论文写作助手算法方案文档

姓名: [周弈成]
学号: [23307130064]
专业: [计算机科学与技术]
学院: [计算与智能创新学院]

2025 年 6 月 8 日

目录

1 算法后端实现方法

1.1 总述

算法后端分为三部分：加载参考论文到向量数据库；根据段落类型和关键词描述，在向量数据库中查找相关文本并写作综述段落；整合生成的段落，生成 LaTeX 和 pdf 格式的全文。

1.2 加载参考论文

本系统通过 `pdf_to_vectordb.py` 脚本实现 PDF 文档的智能处理和向量化存储，主要包含以下核心功能：

1.2.1 PDF 文本和公式提取

系统采用 PyMuPDF（fitz）库对 PDF 文档进行解析，能够同时处理文本内容和数学公式：

- **文本提取**：直接提取 PDF 中的文本块内容
- **公式识别**：通过 Mathpix OCR API 将公式图像转换为 LaTeX 格式，以 `[公式]{latex_code}` 的形式存储
- **内容整合**：将文本和公式内容合并为完整的文档表示

1.2.2 文本分块处理

为了优化检索效果和向量化质量，系统对提取的文本进行智能分块：

- **块大小限制**：每个文本块限制最大字符数量
- **重叠设计**：相邻文本块有重叠字符，保持上下文连续性
- **分割策略**：使用 RecursiveCharacterTextSplitter 实现递归文本分割

1.2.3 向量化和存储

系统使用先进的 embedding 技术将文本转换为向量表示：

- **Embedding 模型**：采用 BGE-M3 模型进行文本向量化，支持多语言和长文本处理
- **语义表示**：向量能够捕捉文本的深层语义信息，相似内容在向量空间中距离较近
- **向量数据库**：使用 ChromaDB 存储向量化后的文档，支持高效的相似度检索

该实现确保了学术论文中的文本内容和数学公式都能被准确识别、向量化并用于后续的语义检索任务。

1.3 段落写作

系统根据规则提示词，进行段落写作：

- **字数限制**：限制段落字数，使段落内容精简
- **向量数据库**：使用 ChromaDB 存储向量化后的文档，支持高效的相似度检索

2 算法后端优化方法

2.1 总述

0.5 实验实现算法后端后，1.0 实验中在多个部分进行优化：二次请求；优化向量数据库分块参数；应用重排序。

2.2 二次请求

在生成文本后，再次把模型生成的文本和原本的规则提示词发送至模型，重新生成文本。

2.2.1 优化向量数据库分块参数

系统对提取的文本进行智能分块时的块大小参数和重叠参数需要调整。实验中，对比“块大小 1000 字，重叠 200 字”和“块大小 200 字，重叠 50 字”两种参数。

2.2.2 应用重排序

通过向量数据库检索到的文本可能相关性不足。我们通过重排序，优化检索文本的精确度。为此，需要提升检索文本数量再进行重排序取前几个。

3 实验

3.1 实验环境

硬件：Macbook Pro，Apple M4 芯片，16GB 内存。

- **操作系统**： [MacOS]
- **编程语言**： [python]

3.2 数据集

论文库数据集：收集 5 种计算机领域的热门论文（50 篇）及无关错误论文（2 篇）。按照领域分类。

问答数据集：包含 50 篇各计算机领域论文大纲问答和 2 篇无关问答。数据集的问题字段和参考数据集的答案字段由 Claude 4 Sonnet 生成，参考文献字段按照论文库领域分类给出。

3.3 实验设计

评估标准：BLEU 和 Rouge 作为答案字段的文本翻译指标；Precision@K, Recall@K, F1@K 为参考文献指标（K 取 1、3、5、8，最后用 K=3）。此外，额外设计 Top-K 包含率（Coverage）指标：若一个问答的前 K 个参考文献全包含在参考文献中，则为 1，否则为 0。统计多个 query 下的平均 InclusionRate 来衡量整体性能。

先使用问题直接连接向量数据库后根据提示词直接接入大模型润色的答案作为对照，再把问题接入算法层后端，依次测试无优化、二次请求优化及向量化参数与重排序优化三个优化方法。

3.4 实验过程

1. 使用 start.sh 启动前后端，加载参考论文到向量数据库
2. 使用 generate_qa_content.py 脚本，调用模型，根据问答数据集问题生成文本和参考文献
3. 使用 evaluate_translation.py 脚本，评估生成文本的翻译质量
4. 使用 evaluate_r_script.py 脚本，评估引用文献的匹配质量

3.5 实验结果

表 1: 文本对比实验结果

算法	BLEU-1	BLEU-4	ROUGE-1 F1	ROUGE-L F1
对照	0.2137	0.0180	0.0423	0.0384
无优化	0.2713	0.0233	0.0688	0.0677
二次请求优化	0.2742	0.0227	0.0562	0.0529
向量化参数与重排序优化	0.2828	0.0260	0.0630	0.0619

表 2: 引用对比实验结果

算法	Precision@3	Recall@3	F1@3	InclusionRate@3
对照	0.6410	0.2289	0.3213	0.3654
无优化	0.6090	0.2205	0.3082	0.3269
二次请求优化	0.6026	0.2188	0.3055	0.3269
向量化参数与重排序优化	0.5705	0.1983	0.2821	0.2115

3.6 结果分析

我们可以发现，算法后端的三个优化对文本生成质量的优化启正面作用，但是对引用的匹配度存在负面作用。有可能是因为引用文献的参考值设置过于死板遵循分类，而忽略每篇题材论文的独特性，有优化的空间。

4 结论

本实验中，我们通过优化算法后端的三个部分，提高了文本生成质量，但是对引用匹配起到了反效果。这启示我们在设计算法后端时，需要考虑到每篇论文的独特性，而不是简单地遵循分类。以及，数据库的设计还有进一步的优化空间。