开发内容

1. 通用处理
    - 通用PDF文件内容解析
    - PDF表格内容提取
2. 招标文件详审部分处理
    - 详审因素位置定位
    - 详审内容等级评定
    - 并行计算处理

项目进度

已完成节点

1. 招标文件PDF的解析
2. 投标文件TextMind解析
3. 解析结果统一转换
4. 商务部分详评
5. 技术部分详评
6. 大模型等级评价生成

进行中节点

1. 商务部分条目范围界定优化
2. 技术部分条目范围界定优化

未开始节点

1. 模块合并调用

环境配置

1. 使用 `poetry==1.8.3` 作为依赖管理和打包工具
2. 运行环境 `Python==3.10.11`
3. 大模型环境 `ollama`
4. 模型使用 `Qwen2.5:7b`
5. 接口使用 `FastAPI`
6. 任务使用 `Celery`
7. 向量化模型使用 `GanymedeNil/text2vec-base-chinese`
8. 后台任务分配使用 `Redis` 中间件
9. 结果保存使用 `Sqlite3 ORM` 数据(开发环境)，正式环境需切换 `MongoDB` 集群
10. 中间过程序列化使用 `json` 数据格式
11. 向量数据库使用 `ChromaDB`

相关功能

通用PDF解析

实现目标

1. 对招标文件进行内容提取
2. 对投标文件进行内容提取

实现方式

1. 使用Python包处理招标文件，使用PDFMiner工具对PDF内容样式进行解析，使用camelot+ghostscript处理PDF表格数据，使用OpenCV+Pillow处理图片数据
2. 使用TextMind处理投标文件，对TextMind的解析结果格式调整为通用格式

实现效果

1. 解析内容包含标题、大纲、图片、表格、正文部分。
2. 每个部分包含页码和文本内容。

内容定位

实现目标

1. 对评分表中的商务部分和技术部分的每个评分因素，找到投标文件对应的部分

实现方式

1. 在招标文件的表格部分中，找到评分因素表
2. 对投标文件的标题、大纲部分，使用标题等级判定，列出标题对应的等级和上下文关系
3. 对每个评分因素使用向量余弦相似度匹配的方式查找投标文件中对应的标题，将其对应的页码作为起始页，将下一个同级标题所在的页码作为终止页
4. 起始页和终止页间的所有正文、表格和图片作为该评分项下的原始内容数据

实现效果

1. 每个评分因素对应投标文件中的部分相关内容

等级评定、内容概要

实现目标

1. 对招标文件中的评审规则，对投标文件中内容判定分数或等级，并输出其概要

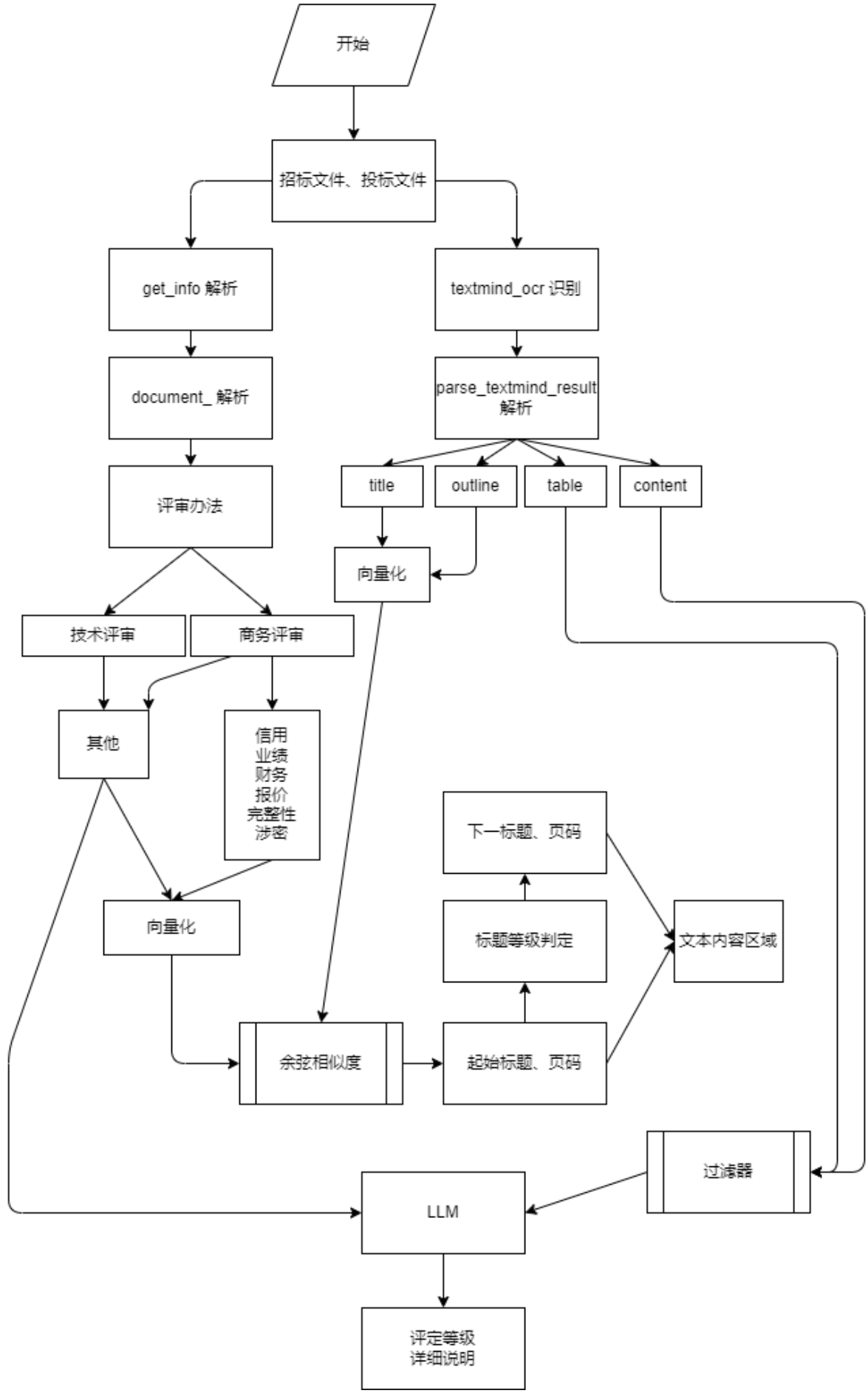
实现方式

1. 商务部分中对常用评价因素(信用、业绩、财务、报价、完整性、涉密)使用独立处理方法，其他使用通用处理方法
2. 技术部分使用通用处理方法
3. 对于固定的评审规则，使用预设的计算方法
4. 对于可变的评审规则，将规则转化为大模型的提示词，并将投标文件中的文本数据转化为大模型的系统数据，进行结果预测
5. 对预测的结果限制输出，将其转化为等级和评价两个部分

版本控制

使用 Git 版本控制
代码 Git 地址: http://192.168.1.202:8087/xzc/pdf_title_image.git

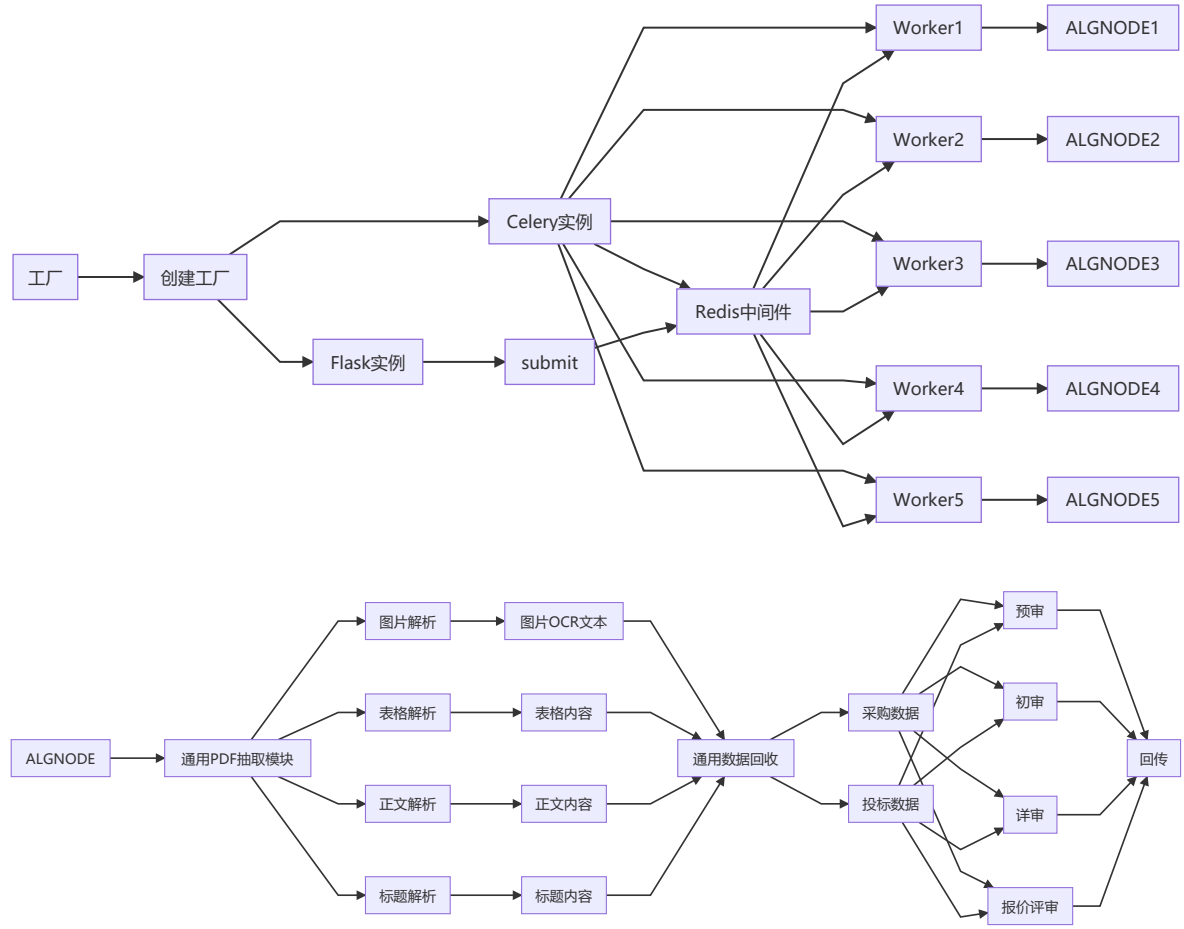
流程图


开始
招标文件、投标文件
get_info 解析
textmind_ocr 识别
document_ 解析
parse_textmind_result
解析
评审办法
title
outline
table
content
向量化
技术评审
商务评审
其他
信用
业绩
财务
报价
完整性
涉密
下一标题、页码
向量化
标题等级判定
文本内容区域
余弦相似度
起始标题、页码
过滤器
LLM
评定等级
详细说明

服务架构

image-20240826152353199

流程结构



代码层

异步配置文件(backend/config.py)

```
class Config:
    CELERY = dict(
        CELERY_BROKER_URL = 'redis://localhost:6379/0',
        CELERY_RESULT_BACKEND = 'redis://localhost:6379/0',
        include = "backend.celery_task",
        task_ignore_result = True,
        timezone = 'Asia/Shanghai',
        enable_utc = False,
        task_track_started = True
    )
```

创建工厂(backend/init.py)

```
from flask import Flask
from celery import Celery, Task

celery_app = Celery(__name__)

def create_app(test_config: dict = None) -> Flask:
    app = Flask(__name__)
```

```
    class FlaskTask(Task):
        def __call__(self, *args: object, **kwargs: object) -> object:
            with app.app_context():
                return self.run(*args, **kwargs)

    if test_config is None:
        app.config.from_pyfile('config.py', silent=True)
    else:
        app.config.from_mapping(test_config)

    celery_app.config_from_object(app.config['CELERY'])
    celery_app.Task = FlaskTask
    celery_app.set_default()

    return app
```

封包脚本(make_celery.py)

```
from backend import (
    create_app,
    celery_app,
)

app = create_app()
```

WEB配置 (gunicorn_config.py)

```
workers = 1
bind = "0.0.0.0:8000"
backlog = 2048
loglevel = "INFO"
daemon = True
pidfile = "/var/run/backend.pid"
accesslog = "/var/log/backend/access.log"
errorlog = "/var/log/backend/error.log"
```

执行脚本(run.sh)

```
gunicorn --config gunicorn_config.py make_celery
celery multi start worker -A make_celery:celery_app -P prefork -E --
loglevel=INFO --logfile=/var/log/celery/%n%I.log --pidfile=/run/celery/%n.pid
celery -A make_celery:celery_app events
```

主要模块描述

1. get_info　　PDF信息抽取模块
2. tools.TitleLevelJudge 标题等级和下一标题识别功能
3. tools.filter_*　　根据页码筛选内容
4. tools.match_*　　正则匹配
5. matcher　　向量相似度匹配

6. busi_instance　　商务部分格式化输出
7. tech_instance　　技术部分格式化输出
8. extract_financial_report 财报抽取
9. instance_locate　　根据标题判定内容位置
10. LLMAgent　　大模型调用模块
11. document_　　招标文件解析模块
12. parse_textmind_result　textmind内容格式化
13. project_loc　　项目业绩的表格定位模块
14. text_extractor　　文本内容相似度匹配
15. responser　　输出数据类定义

PDF中无边框表格内容抽取

```
1. camelot-py git源下载
    git clone https://www.github.com/camelot-dev/camelot
    修改pyproject.toml中 pdfminer-six = "^20231228"
    安装命令： 进入camelot目录下，pip install -e .
2. 在wsl Debian中安装 ghostscript 【模块本身】
    apt install ghostscript
3. ghostscript 下载
    pip install ghostscript==0.7.0 【模块驱动】
4. 代码修改 【CV运行时不需要设置宽高，使用默认即可】
    tables_pro = camelot.read_pdf(
                    self.file_path,
                    # flavor='stream',
                    pages=str(page_number+1),
                    # edge_tol=200,
                    # row_tol=50,
                )
```