广告审核治理系统 — 基于 RAG 的实现方案

作者：为王宇量身定制的系统设计文档

---

1. 目标与背景

目标：构建一套基于检索增强生成（RAG）的广告审核治理系统，目标是： - 提高自动化判定覆盖率与准确性，降低人工审核量； - 为每次判定提供可解释的“违规条款 + 参考案例 + 置信度”； - 能快速响应新型绕过/对抗手法，通过反馈闭环能力持续自我学习。

背景：王宇已实现本地化 RAG 流水线（LangChain + ChromaDB + 本地 Gemma3n/ollama 模型），本方案在此基础上扩展为生产级广告治理系统。

---

2. 总体架构（高层）

三层审核架构：

```
Incoming Ads / Events
        |
+-------v--------+
| 1) 规则引擎     |  <-- 低延迟、确定性规则（关键词、黑名单、正则）
+-------+--------+
        |
+-------v--------+
| 2) RAG 审核层   |  <-- 检索政策条款 + 相似案例 -> LLM 判定
+-------+--------+
        |
+-------v--------+
| 3) 人工复核     |  <-- 置信度中等/边界样本/上诉流程
+-----------------+
        |
    Feedback Loop
        |
Knowledge DB / Model Re-train
```

注：规则引擎作为一层快速过滤以保障低延迟及安全保底；RAG 层做深入理解与可解释输出；人工复核层负责高风险/疑难案件。所有人工复核结果、投诉与判定结果写回知识库作为增量训练/检索语料。

---

3. 关键模块说明

3.1 数据采集层（Ingestion）

职责：接入所有相关数据流：广告提交接口、投放日志、用户投诉、人工审核结果、政策文档、历史违规案例。

技术/格式：Kafka（实时流）、S3/HDFS（批量归档 Parquet）、关系型数据库（配置/黑名单）。

Schema 建议（广告事件）： - `event_id` (string) - `ad_id` (string) - `advertiser_id` (string) - `timestamp` (ISO8601) - `ad_text` (string) - `creative_urls` (array[string]) - `landing_url` (string) - `meta` (json: geo, device, audience) - `label` (int optional) // 人工/投诉标签

3.2 规则引擎（Rule Engine）

职责：做低延迟精确拦截（硬规则）：关键词、正则、域名黑名单、URL 检测（恶意跳转）、基础图像过滤（尺寸/色情 hash）。

实现：使用高性能规则库（如 OpenResty + Lua 或自研 fastmatch service），规则可热更新并记录命中日志。

输出：`decision: {block/pass, reason: [rule_id,...], matched_text}`

3.3 多模态解析与特征服务**

职责：对广告素材做预处理和富化： - 文本清洗（去噪、同音/形变归一化） - 图片 OCR（Tesseract 或 商业 OCR） - 图片 embedding（CLIP） - URL / Landingpage 抓取并解析 - 元数据（历史违规数、广告主信誉分）查询

缓存：常用 creative/广告主信息缓存到 Redis，提高查询速度。

3.4 RAG 审核层

职责：检索政策条款 & 历史案例并调用 LLM 生成判定与解释。

子模块： - **向量检索服务**：ChromaDB / FAISS / Milvus（按政策条款与案例向量化存储） - **Prompt Builder**：将检索结果拼接进 prompt 模板并加入上下文（ad_text + OCR_text + meta） - **LLM Model**：本地部署 Gemma3n / Ollama 或远端 API（若公司政策允许） - **Post-processor**：解析模型输出成结构化结果（`decision`, `reason`, `clauses`, `examples`, `confidence`）

示例 Prompt 模板：

```
You are an ad compliance assistant. Given the ad content and related policy clauses and
historical cases, decide if the ad violates the policy.
Ad Text: {ad_text}
OCR Text: {ocr_text}
Meta: {meta}
Relevant Clauses: {clauses}
Similar Cases: {cases}
```

```
Please output JSON: {"decision": "block/allow/flag", "confidence": 0.0-1.0,
"matched_clause_ids": [], "explanation": "..."}
```

检索策略：先检索 n=5 最近/相关条款 + 3-5 最相似的历史违规案例（基于 cosine similarity）。

置信度计算：LLM 输出概率/信心水平 + 基于检索相似度加权形成最终置信度。

3.5 人工复核平台

职责：呈现疑难样本（模型低置信度或人工上诉），辅助人工快速判定并收集理由标签。

功能点： - 支持批量审核与单条审核 - 显示模型给出的解释/命中条款/历史相似案例 - 审核人可修改判定并标注条款 - 打标签后数据回流训练库

3.6 模型管理与训练平台

职责：管理模型版本、训练流水线、A/B 测试环境与自动化再训练。

关键能力： - 数据切片和版本化（使用 DVC 或数据湖 + 元数据） - 自动化训练 pipeline（Airflow / Kubeflow） - 模型注册与部署（MLflow / Seldon / KFServing） - 在线/离线评估对照

3.7 监控与报警

监控指标： - 流量/延迟/错误率 - 模型指标：线上 AUC/Precision/Recall（通过抽样人工标注校验） - 业务指标：投诉率、误判率、人工负载 - 数据漂移（PSI）与新模式告警

报警策略：当误判率或投诉率超过阈值自动触发回滚与人工检查。

---

4. 数据与知识库设计

4.1 知识库（Policy DB）结构

- `clause_id` (string)
- `text` (string)
- `policy_category` (enum)
- `effective_date` (date)
- `vector` (float[]) // embedding
- `examples` (list of case_ids)

4.2 历史违规案例表

- `case_id`
- `ad_id`
- `ad_text`
- `creative_hash`
- `verdict` (block/allow)
- `clause_ids`
- `annotator_id`

- `timestamp`

---

5. API 设计（简要）

5.1 审核请求（同步）

`POST /api/v1/review` **请求体**: `{ ad_id, ad_text, creative_urls, landing_url, meta }` **返回**: `{ decision, confidence, matched_clause_ids, explanation, examples }`

5.2 批量异步（用于离线批处理）

`POST /api/v1/review_batch` → 返回 task_id，查询 `/status/{task_id}` 获取结果（存 S3 / DB）

5.3 人工复核回传

`POST /api/v1/human_feedback` `{ case_id, final_decision, reasons, annotator_id }` → 写入案例库并触发 retrain 如果达到阈值

---

6. 部署策略与资源（建议）

6.1 早期（PoC）

- 小规模用例：单机部署 ChromaDB + 本地 Ollama/Gemma3n
- LangChain 构建 pipeline，FastAPI 封装服务
- 数据量小用 SQLite/Postgres + Parquet 存档

6.2 生产部署

- 向量库：Milvus / FAISS + 持久化云存储
- LLM：公司合规允许下用 GPU 集群托管模型或调用受控外部 API
- 服务框架：K8s + HPA，Model server（Seldon/KFServing）
- 流量组件：Kafka + Flink/Beam 做流式预处理
- 存储：S3（冷存）+ ClickHouse（实时分析）

---

7. 评估指标与 A/B 测试

线上指标： - 自动化通过率（自动判定占比） - 人工审核量（人/天） - 违规检出率（Recall）与误判率（Precision） - 投诉率、退单率、品牌/平台声誉指标

A/B 测试建议： - 随机分流广告到 control（现有系统） vs treatment（RAG） - 观察 2-4 周后：召回率、误判率、人工负载变化

---

8. 风险与缓解

1. **LLM 误判 / 不稳定输出**：使用结构化 prompt + 输出校验器（JSON schema）；低置信度交给人工。

2. **越狱/对抗攻击**：对抗样本训练、规则层检测混淆字符/拼音/编码；实时举报链路快速回补。
3. **法规合规与隐私**：敏感 PII 需脱敏；政策条款需法律团队审核并归档。
4. **性能瓶颈（延迟）**：规则层先行、RAG 层异步化或缓存常见判定结果。

---

9. 迭代与路线图（6 个月建议）

Month 0-1（PoC）: 本地 RAG + 少量政策/案例，搭建 API 与人工复核工具

Month 2-3: 扩展向量库、引入图片 OCR + CLIP，多模态 RAG，做离线评估

Month 4-5: 接入流式采集（Kafka）、部署 model server、完善监控与数据回流

Month 6: 小范围 A/B 上线，反馈优化，制定自动化 retrain 策略

---

10. 测试计划（示例）

- 单元测试：prompt builder、post-processor、APIs
- 集成测试：从 ingestion 到 decision 的 end-to-end 流程
- 对抗测试：生成对抗样本（谐音、混淆字符、图片嵌入文本）检验鲁棒性
- 人工盲测：抽样人工复核与模型判定比较，计算指标

---

11. 附：Prompt 示例与解析

Prompt（简化）：

```
Policy clauses:
1) Clause 001: ...
2) Clause 002: ...

Ad Text: "{ad_text}"
Similar Cases: {case_snippets}
Question: Does this ad violate any policy? If yes, list clause ids and explain.
Return JSON.
```

解析要点： - 将检索到的条款/案例限制在合理长度，避免超出上下文窗口 - 指导 LLM 输出严格的 JSON，用 post-processor 校验并容错

---

12. 总结

本方案以 **实用落地** 为主线：规则保障低延迟+RAG 提供可解释判定+人工复核闭环保证质量。结合王宇现有本地 RAG 实践，可以快速从 PoC 向生产化推进。文档后续可扩展为详细的 API Spec、ER 图、部署清单与成本估算。

如果你希望，我可以把这份设计文档导出为 PDF/Markdown，或生成更详细的部署清单（包括 k8s manifest、Dockerfile、prometheus 报表模版 等）。告诉我你需要哪个，我直接生成。