王红强

男 | 15031231747 | 15031231747@163.com

7年工作经验 | 求职意向：Java / Agent 工程 | 期望城市：北京

教育经历

长安大学 本科 2015-2019

在校期间认真学习，积极参加老师的项目，有较好的学习能力。

本人性格积极乐观，随和幽默，爱好较为广泛。

专业技能

熟练掌握 Agent 架构设计与实现，包括 LLM Agent、Tool-using Agent、Multi-Agent System，具备构建企业级 Agent 系统的完整经验；

熟悉 Agent 核心机制：Planning（任务规划）、Memory（记忆管理）、Tool Use（工具调用）、Reflection（反思优化）、Multi-Agent Collaboration（多智能体协作）；

掌握主流 Agent 框架：LangChain、LlamaIndex、AutoGPT、MetaGPT、Claude Agent SDK，能够基于业务场景选型并定制开发；

熟练使用 MCP（Model Context Protocol）协议，设计并实现自定义 MCP Server，支持 Agent 与外部系统（数据库、API、文件系统）的标准化集成；

具备 Agent 系统工程化能力：Prompt 模板管理、Tool Registry、Context Window 优化、流式输出、错误重试、幂等性保障、可观测性建设；

熟悉 Agent 安全合规与成本治理：权限控制、调用审计、敏感信息脱敏、Token 消耗优化、缓存策略、超时熔断；

能够设计并实现复杂 Agent Workflow：多步推理、条件分支、循环迭代、并行执行、子任务分解与结果聚合；

具备 Agent 评估与优化能力：设计评估指标（准确率、召回率、响应时长、成本）、A/B 测试、Prompt 迭代优化、Few-shot Learning。

熟悉使用 Spring AI、BGE M3 模型、Qwen3-Plus、通义 TextEmbeddingV2、LangChain4j 链式调用与 Prompt 工程；

熟悉使用智能话术推荐（RAG 核心）：设计并实现核心的智能辅助/话术推荐接口，采用 BM25 检索 + 余弦相似度 + 文本相似度动态决策。RAG 检索增强系统搭建（Milvus 向量库集成）；

熟练掌握 Java 语言特性，熟悉常用的 API 和设计模式，拥有良好的编码习惯，有代码洁癖；

熟练使用 JAVA 常用开发工具，如 IDEA、Postman、Git等；

精通 MySQL 操作，阅读过底层，知道在项目前中后期具体怎么优化和设计，目前熟练使用国产的 TiDB ；

熟练 Redis、MongoDB，阅读过源码，对 Redis 的持久化、集群策略有一定了解；

熟练常用的 Linux 操作系统的基本指令；

熟悉 JVM 内存模型、类的加载过程、GC 垃圾回收机制，知道 JVM 如何性能调优；

精通 RocketMQ、kafka，阅读过源码，用过它的延时队列 + 事务消息，尽量避免少丢消息，以及丢消息后做补偿；

熟悉微服务框架 Spring Boot、Spring Cloud、Spring Cloud Alibaba 及其各种组件等；

熟练使用 Elasticsearch、Logstash or filebeat、Kibana，看过 ES 的源码。

工作经历

携程计算机技术（上海）有限公司 Java 2019.02-2021.04

负责酒店商家综合运营与搜索转化模块的研发工作，重点参与商家运营辅助、价格合规管控及数据可视化子系统的设计与实现。主导开发了基于Flink的实时数据分析流程，通过Python爬虫抓取竞品数据，为中小商家提供个性化标签优化建议，使商家店铺信息完善率从68%提升至92%，PSI指数平均提升18%。重构定价合规校验逻辑，设置动态阈值并开发预警功能，定价合规率从75%上升至98%，因定价佣金引发的投诉下降72%。同时优化数据统计接口与可视化报表，数据准确率提升至99.8%，支撑运营团队实时监控核心指标。

北京沃东天骏信息技术有限公司 武汉分公司 Java 2021.04-至今

深度参与京东大金融交易中台核心系统建设，负责支付网关与智能风控两大核心系统的技术研发与 AI智能化升级，覆盖日均亿级交易量、数百亿资金流转的核心链路。在支付网关项目中，设计动态路由与多商户号切换策略，支付成功率从98.2%提升至99.5%；采用Kafka+Flink重构对账流程，实现实时对账；优化全球购支付能力，接入汇率服务和卡BIN识别，跨境支付失败率从15%降至4%。智能风控参与日均亿笔交易的实时风险识别与拦截系统建设，负责 Drools 规则引擎热加载优化（解决元空间泄漏）、Dubbo 泛化调用性能优化（TPS 从 2000 提升至 4800+）、Flink 实时风控引擎开发（解决 RocksDB 写放大导致 Checkpoint 超时），风险拦截率达 99.5%，决策耗时控制在 50ms 以内。利用 AI Coding 工具（Claude Code、Cursor）主导 15 万行代码重构，文档生成效率提升 90%；基于 AI Agent 技术构建对账差错智能处理系统，开发数据清洗、差错诊断、批量评估、汇总报告 4 大 Agent，差错处理时长从15 分钟降至 5分钟以内，预期可优化财务团队工作量 60%，按当前人力成本测算，完全落地后预计年节省成本 ¥200 万，目前已完成核心功能开发，正在灰度测试阶段。

项目经历

支付对账差错智能处理系统（基于 AI Agent） AI Agent 工程师 2025.07-至今

内容：

支付对账是交易中台的核心环节，日均需处理京东与微信、支付宝、银联等 12 个支付渠道的对账文件，但传统人工处理模式面临三大核心痛点：

1. **对账差错处理效率低**：日均对账差错 500-800 笔（占比 0.05%），每笔差错需人工排查原因（渠道掉单、重复支付、金额不符、时间差异等），平均处理时长 15 分钟，财务团队 8 人每天耗时 100+ 小时处理差错，高峰期（大促后）积压超 2000 笔；
2. **差错原因分析依赖经验**：差错排查需调取支付流水、渠道返回码、用户操作日志、系统监控等多源数据，新人需 3 个月才能独立处理，老员工离职导致知识断层，处理准确率波动大（60%-85%）；
3. **重复性工作占比高**：统计发现 70% 的差错属于高频类型（如"渠道返回成功但我方未收到通知""用户重复点击导致重复扣款"），每次都要重复相同的排查流程，人工效率低且容易出错。

核心目标：基于 AI Agent 技术构建对账差错智能处理系统，实现高频差错自动化排查与修复，差错处理时长从 15 分钟降至 2 分钟以内，人工处理量降低 70%，处理准确率稳定在 95% 以上，财务团队从 8 人优化至 3 人。

业绩：

1.差错智能诊断 Agent（解决“排查慢、依赖经验”）

问题：差错排查需综合分析支付流水（MySQL）、渠道对账文件（FTP）、系统日志（Elasticsearch）、监控数据（Prometheus），传统规则引擎能处理简单场景，复杂差错需人工逐一排查，新人不知从何下手。

解决：基于大模型（Claude Sonnet 3.5）构建差错诊断 Agent，实现四大核心能力：

多源数据自动采集：设计 MCP Server 集成层，实现 MySQL MCP Server（查询支付流水）、FTP MCP Server（读取渠道对账文件）、Elasticsearch MCP Server（检索系统日志）、Prometheus MCP Server（查询监控指标），Agent 根据差错单号自动调用多个 Tool 采集上下文；

差错原因推理：将采集的数据（支付流水状态、渠道返回码、日志关键字、网络监控）输入大模型，结合历史案例库（Few-shot Learning），推理差错根因（如"渠道回调超时导致我方未更新状态""用户网络抖动导致重复提交"）；

修复方案生成：根据差错类型自动生成修复方案（如"重新查询渠道订单状态并补单""标记重复订单并退款""联系渠道核实"），并自动调用修复工具（补单 API、退款 API、工单系统）；

人工审核机制：低置信度差错（<80%）或涉及大额金额（>1000 元）自动转人工复核，Agent 提供诊断报告辅助决策。

效果：Agent 自动处理 70% 的高频差错（如渠道回调超时、重复支付），差错平均处理时长从 15 分钟降至 2 分钟；诊断准确率 95%，误判率从 15% 降至 5%；新人培训周期从 3 个月缩短至 1 周。

2. 批量差错评估与优先级排序 Agent（解决“高峰期积压、处理无序”）

问题：大促后差错积压超2000笔，按时间顺序处理导致大额/投诉类高优先级被延迟，用户满意度低；人工评估影响面耗时长。

解决：设计批量评估 Agent，实现智能调度与排序：

差错影响评估：Agent 自动分析差错影响维度（金额大小、投诉情况、VIP标识、商家结算影响等），综合打分（0-100）；

优先级自动排序：按评分从高到低排序差错工单，高优先级差错（>80 分）自动标红并推送至处理队列首位；

批量处理策略：对同类型差错（如"某渠道批量回调超时"）自动聚合，Agent 生成批量修复脚本，一键执行；

处理进度看板：实时展示差错处理进度（待处理、处理中、已完成），剩余处理时长预估（基于历史数据）。

效果：处理效率提升 60%；高优差错耗时由 4 小时降至 30 分钟；批量处理耗时缩短 80%；整体用户投诉率下降 50%。

京东大金融 支付网关系统（交易中台） java 2023.02-至今

内容:

随着京东零售业务覆盖自营、POP商家、小时购、企业购、全球购等15条业务线，支付环节逐渐暴露出制约业务发展的四大瓶颈：

一是支付渠道碎片化，各业务线独立对接微信、支付宝、银联、京东支付、数字人民币等12类支付渠道，存在“自营用老版SDK、POP用定制接口”的差异，跨业务支付能力复用率不足30%，新渠道接入平均耗时25天；

二是支付响应滞后，大促高峰期（如618、双11）支付接口超时率达1.5%，导致用户“支付成功但订单未确认”的投诉日均超1500单，用户“支付相关投诉量减少70%”。

三是异常处理混乱，支付超时、重复支付、渠道故障等场景缺乏统一预案，资金退回时效最长达18小时，用户满意度仅70%。

四是成本管控薄弱，未结合渠道费率、补贴政策动态选择最优渠道，年均额外支付渠道成本超1500万元。

核心指标设定为：支付接口平均响应时间≤80ms，超时率降至0.5%以下，渠道接入周期缩短至7天，年支付成本降低25%，用户支付相关投诉量减少70%。

业绩:

1. 原有路由策略仅基于静态权重，导致某些渠道在高峰期过载或者被灰黑产攻击被临时风控。

解决：设计并实现了基于多商户号分流+异步动态探活检测，实时成功率的动态路由算法，用redis zset写了个滑动窗口算法，3秒窗口期内，统计各渠道成功率，自动降低失败率高的渠道权重。上线后渠道负载均衡度提升40%，整体支付成功率提升2.1%。支付整体成功率提升至99.5%

2.老版本对账在数据库里面多表join，多线程对账出现过死锁导致cpu彪高情况，换用kafka+Fink重构前半部分对账流程，随意参与后半部分，对账差错处理中心的业务研发，上线后人工处理量减少90%

3.原有支付服务代码历时多年，耦合了风控、营销、会员等逻辑，单次部署上线需40分钟，变更风险高。老板随后让我负责牵头按业务域进行模块重构与解耦。首先，通过静态代码分析梳理调用链路，定义清晰的上下文边界。然后，将营销立减、会员积分等非核心支付逻辑，通过DDD领域事件发布出去，由下游业务方订阅处理。重构后，支付核心服务部署时间降至8分钟，相关功能迭代速度提升一倍。

4. 参与全球购支付能力建设，解决“汇率波动与支付失败”问题

对接第三方汇率服务API ，开发实时汇率获取组件，在支付发起前向用户展示预估金额，支付时按实时汇率计算实际费用；针对跨境支付中“银行卡地区限制”问题，开发卡BIN识别模块，通过解析银行卡前6位判断卡种与地区，提前拦截不可用卡；上线后全球购订单支付失败率从15%降至4%，用户汇率投诉量减少85%

京东零售智能风控中台 Java 2021.06-2023.01

内容:

日活用户超1亿，日均订单量峰值突破亿笔。交易场景覆盖自营、三方商家、拼购、直播、秒杀、企业购、虚拟充值、本地生活、跨境电商等数十种复杂业务，交易链路贯穿用户下单、支付、退款、商家结算、京豆/优惠券核销、售后、资金结算等全流程，平台日均资金处理规模达数百亿元。

随着业务高速发展与模式创新，交易风险（如欺诈交易、恶意刷单套利、商家违规套现、用户信息盗用、批量虚假下单、异常退款/提现等）呈现多元化、隐蔽化、规模化特征，对平台资金安全与用户体验构成严峻挑战：

以2024年Q3为例，系统日均识别到的异常交易行为超过千万次，相关风险涉及的预估资金损失月均达数千万元级别，不仅造成直接经济损失，还引发正常用户下单拥堵、商家结算延迟、平台声誉受损等连锁问题。

原有风控系统采用“独立规则引擎+人工审核”模式，在支撑新业态时面临以下痛点：

1. 风控逻辑与核心交易链路耦合过深，风控校验过程阻塞主流程，高峰期支付环节延迟增加超过2秒，用户支付流失率提升超过10%；

2. 风险识别以离线T+1分析为主，缺乏实时决策与拦截能力，风险行为发生后方可处置，资金追回难度与成本极高；

3.项目目标在于实现风险毫秒级实时感知与拦截、对正常业务无感、支持策略可视化配置与快速迭代、全链路风险可追溯。最终将核心风险拦截率提升至99.5%以上，风控决策耗时控制在50ms以内，降低业务风险资金损失超过80%，并大幅提升风控策略的运营与研发效率。

业绩：

一、Drools规则热加载元空间泄漏治理

问题：风控规则动态更新时，原方案全量重建KieBase并创建新ClassLoader，旧ClassLoader与类元数据无法回收，元空间持续膨胀，引发Full GC乃至OOM。

方案：

1.引入KieScanner监听Maven仓库规则包版本变化，实现增量热更新，减少全量重建；2.对历史KieBase与ClassLoader使用WeakReference，新规则加载成功后主动触发System.gc()提示回收元空间。

效果：元空间稳定在512MB以内，热加载时间从分钟级缩至5秒内，系统稳定性显著提升。

二、高并发Dubbo泛化调用性能优化

问题：规则引擎高并发泛化调用下游数百个特征服务，反射解析与Hessian2序列化成为瓶颈，CPU飙升，单机TPS从5000降至2000。

方案：

1.服务启动时缓存高频泛化调用的方法元数据，避免重复反射；

2.以Kryo替换Hessian2，并引入对象池降低序列化开销；

3.对调用最频繁的TOP10服务，推动业务方提供轻量级API二方包改为直调。

效果：单机TPS恢复至4800+，CPU使用率下降35%，YoungGC频率降低，GC暂停时间减至15ms左右。

三、Flink RocksDB写放大导致Checkpoint超时

问题：TB级状态数据持续高速写入，RocksDB频繁Compaction产生严重写放大，Checkpoint耗时从1分钟飙升至5分钟以上，作业反复重启。

方案：

1.状态数据治理：重新设计状态结构，按时间维度分桶管理状态并设置TTL，实现过期状态自动清理，从源头控制状态总量；

2.RocksDB参数调优：结合硬件特性（SSD），调整write_buffer_size、level0_slowdown_writes_trigger等关键参数，优化Compaction策略，平衡写入性能与空间放大；

3.Checkpoint机制优化：将全量Checkpoint切换为异步增量Checkpoint模式，利用多线程提升快照上传并发度。

效果：Checkpoint耗时稳定在30秒以内，作业SLA达99.9%以上，后端存储成本降低约60%。

携程大住宿部门-酒店商家综合运营+搜索转化模块　数字后端工程师　2019.05-2021.04

内容：

随着美团、飞猪等竞品冲击，以及抖音、小红书等内容平台的流量分流，携程酒店业务面临两大核心痛点：
一是商家运营效率低下，多数中小酒店商家缺乏专业运营能力，无法有效优化店铺信息、调控价格与库存，导致平台酒店PSI（综合竞争力指数）偏低，搜索排名靠后，平均曝光转化率不足3.2%；
二是用户筛选决策成本高，携程80%的酒店订单来自搜索页，但用户在搜索后需在14.3家酒店中反复对比，核心诉求（价格、房型、设施、评价）无法快速匹配，导致订单流失率高达45%；
三是平台与商家的协同存在短板，携程原有调价助手存在不合理干预商家定价的问题，且商家佣金率偏高（部分品类达25%），加上信息同步不及时，导致商家投诉率居高不下（月均投诉超800起），影响平台生态稳定性。
为解决上述问题，提升平台酒店商家活跃度、搜索转化效率，优化平台与商家的协同生态，增强携程大住宿业务的核心竞争力，同时响应市场监管局对平台定价合规性的整改要求，启动本次酒店商家综合运营与搜索转化优化模块项目。

业绩：

1.参与商家运营辅助子模块的功能开发与优化，重点负责店铺信息优化辅助、智能运营诊断两个核心功能的业务逻辑实现。针对中小商家无法精准优化店铺标签、标题的问题，使用Python编写数据爬虫脚本，抓取携程平台内同品类优质商家的标签、标题及用户搜索热词数据，通过Flink进行数据清洗与分析，筛选出高转化标签与关键词，设计标签推荐算法（非复杂算法，侧重业务逻辑匹配），为商家提供个性化的标题、标签优化建议；同时，针对商家运营诊断报告过于笼统的问题，优化报告生成逻辑，整合商家曝光、转化、好评等核心数据，对比同区域竞品数据，明确指出商家短板，并提供可直接落地的优化方案（如阶梯满减、连住优惠设置建议）上线后，商家店铺信息完善率从68%提升至92%，标签匹配准确率达88%，助力中小商家PSI指数平均提升18%。
2.参与价格与库存合规管控子模块的合规优化，重点解决原有调价助手不合理干预商家定价、定价合规性不足的问题。结合市场监管局整改要求，梳理平台定价管控规则，设计价格合规校验逻辑，通过Java开发合规校验接口，设置定价阈值（基于商家成本价、同区域竞品价格），禁止调价助手强制修改商家定价，改为向商家推送竞品价格参考与市场需求预警，同时开发违规定价预警功能，当商家定价超出合规阈值时，自动触发预警并提醒整改，同步留存定价记录，确保合规可追溯；此外，优化商家佣金展示模块，明确不同品类商家的佣金比例，设计佣金减免规则（运营达标商家可享受5%-10%的佣金减免），减少商家因定价、佣金问题产生的投诉，项目上线后，商家定价合规率从75%提升至98%，因定价、佣金问题导致的投诉量下降72%，顺利通过市场监管局合规检查。
3.参与数据统计与可视化子模块的开发与优化，负责商家端数据统计功能的业务逻辑实现与数据校验。基于tidb、Hive数据库，开发商家核心数据（曝光、转化、订单、好评率、PSI指数）的统计接口，通过Flink实现数据实时同步，确保商家可实时查询自身运营数据；针对商家反馈的数据统计不准确、报表不直观的问题，优化数据统计逻辑，修正3处数据统计漏洞（如订单量统计重复、好评率计算错误），优化可视化报表设计，新增竞品对比图表、数据趋势预警功能，支持商家自定义报表筛选维度，提升商家数据解读效率；同时，协助平台运营团队，开发平台端核心指标统计报表，实时监控项目达成情况，为运营决策提供数据支撑，数据统计准确率提升至99.8%，商家对数据统计功能的满意度达90%以上。

个人优势

• 具备Java面向对象编程思想及系统设计能力，熟练掌握Java基础，熟悉Java内存结构及内存模型，熟悉类加载机制，垃圾回收与GC垃圾回收机制，JVM性能调优；
• 熟练掌握Java框架，Spring、SpringMVC、MyBatis、SpringBoot、SpringCloud等框架项目开发；
• 熟练掌握MySQL、Oracle等关系型数据库，熟练掌握MySQL 优化手段，分库分表从设计方案到实施；
• 熟练掌握非关系型数据库redis 其底层数据结构及核心原理；对Redis 持久化、集群策略有一定了解；
• 熟悉FTP文件存储、阿里云OSS对象存储、minio分布式文件存储使用，阿里云短信服务平台（SMS）等第三方接口平台的使用；
• 熟练使用Nginx反向代理及负载均衡，了解DNS、CDN；
• Devops：熟练使用Docker，jenkins，经常写一些dockerFile，写过K8S的一些自动压测脚本；
• 云原生：熟练掌握Kubernetes 编排以及周边生态/Ranncher/Docker/Habor，搭建持续集成环境；
• 熟练使用Git、SVN、Maven等版本控制；有linux常用命令及Linux下部署项目的实际经验，熟练业务分析、功能开发、有系统架构设计及运维系统经验；
• 熟悉Kibana+Logstash；熟练使用Elasticsearch、Logstashor filebeat、Kibana，看过ES的源码；
• 具备带新人能力及一定的项目管理经验，具有较强的文档编写能力；