

智能垃圾分类系统详细设计

项目组：lajifenlei

版本：V1.0

发布日期：2025 年 11 月 19 日

评审日期: 2025 年 11 月 19 日
项目编号: LAJI-2025-001
文档编号: DD-2025-11-01
密级: 内部
编制: 北京长江软件 (示例)

目录

第一章 导言	4
1.1 目的	4
1.2 范围	4
1.3 缩写说明	4
1.4 术语定义	4
1.5 引用标准	4
1.6 参考资料	5
1.7 版本更新信息	5
第二章 系统设计概述	6
第三章 详细设计概述	7
3.1 总体结构	7
3.2 运行流程	7
第四章 文本与批量分类模块详细设计	9
4.1 视图层	9
4.2 控制层	9
4.2.1 接口定义	9
4.2.2 异常处理	10
4.2.3 批量接口实现	10
4.3 模型层	11
4.3.1 数据管理器	11
4.3.2 分类器	12
第五章 图片识别模块详细设计	15
5.1 视图层	15
5.2 控制层	15
5.3 服务层	16
5.3.1 核心代码	16

第六章 规则管理模块详细设计	18
6.1 视图层	18
6.2 控制层	19
6.2.1 API 核心实现	19
6.3 模型层	20
第七章 统计分析与可视化模块	21
7.1 视图层	21
7.2 控制层	21
7.3 数据处理	22
第八章 运行与部署设计	23
8.1 配置管理	23
8.2 部署方案	23
第九章 数据、日志与安全	24
9.1 数据备份	24
9.2 日志策略	24
9.3 安全与性能	24
附录 A 附录	25
A.1 关键代码示例	25
A.2 流程图说明	25
A.3 界面截图补充	25

第一章 导言

1.1 目的

本详细设计文档用于说明智能垃圾分类系统的总体设计方案、模块内部结构、接口定义和关键算法实现，指导开发、测试与后续维护工作。

1.2 范围

项目覆盖文本分类、图片识别、规则管理、统计分析和开放 API。本文档面向研发、测试、运维及项目干系人。

1.3 缩写说明

缩写	说明
API	Application Programming Interface，应用程序接口
CORS	Cross-Origin Resource Sharing，跨域资源共享
REST	Representational State Transfer

1.4 术语定义

- **规则库：**存储在 `garbage_rules.csv` 中的标准垃圾分类条目。
- **图片识别模型：**基于 CLIP 的中文预训练模型集合，用于推断图片对应的垃圾类型。

1.5 引用标准

遵循《GB/T 8567-2006 计算机软件文档编制规范》和 Flask 官方最佳实践。

1.6 参考资料

参考《网上招聘系统详细设计》模板、项目 README.md、config.py 及 app 目录源码。

1.7 版本更新信息

版本	日期	说明
1.0	2025-11-19	初版，覆盖核心模块设计、流程图与截图

第二章 系统设计概述

系统采用前后端一体化的 Flask 架构，前端静态资源托管在 `app/static`，后端通过 Flask-RESTful 暴露 API。数据源采用 CSV 文件，必要时可替换为数据库。图片识别服务独立于文本规则，使用延迟加载模式以降低启动成本。整体逻辑分为视图层、控制层、模型层和服务层。

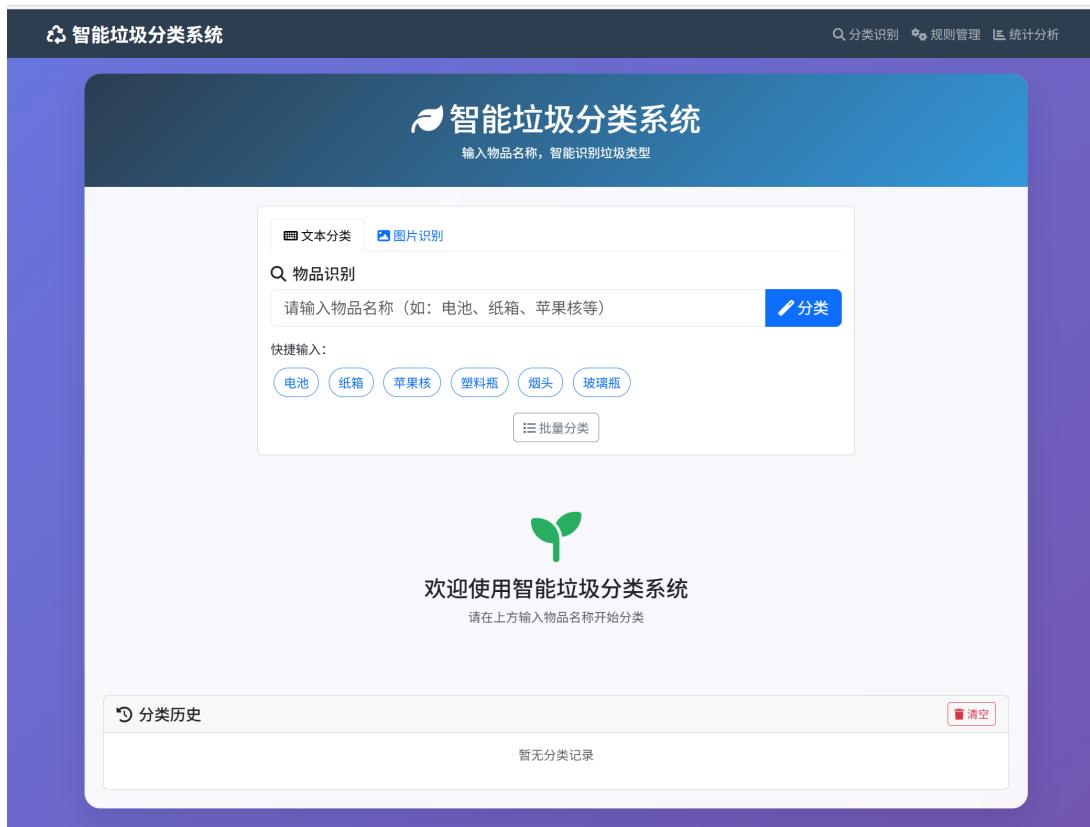


图 2.1: 系统主界面示意

第三章 详细设计概述

3.1 总体结构

层次	说明
视图层	app/static 中的 HTML/JS，负责交互与结果展示。
路由控制层	app/routes，负责请求路由、入参校验与响应封装。
模型层	app/models，包含规则读取、分类策略与批量处理能力。
服务层	app/services，实现图片识别、第三方依赖和扩展能力。
配置层	config.py，负责环境配置、文件路径、日志策略等。

3.2 运行流程

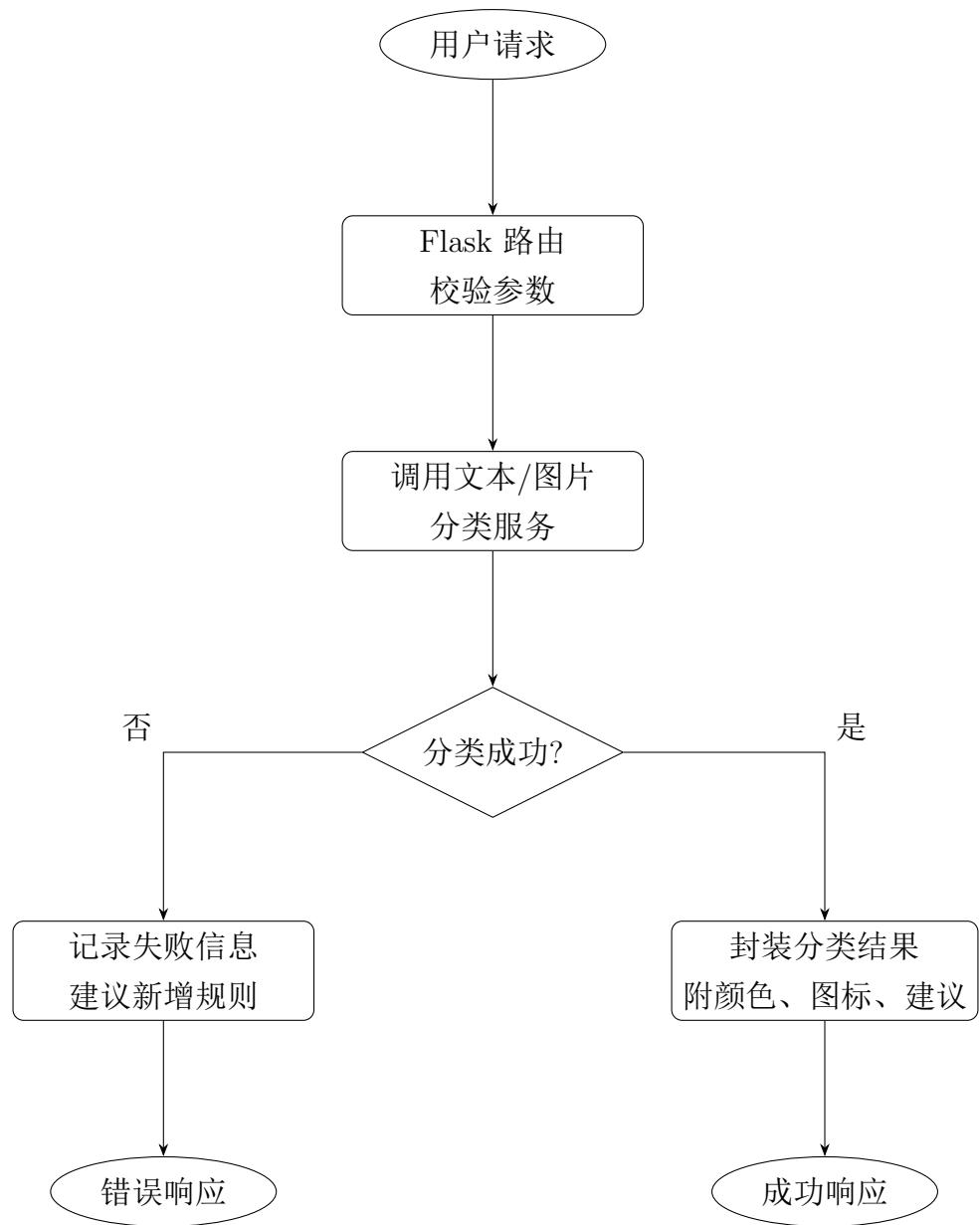


图 3.1: 分类请求处理流程图

第四章 文本与批量分类模块详细设计

4.1 视图层

- 文本分类页面包含输入框、分类按钮和结果卡片，结果展示类型、依据、建议。
- 支持批量输入对话框，逐行输入多个物品，后端返回数组渲染成表格。
- 相关截图如图 4.1 所示。



图 4.1: 文本分类交互界面

4.2 控制层

4.2.1 接口定义

接口	描述
POST /api/classify	单条文本分类，参数 <code>item_name</code> 。
POST /api/batch-classify	批量分类，参数 <code>items[]</code> 。

GET 智能相似物品建议。
/api/similar-items

4.2.2 异常处理

- 缺少参数：返回 400 与错误描述。
- 服务器异常：日志记录后返回 500。
- 批量接口在任何单项失败时仍返回完整结果，便于前端逐条提示。

4.2.3 批量接口实现

批量接口通过复用 `GarbageClassifier.batch_classify`，统一封装颜色、图标和统计信息，确保与单条接口一致的响应格式。核心实现如下：

为避免 `listings` 在 UTF-8 下的局限，以下代码示例中的中文提示字符串以英文展示，实际实现仍返回本地化文案。

Listing 4.1: Batch classify API snippet (app/routes/api.py)

```
1 class BatchClassifyAPI(Resource):
2     """Batch garbage classification API"""
3
4     def post(self):
5         try:
6             data = request.get_json()
7             if not data or 'items' not in data:
8                 return {'error': 'Please provide the items list'}, 400
9
10            items = data['items']
11            if not isinstance(items, list):
12                return {'error': 'Items list must be an array'}, 400
13
14            clf = get_classifier()
15            results = clf.batch_classify(items)
16
17            formatted_results = []
18            for item_name, success, garbage_type, reason, suggestion in results:
19                formatted_results.append({
20                    'item_name': item_name,
21                    'success': success,
22                    'garbage_type': garbage_type,
23                    'reason': reason,
24                    'suggestion': suggestion,
```

```

25         'color': clf.get_type_color(garbage_type) if success else '#666666',
26         'icon': clf.get_type_icon(garbage_type) if success else '?'
27     })
28
29     return {
30         'results': formatted_results,
31         'total': len(items),
32         'successful': sum(1 for r in formatted_results if r['success']),
33         'timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S')
34     }
35
36 except Exception as e:
37     logger.error(f"Batch classification error: {e}")
38     return {'error': f'Batch classification failed: {str(e)}'}, 500

```

4.3 模型层

4.3.1 数据管理器

通过 GarbageDataManager 维护 CSV 规则，支持加载、增删改、模糊匹配和统计聚合。模糊匹配逻辑在未命中 exact key 时遍历关键字。

代码摘录

Listing 4.2: Rule loading snippet (app/models/data_manager.py)

```

1 class GarbageDataManager:
2     """Garbage classification data manager"""
3
4     def __init__(self, csv_file: str = None):
5         if csv_file is None:
6             from flask import current_app
7             csv_file = current_app.config['DATA_FILE']
8
9         self.csv_file = csv_file
10        self.rules_dict = {}
11        self.load_rules()
12
13    def load_rules(self) -> None:
14        try:
15            if not os.path.exists(self.csv_file):
16                print(f"Warning: CSV file {self.csv_file} is missing, creating empty
rule set")

```

```
17     return
18
19     with open(self.csv_file, 'r', encoding='utf-8') as file:
20         reader = csv.DictReader(file)
21         for row in reader:
22             item_name = row['item_name'].strip()
23             garbage_type = row['garbage_type'].strip()
24             reason = row['reason'].strip()
25
26             self.rules_dict[item_name] = {
27                 'type': garbage_type,
28                 'reason': reason
29             }
30
31     print(f"Loaded {len(self.rules_dict)} garbage classification rules")
32
33 except Exception as e:
34     print(f"Error while loading rules: {e}")
35     self.rules_dict = {}
36
37 def get_classification(self, item_name: str) -> Optional[Tuple[str, str]]:
38     item_name = item_name.strip()
39
40     if item_name in self.rules_dict:
41         rule = self.rules_dict[item_name]
42         return rule['type'], rule['reason']
43
44     for stored_name, rule in self.rules_dict.items():
45         if item_name in stored_name or stored_name in item_name:
46             return rule['type'], f"Matched similar item '{stored_name}': {rule['reason']}"
47
48     return None
```

4.3.2 分类器

GarbageClassifier 负责：

1. 先查询规则库。
2. 未命中时执行关键字推理，基于材料/特征词映射。
3. 生成颜色、图标和处理建议，供前端渲染。

代码摘录

Listing 4.3: Classifier keyword reasoning (app/models/classifier.py)

```
1 class GarbageClassifier:
2     def classify(self, item_name: str) -> Tuple[bool, str, str, str]:
3         if not item_name or not item_name.strip():
4             return False, "", "Please enter an item name", ""
5
6         result = self.data_manager.get_classification(item_name)
7
8         if result:
9             garbage_type, reason = result
10            suggestion = self._get_disposal_suggestion(garbage_type)
11            return True, garbage_type, reason, suggestion
12        else:
13            predicted_result = self._keyword_analysis(item_name)
14            if predicted_result:
15                garbage_type, reason = predicted_result
16                suggestion = self._get_disposal_suggestion(garbage_type)
17                return True, garbage_type, f"Predicted: {reason}", suggestion
18            else:
19                return False, "unknown", f"Sorry, no rule found for '{item_name}'",
20                      "Please consult ops or extend the rule set"
21
22    def _keyword_analysis(self, item_name: str) -> Optional[Tuple[str, str]]:
23        recyclable_keywords = ['paper', 'plastic', 'glass', 'metal', 'iron', 'aluminum',
24                               , 'copper', 'steel', 'bottle', 'can']
25        hazardous_keywords = ['battery', 'lamp', 'thermometer', 'medicine', 'paint',
26                               , 'pesticide', 'chemical', 'mercury', 'lead']
27        kitchen_keywords = ['vegetable', 'fruit', 'meat', 'fish', 'shrimp', 'egg',
28                           , 'rice', 'noodle', 'bean', 'milk', 'leftover', 'peel']
29        other_keywords = ['cigarette', 'ash', 'diaper', 'sanitary', 'ceramic', 'brick',
30                          , 'tile', 'dust', 'hair', 'textile']
31
32        for keyword in hazardous_keywords:
33            if keyword in item_name:
34                return "hazardous", f"Keyword '{keyword}' implies hazardous material"
35
36        for keyword in kitchen_keywords:
37            if keyword in item_name:
38                return "kitchen", f"Keyword '{keyword}' implies organic waste"
39
40        for keyword in recyclable_keywords:
41            if keyword in item_name:
42                return "recyclable", f"Keyword '{keyword}' implies recyclable material"
```

```
38  
39     for keyword in other_keywords:  
40         if keyword in item_name:  
41             return "other", f"Keyword '{keyword}' implies residual waste"  
42  
43     return None
```

第五章 图片识别模块详细设计

5.1 视图层

图片识别页面支持拖拽上传、阈值滑杆、识别结果列表与详细 Top-K 置信度图，如图 5.1。



图 5.1: 图片识别界面

5.2 控制层

POST /api/classify-image 入口负责：

- 校验 multipart/form-data，限制类型为 png/jpg/jpeg/gif/bmp。


```
21
22     reason = f"Image classification result: {object_name} (confidence: {
23         confidence}%)"
24
25     return True, garbage_type, reason, object_name, detailed_results
26
27 if detailed_results:
28     best = detailed_results[0]
29     return False, "unknown", f"Detected {best['object_name']} but cannot
30         determine garbage type", best['object_name'], detailed_results
31
32 else:
33     return False, "unknown", "Image content could not be recognized", "", []
34
35 except Exception as e:
36     return False, "error", f"Image classification failed: {str(e)}", "", []
```

第六章 规则管理模块详细设计

6.1 视图层

规则管理页面提供过滤、分页、增删改对话框，截图如图 6.1。

The screenshot shows the 'Rule Management System' interface. At the top, there is a header with a gear icon and the text '规则管理系统'. Below the header, a sub-header reads '管理垃圾分类规则，添加、修改、删除规则'. The main area contains a table with the following data:

垃圾类别	可回收垃圾	操作
塑料瓶	塑料材质可回收再利用	<input type="checkbox"/> <input type="button" value="删除"/>
纸箱	纸质材料可回收利用	<input type="checkbox"/> <input type="button" value="删除"/>
玻璃瓶	玻璃材质可回收再造	<input type="checkbox"/> <input type="button" value="删除"/>
易拉罐	铝金属可回收利用	<input type="checkbox"/> <input type="button" value="删除"/>
报纸	纸质材料可回收利用	<input type="checkbox"/> <input type="button" value="删除"/>
杂志	纸质材料可回收利用	<input type="checkbox"/> <input type="button" value="删除"/>
书本	纸质材料可回收利用	<input type="checkbox"/> <input type="button" value="删除"/>
纸板	纸质材料可回收利用	<input type="checkbox"/> <input type="button" value="删除"/>
塑料袋	难以回收的塑料制品	<input type="checkbox"/> <input type="button" value="删除"/>
矿泉水瓶	塑料材质可回收再利用	<input type="checkbox"/> <input type="button" value="删除"/>

At the top left of the main area, there are buttons for '+ 添加规则' (Add Rule) and '刷新' (Refresh). At the top right, there is a search bar labeled '搜索规则...' and a dropdown menu labeled '所有类型'.

图 6.1: 规则管理页面

6.2 控制层

- GET /api/rules: 返回规则列表及颜色、图标元数据。
 - POST /api/rules: 新增前校验垃圾类型是否属于四大分类。
 - PUT /api/rules: 与新增逻辑统一，覆盖式写入。
 - DELETE /api/rules?item_name=: 精确匹配删除。

6.2.1 API 核心实现

新增接口在写入前做字段、类型和约束校验，并复用 `GarbageDataManager.add_rule` 完成持久化，关键代码如下：

Listing 6.1: Rule create API snippet (app/routes/api.py)

```
1 class RulesAPI(Resource):
2     def post(self):
3         try:
4             data = request.get_json()
5             required_fields = ['item_name', 'garbage_type', 'reason']
6
7             if not data or not all(field in data for field in required_fields):
8                 return {'error': 'Please provide all rule fields'}, 400
9
10            item_name = data['item_name'].strip()
11            garbage_type = data['garbage_type'].strip()
12            reason = data['reason'].strip()
13
14            if not all([item_name, garbage_type, reason]):
15                return {'error': 'Fields cannot be empty'}, 400
16
17            valid_types = ['recyclable', 'hazardous', 'kitchen', 'other']
18            if garbage_type not in valid_types:
19                return {'error': f'garbage_type must be one of: {" ".join(valid_types)}'},
20                400
21
22            dm = get_data_manager()
23            success = dm.add_rule(item_name, garbage_type, reason)
24
25            if success:
26                logger.info(f"Rule added: {item_name} -> {garbage_type}")
27                return {
28                    'success': True,
29                    'message': 'Rule added successfully',
30                }
31        except Exception as e:
32            logger.error(f"An error occurred: {e}")
33            return {'error': str(e)}, 500
```

```
29         'rule': {
30             'item_name': item_name,
31             'garbage_type': garbage_type,
32             'reason': reason
33         }
34     }
35     else:
36         return {'error': 'Failed to add rule'}, 500
37
38 except Exception as e:
39     logger.error(f"Rule creation error: {e}")
40     return {'error': f'Rule creation failed: {str(e)}'}, 500
```

6.3 模型层

规则更新直接写回 CSV 并打印操作日志，保证与批量分类共用同一数据源。为防止并发写冲突，可后续引入文件锁或数据库。

第七章 统计分析与可视化模块

7.1 视图层

统计页面展示各类垃圾数量柱状图与整体比例环形图（图 7.1）。



图 7.1: 统计分析页面

7.2 控制层

GET /api/statistics 汇总规则库条目，计算数量与占比，并附带颜色、图标元数据。前端根据返回的 percentage 进行图表绘制。

7.3 数据处理

统计数据保存在内存中，不写入文件。调用方需要定时刷新以获取最新规则变化。

第八章 运行与部署设计

8.1 配置管理

配置项	说明
SECRET_KEY	Flask 会话密钥， 默认 garbage-classification-system-2024。
DATA_FILE	规则库位置， 默认仓库根目录 garbage_rules.csv。
MAX_CONTENT_LENGTH	上传文件大小限制， 默认 16MB。

8.2 部署方案

1. 开发环境：运行 `python run.py`，启用调试模式。
2. 生产环境：通过 Gunicorn 启动 `app:create_app('production')`，日志写入 `logs/garbage_clas`
3. Docker：可根据 README 中示例构建镜像。

第九章 数据、日志与安全

9.1 数据备份

- CSV 文件需纳入备份计划，建议每日定时复制至对象存储。
- 可设计 API 导出接口，方便外部系统校对。

9.2 日志策略

- 开发环境：标准输出，适合调试。
- 生产环境：RotatingFileHandler，10 个轮转文件，每个 9.7MB。
- 日志记录分类成功/失败、规则修改、图片识别模型加载情况。

9.3 安全与性能

- 通过 CORS 限制来源，可在生产配置中指定允许域。
- 对批量接口设置合理的 items 长度限制，避免阻塞。
- 图片上传需结合 WAF/杀毒策略，防止恶意文件。

附录 A 附录

A.1 关键代码示例

Listing A.1: Classify API snippet (app/routes/api.py)

```
1 class ClassifyAPI(Resource):
2     def post(self):
3         data = request.get_json()
4         if not data or 'item_name' not in data:
5             return {'error': 'Please provide item_name'}, 400
6
7         item_name = data['item_name'].strip()
8         clf = get_classifier()
9         success, garbage_type, reason, suggestion = clf.classify(item_name)
10
11     return {
12         'success': success,
13         'item_name': item_name,
14         'garbage_type': garbage_type,
15         'reason': reason,
16         'suggestion': suggestion,
17         'color': clf.get_type_color(garbage_type) if success else '#666666',
18         'icon': clf.get_type_icon(garbage_type) if success else '?',
19         'timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S')
20     }
```

A.2 流程图说明

图 A.1 展示图片识别服务内部调用链。

A.3 界面截图补充

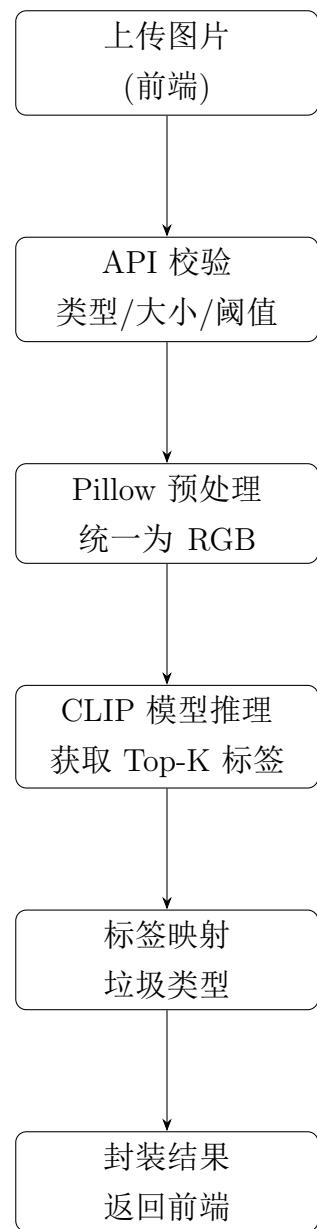


图 A.1: 图片识别服务流程



图 A.2: 文本分类结果示例



图 A.3: 图片识别结果示例