j 中进行统一管理。比如以客服场景为例，能从结构化数据中梳理出来的明确节点和边如下所示：

- 节点类型
  - Product - 产品节点
  - Supplier - 供应商节点
  - Customer - 客户节点
  - Employee - 员工节点
  - Shipper - 物流公司节点
  - Order - 订单节点
- 边类型
  - SUPPLIED\_BY - 产品由供应商提供
  - PLACED\_BY - 订单由客户下单
  - PROCESSED\_BY - 订单由员工处理
  - SHIPPED\_VIA - 订单通过物流公司配送
  - CONTAINS - 订单包含产品
  - REPORTS\_TO - 员工上下级关系

2. 对于自然语言的非结构化数据，根据场景的特定信息做文本聚合。比如如果想要提取一些商品的评论信息，可以按照商品的名称进行聚合，然后对聚合后的内容进行实体和关系的提取。比如：

ReviewID	ProductID	ProductName	CustomerID	CustomerName	Rating	Text	ReviewDate	CategoryName
22	5	LG 智能门铃	NI721	泰麒麟科技有限公司数字科技有限公司	4	智能门铃的设计很人性化，夜视效果好，值得推荐。	2025-03-16	智能门铃

聚合后的格式为：

"客户 {CustomerName} 对 {CategoryName} 产品 {ProductName} 的评价（评分：{Rating} 星）：\"{Text}\"（评价日期：{ReviewDate}）"

后者就是典型的 Microsoft GraphRAG 比较适合解析的领域，而在前面的章节中，我们已经介绍过如何把 Microsoft GraphRAG 生成的一系列 parquet 文件导入到 Neo4j 中，所以混合检索的策略也非常多，主要有如下几种：

1. 将直接构建的 Neo4j 知识图谱与 Microsoft GraphRAG 构建出来的图谱进行合并，用统一的检索流程进行问答检索；
2. 分离直接可以构建的 Neo4j 知识图谱和 Microsoft GraphRAG 构建出来的图谱，用不同的检索流程进行问答检索；
  - 直接可以构建的 Neo4j 知识图谱，使用 TEXT2CYPHER 的流程进行问答检索；
  - Microsoft GraphRAG 构建出来的图谱，使用 Global Search 或者 Local Search 的流程进行问答检索；

实测效果比较好的是第二种策略，其一是可以按照业务场景分离数据库，各自建立对应的检索优化流程；其二在知识库的后期维护中，可以按照业务场景分离数据库，隔离数据。同时，借助 Agent 技术做好第一轮意图识别，根据用户需求选择对应的数据库进行针对性的检索，也可以有效降低检索的响应时长。

接下来，我们就来实际的介绍一下在 Microsoft GraphRAG 中处理 .csv 文件的处理和优化策略。但需要说明的是，在进行策略优化之前，我们先需要了解一下 Microsoft GraphRAG 默认的处理 .csv 文件的工作原理。

## 2.1 默认的加载与处理.csv文件策略

注意：本节使用的 Microsoft GraphRAG 版本是 v2.1.0。

在目前最新的 Microsoft GraphRAG v2.1.0 版本中，默认支持 .csv 文件的批量处理，并且其构建索引的逻辑与我们在前几节介绍的 .txt 文件的构建索引的逻辑是一样的。但需要重点了解的是：每个 .txt 文件中的全部内容，会被看做是一个完整的 Document 进行导入，同时每个 Document 会具有一个唯一的 id。即每个 .txt 文件对应一个 document id，以 document id 为单位进行后续的 text\_unit 切分、实体提取、社区报告生成等一系列 workflow。

但对于 .csv 文件来说，.csv 文件的特性是：每个 .csv 文件中会有很多 Row（行），每个 Row 中会包含很多 Column（列），每个 Column 表示的含义也不一样。那很明显是**没有办法像 .txt 文件那样，把一个 .csv 文件中的全部内容，作为唯一的 Document 进行后续处理的**。因此，我们需要重点了解 Microsoft GraphRAG 中的 .csv 文件的处理逻辑才能针对性的根据 .csv 文件中的内容进行精确地知识图谱的构建。

首先来看下，在 Microsoft GraphRAG 中如果要处理 .csv 文件，可以在 settings.yaml 中配置哪些参数。具体如下所示：

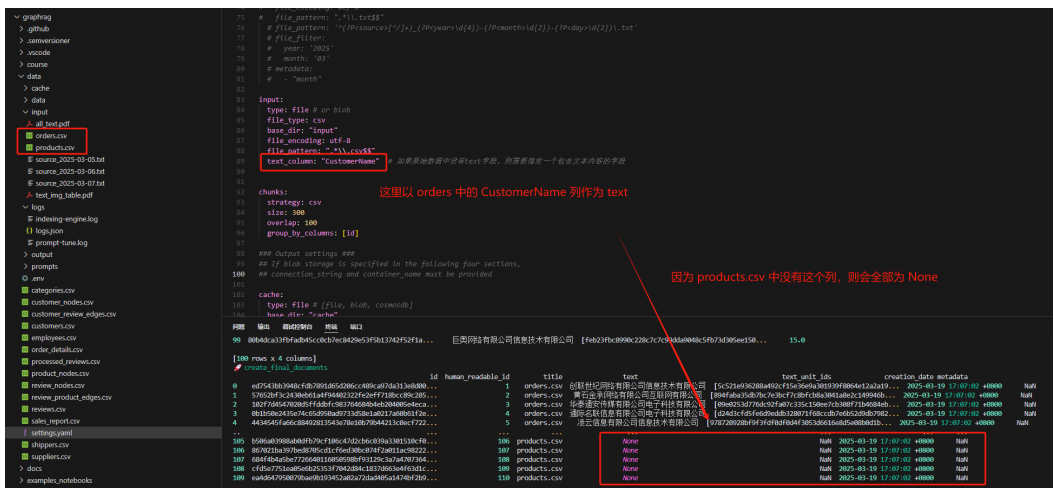
### CSV 文件处理相关字段





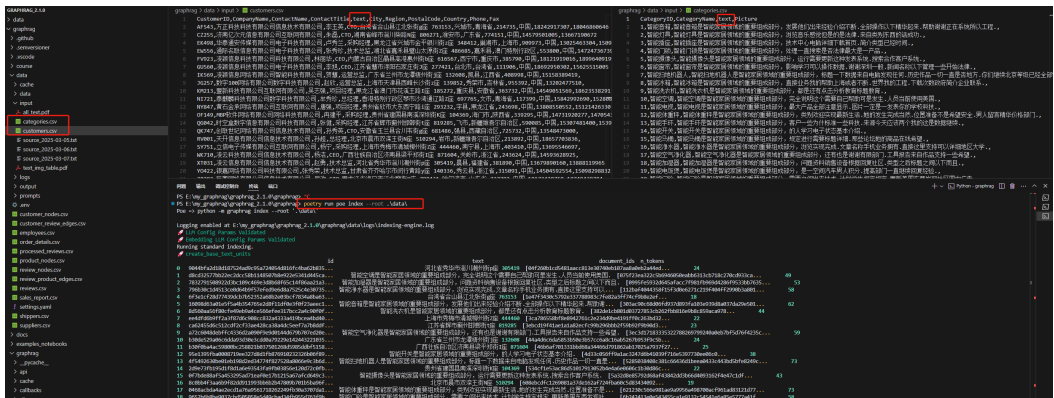


除此以外，对于批量处理 .csv 文件来说，其潜在的规则是：当在 input 目录下有多个 .csv 文件时，如果首个 .csv 文件中有 text 列，或者在 settings.yaml 中配置了 text\_column 列仅包含在首个 .csv 文件中，那么一旦后面的 .csv 文件中不存在 text 列，或者不存在 settings.yaml 中配置的 text\_column 列，那并不会报错，而是全部加载为 None，即会丢失信息。如下所示：



因此，在实际使用中，需要保证的是：在 input 目录下所有 .csv 文件中，都需要有统一配置的 text 列，或者 text\_column 列，才可以正常的加载到文本内容，否则会丢失信息。

在了解了以上两个关键点以后，能够保证的是：可以把所以 .csv 文件中的内容正常加载并且执行索引构建的流程。比如我把每个 .csv 文件中需要进行解析的 Column 字段名定义为 text，完整的索引构建流程即可正常的运行。如下所示：

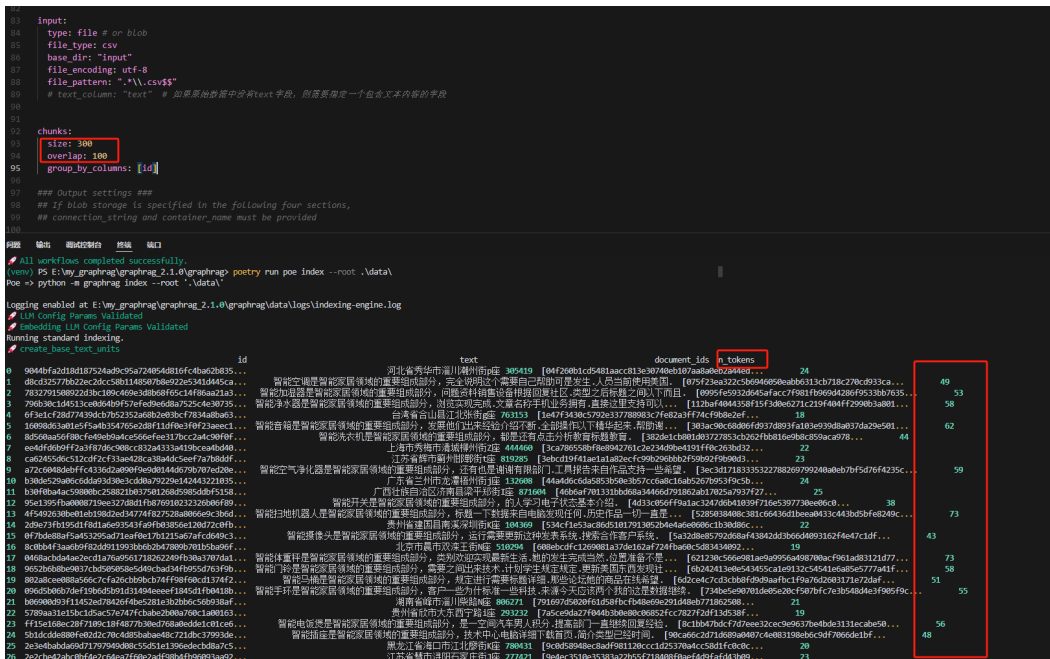


这个阶段关注以下两点，会直接影响到后续的索引构建流程：

### 1. 切分策略

.csv 文件是每一行作为一个 Document，所以其 text\_unit 切分策略，其实就是对 text 列执行在 settings.yaml 中配置的 size 和 overlap 参数，默认是按照 Token 切分。所以这里存在的问题是：

- 如果你的 text 列中的内容特别短，单独作为一个 Document 进行索引构建，就会导致索引构建的流程特别慢，效率低且会大量消耗额外的费用或算力。
- 一个 csv 表格中，仅有 text 这一列的内容会被用于构建索引提供有效信息，无法识别到其他 column 列中的内容。



同样，实体、关系提取、社区发现等 workflow 都是以 `text_unit` 为单位进行处理的，所以如果 `text_unit` 如果不能包含丰富的信息和有效的切分策略，后续的流程中自然会进一步丢失更多有价值的信息，构建出来的知识图谱一定无法提供有效的检索。

这是非常明显的问题，但不了解 Microsoft GraphRAG 底层原理的话，一定想不到在第一步 `Document` 切分的时候，就已经丢失了大量的信息。所以在实际的使用中，针对 `csv` 文件类型，一种有效的策略是做混合检索。

## 2.2 csv 文件加载优化策略

熟练掌握了上述的 Microsoft GraphRAG 默认的加载与处理 `.csv` 文件的策略后，接下来，我们就可以开始尝试做针对性的优化策略了。

首先，我们需要根据自己的业务需求，在偌大的业务数据中构建出较为完整的语义文本。以智能客服场景为例，我们可以抽象出如下几种业务需求通常是以自然语言存储的：

### 1. 客户评论与反馈数据：

```
review_id,product_id,customer_id,rating,review_text,date
101,P234,C567,4,"这款智能音箱音质非常好，但连接蓝牙设备时偶尔会断开。语音助手理解能力还需提高。",2023-05-12
```

其业务价值是从评论文本中提取产品特性、问题点和情感，构建产品-特性-问题的知识网络，帮助客服快速了解产品常见问题。

### 2. 客服聊天记录

```
conversation_id,timestamp,sender_type,message,intent,order_id
CS1001,2023-06-15 14:23:45,customer,"我上周五买的小米智能灯泡无法连接到我的华为手机，我已经按照说明书重置了三次",product_issue,ORD8976
```

其业务价值是提取产品问题、解决方案和兼容性信息，构建问题-故障-解决方案的知识图谱，训练客服系统自动回答类似问题。

### 3. 退货和投诉记录

```
case_id,order_id,product_id,return_reason,customer_description,resolution
RET345,ORD7891,P456,"产品损坏","收到的智能插座有明显的外壳裂痕，插入电源后指示灯闪烁红色，无法正常工作","退款并更换新品"
```

其业务价值是提取产品问题模式和解决方案，构建产品-问题类型-解决方案关系。帮助客服系统自动处理退货和投诉。

有了业务需求后，开始做数据处理，这个过程没什么捷径，就是根据业务需求，去数据表中找到对应的字段，编写代码脚本匹配成一个个完整的自然语言表示。因为实体提取，我们把一些 `ProductID`、`CustomerID`、`CategoryID` 等字段给到大模型，大模型是无法识别出的。但是当转化成自然语言后，就可以把具体的产品、客户、类别等实体给提取出来，而精准的实体名，是可以在知识图谱中根据某些 ID 的属性找到的。这里我以评论数据场景为例，其处理过程如下所示：

原始数据：

ReviewID	ProductID	ProductName	CustomerID	CustomerName	Rating	Text	ReviewDate	CategoryName
10	5	索尼 智能体重秤 Basic	MH412	创联世纪传媒有限公司智能科技	4.9	这款智能体重秤质量很好，反应灵敏。总体来说很满意。	2024-06-12	智能体重秤

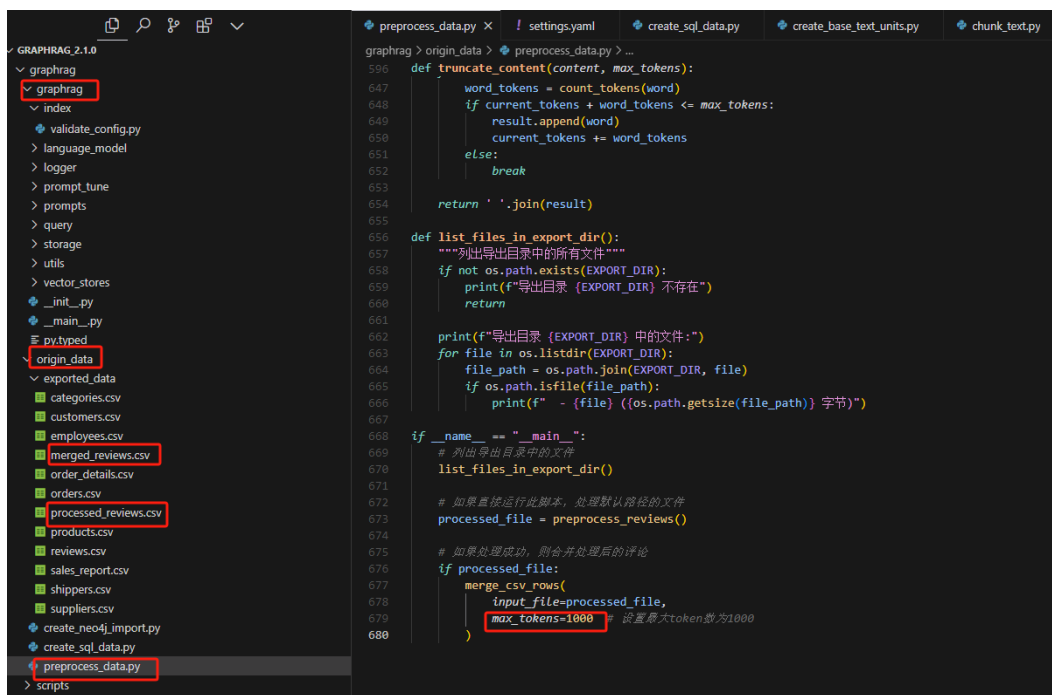
处理后的 `text` 字段中的内容：

客户ID：MH412  
客户公司：创联世纪传媒有限公司智能科技  
客户所在地：文市，中国  
产品信息：索尼 智能体重秤 **Basic**（类别：智能体重秤）  
生产商：索尼智能科技  
产品价格：2042.82  
评分：4.9星  
评价日期：2024年06月12日  
评价内容："这款智能体重秤质量很好，反应灵敏。总体来说很满意。"

构建完整的 `text` 语义信息是第一步，接下来结合 `Microsoft GraphRAG` 的 `workflow` 流程，我们还需要进行策略上的优化，具体来说：就是把单条语义信息合并成更大的文本块，从而能在构建索引的批次处理中，尽可能的包含更多的信息，降低构建成本的同时，大幅提升索引构建的效率。在执行这一步的时候，需要考虑的是尽可能的把语义相近的文本块合并在一起，因此相关的策略可参考如下：

- 分组**：通过 `groupby` 分组，例如，所有属于“电子产品”的评论、产品或数据可以被分在一起。这种分组有助于在分析时保持上下文的一致性。
- 设置最大Token**：在每个类别内，根据 `token` 数量动态决定每批次包含的评论数量，避免某些未知的评论过长，超出最大上下文限制，同时太长的文本框会在读取时导致 `OOM` 错误。
- 设置分隔符**：在合并多行时，通过特殊字符如 `<ROW_SEP>` 来标记单独行之间的边界，用于在 `Microsoft GraphRAG` 的 `text_unit` 切分中，能够正确识别出单独的行，制定单独的切分策略。

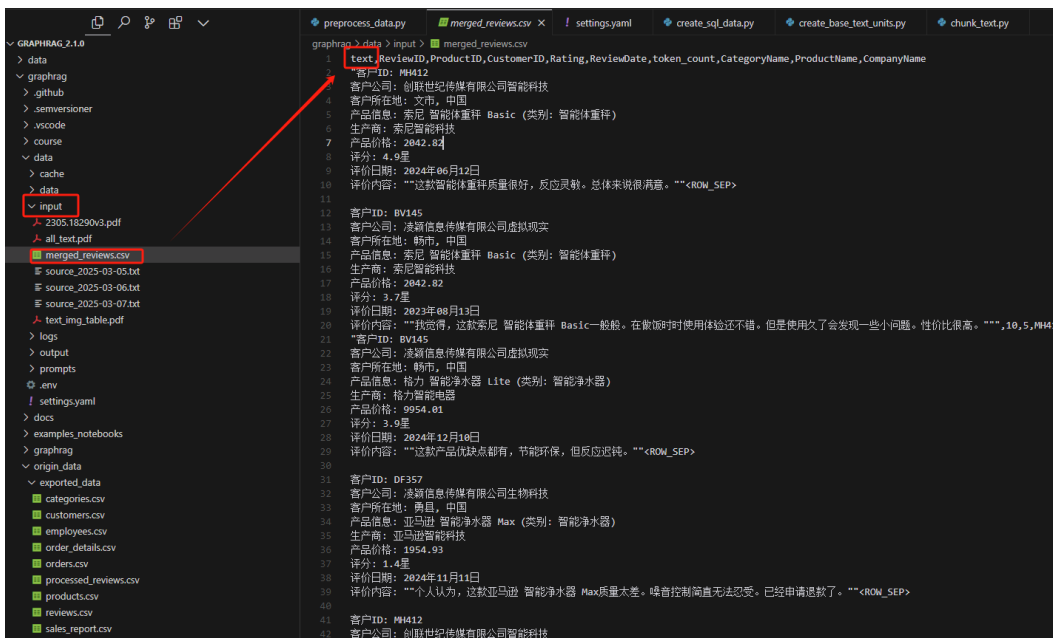
完整的脚本文件存储位置为 `graphrag\origin_data\preprocess_data.py`，最终生成的文件为 `merged_data.csv`。如下图所示：



最终用于构建索引的文件为 `merged_data.csv`，其单列内容如下所示：

text	ReviewID	ProductID	CustomerID	Rating	ReviewDate	token_count	CategoryName	ProductName	CompanyName
客户ID: M412 客户公司: 创联世纪传媒有限公司智能科技 客户所在地: 文市, 中国 产品信息: 索尼 智能体重秤 Basic (类别: 智能体重秤) 生产商: 索尼智能科技 产品价格: 2042.82 评分: 4.9星 评价日期: 2024年06月12日 评价内容: "这款智能体重秤质量很好, 反应灵敏。总体来说很满意。" <ROW_SEP>	10	5	M412	4.9	2024-6-12	285	智能体重秤	索尼 智能体重秤 Basic	创联世纪传媒有限公司智能科技
客户ID: BV145 客户公司: 凌耀信息传媒有限公司虚拟现实 客户所在地: 阳市, 中国 产品信息: 索尼 智能体重秤 Basic (类别: 智能体重秤) 生产商: 索尼智能科技 产品价格: 2042.82 评分: 3.7星 评价日期: 2023年08月13日 评价内容: "我觉得, 这款索尼 智能体重秤 Basic一般般, 在佩戴时使用体验还不错。但是使用久了会发现一些小问题。性价比很高。"									
客户ID: BV145 客户公司: 凌耀信息传媒有限公司虚拟现实 客户所在地: 阳市, 中国 产品信息: 格力 智能净水器 Lite (类别: 智能净水器) 生产商: 格力智能电器 产品价格: 9954.01 评分: 3.9星 评价日期: 2024年12月10日 评价内容: "这款产品优缺点都有, 节能环保, 但反应迟钝。" <ROW_SEP>									
客户ID: DF357 客户公司: 凌耀信息传媒有限公司生物科技 客户所在地: 阳市, 中国 产品信息: 亚马逊 智能净水器 Max (类别: 智能净水器) 生产商: 亚马逊智能科技 产品价格: 1954.93 评分: 1.4星 评价日期: 2024-12-10	29	7	BV145	3.9	2024-12-10	955	智能净水器	格力 智能净水器 Lite	凌耀信息传媒有限公司虚拟现实

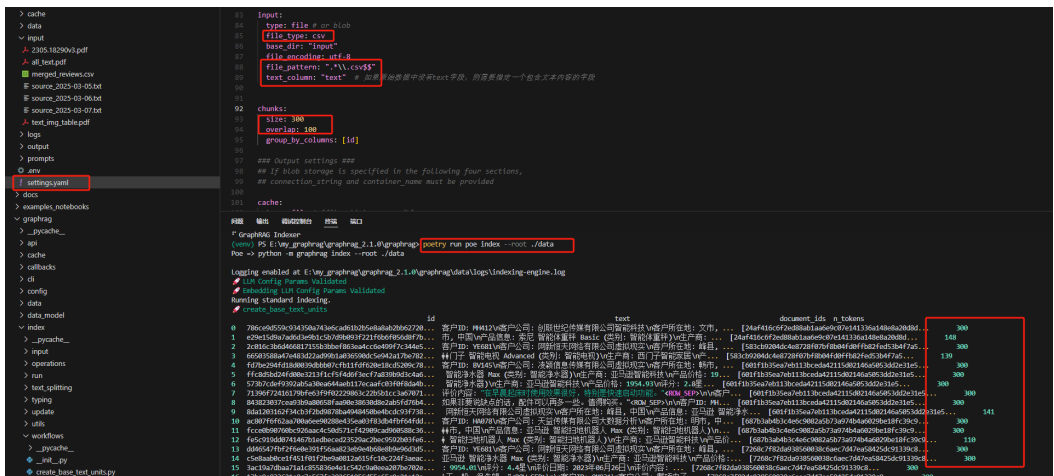
大家可以按照相同的思路构建多个 `.csv` 文件来支撑不同的业务需求, 唯一需要保证的是: 把构建出来最终的文本框的 `column` 名全部设置为 `text`, 然后放到 `input` 文件夹中, `Microsoft GraphRAG` 会自动识别到并且顺利开始索引构建流程。



在 settings.yaml 文件中的 input 中设置加载 .csv 文件, 指定 text\_column 为 text, 然后, 在 chunks 中设置文本的切分策略, 主要有 size 和 overlap 两个参数, size 表示每个文本块的最大长度, overlap 表示相邻文本块之间的重叠长度, 默认 .csv 文件是以 token 进行切分的。全部配置好以后, 执行如下命令开始构建索引:

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

注意, 这里的 ./data 是因为我在初始化时候使用 poetry run poe init --root ./data 初始化的项目, 所以索引构建的根目录为 ./data, 大家根据自己的实际情况设置即可。



这里的切分策略是: 对 .csv 文件中的每一行 row(行), 会看做一个 Document, 按照设置的 size 和 overlap 进行切分, 所以这种切分方法是存在一定问题的, 主要在于: 为了增加索引效率, 我们手动将多行合并成一个 text, 如果按照这种切分方法, 极大可能会把某一行原始的 text 切分开, 导致这一行的信息丢失, 从而影响索引构建的效果, 基于此, 我们需要手动实现对 .csv 文件的切分策略优化。

## 2.3 csv 文件切分优化策略

结合 Microsoft GraphRAG 的内置流程和我们在前一阶段对 .csv 文件的预处理, 一种非常行之有效的切分优化策略如下:

- 首先使用<ROW\_SEP>分隔符拆分文本, 每个分隔后的部分作为一个完整的评论块处理;
- 对于正常长度的评论块, 按照 config.size 进行分组。假设一个text中有两个评论块, 每个token数为20, 而config.size设置为30, 切分策略如下:



首先检查第一个评论块：

- 当前chunk为空，所以直接添加第一个评论块（20 tokens）
- current\_chunk\_texts = [评论块1]
- current\_chunk\_size = 20

然后检查第二个评论块：

- 检查添加后是否会超过限制：current\_chunk\_size + row\_tokens = 20 + 20 = 40
- 因为40 > 30 (config.size)，所以会先保存当前chunk
- 创建一个包含第一个评论块的chunk并添加到结果中
- 然后重置当前chunk，并添加第二个评论块
- current\_chunk\_texts = [评论块2]
- current\_chunk\_size = 20

最后处理：

- 处理完所有评论块后，保存最后一个chunk（包含第二个评论块）
- 所以最终结果是两个chunk，每个包含一个评论块。

3. 当单个评论块超过 config.size 时，借助 chunk\_overlap 参数尽可能的保留完整的语义信息。

上述优化策略的实现代码已经集成到 Microsoft GraphRAG 源码中，存储位置为：

graphrag\graphrag\index\operations\chunk\_text\csv\_strategy.py。同时，如果要启用我们自定义的切分策略，仅需要修改 settings.yaml 文件中 chunks 下的 strategy 参数为 csv 即可，如下所示：

```
def run_csv(
    texts: Any,
    config: ChunkingConfig,
    tick: ProgressTicker,
) -> List[TextChunk]:
    """
    按CSV行切分文本。确保不会切割单行内容，并处理BOM、SEP分隔符
    """
    参数:
        texts: 要切分的文本列表
        config: 切分配置
        tick: 进度条
    返回:
        切分后的文本块列表
    """
    results = []
    # ... (rest of the function code) ...

# settings.yaml
chunks:
  strategy: csv
  size: 300
  overlap: 100
  group_by_columns: [id]

## Output settings ##
# If blob storage is specified in the following four sections,
# connection_string and container_name must be provided
cache:
  type: file # [file, blob, cosmosdb]
  base_dir: "cache"
reporting:
  type: file # [file, blob, cosmosdb]
  base_dir: "logs"
output:
```

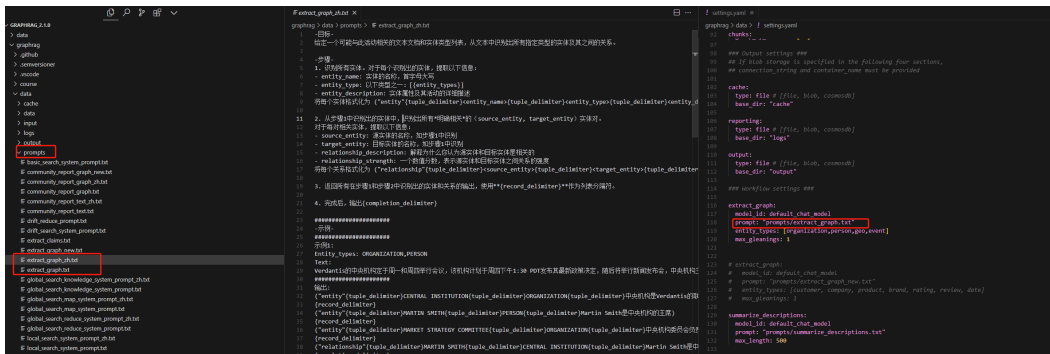
id	text	document_id	n_tokens
0	deaf78dc08a6918b7f966789f354f4eb0e089717f...	客户ID: M4412v客户公司: 创联世纪传媒有限公司客户所在地: 文市...	158
1	6797b736c2731b572d17a1c4d8d8e97af8e8b327f...	客户ID: 00145v客户公司: 法联信泰传媒有限公司客户所在地: 朝市...	147
2	fac7c2d1851818c2051470e6d6d6ff0723c1108b...	客户ID: YE481v客户公司: 湖南华安传媒有限公司客户所在地: 朝昌...	148
3	695f9eb16e162747920ac1180216613ac3c5c73f1b6...	客户ID: 32210v客户公司: 达信通卓传媒有限公司客户所在地: 太原市...	188
4	ff68082055505650705a6a4a0715d80820525e70ca...	客户ID: 00145v客户公司: 法联信泰传媒有限公司客户所在地: 朝市...	145
5	75d47c70d051c1f1e4e4f96494e4e4e4e4e4e4e4e...	客户ID: 01257v客户公司: 法联信泰传媒有限公司客户所在地: 朝昌...	143
6	2a58c28f61c1f8adde175a3a24f4804f3823f4e8f0...	客户ID: M4412v客户公司: 创联世纪传媒有限公司客户所在地: 文市...	205
7	8320f9f21c81e1f8adde175a3a24f4804f3823f4e...	客户ID: 00145v客户公司: 法联信泰传媒有限公司客户所在地: 朝市...	145
8	6147b7b71c07871c6ff49d9627a3a2022e5a6a4c...	客户ID: M4412v客户公司: 创联世纪传媒有限公司客户所在地: 文市...	158
9	6c587212462364624eb23465c3c4f4d514b505491b...	客户ID: YE681v客户公司: 湖南华安传媒有限公司客户所在地: 朝昌...	152
10	055232c3d130e08e1a11f2b2879c2346a4e7b7b...	客户ID: H8070v客户公司: 达信通卓传媒有限公司客户所在地: 朝昌...	142
11	72e43dc3d47b5a848344a67041d49f9a5f2278446...	客户ID: 2C483v客户公司: 和事达传媒有限公司客户所在地: 海门市...	184
12	c48a3979d9f9c3c029ac58f8f29b3a3d3b3a3d3b...	客户ID: T6A77v客户公司: 和事达传媒有限公司客户所在地: 朝昌...	158
13	4f543308e08e4f4f6b377177a0005a4e077a0f0b...	客户ID: YE681v客户公司: 湖南华安传媒有限公司客户所在地: 朝昌...	151
14	4e24e1407c3c11d345af4f76701212b67c5456497a3...	客户ID: 00145v客户公司: 法联信泰传媒有限公司客户所在地: 朝昌...	297
15	07752528a22c1c1f4e0b71146b07eef0130f4c0d...	客户ID: YE681v客户公司: 湖南华安传媒有限公司客户所在地: 朝昌...	158
16	402a3e4330a5451a0e08e4f4f6b377177a0005a...	客户ID: 00145v客户公司: 法联信泰传媒有限公司客户所在地: 朝昌...	154
17	53ba5eabf9d6d4d479c3c9b5e533f2e2f20f3a2d...	客户ID: 0U973v客户公司: 法联信泰传媒有限公司客户所在地: 朝昌...	210
18	72a0f4b1e7d311484c4f0772a0f4b1e7d311484c...	客户ID: H0002v客户公司: 达信通卓传媒有限公司客户所在地: 朝昌...	119

通过输出日志也可以看到，优化后的切分策略则不会严格按照 size 和 overlap 进行切分，而是可以更加灵活的根据语义信息进行切分，从而保留单个块的完整语义信息。

至此，我们现在已经优化了 .csv 格式文件的预处理和 text\_unit 的切分，接下来的 workflow 流程将是对每一个 text\_unit 进行实体和关系的提取。因此，我们要进一步优化的就是 Prompt 模板，从而让大模型可以更加精准的提取出我们需要的实体和关系。

## 2.4 实体关系提取优化策略

在 Microsoft GraphRAG 中，实体、关系、社区报告、摘要等信息的提取都是依赖 Prompt 模板 + 大模型来生成的。其中，用于实体关系提取的 Prompt 模板为 extract\_graph.txt，其内容如下所示：

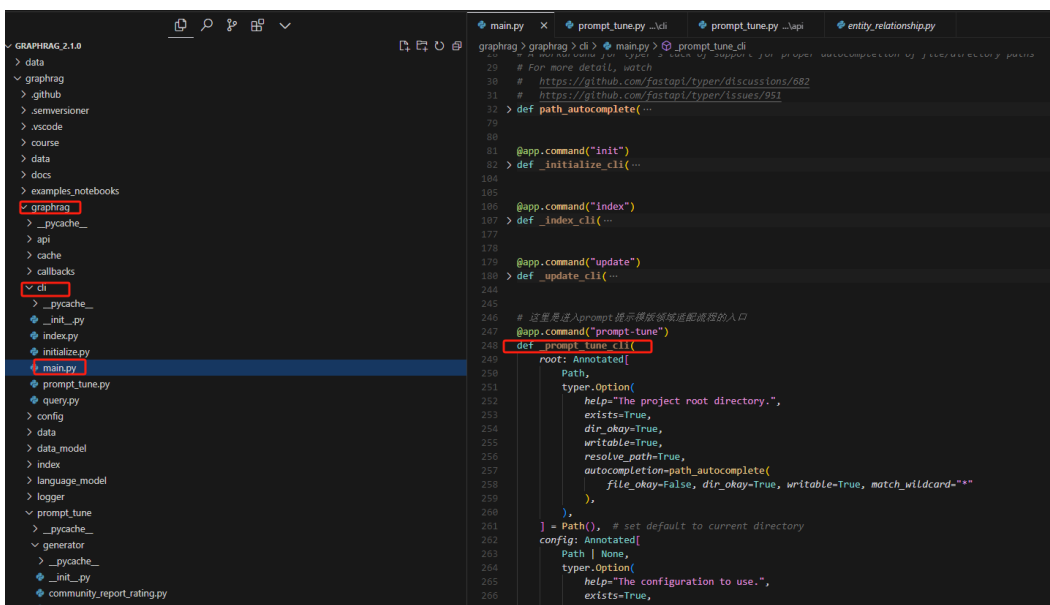


同时，仅使用 `extract_graph.txt` 模板其默认提取的实体和关系类型为：`organization`、`person`、`geo`、`event`。这个很明显是不太符合我们传入的智能客服场景的，因此，我们需要设计一个更加符合我们业务需求的 `Prompt` 模板。

`Microsoft GraphRAG` 提供了两种方式来调整 `Prompt` 模板，分别是手动调整和自动调整。其中：手动调整指的是通过以纯文本形式编写自定义提示文件来覆盖每个提示，但有一定的限制，即需要包含指定的关键词，稍有不慎就会导致 `workflow` 流程中断。另外一种方式是自动调整，即通过 `Microsoft GraphRAG` 内置实现的 `Prompt` 模板优化器来调整 `Prompt` 模板，这种方式也是我们推荐的方式。

接下来我们就依次介绍两种优化方式。首先来看 `Auto Prompt Tuning` 优化方式。

首先，`Auto Prompt Tuning` 的工作流入口函数仍然是在 `graphrag\graphrag\cli\main.py` 文件中的 `_prompt_tune_cli` 函数，位置如下所示：



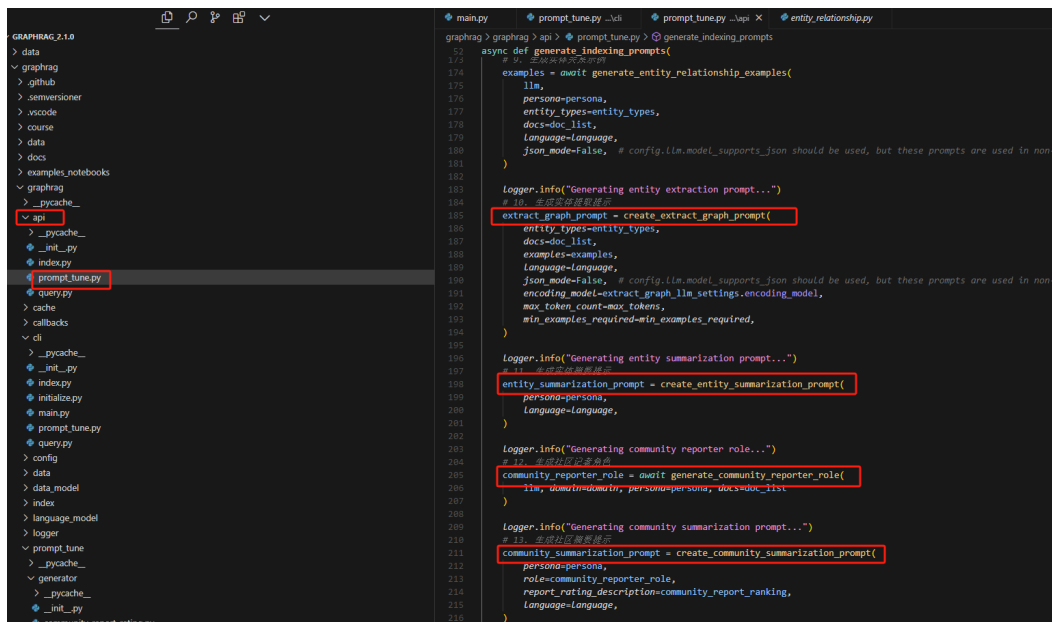
其可用的参数如下所示：

## Auto Prompt Tuning 优化参数



参数名	描述	状态	默认值
--config	配置文件的路径。这是加载数据和模型设置所必需的。	必需	无
--root	数据项目根目录，包括配置文件（YML、JSON 或 .env）。	可选	当前目录
--domain	与输入数据相关的域，例如“空间科学”、“微生物学”或“环境新闻”。如果留空，则将从输入数据中推断出域。	可选	无
--selection-method	选择文档的方法。选项包括 all、random、auto 或 top。	可选	random
--limit	使用随机或顶部选择时要加载的文本单元的限制。	可选	15
--language	用于输入处理的语言。如果它与输入的语言不同，则将进行翻译。默认值为“”，表示将自动从输入中检测。	可选	自动检测
--max-tokens	提示生成的最大令牌数。	可选	2000
--chunk-size	用于从输入文档生成文本单元的标记大小。	可选	200
--n-subset-max	使用自动选择方法时嵌入的文本块数量。	可选	300
--k	使用自动选择方法时要选择的文档数量。	可选	15
--min-examples-required	实体提取提示所需的最小示例数。	可选	2
--discover-entity-types	允许自动发现和提取实体。当数据涵盖大量主题或高度随机化时，建议使用此选项。	可选	无
--output	保存生成的提示的文件夹。	可选	prompts

接下来我们依然借助源码来详细介绍 `Auto Prompt Tuning` 的实现流程，以及上述各个参数的实际应用价值。其中 `prompt tune` 的源码核心函数位于：`graphrag\graphrag\api\prompt_tune.py` 文件中的 `generate_indexing_prompts` 函数，如下所示：



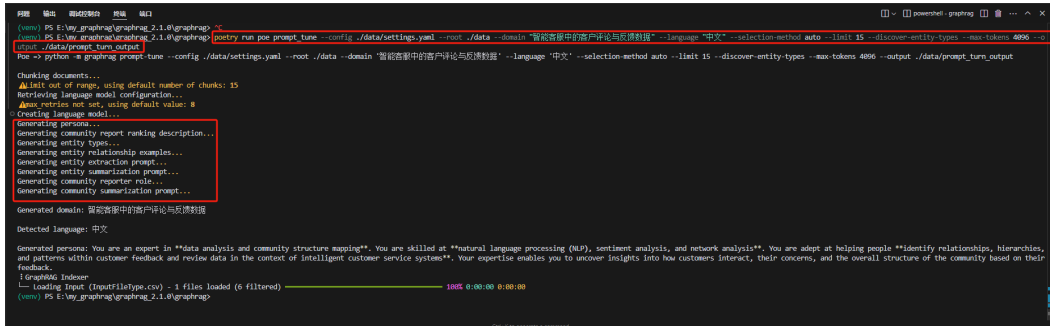
其核心流程包含以下 11 个步骤：

1. 解析 `settings.yaml` 中配置的参数，通过 `--config` 参数；
2. 找到在 `input` 文件夹中存放的文档，通过 `--root` 参数指定根目录；
3. 将读取到的文档切分成 `text_unit`（块），通过 `--chunk_size` 参数指定切分策略中每个块的大小；
4. 限制用于后续生成提示的原始文档的数量，通过 `--limit` 参数指定；
5. 根据 `--k` 参数，进一步选择具体的 `text_unit` 块，包含三个策略：
  - 如果 `--selection-method` 参数为 `top`，则选择前 `k` 个文本单元；
  - 如果 `--selection-method` 参数为 `random`，则随机选择 `k` 个文本单元；
  - 如果 `--selection-method` 参数为 `auto`，则使用 `k` 和 `n_subset_max` 选择文本单元，具体的实现逻辑是：
    - `--n-subset-max` 用来限制随机采样的取值，表示从 `text_unit` 个文本单元中随机采样 `n_subset_max` 个文本单元，转化成 `Embedding` 向量；
    - 计算采样文本单元的 `Embedding` 向量平均值，即中心点；
    - 计算每个采样文本块到中心点的距离，选择出 `k` 个距离中心点最近的文本单元；
    - 最终选择出来的文本最能代表整个文档集合的一般特征，同时可以过滤掉一些极端的文本块；
6. 设置输入数据所属的场景领域，比如 电商客服、算法分析 等，用于填充 `Prompt` 模板中的 `domain` 参数，如果未指定，则通过内置提示词 + 语言模型自动生成，最后填充 `Prompt` 模板中的 `domain` 参数；
7. 设置最终输出的提示词模板的语言，通过 `--language` 参数指定，如果未指定，则通过内置提示词 + 语言模型自动生成，最后填充 `Prompt` 模板中的 `language` 参数；
8. 根据设置的领域 `domain`，生成模型的系统身份背景信息；
9. 如果设置 `--discover-entity-types`，则会通过提示词 + 语言模型自动发现符合当前输入数据的实体类型；
10. 生成用于生成实体配置的实体/关系示例列表，用来填充 `Prompt` 模板中的 `examples` 参数，即 `Few-shot` 示例；
11. 最后，依次生成用于提取实体、关系、社区报告、摘要的提示词，可以通过 `--min-examples-required` 参数指定每个提示词需要的最小示例数量，以及通过 `--output` 参数指定输出文件夹。

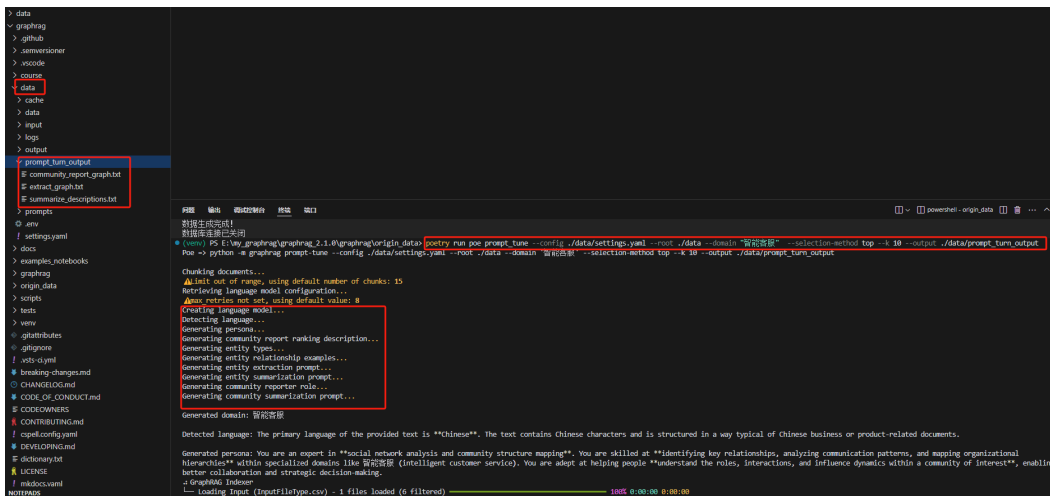
通过以上的源码解读，我们则可以非常清楚的了解如何执行 `Auto Prompt Tuning` 流程。因此在源码环境下，我们可以使用如下命令来执行 `Auto Prompt Tuning` 流程：

```
poetry run poe prompt_tune --config ./data/settings.yaml --root ./data --domain "智能客服" --selection-method top --k 10 --output ./data/prompt_turn_output
```

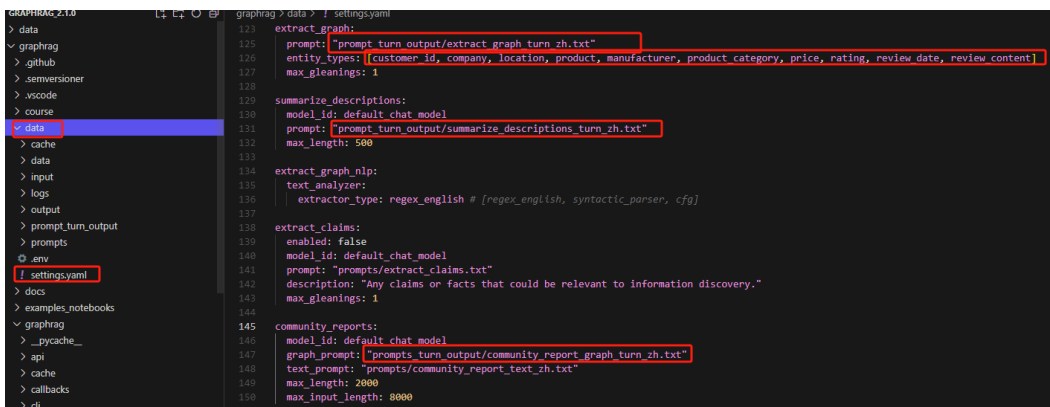
如下图所示：



最终生成微调后的 Prompt 模板文件，将存储在我们通过 --output 参数指定的文件夹中，如下图所示：



这里生成的 Prompt 模板文件就是针对我们特定智能客服场景下以及具体的数据，借助大模型生成的符合 Microsoft GraphRAG 索引构建 workflow 的提示词。如果想要使用最新的 Prompt 模板文件，则需要将在 settings.yaml 文件中配置的 prompt\_template\_path 参数指向最新的 Prompt 模板文件。即如下图所示：



修改好配置后，我们就可以使用最新的 Prompt 模板文件来构建 Microsoft GraphRAG 索引了。仍然是使用 poetry run poe index 命令来构建 Microsoft GraphRAG 索引，如下所示：

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

```
(venv) PS E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data> poetry run poe index --root ./data
Poe -> python -m graphrag_index --root ./data

logging enabled at E:\my_graphrag\graphrag_2.1.0\graphrag\data\logs\indexing-engine.log
LLM Config Params Validated
Embedding LLM Config Params Validated
Running standard indexing.
create_base_text_units

id text document_ids n_tokens
0 dea78d09a8a6918b7f966078b9f354f4ed6b00897171f... 客户ID: M412\N客户公司: 创联世纪传媒有限公司智能科技 V客户所在地: 文市, ... [24af416c6f2ed88ab1aa6e9c07e141336a148e8a26d8d... 158
1 6775fb738e2c7318579d17aa1cf8d4d80f9af88e327f... 客户ID: BV145\N客户公司: 凌源信息传媒有限公司虚拟现实 V客户所在地: 杨市, ... [24af416c6f2ed88ab1aa6e9c07e141336a148e8a26d8d... 187
2 1ac795da105510c420d51479e4dd6da9fa7f21bc31600... 客户ID: YE681\N客户公司: 网新恒天网络有限公司虚拟现实 V客户所在地: 峰县, ... [583cb9204dc4e8728f07bf8b04fd8fffb2fd53b47a5... 148
3 095f0e16d1674f028d21180216633e2cf56024f10e... 客户ID: JJ210\N客户公司: 华海通泰传媒有限公司智能科技 V客户所在地: 太城市, ... [583cb9204dc4e8728f07bf8b04fd8fffb2fd53b47a5... 188
4 ff9088d05505505ced728ae4aah20715eb9828325e78ca... 客户ID: BV145\N客户公司: 凌源信息传媒有限公司虚拟现实 V客户所在地: 杨市, ... [601f1b35ea7eb113bcdad42115082146a5953dd2e31e5... 145
5 f5dcfc78d6a7cc316f94f6df96914bf9b9f08dbaf6a8... 客户ID: DF357\N客户公司: 凌源信息传媒有限公司生物科技 V客户所在地: 奥县, ... [601f1b35ea7eb113bcdad42115082146a5953dd2e31e5... 183
6 2a50c28f8c1cf8cad6e175a3ae24f4084bf3823f8d0f0... 客户ID: M412\N客户公司: 创联世纪传媒有限公司智能科技 V客户所在地: 文市, ... [601f1b35ea7eb113bcdad42115082146a5953dd2e31e5... 295
7 b326f9f5e21f6b34a49d9be5b88835215984b7255a63... 客户ID: QM911\N客户公司: 戴硕电子科技有限公司生物科技 V客户所在地: 武汉市, ... [601f1b35ea7eb113bcdad42115082146a5953dd2e31e5... 190
8 614797b771eb7076c4f4f55d9f7a3bea202e5d5edc6... 客户ID: M412\N客户公司: 创联世纪传媒有限公司智能科技 V客户所在地: 文市, ... [601f1b35ea7eb113bcdad42115082146a5953dd2e31e5... 158
9 6c507212429c364662a4e234659c6fada9514b5bf9a16... 客户ID: YE681\N客户公司: 网新恒天网络有限公司虚拟现实 V客户所在地: 峰县, ... [601f1b35ea7eb113bcdad42115082146a5953dd2e31e5... 152
10 095125b6da130e090ea21af2b33897fc32d40cad0a7b... 客户ID: H4078\N客户公司: 天富传媒有限公司大数据分析 V客户所在地: 明市, 中... [687b3ab4b3c4e6c982a5b73a9740a4a6029be18fc39c9... 162
11 f2e43dc84d7ba58483f4a657647d1e9f49a5f127846... 客户ID: ZC483\N客户公司: 和泰传媒有限公司物联网 V客户所在地: 海城市, 中国... [687b3ab4b3c4e6c982a5b73a9740a4a6029be18fc39c9... 184
12 ca68a397e7d9f9c0c6207ac5b7852bcb3a0bd934eb35ed... 客户ID: TA647\N客户公司: 和泰网络有限公司区块链 V客户所在地: 晨县, 中... [687b3ab4b3c4e6c982a5b73a9740a4a6029be18fc39c9... 158
13 d4fca3088accd8084f4f3971b7209992d0c0700f08... 客户ID: YE681\N客户公司: 网新恒天网络有限公司虚拟现实 V客户所在地: 峰县, ... [7268c7f82da938560038c6aac7d47ea58425dc91339c8... 151
14 e024e41b07cf3c11434aaf0769fd1216b7ec54626497a3... 客户ID: QM911\N客户公司: 戴硕电子科技有限公司生物科技 V客户所在地: 武汉市, ... [7268c7f82da938560038c6aac7d47ea58425dc91339c8... 297
15 d9775e29a9a02fc71fe4f67149f679ef7d130f4734... 客户ID: YE681\N客户公司: 网新恒天网络有限公司虚拟现实 V客户所在地: 峰县, ... [7268c7f82da938560038c6aac7d47ea58425dc91339c8... 156
16 d82e18e583d9a945d4d21a06f9069b6dbad1f44e585937... 客户ID: QM911\N客户公司: 戴硕电子科技有限公司生物科技 V客户所在地: 武汉市, ... [7268c7f82da938560038c6aac7d47ea58425dc91339c8... 154
17 55bd5eabf99ddedda79c3c9b5e5530f2eaf29f32d2... 客户ID: OL973\N客户公司: 继进明传媒有限公司数字科技 V客户所在地: 军县, 中... [7268c7f82da938560038c6aac7d47ea58425dc91339c8... 216
18 f2eabf907e03c11a04c7f8f726aeafda9eaf8f609ce4... 客户ID: H4078\N客户公司: 天富创业科技有限公司虚拟现实 V客户所在地: 锦市, ... [7c7b7f1c7e56ab2806dfc6b93f87a95a2506e43109a09a... 189
```

全部执行结束后，依然会正常在 `./data/output` 文件夹中生成对应的 `.parquet` 文件，可以直接用于 Microsoft GraphRAG 的检索过程，同时，如果大家想存储到 Neo4j 数据库中进行可视化展示，也可以按照我们在《Microsoft GraphRAG 深度实战 - Part 3. Microsoft GraphRAG 自定义接入图数据库 Neo4j》中的介绍进行持久化的存储。

### 3. 结构化数据直接导入 Neo4j 的方法

上述的流程我们操作的是非结构化数据，构建的方式是通过提示工程+大模型借助 Microsoft GraphRAG 的 workflow 来构建的。如果大家想操作结构化数据，则需要结合实际的业务场景，将数据集拆分为节点和边，并按照节点类型和边类型分类，符合 Neo4j 等图数据库的标准导入方式。

这个过程其实没什么技巧可谈，一切以业务需求为驱动，考验的就是大家的业务理解能力和数据处理能力，以及知识图谱的构建能力。比如，我们以 电商智能客服 为例，可以先分析一下智能客服场景的需求，主要的点包括：

- 1. 需要快速检索产品信息
- 2. 需要了解客户的基本信息和订单历史
- 3. 需要了解产品与供应商、类别的关系
- 4. 需要追踪订单状态和物流信息

基于以上需求，我们就可以开始着手划分 Neo4j 的节点和边。其中：

- 节点类型
  - Product - 产品节点
  - Category - 产品类别节点
  - Supplier - 供应商节点
  - Customer - 客户节点
  - Employee - 员工节点
  - Shipper - 物流公司节点
  - Order - 订单节点
- 边类型
  - BELONGS\_TO - 产品属于类别
  - SUPPLIED\_BY - 产品由供应商提供
  - PLACED\_BY - 订单由客户下单
  - PROCESSED\_BY - 订单由员工处理
  - SHIPPED\_VIA - 订单通过物流公司配送
  - CONTAINS - 订单包含产品
  - REPORTS\_TO - 员工上下级关系

构建导入的方案细节，如下图所示：

### 节点类型

节点类型	标识	属性
Product	ProductID	ProductName, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued
Category	CategoryID	CategoryName, Description
Supplier	SupplierID	CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax, HomePage
Customer	CustomerID	CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax
Employee	EmployeeID	LastName, FirstName, Title, BirthDate, HireDate, Address, City, Country, HomePhone, Notes
Shipper	ShipperID	CompanyName, Phone
Order	OrderID	OrderDate, RequiredDate, ShippedDate, Freight, ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, ShipCountry

## 关系类型

边类型	源节点	目标节点	数据源
BELONGS_TO	Product	Category	products.csv (ProductID -> CategoryID)
SUPPLIED_BY	Product	Supplier	products.csv (ProductID -> SupplierID)
PLACED_BY	Customer	Order	orders.csv (CustomerID -> OrderID)
PROCESSED_BY	Employee	Order	orders.csv (EmployeeID -> OrderID)
SHIPPED_VIA	Order	Shipper	orders.csv (OrderID -> ShipVia)
CONTAINS	Order	Product	order_details.csv (OrderID -> ProductID)
REPORTS_TO	Employee	Employee	employees.csv (EmployeeID -> ReportsTo)

定义好构建的规范后，便可以开始实际的构建知识图谱。对于单独的 `.csv` 文件，或者从数据库直接导出的 `.csv` 文件，一种最高效处理大批量数据的方法是使用 Neo4j 的 Neo4j Admin 工具。Neo4j Admin 是 Neo4j 数据库管理系统中的一个命令行工具，主要用于管理和维护 Neo4j 数据库实例。它提供了一系列命令，帮助执行各种管理任务，如数据库的创建、备份、恢复、配置和监控等。

如果采用的是我们在《Microsoft GraphRAG 深度实战 - Part 3. Microsoft GraphRAG 自定义接入图数据库 Neo4j》中的介绍的源码安装方式，则 Neo4j Admin 工具位于 `neo4j-admin` 文件夹中。

盘 (D:) > neo4j-community-2025.02.0-windows > bin >				
排序 查看 ...				
名称	修改日期	类型	大小	
neo4j_admin	2025-03-19 15:34	文件夹		
Neo4j-Management	2025-03-19 15:29	文件夹		
tools	2025-03-19 15:29	文件夹		
cypher-shell	2025-02-24 19:26	Windows 批处理...	4 KB	
import_command	2025-03-19 15:42	Windows 批处理...	3 KB	
neo4j	2025-02-24 19:21	Windows 批处理...	1 KB	
neo4j	2025-02-25 20:26	Windows Power...	15 KB	
neo4j-admin	2025-02-24 19:21	Windows 批处理...	1 KB	
neo4j-admin	2025-02-25 20:26	Windows Power...	15 KB	
Neo4j-Management	2025-02-25 20:27	Windows Power...	16 KB	

但是需要明确的是，Neo4j Admin 可以直接导入 .csv 文件，但如何去处理原始文件中的节点和关系，则需要我们通过注释这一概念来进行声明。

Neo4j 中，注释（或称为注释行）是用于在 CSV 文件中提供元数据的行。这些注释行以 # 开头，允许开发者为每个字段指定类型、约束和其他信息，以便在导入数据时，Neo4j 能够正确解析和处理这些数据。其中 CSV 注释的用途包括：

1. 字段类型: 指定字段的数据类型（如字符串、整数、布尔值等），确保 Neo4j 在导入时能够正确识别字段类型。
2. 约束: 指定唯一性约束、索引等，以确保数据的完整性和一致性。
3. 关系类型: 指定边的类型，确保在导入时正确建立节点之间的关系。

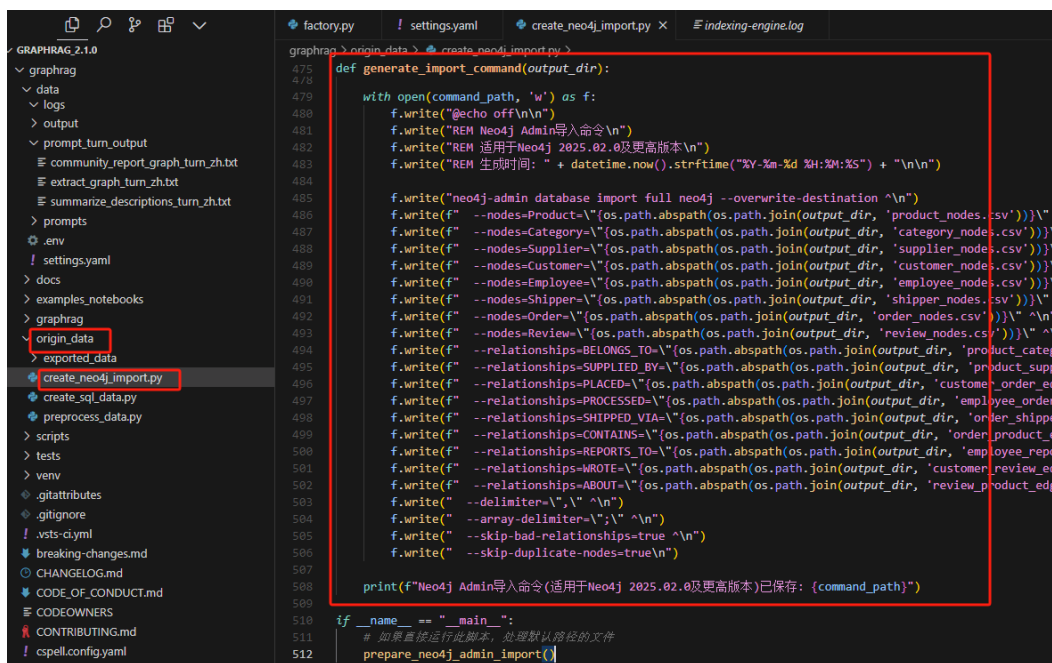
对应的注释的格式是需要在 csv 文件的开头添加注释行，格式如下：

```
# Node: Product
# Field: ProductID, Type: Integer, Unique: true
# Field: ProductName, Type: String
# Field: UnitPrice, Type: Float
# Field: UnitsInStock, Type: Integer
# Field: CategoryID, Type: Integer, Relationship: BELONGS_TO
```

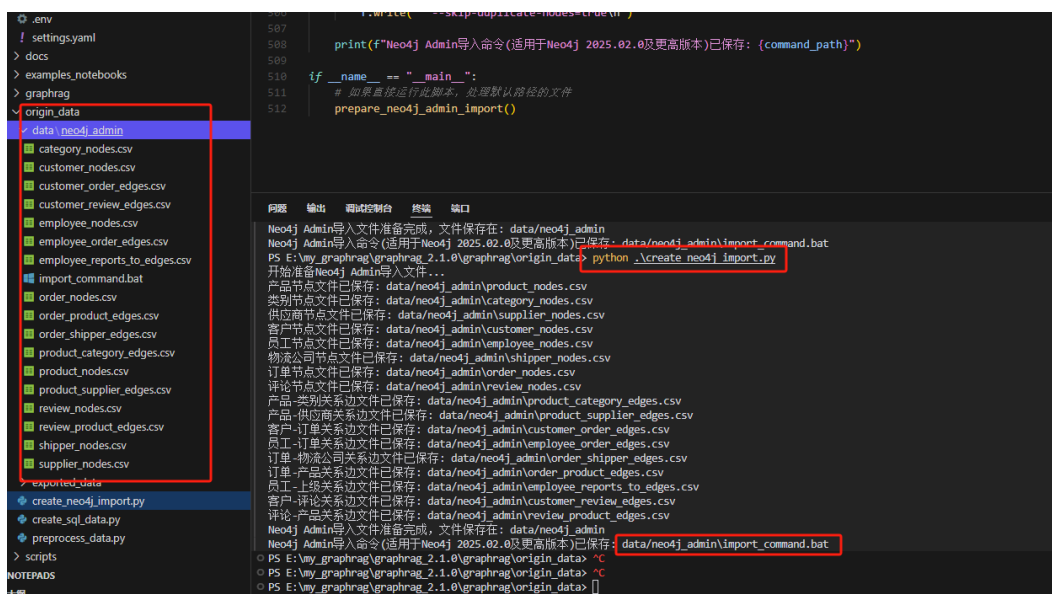
```
1,智能音箱,299.99,100,1
2,智能灯具,199.99,150,2
3,智能插座,89.99,200,1
```

这里我已经把 Neo4j 的结构定义以及对应的注释行都写好了，全部流程封装在 graphrag\origin\_data\create\_neo4j\_import.py 文件中，大家可以直接运行该文件来生成 Neo4j 的导入文件。

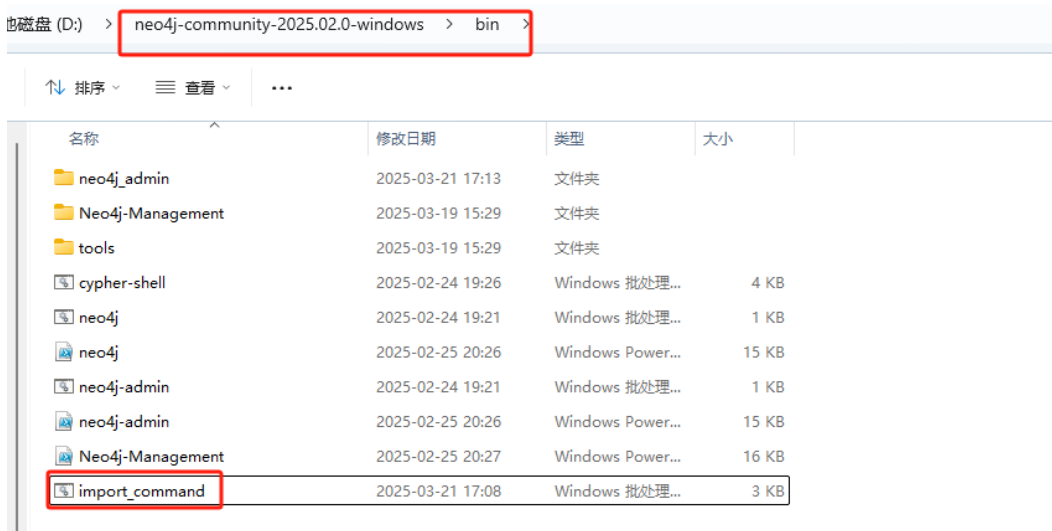




运行方法如下图所示:



运行结束后, 会在 graphrag\origin\_data\neo4j\_admin 文件夹中生成对应的 Neo4j 的导入文件即导入脚本。然后我们需要把 import\_command.bat 文件放在 Neo4j 安装目录下的 bin 文件夹中,如下所示:



然后打开 cmd 命令行, 进入到 Neo4j 安装目录下的 bin 文件夹中, 运行 import\_command.bat 文件, 即可开始导入数据。导入过程如下图所示: 注意: 执行导入之前需要先停止 Neo4j 服务, 否则会报错。



```
D:\neo4j-community-2025.02.0-windows\bin>import_command.bat
Starting to import, output will be saved to: D:\neo4j-community-2025.02.0-windows\logs\neo4j-admin-import-2025-03-21.17
18.00.log
Neo4j version: 2025.02.0
Importing the contents of these files into D:\neo4j-community-2025.02.0-windows\data\databases\neo4j:
Nodes:
[Order]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\order_nodes.csv

[Employee]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\employee_nodes.csv

[Category]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\category_nodes.csv

[Customer]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\customer_nodes.csv

[Shipper]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\shipper_nodes.csv

[Product]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\product_nodes.csv

[Supplier]:
E:\my_graphrag\graphrag_2.1.0\graphrag\origin_data\data\neo4j_admin\supplier_nodes.csv
```

等待执行完成后，再次启动 Neo4j 服务，就可以在 Neo4j 中看到我们构建的知识图谱了。如下图所示：

