论文搜索助手

【基础要求】根据用户提问，系统从论文（PDF文档）库中搜索论文，并总结回答用户问题 【主要界面】主页含对话框（右上），聊天页含对话框、回答和论文列表（右下），点击论文查看论文详情

要求

除本项目提供的模型API外，不得调用其他外部API 系统符合分层架构设计要求 界面使用Vue，业务层和算法层使用FastAPI
三层架构：

- 界面：`frontend`
- 业务层：`backend`
- 算法层：`backend_algo`

部署和使用方法

配置python环境

建立环境，命名为fastapi，并安装依赖：

```
conda create -n fastapi python=3.12
conda activate fastapi
cd backend
pip install -r requirements.txt
cd ..
cd backend_algo
pip install -r requirements.txt
```

安装前端依赖

先下载安装node.js，再执行下面命令：

```
cd frontend
npm install
```

配置MySQL数据库

MySQL下载并安装：https://dev.mysql.com/downloads/installer/
打开MySQL Command Line Client
输入安装时设置的 root 用户密码
创建数据库：test

```
CREATE DATABASE test;
```

验证数据库是否创建成功

```
SHOW DATABASES;
```

如果看到 test 出现在列表中，表示创建成功。
安装依赖(fastapi环境)：

```
pip install PyMySQL[rsa]
```

修改backend/database.py:

```
SQLALCHEMY_DATABASE_URL = "mysql+pymysql://root:你的数据库密码@localhost:3306/test"
```

更新数据库

先启动一次项目，进行相应初始化，不要关闭：

```
chmod +x start_dev.sh
./start_dev.sh
```

爬取论文：

```
python backend_algo/arxiv_crawler.py
```

处理论文向量并存储到ChromaDB：

```
python backend_algo/batch_embed.py
```

这时再关闭项目

启动命令

在git bash运行

```

```

```
chmod +x start_dev.sh
./start_dev.sh
```

在前端启动好之后（跳出前端链接），即可使用。

打开后，注册并登录，进入“聊天”页面，在对话框输入问题（最好和NLP, AI, ML相关，因为目前只爬取了这些领域的论文），搜索后等待片刻，即可得到大模型文字回答和右方相关论文列表，点击论文即可查看详情，并且能得到其它论文推荐（根据用户最近的论文查看行为推荐）。

代码结构

系统架构

采用三层架构设计：

1. **前端层(frontend)**: Vue 3 + TypeScript实现用户界面
2. **业务层(backend)**: FastAPI实现业务逻辑和API接口
3. **算法层(backend_algo)**: 论文爬取、向量化和推荐算法

目录结构

前端层(frontend/src)

- `components/`: Vue组件
  - `Chat.vue`: 聊天和论文搜索界面
  - `PaperDetail.vue`: 论文详情展示
  - `LoginForm.vue`: 登录表单
  - `RegisterForm.vue`: 注册表单
  - `HighlightText.vue`: 处理模型回答的高亮显示和论文匹配交互
- `pages/`: 页面路由组件
  - `Index.vue`: 主界面
  - `Login.vue`: 登录页
  - `Register.vue`: 注册页
- `router/`: 路由配置
- `request/`: API请求封装
  - `api.ts`: 接口定义
  - `http.ts`: HTTP请求封装
- `store/`: 状态管理(Pinia)
  - `user.ts`: 用户状态管理

业务层(backend)

- `models.py`: 数据模型定义(User, Paper等)
- `main.py`: FastAPI应用和路由
- `database.py`: 数据库连接配置
- `crud.py`: 数据库操作
- `schemas.py`: Pydantic模型

- `security.py`: 认证和安全

主要API接口:

- 用户认证: `/api/token`
- 论文搜索: `/api/papers/search`
- 论文详情: `/api/papers/{paper_id}`
- 论文推荐: `/api/recommendations/{paper_id}`
- 用户行为记录: `/api/papers/{paper_id}/interact`

算法层(backend_algo)

- `arxiv_crawler.py`: arXiv论文爬取
  - 从arXiv API获取论文元数据
  - 支持NLP/AI/ML领域论文
  - 保存到SQL数据库
- `batch_embed.py`: 论文向量化处理
  - 使用bge-m3模型生成嵌入向量
  - 存储到ChromaDB向量数据库
- `test_*.py`: 各模块测试

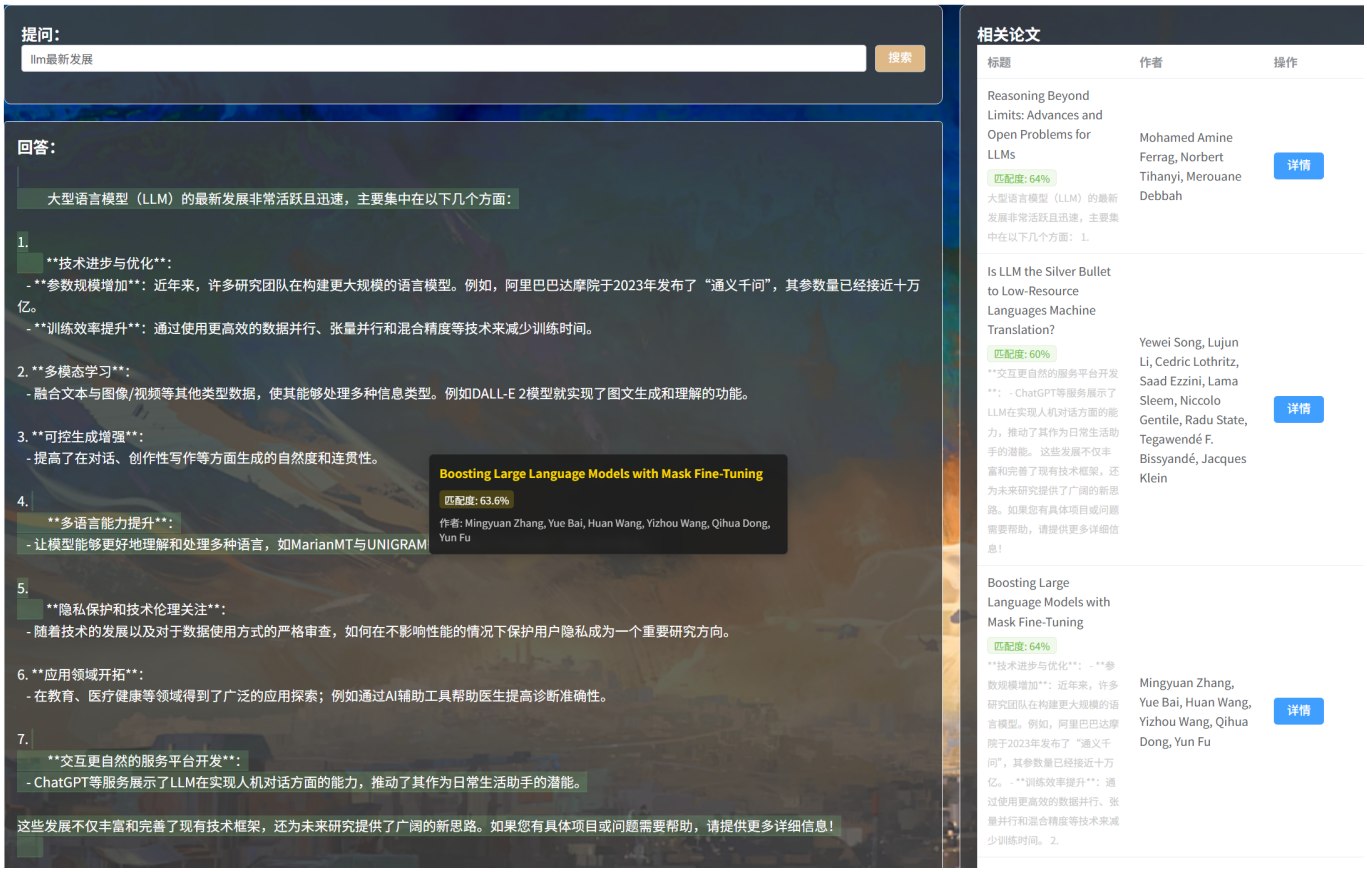
数据流

1. 用户在前端输入问题
2. 前端调用`/api/chat`接口
3. 后端处理流程:
   - 回答生成阶段:
     - 调用本地LLM生成回答
   - 论文搜索阶段:
     - 将模型回答向量化
     - 从ChromaDB搜索相似论文
   - 匹配分析阶段:
     - 分析回答文本，标记与论文相关的部分
     - 为每个匹配生成: · 匹配的论文ID · 匹配分数(0-1) · 匹配的文本片段
4. 返回数据结构包含:
   - LLM生成的回答文本
   - 匹配信息数组
   - 相关论文列表
5. 前端处理:
   - 使用HighlightText组件渲染回答文本
   - 根据匹配分数动态高亮相关文本
   - 鼠标悬停时显示匹配论文信息
6. 用户交互:
   - 点击论文时记录行为到数据库
   - 基于最近查看的论文生成推荐
   - 高亮文本可点击，增强交互体验

效果演示

搜索结果

包含文字回答、相关论文列表、回答与论文匹配的高亮，支持鼠标悬浮在高亮时显示论文标题。



论文详情

显示论文作者、日期、摘要等信息，能预览pdf原文，并且在最下方有根据用户喜好生成的推荐（同样支持点开查看详情）。

Reasoning Beyond Limits: Advances and Open Problems for LLMs

作者: Mohamed Amine Ferrag, Norbert Tihanyi, Merouane Debbah

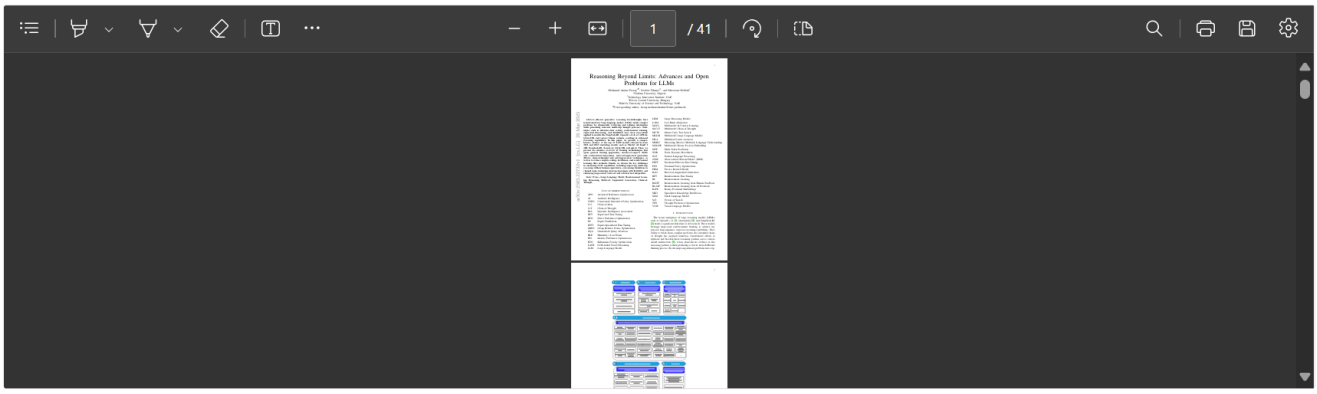
发表日期: 2025-03-26T12:29:40

关键词: cs.LG, cs.CL

摘要

Recent generative reasoning breakthroughs have transformed how large language models (LLMs) tackle complex problems by dynamically retrieving and refining information while generating coherent, multi-step thought processes. Techniques such as inference-time scaling, reinforcement learning, supervised fine-tuning, and distillation have been successfully applied to models like DeepSeek-R1, OpenAI's o1 & o3, GPT-4o, Qwen-32B, and various Llama variants, resulting in enhanced reasoning capabilities. In this paper, we provide a comprehensive analysis of the top 27 LLM models released between 2023 and 2025 (including models such as Mistral AI Small 3 24B, DeepSeek-R1, Search-o1, QwQ-32B, and phi-4). Then, we present an extensive overview of training methodologies that spans general training approaches, mixture-of-experts (MoE) and architectural innovations, retrieval-augmented generation (RAG), chain-of-thought and self-improvement techniques, as well as test-time compute scaling, distillation, and reinforcement learning (RL) methods. Finally, we discuss the key challenges in advancing LLM capabilities, including improving multi-step reasoning without human supervision, overcoming limitations in chained tasks, balancing structured prompts with flexibility, and enhancing long-context retrieval and external tool integration.

PDF全文



相关推荐

debug-gym: A Text-Based Environment for Interactive Debugging

Xingdi Yuan, Morgane M Moss, Charbel El Feghali, Chinmay Singh, Darya Moldavskaya, Drew MacPhee, Lucas Caccia, Matheus Pereira, Minseon Kim, Alessandro Sordoni, Marc-Alexandre Côté

关闭