RAG Pipeline API 接口文档

目录

---

1. 简介

本API提供了三种对话接口：

- 1. 基于向量检索的RAG对话系统
- 2. 基于知识图谱的RAG对话系统
- 3. 智能路由的Agent系统

所有接口都支持流式输出和普通输出两种模式。

接口采用fastapi的方式来进行， 代码见 git api.py

- https://git.imooc.com/coding-920/RAG_erag/src/master/api.py

---

2.fastapi介绍

2.1 FastAPI 简介与特点

FastAPI 是近年来 Python Web 开发领域最受关注的框架之一，它具有以下显著特点：

- **极高性能**：可与 NodeJS 和 Go 比肩，是最快的 Python web 框架之一
- **高效编码**：提高功能开发速度约 200%至 300%
- **自动文档**：内置 Swagger UI 和 ReDoc 文档生成
- **类型安全**：基于 Python 类型提示，减少约 40%的人为错误
- **异步支持**：原生支持 async/await 语法
- **数据验证**：通过 Pydantic 提供强大的数据验证功能

安装 FastAPI 非常简单，只需运行以下命令：

```
pip install fastapi uvicorn
```

其中 Uvicorn 是一个支持 ASGI 的轻量级高性能 Web 服务器，是部署 FastAPI 应用的推荐选择。

2.2 基础功能与代码示例

2.2.1 创建基本 API 端点

FastAPI 最基本的功能是创建 API 端点。以下是一个简单的 “Hello World” 示例：

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def read_root():
    return {"message": "Hello World"}

@app.get("/items/{item_id}")
async def read_item(item_id: int, q: str = None):
```

```
    return {"item_id": item_id, "q": q}
```

代码说明：

- `app = FastAPI()` 创建 FastAPI 实例
- `@app.get("/")` 定义 GET 方法的路由
- 路径参数 `item_id` 会自动转换为指定的类型（这里是 int）
- 查询参数 `q` 是可选的，默认值为 None

启动服务：

```
uvicorn main:app --reload
```

访问 `http://127.0.0.1:8000/docs` 可以看到自动生成的交互式 API 文档。

2.2.2 请求体与 Pydantic 模型

FastAPI 使用 Pydantic 模型来定义请求体的数据结构：

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Item(BaseModel):
    name: str
    price: float
    is_offer: bool = None
```

```python
@app.post("/items/{item_id}")
async def update_item(item_id: int, item: Item):
    return {"item_name": item.name, "item_id": item_id}
```

代码说明：

- `Item` 类继承自 `BaseModel`，定义了请求体的结构
- 字段类型提示确保了输入数据的类型安全
- `is_offer` 是可选字段，默认值为 None
- FastAPI 会自动验证请求体是否符合模型定义

2.2.3 文件响应与下载

FastAPI 可以方便地实现文件下载功能：

```python
import os
from fastapi import FastAPI
from starlette.responses import FileResponse

app = FastAPI()

@app.get('/download/{filename}')
async def get_file(filename: str):
    path = os.path.join(os.getcwd(), filename)
    if not os.path.exists(path):
        return {"success": False, "msg": "文件不存在！"}
    return FileResponse(path)
```

代码说明：

- `FileResponse` 用于返回文件内容
- 安全性考虑：只允许下载已知文件名的文件
- 可以进一步扩展为需要认证的文件下载服务

2.2.4 RAG接口实例（详见api.py）

```
# 对应调用的http url:  http://localhost:8000/rag_pipeline
@app.post("/rag_pipeline",
          summary="基于向量检索的RAG对话接口",
          response_description="返回AI助手的回答和相关上下文")
def rag_pipeline(request: RagPipelineRequest):
    try:
        # 在直接调用rag_pipline
        response, context = run_rag_pipline(
            query=request.query,
            context_query=request.query, # 使用query作为上下文查询
            k=request.k,
            context_query_type=request.context_query_type,
            stream=request.stream,
            temperature=request.temperature
        )

        # 涉及流式输出
        if request.stream:
            # 使用FastAPI的StreamingResponse创建流式响应
            # StreamingResponse允许逐步发送数据，而不是等待所有数据准备好后一次性返回
```

下一节：RAG Pipeline API 接口文档-【环境准备&API接口说明】

下一节