项目申请书

项目名称：开源软件 API 图谱构建与基于图谱的开源软件生态评估

导师：齐国强

申请人：代睿祺

日期：2025/5/22

1 项目背景

1.1 项目简介

随着开源软件在全球范围内的广泛应用，理解和评估开源软件、洞悉开源软件生态的结构与动态十分关键。本项目计划构建API签名并搭建开源软件 API 图谱，覆盖功能、参数类型、来源三方库、编程语言等信息。并基于该图谱对开源软件生态进行评估，帮助开发者和企业更好地了解开源软件的内部结构、依赖关系，以及整个开源生态的健康状况。这不仅有助于提升软件的开发效率，也为开源项目的可持续发展提供数据支持和决策依据。基于开源API图谱还能实现相似开源软件的精准识别与推荐，助力开发者高效发掘适用的开源资源，推动开源生态的蓬勃发展。

1.2 项目提出目的

本项目旨在响应当前开源软件复杂性日益提升与智能化分析需求增长的双重趋势，构建一个融合意图识别、自动执行与知识工程能力的通用大模型平台（LLM能力平台），以系统性地理解和分析开源生态。在此平台基础上，项目将开发一套基于LLM的 API 图谱分析工具，自动解析海量开源代码中的接口调用关系、参数特征与语义功能，并将其结构化地沉淀到统一的图谱知识库中。该图谱不仅提升了开源软件的可观测性与可理解性，也为质量评估与研究分析提供了强有力的支撑数据。

项目还将面向用户提供智能化的开源软件生态评估服务，利用大模型的知识推理能力结合图谱数据，生成个性化的软件推荐、影响力评估及依赖风险分析报告，从而赋能开发者、研究人员与企业 IT 决策者更科学地选择、使用和管理开源资源，推动 OSS 生态的可持续健康发展。

1.3 项目产出要求

1. API 图谱数据集：构建涵盖多种主流编程语言的开源软件 API 图谱数据集，数据集包含开源项目信息、API 调用关系、API 功能描述及相关元数据，为后续分析与应用筑牢数据根基。
2. 图谱可视化与应用平台：开发基于 Web 的图谱可视化与应用平台，用户可在平台上直观查看开源软件的 API 图谱，进行节点搜索、关系查询、图谱缩放等操作。同时，平台支持相似开源软件的识别与推荐，以及开源软件生态健康状况的可视化展示。
3. 生态评估与应用报告：基于图谱分析结果，撰写开源软件生态评估与应用报告，内容包括开源软件生态健康状况评估、关键项目与开发者识别、相似开源软件推荐策略，以及对未来发展趋势的预测。报告需提供数据支撑与案例分析，为开源软件生态的优化提供参考。

1.4 项目技术要求

1. 技术能力：实习生需熟练掌握 Python 编程语言，具备数据采集、处理与分析能力，了解图数据库和知识图谱的基本概念与应用。熟悉 Web 开发技术，如 Flask、Django 等，能够开发简单 Web 应用。掌握机器学习和自然语言处理基础知识，具备算法实现与调优能力者优先。
2. 沟通协作：具备良好的沟通能力与团队协作精神，能与团队成员和导师保持密切沟通，及时汇报项目进展。积极参与团队讨论，分享想法与经验，共同解决项目难题。
3. 学习能力：拥有较强的学习能力与自我驱动力，能够快速掌握项目所需的新技术与工具。在实习过程中，持续探索与尝试新方法，提升技术水平与问题解决能力。

1.5 项目相关

本项目依托于`OSS-Compass`社区开源项目，该项目由开源社区共同推动，旨在构建面向开源生态的可观测性平台，已在开源软件质量评估、健康度分析、开发者行为建模等方面形成初步成果。作为一个活跃的开源协作项目，OSS-Compass 聚集了来自多个高校、研究机构和企业的贡献者，具备良好的协作基础和技术积累。

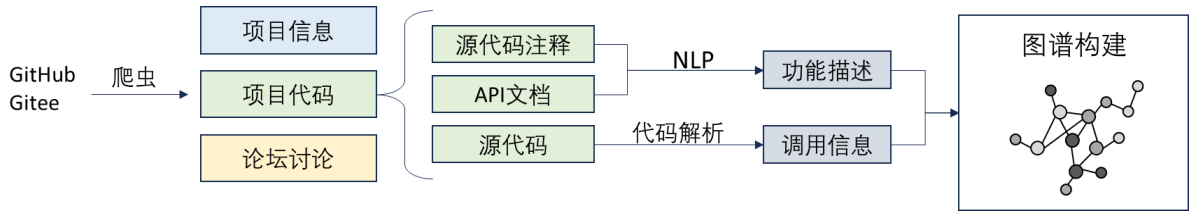
当前，OSS-Compass 已搭建了以数据指标驱动的评估体系，并构建了包括仓库活跃度、贡献者结构、社区响应性等多维度指标的评估模型，为本项目提供了宝贵的数据源、架构支持与用户基础。通过本项目的推动，我们将进一步拓展 OSS-Compass 的技术边界，引入通用大语言模型（LLM）、知识图谱与语义解析等智能化手段，补充其在 API 层级理解与语义分析方面的能力短板。

此外，本项目与 OSS-Compass 在理念上保持一致，即以开源方式促进开源生态可持续发展，拟继续采用开放协作的开发模式，持续向社区贡献核心组件、图谱数据与评估能力，推动平台与生态共建共赢。

---

2 项目实现方案

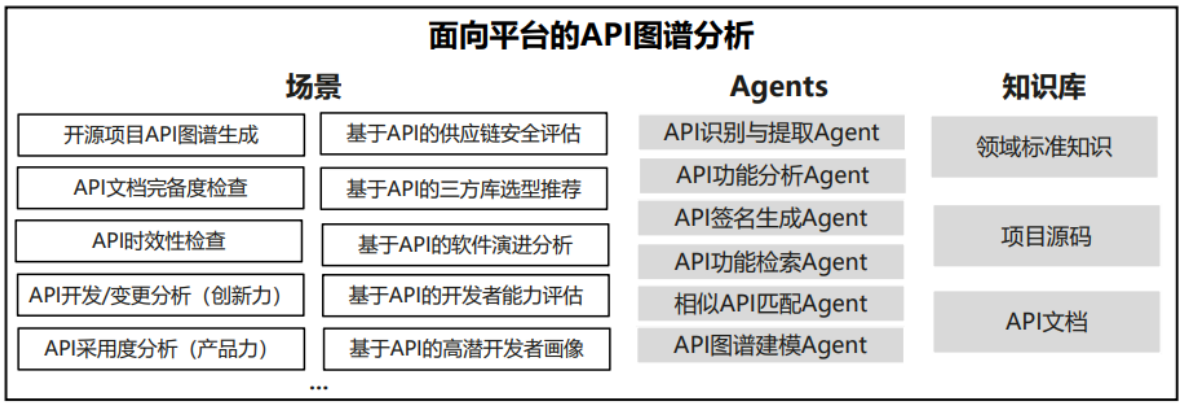
项目主体开发如下



在完成图谱构建之后，我们可以充分利用获取到的API图谱和相关信息，再充分运用搭建的通用LLM工程平台进行Agents构造，实现以下类型的Agents: 1.API识别与提取；2. API功能分析; 3.API签名生成; 4. API功能检索; 5.相似API匹配; 6.API图谱建模；

本次项目可扩展的场景如下：1. API文档完备度检查；2. API时效性检查; 3.API开发/变更分析; 4. API采用度分析

综上总结如下：

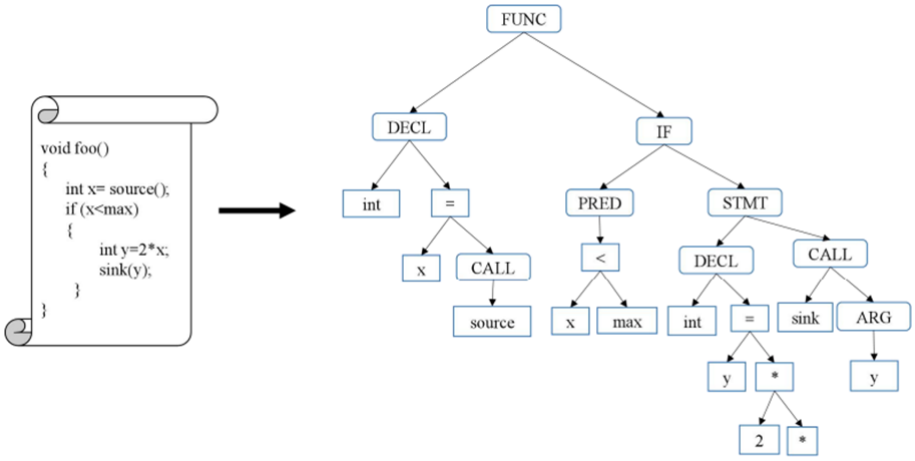


2.1 相关技术

2.1.1 数据采集与预处理

为实现对开源软件生态的全面建模，本项目将从三类关键数据源进行采集与建模：**项目源代码（Source Code）、API 文档（API Documentation）与领域知识标准（Domain Knowledge Standards）**，构建具有语义信息的 API 图谱。

- **项目源代码层**：通过设计高效的 GitHub/GitLab 爬虫系统，自动化采集主流开源项目的代码仓库，包括函数定义、类结构、模块导入、依赖路径等结构化信息。对不同语言的源文件，采用语言识别器进行自动分类，并利用语法分析工具（如 Python 的 `ast`、Java 的 `Eclipse JDT` 或 `Spoon`）构建抽象语法树（AST），提取函数签名、参数列表、返回类型及内部调用关系，形成调用链初步图谱。



- **API 文档层**：系统将自动抓取与解析项目提供的 README、文档网站（如 Sphinx、Javadoc、Typedoc）及内联注释（docstring），并结合自然语言处理技术（如依存句法分析、关键词抽取、句向量嵌入）提炼 API 的功能描述、适用场景、使用示例等语义信息。此语义层为后续功能理解与推荐系统提供关键基础。
- **领域标准知识层**：结合如 OpenAPI/Swagger 规范、JSON Schema、Programming Language Specification、软件架构文献及功能标签体系（如 NLP、ML、Auth、Logging 等），引入领域知识图谱与术语体系，用于对不同 API 的语义归类、标签标准化与功能对齐，从而提升图谱的通用性与语义完整度。

多源数据经统一预处理流程（编码统一、冗余去除、字段补全）与标准化建模，将被导入图谱构建模块，并在图数据库中映射为包含调用路径、功能语义、项目归属、标准标签等多属性的复合图结构，为 API 的理解、推荐与生态评估奠定基础。

2.1.2 API抽取与功能理解

在获取源代码语法结构的基础上，本项目将利用 `tree-sitter` 将自动识别函数、类、模块及其相互调用关系，从而提取 API 级别的调用图。针对不同语言的编程特性，采用定制化抽取策略，如静态分析与符号解析，对接口的名称、参数类型、返回值及依赖关系进行结构化表达。

为增强 API 的语义可解释性，项目将进一步引入自然语言处理技术，从代码注释、函数签名、项目文档中提取 API 的功能描述，结合上下文信息进行语义归一化处理，实现对 API 功能的准确理解与标签化，为图谱的知识化构建提供语义支撑。

本申请书将以https://github.com/oss-compass/openchecker开源库为例，提取代码里涉及的API，初步整理成json格式。部分如下

```
{
  "./openchecker/main.py": [
    ["requests.get", 42],
    ["json.loads", 45],
    ["logger.info", 50]
  ],
  "./openchecker/utils/helper.py": [
    ["os.path.join", 10],
    ["re.match", 20]
  ]
}
```

2.1.3 图谱构建技术

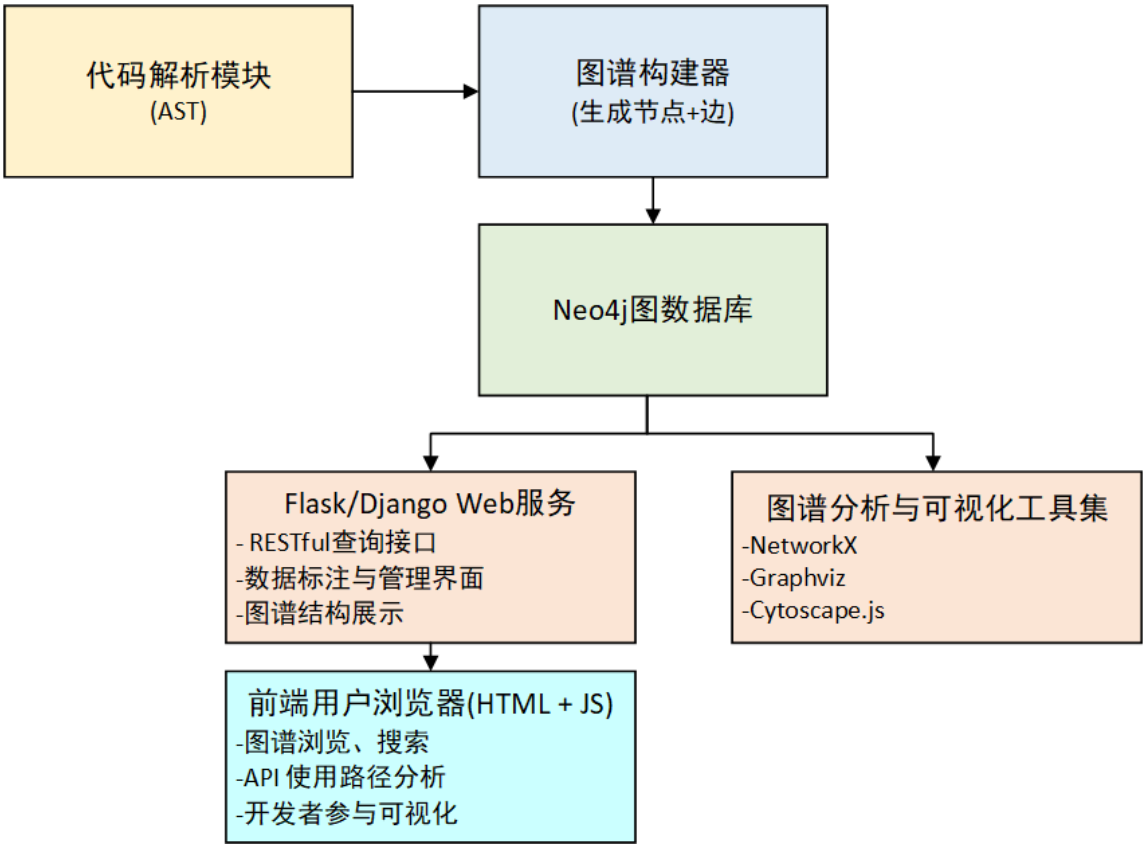
本项目基于提取的 API 元信息与调用关系，采用 Neo4j 图数据库对开源软件生态进行图谱建模。图谱中的节点包括多类实体，如开源项目、模块、API 接口、开发者、依赖库等，边则表示调用关系、版本依赖、贡献路径、语义相似性等多维连接。

为了增强图谱的语义表达与扩展能力，每个节点将通过属性标注记录功能标签、所属语言、项目来源、版本历史、调用频率等关键元信息。这些结构性与语义性信息将支持后续的图挖掘、影响分析和相似性计算。

在系统架构层面，将使用 Flask（或 Django）构建 Web 服务，负责提供：

- 图谱数据的 API 接口；
- 图谱的查询与可视化交互；
- 图谱更新与调度控制面板。

可视化层将结合 NetworkX 与 Graphviz 进行静态结构图生成，同时引入 Cytoscape.js 实现浏览器端的交互式图谱探索。



2.1.4 自然语言处理与知识增强

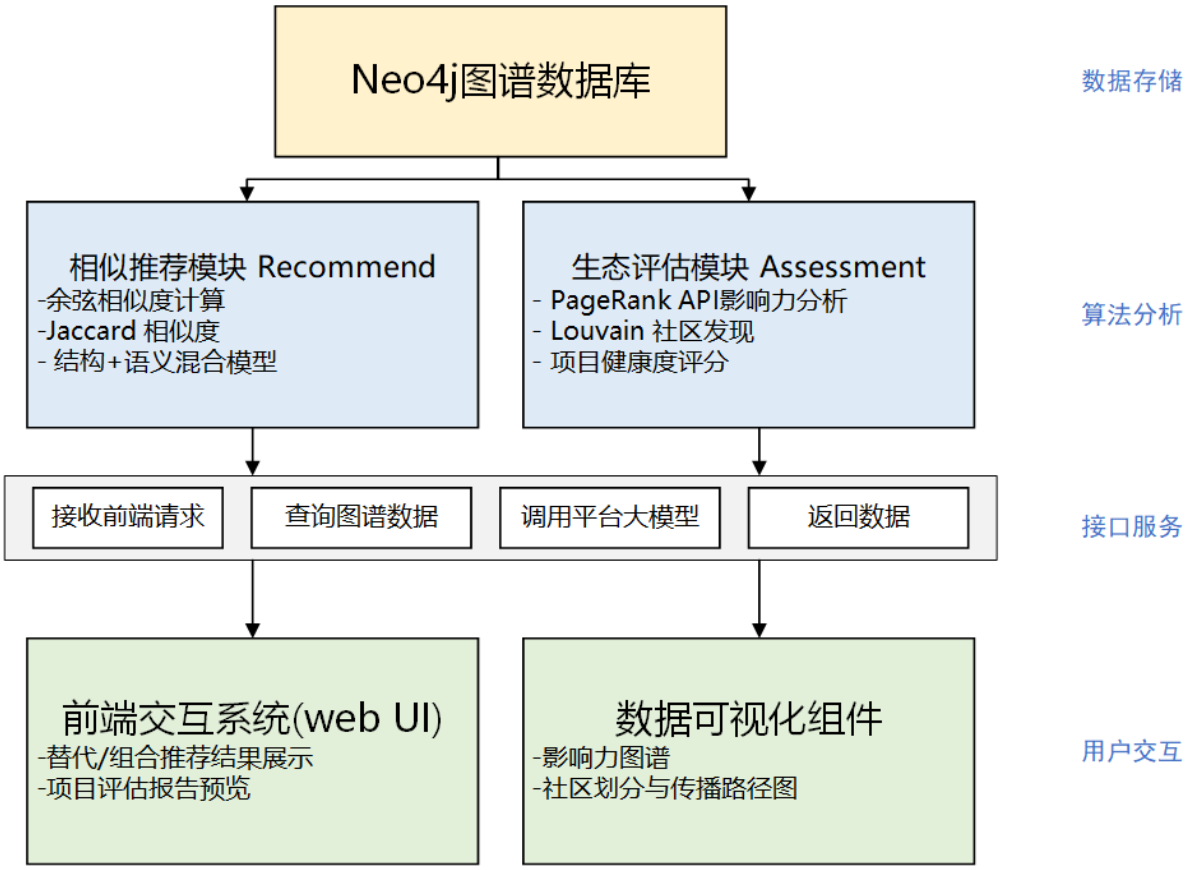
为进一步提升图谱的知识表达能力与推理价值，本项目将引入自然语言处理（NLP）与知识增强机制，对 API 文本数据进行深入挖掘。首先，对 API 的功能描述进行文本分类与语义聚类，生成统一的功能标签体系；其次，对开源项目的说明文档、社区讨论、Issue 等文本数据进行情感分析与观点抽取，辅助构建开发者对项目的主观反馈画像。

此外，将结合预训练语言模型（如 BERT、GPT）进行知识补全与功能关系识别，从而在图谱中加入“相似API”“扩展功能”“同源项目”等语义连接，增强图谱的知识联想与可拓展性。

2.1.5 推荐与评估模型

图谱构建完成后，将在其上部署多种推荐与评估算法，以支持面向用户的智能服务。在相似推荐方面，将基于 API 调用模式、参数结构、语义标签等维度，计算余弦相似度、Jaccard 系数，辅助识别功能重合或结构相近的开源组件，实现替代推荐与组合推荐。

在生态评估方面，项目将借助 PageRank 算法度量 API 节点的影响力，结合社区发现算法（如 Louvain、Infomap）识别开源项目之间的协作网络与功能社群。多种维度的分析结果将综合生成质量评价报告，涵盖依赖稳定性、更新频率、社区活跃度、风险传播性等指标，全面反映开源项目的生态健康度与可信度。



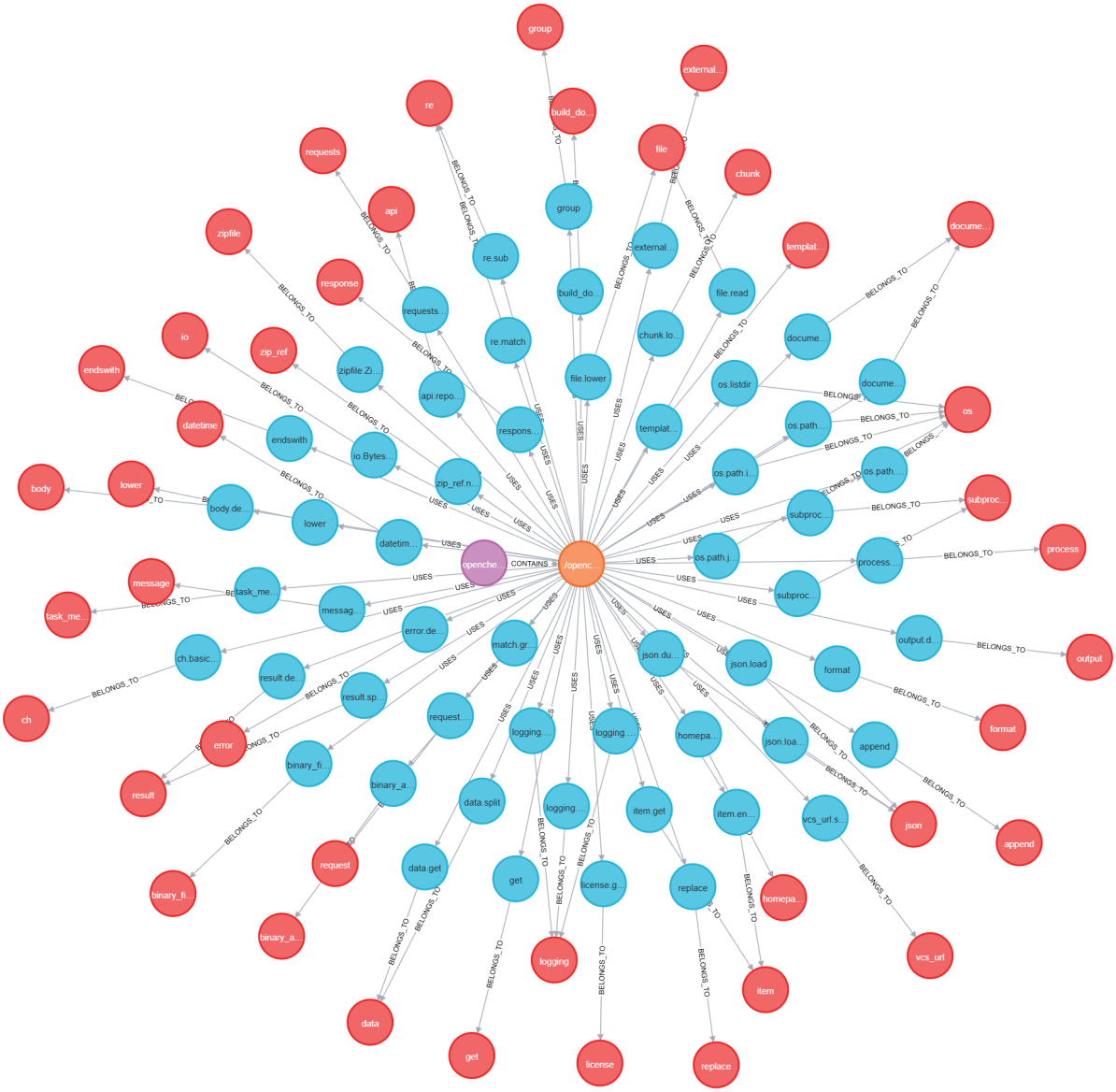
2.2 技术细节

2.2.1 API图谱数据模型设计

本项目的图谱数据模型将以多实体、多关系的异构图结构构建，具备可扩展的节点和边类型定义。节点（Node）包括但不限于：开源项目、API 接口、函数、模块、开发者、第三方库、功能标签、语义类别等。边（Edge）类型涵盖：调用关系、依赖关系、版本演化、贡献关系、功能相似、语义归属等。

每个节点将包含丰富的属性集合，如 API 节点包含名称、参数、返回值、文档描述、所属项目、调用频次、语言类型等；开发者节点包含用户名、历史提交、参与项目等属性。图结构设计将满足后续图挖掘、相似性计算和语义分析等多种应用需求。

初步生成的API图谱如下所示，以本项目仓库所涉及的代码为例




```
{
  "nodes": [
    {
      "id": "api:requests.get",
      "type": "API",
      "attributes": {
        "name": "requests.get",
        "params": ["url", "params", "headers"],
        "returns": "Response",
        "description": "Sends a GET request.",
        "project": "openchecker",
        "call_frequency": 18,
        "language": "Python"
      }
    },
    {
      "id": "dev:alice",
      "type": "Developer",
      "attributes": {
        "username": "alice",
        "commits": 124,
```

```
        "projects": ["openchecker", "another_project"]
      }
    },
    {
      "id": "project:openchecker",
      "type": "Project",
      "attributes": {
        "name": "openchecker",
        "language": "Python",
        "license": "Apache-2.0",
        "stars": 350
      }
    }
  ],
  "edges": [
    {
      "source": "project:openchecker",
      "target": "api:requests.get",
      "type": "调用关系",
      "attributes": {
        "times": 18,
        "locations": ["main.py:42", "utils.py:88"]
      }
    },
    {
      "source": "dev:alice",
      "target": "project:openchecker",
      "type": "贡献关系",
      "attributes": {
        "commits": 72,
        "roles": ["developer", "reviewer"]
      }
    }
  ]
}
```


2.2.2 图数据库搭建与优化

项目将采用 Neo4j 图数据库作为核心存储与查询引擎。通过设计自动化的数据导入管道，将提取的 API 及其关系结构以 Cypher 语言或 `neo4j-admin import` 工具批量导入。为提高查询效率，将对常用查询路径建立索引，如基于项目名、功能标签、调用路径等字段的快速定位。

图遍历优化方面，将采用图分区策略和缓存机制，缩短复杂路径查询（如跨模块调用链）时的响应时间。此外，提供 RESTful API 与 Cypher 查询接口的二次封装，支持 Web 服务、评估引擎与可视化模块进行高效交互，确保整个系统的响应性能与并发访问能力。

2.2.3 多语言支持与兼容性处理

由于开源生态涵盖多种主流语言，项目将针对不同语言构建差异化的 API 提取与解析模块。例如：

- **Python** 使用 `ast`、`inspect` 模块进行函数结构解析和注释抽取；
- **Java** 利用反射机制与基于源码的语法分析器（如 Eclipse JDT）提取类与方法信息；

- **JavaScript/TypeScript** 结合 Babel 或 TypeScript 编译器 API，支持复杂模块化结构与异步调用分析。

所有语言模块将通过统一的接口规范输出标准化 API 实体，确保图谱数据结构一致性，便于后续知识融合和图数据库统一接入。

2.2.4 相似度计算与推荐逻辑

为支持 API 层级和项目层级的推荐功能，项目将集成多种相似度计算与推荐算法：

- **余弦相似度（Cosine Similarity）**：基于 API 的调用向量、功能词嵌入或语义向量，度量相似 API 的距离。
- **Jaccard 系数**：基于调用集合、参数集或依赖库集合评估两个 API 的交集占比。
- **K-近邻推荐模型（KNN）**：基于图结构、行为路径或嵌入空间，为用户推荐相似项目或替代 API。
- **图嵌入算法（如 Node2Vec）**：将 API 节点向量化，结合语义信息实现深度相似度计算。

推荐逻辑将根据上下文动态调整，如用户当前使用语言、项目类型、功能偏好等，以提供个性化、可解释的推荐结果。

2.2.5 情感与反馈分析引擎

为了增强对开源软件质量与社区活跃度的认知，本项目将设计情感与反馈分析引擎，自动采集并分析以下数据源：

- **项目讨论（GitHub Issues/Discussions）**
- **开发者评论与 Pull Request 记录**
- **开源社区论坛（如 Stack Overflow）**
- **文档与 README 中的用户语句**

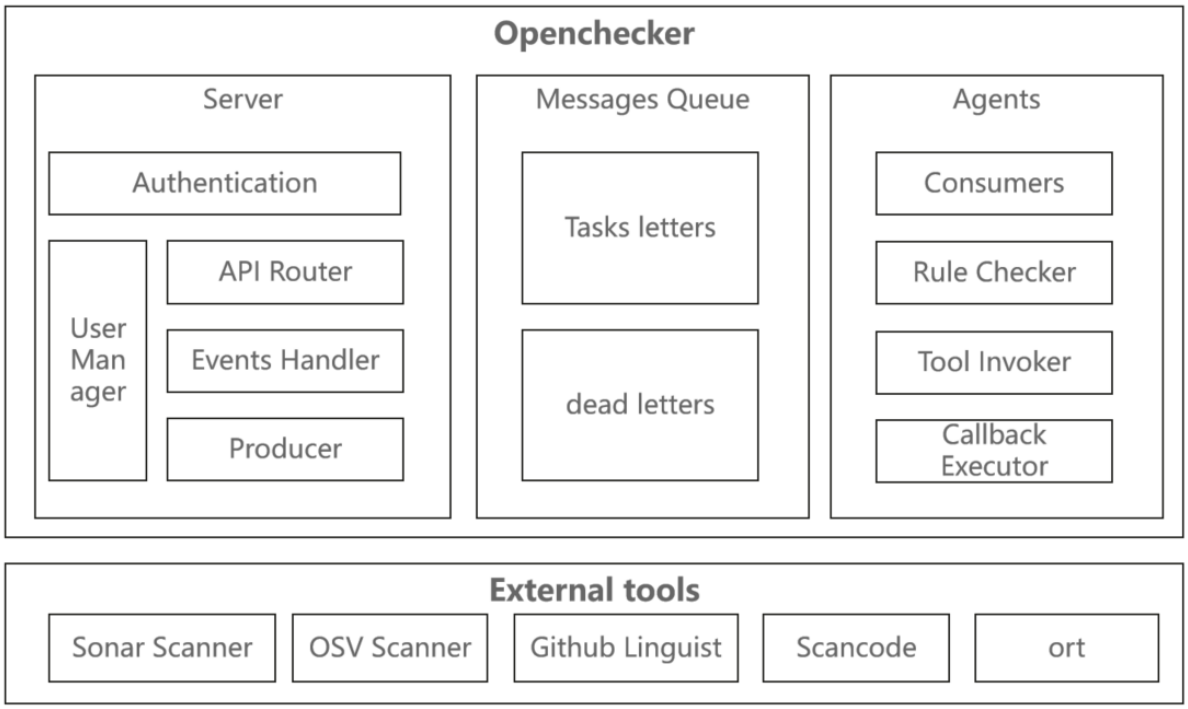
通过使用情感分类模型（如 BERT + Fine-tuned Sentiment Classifier）分析文本的情绪倾向，结合关键词抽取与实体识别技术，提取用户对特定功能、API 或项目的正面或负面意见。结果将反馈至图谱属性，辅助质量评估与信任机制设计。

2.2.6 可视化界面与交互平台设计

项目将开发基于 Web 的图谱交互平台，支持用户通过可视化方式探索 API 调用结构、项目依赖网络和社区分布图。界面功能设计如下：

- **图谱浏览与节点搜索**：支持按项目、功能、语言等条件筛选与聚焦；
- **相似项目推荐可视化**：动态展示推荐路径与相似性指标；
- **生态指标仪表板**：图表化呈现影响力、活跃度、更新频率等评估结果；
- **用户自定义查询界面**：集成 Cypher 编辑器与图形查询辅助工具。

平台将结合前端技术（如 React、D3.js）与图数据库可视化工具（如 Neo4j Bloom），实现高交互性与良好用户体验。同时，系统架构将支持基于角色与权限的扩展机制，便于后续集成多角色访问控制与数据管理功能。



3 项目规划

3.1 第一阶段（7月1日 – 8月25日）

- 第1-2周：需求确认与系统框架设计；
- 第3-4周：完成数据采集模块和代码解析工具；
- 第5-6周：搭建初步 API 抽取逻辑和语义分析机制；
- 第7-8周：图谱初始建模与 Neo4j 数据结构部署；
- 第9周：开发消息中间件结构与任务生产者模块；
- 第10周：实现部分算法逻辑（相似度计算、影响力评估）与模型验证。

阶段性目标交付：

- 完成数据底座搭建、图谱建模框架；
- 发布初版图谱结构样本与 API 抽取工具。

3.2 第二阶段（8月26日 – 9月30日）

- 第11-12周：开发图谱交互 UI 与用户查询接口；
- 第13周：完成评估算法集成与结果可视化仪表板；
- 第14周：构建外部工具集成模块与 Callback 支持；
- 第15周：完成系统测试、部署上线与文档撰写；
- 第16周：整理阶段成果，发布开源版本并准备社区推广。

最终交付成果：

- 全流程可用的 API 图谱分析平台；
- 支持自定义查询、相似推荐与生态评估；
- 完整文档、样例图谱与用户服务指南。