王佳炜

+86 15277198576 | q1428030957@163.com
github.com/jiawei-wang-dev
求职意向：Go 后端 / AI Agent 应用开发



教育背景

莫纳什大学（QS 31） 信息技术硕士 **2025.07 - 2028.02**
广西大学（211） 土木工程本科 **2019.09 - 2023.06**

相关课程：云计算与安全、移动与分布式计算系统、互联网应用开发、数据科学基础

项目经历

MonashMate | 校园行政助手 Agent | 个人项目

基于 Harness Engineering 的可信 RAG 与反馈自进化系统

项目背景：面向 Monash 校园行政与学生服务场景，独立设计并实现多部门可信行政助手 Agent，围绕制度版本治理、跨部门检索、上下文控制、Agent 执行治理与反馈闭环，解决普通 RAG 中旧制度误召回、证据不可控及复杂任务执行不稳定等问题。

技术栈：Go、Gin、CloudWeGo Eino、MySQL、Elasticsearch、Redis / Redis Stream、BM25、Dense Vector、RRF、Reranker、Prometheus

- **Agent Harness 编排治理：**基于 Go、Gin、CloudWeGo Eino 构建 Graph / ReAct Harness，将主链设计为 Intent → Route → Memory / RAG / Evidence → Context → ReAct → Verify → Finalize；由 Runtime 统一约束部门权限、Tool Allowlist、MaxSteps / Budget / Timeout、重复调用与 Citation 合法性，实现受控 Tool Calling 与失败降级。
- **可信 RAG 与事实治理：**以 MySQL 作为 Fact Plane 管理制度文档、Chunk、版本链及 active / archived 状态，Elasticsearch 承担 BM25 + Dense Vector 混合召回，经 RRF + Reranker 排序后回查 MySQL 校验 active / version / department，避免旧版本索引被误当成当前事实。
- **Evidence 治理与精确检索：**结合 Query Rewrite、BM25 / Dense Vector 并行召回、RRF 与 Reranker 处理数字、条款号、时间节点及自然语言查询；将 Retrieval / Tool Result 统一转为 Evidence，通过有效性校验、Authority / Freshness 评分与 Citation Allowlist 约束最终回答。
- **Memory 与 Context Engineering：**使用 Redis 管理 Session Focus，并结合 Episodic、User、Organization Memory；由 ContextNode 按权限、当前输入优先级、相关性及 Token Budget 裁剪并组装 Agent Context，避免历史摘要、组织经验或模型推断覆盖权威制度事实。
- **反馈自进化与 Skill Workflow：**基于 Business Trace 将用户反馈与 Verifier 失败沉淀为 Bad Case，按 Intent / Retrieval / Knowledge / Generation Gap 归因并生成 Skill / Rule / Hook 候选；经 Replay / Regression、Review、灰度发布与回滚后进入正式版本。
- **多部门 Agent Service：**按部门拆分独立 Agent Service，由 Orchestrator 并行调度跨部门任务并支持 Partial Success、Timeout 降级与故障隔离；底层共享 Elasticsearch，通过部门 Index / dept_id Scope 隔离检索范围。

WatchOps-Lite | 智能排障 Agent | 个人项目

github.com/jiawei-wang-dev/WatchOps-Lite

项目背景：面向 OnCall 场景构建智能排障 Agent，将指标、日志、Trace 与知识库接入统一诊断流程，支持多轮故障排查与证据驱动的诊断输出。

技术栈：Go、Gin、CloudWeGo Eino、Redis、Elasticsearch、Prometheus、Jaeger、OpenTelemetry、Docker

- **Agent 执行链路：**基于 Eino Graph + ReAct 分层处理任务理解、参数校验、知识检索、工具取证与答案生成；service、trace_id 等关键条件不足时优先澄清，避免错误查询并减少无效 Tool Call。
- **多轮上下文：**使用 Redis 管理近期消息、历史摘要与当前任务状态，对服务、时间范围等已确认信息受控继承；当前输入优先于历史状态，并限制旧摘要覆盖新事实，降低长对话中的上下文漂移。
- **Hybrid RAG + Evidence：**结合 BM25 + Vector 处理服务名、错误码与自然语言故障描述；将 Metrics / Logs / Trace / RAG 结果统一为 Evidence，通过 Evidence ID 约束诊断结论引用，并在证据不足时继续检索或安全降级。
- **评测与回归：**围绕 Intent / Slot、Retrieval、Tool / Evidence 与 Agent E2E 建立分层评测，失败用例沉淀为 Bad Case，通过 Replay / Regression 形成可定位、可复现、可回归的质量闭环，并持续补充回归样本集。

RelayIM-Go | 分布式实时消息后端 | 个人项目

github.com/jiawei-wang-dev/relayim-go

项目背景：独立设计多节点即时通讯后端，围绕长连接、跨节点路由、消息可靠投递、离线恢复及节点故障场景进行工程化实现与验证。

技术栈：Go、go-zero、gRPC、WebSocket、Kafka、Redis、MySQL、etcd、Prometheus、Grafana

- **长连接与路由：**WebSocket 管理客户端长连接，etcd 感知服务节点、Redis 保存在线路由；通过心跳、连接状态与发送队列处理断线重连、失效连接及慢客户端，并限制异常连接持续占用资源。
- **可靠投递与幂等：**Kafka 解耦消息持久化与在线投递，Redis Stream 承接跨节点消息；以客户端消息 ID + MySQL 唯一约束处理网络重试和重复消费，并设计消费者异常接管与未完成任务恢复机制，降低节点故障下的消息丢失风险。
- **离线与历史恢复：**分别维护消息同步进度与用户阅读进度，Redis 缓存近期消息、MySQL 保存完整记录；用户重连后按同步位置补齐离线消息、送达与已读状态，并避免同步进度与阅读进度相互覆盖。
- **故障验证与可观测性：**围绕重复消费、断线重连、节点故障和消费者接管设计 Unit、Race、E2E 与故障测试，并通过 Prometheus / Grafana 观测连接、消息积压与投递状态。

技术能力

- **Go 后端：**Go、Gin、go-zero、gRPC / Protobuf、WebSocket；熟悉 Goroutine / Channel、并发控制、长连接、REST API、服务通信与异步任务。
- **数据库与缓存：**MySQL、Redis / Redis Stream；熟悉索引、事务、MVCC、缓存一致性、高可用、分布式锁及热点 Key 等常见场景。
- **消息与分布式：**Kafka、etcd；熟悉消息可靠投递、幂等消费、Consumer Group、服务注册发现及多节点服务协作。
- **AI Agent：**CloudWeGo Eino、Graph / ReAct、Tool Calling、MCP、Hybrid RAG、Memory、Skills、Agent Harness、Evidence、Agent Evaluation、Replay / Regression。
- **检索与可观测性：**Elasticsearch、BM25 / Dense Vector、RRF / Reranker；Prometheus、Grafana、OpenTelemetry、Jaeger，具备 Metrics / Logs / Trace 排查经验。
- **工程化与测试：**Linux、Git、Docker / Compose；具备 Unit、Race、E2E、故障验证及 CI/CD 基本实践。