可泛化知识计算引擎接口文档

V1.0

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 发布日期 | 修改人 | 修改说明 |
| V0.1 | 2020/01/14 |  | 接口初步定义 |
| V0.2 | 2020/01/17 |  | 接口讨论后同步 |
| V1.0 | 2020/03/26 |  | 各课题结构整合 |

目录

[1. 接口说明 4](#_Toc37157457)

[1.1. 接口描述说明 4](#_Toc37157458)

[1.2. 接口风格规范 5](#_Toc37157459)

[1.3. 接口安全机制 5](#_Toc37157460)

[1.4. 接口返回码说明 6](#_Toc37157461)

[2. 接口规范 6](#_Toc37157462)

[2.1. 大数据平台微服务接口 6](#_Toc37157463)

[2.1.1. 大数据获取 6](#_Toc37157464)

[ 在线少量数据查询 6](#_Toc37157465)

[ 离线批量数据获取 7](#_Toc37157466)

[2.2. 动态概念体系构建接口 7](#_Toc37157467)

[2.2.1. 领域概念获取与层次划分接口 8](#_Toc37157468)

[ 实体概念获取 8](#_Toc37157469)

[ 概念上下位关系获取 9](#_Toc37157470)

[ 实体-关系-实体三元组关系获取 10](#_Toc37157471)

[2.2.2. 概念补全接口 11](#_Toc37157472)

[2.3. 多模态实体识别和链接接口 12](#_Toc37157473)

[2.3.1. 视觉实体识别和链接接口 12](#_Toc37157474)

[ 视觉实体识别模型训练 12](#_Toc37157475)

[ 视觉实体识别模型调用 12](#_Toc37157476)

[ 视觉实体链接模型训练 13](#_Toc37157477)

[ 视觉实体链接模型调用 14](#_Toc37157478)

[2.3.2. 文本实体识别和链接接口 15](#_Toc37157479)

[ 文本实体识别模型训练 15](#_Toc37157480)

[ 文本实体识别模型调用 15](#_Toc37157481)

[ 文本实体链接模型训练 16](#_Toc37157482)

[ 文本实体链接模型调用 17](#_Toc37157483)

[2.4. 多模态事件识别和链接接口 18](#_Toc37157484)

[2.4.1. 视觉事件识别和链接接口 18](#_Toc37157485)

[ 视觉事件识别模型训练 18](#_Toc37157486)

[ 视觉事件识别模型调用 18](#_Toc37157487)

[ 视觉事件链接模型训练 19](#_Toc37157488)

[ 视觉实体链接模型调用 20](#_Toc37157489)

[2.4.2. 文本概念事件挖掘接口 20](#_Toc37157490)

[ 事件触发词获取 20](#_Toc37157491)

[ 事件元素获取 21](#_Toc37157492)

[2.5. 属性填充和关系抽取接口 22](#_Toc37157493)

[2.5.1. 属性填充接口 22](#_Toc37157494)

[ 文本属性填充模型训练 22](#_Toc37157495)

[ 文本属性填充模型调用 23](#_Toc37157496)

[2.5.2. 关系抽取接口 24](#_Toc37157497)

[ 文本关系抽取模型训练 24](#_Toc37157498)

[ 文本关系抽取模型调用 24](#_Toc37157499)

[2.6. 知识提及读写接口 26](#_Toc37157500)

[2.6.1. 提及读取 26](#_Toc37157501)

[2.6.2. 提及写入 27](#_Toc37157502)

[2.7. 知识图谱设计构建接口 28](#_Toc37157503)

[2.7.1. 类相关接口 28](#_Toc37157504)

[ 获取实体类 28](#_Toc37157505)

[ 获取事件类 29](#_Toc37157506)

[ 获取关系类 30](#_Toc37157507)

[ 添加实体类 30](#_Toc37157508)

[ 添加事件类 32](#_Toc37157509)

[ 添加关系类 33](#_Toc37157510)

[ 导入实体类 34](#_Toc37157511)

[ 导入事件类 35](#_Toc37157512)

[ 导入关系类 36](#_Toc37157513)

[ 删除实体类 37](#_Toc37157514)

[ 删除事件类 38](#_Toc37157515)

[ 删除关系类 38](#_Toc37157516)

[2.7.2. 图相关接口 39](#_Toc37157517)

[ 获取所有图schema 39](#_Toc37157518)

[ 创建图 40](#_Toc37157519)

[ 获取某一图schema 41](#_Toc37157520)

[ 删除图 42](#_Toc37157521)

[2.8. 知识图谱访问计算接口 42](#_Toc37157522)

[2.8.1. 实例访问接口 43](#_Toc37157523)

[ 写入实体实例 43](#_Toc37157524)

[ 写入事件实例 44](#_Toc37157525)

[ 写入关系实例 44](#_Toc37157526)

[ 获取实体实例 45](#_Toc37157527)

[ 获取事件实例 46](#_Toc37157528)

[ 获取关系实例 47](#_Toc37157529)

[ 获取实体实例集合（QL） 48](#_Toc37157530)

[ 获取事件实例集合（QL） 49](#_Toc37157531)

[ 获取关系实例集合（QL） 49](#_Toc37157532)

[ 更新实体实例 50](#_Toc37157533)

[ 更新事件实例 51](#_Toc37157534)

[ 更新关系实例 52](#_Toc37157535)

[ 更新实体实例集合（QL） 53](#_Toc37157536)

[ 更新事件实例集合（QL） 54](#_Toc37157537)

[ 更新关系实例集合（QL） 55](#_Toc37157538)

[ 删除实体实例 56](#_Toc37157539)

[ 删除事件实例 57](#_Toc37157540)

[ 删除关系实例 58](#_Toc37157541)

[ 删除实体实例集合（QL） 59](#_Toc37157542)

[ 删除事件实例集合（QL） 60](#_Toc37157543)

[ 删除关系实例集合（QL） 60](#_Toc37157544)

[2.8.2. 实体链接接口 61](#_Toc37157545)

[ 实体消歧 61](#_Toc37157546)

[ 实体融合 62](#_Toc37157547)

[ 事件链接 63](#_Toc37157548)

[2.8.3. 图算法接口 64](#_Toc37157549)

[ 图的遍历 64](#_Toc37157550)

[ 路径检索（升级子图相关性排序） 65](#_Toc37157551)

[ 获取子图 67](#_Toc37157552)

[ 计算实例的关系数 68](#_Toc37157553)

[ 获取实例的邻居 69](#_Toc37157554)

[ 获取实例之间的共同邻居 70](#_Toc37157555)

[ 获取实例之间的最短路径 71](#_Toc37157556)

[ 计算实例的接近中心性 72](#_Toc37157557)

[ 计算实例的中介中心性 73](#_Toc37157558)

[2.9. 图表示计算与推理接口 74](#_Toc37157559)

[2.9.1. 图定义接口 74](#_Toc37157560)

[ 图定义 74](#_Toc37157561)

[ 子图定义 75](#_Toc37157562)

[ 异构图定义 75](#_Toc37157563)

[ 图封装 76](#_Toc37157564)

[ 异构图封装 76](#_Toc37157565)

[2.9.2. 图数据导入接口 76](#_Toc37157566)

[ 特征导入 77](#_Toc37157567)

[ 邻接矩阵导入 77](#_Toc37157568)

[2.9.3. 图神经网络表征接口 77](#_Toc37157569)

[ 图卷积神经网络GCN 77](#_Toc37157570)

[ 图注意力神经网络GAT 78](#_Toc37157571)

[ 图向量表征模型 78](#_Toc37157572)

[ 图节点向量表征获取 79](#_Toc37157573)

[ 图关系向量表征获取 79](#_Toc37157574)

[ 图表征局部动态更新 79](#_Toc37157575)

[ 图表征全局更新 80](#_Toc37157576)

[2.9.4. 图推理应用接口 80](#_Toc37157577)

[ 实体或事件分类模型训练 80](#_Toc37157578)

[ 实体或事件分类 80](#_Toc37157579)

[ 相似实体或事件搜索 81](#_Toc37157580)

[ 实体关系预测模型训练 82](#_Toc37157581)

[ 实体关系预测 83](#_Toc37157582)

[ 子图模式搜索 84](#_Toc37157583)

[ 图的聚类 85](#_Toc37157584)

[ 主动学习包装器 86](#_Toc37157585)

[ 主动学习模型接收新数据 86](#_Toc37157586)

[ 主动学习模型请求新数据 87](#_Toc37157587)

[ 导出当前学习模型 87](#_Toc37157588)

[2.10. 人机交互接口 87](#_Toc37157589)

# 接口说明

## 接口描述说明

整体接口设计按照资源层级展开，分为对底层大数据资源（包括结构化数据和非结构化数据）的获取访问、对过渡层提及数据资源的读写、对中间层知识资源（如类、实体、事件、关系）的管理操作、对上层知识计算推理任务的执行等。

|  |  |  |
| --- | --- | --- |
| **层级序号** | **接口层级** | **涉及方法** |
| **1** | **多媒体大数据获取接口** | **读取** |
| **2** | **跨媒体信息抽取接口** | **抽取** |
| **3** | **知识提及读写接口** | **读取、写入** |
| **4** | **知识图谱构建接口** | **读取、写入、生成、修改、删除、匹配** |
| **5** | **图谱数据管理接口** | **读取、写入、修改、删除、计算** |
| **6** | **图表征计算接口** | **读取、写入、生成、计算** |
| **7** | **计算图推理接口** | **分类、预测、聚类、计算** |
| **8** | **人机交互接口** |  |

## 接口风格规范

接口提供两种格式，一是HTTP请求接口，可按需求遵循RESTful风格或GraphQL规范，请求方式根据对资源的操作类型可提供GET、POST、PUT、PATCH、DELETE等，请求的参数和响应的数据均为JSON格式；二是编程接口，可根据开发者用户的需要支持主流编程语言如Java和Python（以下示例以Java说明），提供面向多层级数据对象的操作方法，其底层仍然以HTTP接口为基础进行封装。

## 接口安全机制

1. 分配接口请求的token作为请求参数，验证通过才可返回数据；
2. 要求请求附带发起服务器公网IP地址，以进行白名单验证；
3. 接口请求日志完整保留，定期进行检查并对异常访问加以限制。

## 接口返回码说明

|  |  |
| --- | --- |
| 代码 | 信息 |
| 200 | 成功 |
| 201 | 成功并创建 |
| 304 | 使用缓存数据 |
| 400 | 请求格式错误 |
| 403 | 请求权限不足 |
| 404 | 资源未被找到 |
| 409 | 插入资源冲突 |
| 500 | 服务器内部错误 |
| 501 | 方法尚未被实现 |

# 接口规范

## 大数据平台微服务接口

说明：

结构化与非结构化大数据存储方案由开源大数据平台或分布式数据库提供，访问方式以数据的获取操作为主，可对多种大数据计算框架查询语言和数据库查询语言进行封装，统一数据资源暴露方式，返回文本、图像、视频、语音、网页等多种类型的文档。

### 大数据获取

#### 在线少量数据查询

* HTTP接口（少量数据查询）

|  |  |
| --- | --- |
| 功能 | 以多种Query Language格式获取满足一定条件的限定数量的数据集合，可按需支持如MySQL、Mongodb、ES、HBase、Hive、Spark SQL等。 |
| 方法 | HTTP POST /\_api/ckc/documents/query |
| 参数 | 无 |
| Body | {  “query:”: “db.collection.find({},{content:1})”,  “source”: {  “type”: “mongodb”,  “url”: “mongodb connection url address”  },  “limit”: 1000  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {  “type”: “text”,  “content”: “text content”  }  ]  “status”: “document retrieved”  }  } |

#### 离线批量数据获取

* 编程接口（批量数据获取）

|  |  |
| --- | --- |
| 功能 | 以多种Query Language格式获取满足一定条件的数据集合，可按需支持如MySQL、Mongodb、ES、HBase、Hive、Spark SQL等。 |
| 方法 | CKCApi.getDocuments(String sql, DBSource source, int limit) |
| 参数 | sql – 查询语句字符串  source – 数据源来源配置实例  limit – 返回数据实例数量 |
| 返回 | 包含多条文档对象的集合  Collection<CKCDocument<String, Object>>  CKCDocument子类：CKCTextDocument, CKCImageDocument, CKCVideoDocument, CKCAudioDocument, CKCWebDocument |
| 异常 | CKCApiException |

## 动态概念体系构建接口

说明：根据输入的实体能够动态的获取实体的概念类型，同时自动将概念间的上下位关系体系构建出来。

### 领域概念获取与层次划分接口

#### 实体概念获取

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段文本，调用概念挖掘模型获取实体概念 |
| 方法 | HTTP POST /\_api/ckc/ner/{modelPath} |
| 参数 | 无 |
| Body | {  NER1,  NER2,  ……..  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {NER1:CON1;  NER2:CON2;  ……:……;}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 从网络中获取实体的概念。 |
| 方法 | CKCApi.exConcept(String nerlist, String filePath, String fileType，String encoding) |
| 参数 | nerlist - 获取概念的实体列表  filePath – 数据来源  fileType – 文件格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  enconceptlist – 实体概念列表  modelPath – 模型路径 |
| 异常 | CKCApiException |

NERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| nerlist | string | 是 | 实体列表 |

#### 概念上下位关系获取

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传概念列表，调用上下位关系模型获取概念间的上下位关系 |
| 方法 | HTTP POST /\_api/ckc/ner/{modelPath} |
| 参数 | 无 |
| Body | {  CON1,  CON2,  ……..  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {CON1:CONXX;  CON2:CONXX;  ……:……;}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入概念列表，获取概念中的上下位关系 |
| 方法 | CKCApi.exHyper(String conceptlist, String fileType，String encoding) |
| 参数 | conceptlist– 概念列表  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的上下位关系元组  modelPath – 模型路径 |
| 异常 | CKCApiException |

NERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| conlist | string | 是 | 概念列表 |

#### 实体-关系-实体三元组关系获取

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传实体二元组列表，调用关系挖掘模型获取实体间的关系 |
| 方法 | HTTP POST /\_api/ckc/ner/{modelPath} |
| 参数 | 无 |
| Body | {  HNER1: ENER1;  HNER2: ENER2;  ……..;  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {HER1:RELATION1:ENER1;  HER2:RELATION2:ENER2;  ……:……:……;}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 挖掘实体间的关系 |
| 方法 | CKCApi.exRelation(String ner1, String ner2, String filePath, String fileType，String encoding) |
| 参数 | ner1 – 头实体  ner2 – 尾实体  filePath – 训练数据集访问路径  fileType – 文件格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的关系三元组  modelPath – 模型路径 |
| 异常 | CKCApiException |

NERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| conlist | string | 是 | 头实体、尾实体列表 |

### 概念补全接口

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传概念二元组列表，调用关系挖掘模型获取实体间的关系 |
| 方法 | HTTP POST /\_api/ckc/ner/{modelPath} |
| 参数 | 无 |
| Body | {  CON1: CON2;  ……..;  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {CON1:is-a:CON2;  ……:is-a:……;}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 确定概念间的上下位关系 |
| 方法 | CKCApi.completeHyper(String con1, String con2, String filePath, String fileType，String encoding) |
| 参数 | con1 – 概念1  con2 – 概念2  filePath – 训练数据集访问路径  fileType – 文件格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的概念三元组，以is-a作为概念间上下位关系标识符  modelPath – 模型路径 |
| 异常 | CKCApiException |

NERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| conlist | string | 是 | 概念1、概念2列表 |

## 多模态实体识别和链接接口

说明：

结合自顶向下和自底向上的实体抽取技术，实现一套开放式多模态实体定义、抽取软件接口。

### 视觉实体识别和链接接口

#### 视觉实体识别模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 根据训练数据集，训练视觉实体识别模型。 |
| 方法 | CKCApi.trainVisualNER(String modelType, String filePath) |
| 参数 | modelType – VNER模型类别  filePath – 训练数据集访问路径 |
| 返回 | success - 训练是否成功  modelPath – 训练得到模型的路径 |
| 异常 | CKCApiException |

#### 视觉实体识别模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一份视觉文件，调用VNER模型识别，返回识别结果 |
| 方法 | HTTP POST /\_api/ckc/vner/{modelPath} |
| 参数 | 无 |
| Body | {  “vis\_obj”:,  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {VNERAnnotationFormat}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一段视觉文件，调用VisualNER模型识别，返回识别结果 |
| 方法 | CKCApi.VisualNER(String modelPath, String modelType, String filePath) |
| 参数 | modelPath – VNER模型路径  modelType - VNER模型类别  filePath – 输入视觉文件路径 |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

VNERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| visPath | string | 是 | 视觉文件路径 |
| entities | array of objects | 是 | 词汇数组，每个元素对应结果中的一个词 |
| +entityType | Int | 是 | 命名实体类型编号 |
| …… | …… | …… | …… |

#### 视觉实体链接模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 根据训练数据集，训练视觉实体链接模型。 |
| 方法 | CKCApi.trainVisualEntityLinkingModel(String modelType, String filePath) |
| 参数 | modelType – 视觉实体链接模型类别  filePath – 训练数据集访问路径 |
| 返回 | success - 训练是否成功  modelPath – 训练得到的模型路径 |
| 异常 | CKCApiException |

#### 视觉实体链接模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段视觉文件，调用视觉实体链接模型，返回识别结果 |
| 方法 | HTTP POST /\_api/ckc/visualentitylinking/{modelPath} |
| 参数 | 无 |
| Body | {  “vis\_obj”:,  } |
|  | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {“mid”: 986546,  “eid”:4255},  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一份视觉文件，调用视觉实体链接模型，返回链接结果 |
| 方法 | CKCApi.visualEntityLinking(String modelPath, String filePath) |
| 参数 | modelPath – VisualNER 模型路径  filePath – 输入文本路径 |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

### 文本实体识别和链接接口

#### 文本实体识别模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 根据训练数据集，训练命名实体识别模型。 |
| 方法 | CKCApi.trainNER(String modelType, String filePath, String fileType，String encoding) |
| 参数 | modelType – NER模型类别  filePath – 训练数据集访问路径  fileType – 文件格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  modelPath – 训练得到模型的路径 |
| 异常 | CKCApiException |

#### 文本实体识别模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段文本，调用NER模型识别，返回识别结果 |
| 方法 | HTTP POST /\_api/ckc/ner/{modelPath} |
| 参数 | 无 |
| Body | {  “text”: “研发多模态、细粒度、特定领域、多层次知识图谱的自动构建方法……”,  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {NERAnnotationFormat}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一段文本，调用NER模型识别，返回识别结果 |
| 方法 | CKCApi.NER(String modelPath, String filePath, String fileType，String encoding) |
| 参数 | modelPath – NER模型路径  filePath – 输入文本路径  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

NERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| text | string | 是 | 原始文本 |
| entities | array of objects | 是 | 词汇数组，每个元素对应结果中的一个词 |
| +entityMention | string | 是 | 词汇的字符串 |
| +entityType | Int | 是 | 命名实体类型编号 |
| +byte\_offset | int | 是 | 在text中的字节级offset |
| +byte\_length | int | 是 | 字节级length |
| …… | …… | …… | …… |

#### 文本实体链接模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 根据训练数据集，训练文本实体链接模型。 |
| 方法 | CKCApi.trainEntityLinkingModel(String modelType, String filePath, String fileType，String encoding) |
| 参数 | modelType – 文本实体链接模型类别  filePath – 训练数据集访问路径  fileType – 文件格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  modelPath – 训练得到的模型路径 |
| 异常 | CKCApiException |

#### 文本实体链接模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段JSON格式文本，调用实体链接模型，返回识别结果 |
| 方法 | HTTP POST /\_api/ckc/entitylinking/{modelPath} |
| 参数 | 无 |
| Body | {  “text”: “After a standout at the University, Michael Jordan joined the Bulls in 1984.……”,  “entities”:[{  ”mid”: 986546,  “entityMention”: “Michael Jordan”,  “entityType”: 90,  ……  },  ……  ]  } |
|  | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {“mid”: 986546,  “eid”:4255},  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一段文本，调用文本实体链接模型，返回链接结果 |
| 方法 | CKCApi.entityLinking(String modelPath, String filePath, String fileType，String encoding) |
| 参数 | modelPath – NER模型路径  filePath – 输入文本路径  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

## 多模态事件识别和链接接口

说明：

结合自顶向下和自底向上的事件抽取技术，实现一套开放式多模态事件定义、抽取软件接口。

### 视觉事件识别和链接接口

#### 视觉事件识别模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 根据训练数据集，训练视觉事件识别模型。 |
| 方法 | CKCApi.trainVisualEventRecognition(String modelType, String filePath) |
| 参数 | modelType – VNER模型类别  filePath – 训练数据集访问路径 |
| 返回 | success - 训练是否成功  modelPath – 训练得到模型的路径 |
| 异常 | CKCApiException |

#### 视觉事件识别模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一份视觉文件，调用视觉事件模型识别，返回识别结果 |
| 方法 | HTTP POST /\_api/ckc/visualeventrecognition/{modelPath} |
| 参数 | 无 |
| Body | {  “vis\_obj”:,  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {VERAnnotationFormat}  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一段视觉文件，调用VisualEventRecognition模型识别，返回识别结果 |
| 方法 | CKCApi.visualEventRecognition(String modelPath, String modelType, String filePath) |
| 参数 | modelPath – VNER模型路径  modelType - VNER模型类别  filePath – 输入视觉文件路径 |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

VERAnnotationFormat JSON格式：

| 参数名称 | 类型 | 必需 | 详细说明 |
| --- | --- | --- | --- |
| visPath | string | 是 | 视觉文件路径 |
| events | array of objects | 是 | 词汇数组，每个元素对应结果中的一个词 |
| +eventType | Int | 是 | 事件触发词类型编号 |
| …… | …… | …… | …… |

#### 视觉事件链接模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 根据训练数据集，训练视觉事件链接模型。 |
| 方法 | CKCApi.trainVisualEntityLinkingModel(String modelType, String filePath) |
| 参数 | modelType – 视觉实体链接模型类别  filePath – 训练数据集访问路径 |
| 返回 | success - 训练是否成功  modelPath – 训练得到的模型路径 |
| 异常 | CKCApiException |

#### 视觉实体链接模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段视觉文件，调用视觉事件链接模型，返回识别结果 |
| 方法 | HTTP POST /\_api/ckc/visualeventlinking/{modelPath} |
| 参数 | 无 |
| Body | {  “vis\_obj”:,  } |
|  | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {“trigger”:,  “person”:,},  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一份视觉文件，调用视觉事件链接模型，返回链接结果 |
| 方法 | CKCApi.visualEventLinking(String modelPath, String filePath) |
| 参数 | modelPath – VisualNER 模型路径  filePath – 输入文本路径 |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

### 文本概念事件挖掘接口

#### 事件触发词获取

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段JSON格式文本，调用事件挖掘模型，返回事件触发词列表 |
| 方法 | HTTP POST /\_api/ckc/extracttrigger/{modelPath} |
| 参数 | 无 |
| Body | {  “text”: “After a standout at the University, Michael Jordan joined the Bulls in 1984.……”,  },  ……  ]  } |
|  | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {“trigger”: join},  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入文本片段，调用事件挖掘模型，返回事件触发词 |
| 方法 | CKCApi.exTrigger(String modelPath, String filePath, String fileType，String encoding) |
| 参数 | modelPath – 模型路径  filepath – 输入文本片段路径  filetype – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

#### 事件元素获取

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传一段JSON格式文本，调用事件挖掘模型，根据触发词，返回事件元素 |
| 方法 | HTTP POST /\_api/ckc/extractargument/{modelPath} |
| 参数 | 无 |
| Body | {  “text”: “After a standout at the University, Michael Jordan joined the Bulls in 1984.……”,  trigger: join;},  ……  ]  } |
|  | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {“trigger”: join  “person”: Michael Jordan  “location”: Bulls},  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入文本片段，调用事件挖掘模型，根据触发词，返回事件元素 |
| 方法 | CKCApi.exArgument(String modelPath, String eventword, String filePath, String fileType，String encoding) |
| 参数 | modelPath – 模型路径  eventword – 触发词  filepath – 输入文本片段路径  filetype – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

## 属性填充和关系抽取接口

说明：克服数据噪音大、信息不完全、带标数据少等挑战，实现一套文本属性填充和关系抽取软件接口，支撑构建领域知识图谱。

### 属性填充接口

#### 文本属性填充模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入带标文本，训练文本属性填充模型 |
| 方法 | CKCApi.trainSlotFilling(String modelType, String filePath, String fileType，String encoding) |
| 参数 | modelType – 属性填充模型类型  filePath – 输入文本路径  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  modelPath – 模型访问路径 |
| 异常 | CKCApiException |

#### 文本属性填充模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传JSON格式文本，调用文本属性填充模型，返回填充结果 |
| 方法 | HTTP POST /\_api/ckc/slotfilling/{modelPath} |
| 参数 | 无 |
| Body | {  “text”: “After a standout at the University, Michael Jordan joined the Bulls in 1984.……”,  “entityTypes”:[{  “entityType”: 90  “slots”:[{  “sid”:19011  },{  “sid”:90011  }……  ]},  ……  ]  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [{  “left\_mid”: 986546,  “right\_mid”: 999999,  “sid”:19011  },{  },  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一段文本，调用文本属性填充模型识别，返回填充结果 |
| 方法 | CKCApi.slotFilling(String modelPath, String filePath, String fileType，String encoding) |
| 参数 | modelPath – 文本属性填充模型路径  filePath – 输入文本路径  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

### 关系抽取接口

#### 文本关系抽取模型训练

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入带标文本，训练关系抽取模型 |
| 方法 | CKCApi.trainRelationExtractionModel(String modelType, String filePath, String fileType，String encoding) |
| 参数 | modelType – 文本关系抽取模型类型  filePath – 输入文本路径  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  modelPath – 训练得到模型的路径 |
| 异常 | CKCApiException |

#### 文本关系抽取模型调用

* HTTP接口：

|  |  |
| --- | --- |
| 功能 | 上传JSON格式文本，调用文本关系抽取模型，返回抽取结果 |
| 方法 | HTTP POST /\_api/ckc/relationextraction/{modelPath} |
| 参数 | 无 |
| Body | {  “text”: “After a standout at the University, Michael Jordan joined the Bulls in 1984.……”,  “entities”:[{  “mid”: 986546  “entityMention”: ”Michael Jordan”,  “entityType”: 90，  “byte\_offset“：36,  ……  },  ……  ]  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [{  “left\_mid”: 986546,  “right\_mid”: 999999,  “sid”:19011  },{  },  ……  ]  }  } |

* 编程接口：

|  |  |
| --- | --- |
| 功能 | 输入一段文本，调用文本关系抽取模型，返回抽取结果 |
| 方法 | CKCApi.relationExtraction(String modelPath, String filePath, String fileType，String encoding) |
| 参数 | modelPath – 文本关系抽取模型路径  filePath – 输入文本路径  fileType – 输入文本格式，比如JSON、XML等  encoding - 文本文件编码，比如UTF-8、GBK |
| 返回 | success - 训练是否成功  JSON – JSON格式的分析结果 |
| 异常 | CKCApiException |

## 知识提及读写接口

说明：

通过自然语言处理和图像处理等技术产出的知识提及进行构建写入和查询读取操作，包括类（概念）的提及、实体的提及、事件的提及、关系的提及等类型，用于连接底层数据资源和中间层知识资源。

### 提及读取

* HTTP接口

|  |  |
| --- | --- |
| 功能 | 以一定条件筛选获取提及数据结构，可支持SQL查询或条件筛选 |
| 方法 | HTTP POST /\_api/ckc/mentions/query |
| 参数 | 无 |
| Body | {  “query”: “select mentions from db.entity\_mention where condition”,  }  Or  {  “type”: “entity”,  “condition”: “value”  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “data”: [  {MentionStandardFormat}  ]  “status”: “mention retrieved”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 以一定条件筛选获取提及数据结构，可支持SQL查询或条件筛选。 |
| 方法 | CKCApi.getMentions(String sql, Map<String,Object> conditions) |
| 参数 | sql – 查询语句字符串  conditions – 提及筛选条件 |
| 返回 | 包含多条提及对象的集合  Collection<CKCMention<String, Object>>  CKCMention子类：CKCClassMention, CCKCntityMention, CCKCventMention, CKCRelationMention |
| 异常 | CKCApiException |

### 提及写入

* HTTP接口

|  |  |
| --- | --- |
| 功能 | 以约定的提及标准格式构建请求并写入数据库中。 |
| 方法 | HTTP POST /\_api/ckc/mentions/db/{dbName}/table/{tableName} |
| 参数 | dbName：数据库名称  tableName：表名称 |
| Body | {  “mentions”: [  { MentionStandardFormat }  ],  } |
| 返回 | {  “code”: “200”,  “message”: “success”,  “result”: {  “created”: true,  “status”: “mention created”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 以约定的提及标准格式构建请求并写入数据库中。 |
| 方法 | CKCApi.createMentions(  Collection<CKCMention> mentions,  String dbName,  String tableName  ) |
| 参数 | mentions – 提及接口数据集合  dbName – 数据库名称  tableName – 表名称 |
| 返回 | 是否成功创建提及  boolean |
| 异常 | CKCApiException |

## 知识图谱设计构建接口

**说明：**

定义或导入知识图谱的多层级、多类型的类（概念）体系，并进一步构建知识图谱模型体系，以知识提及的资源为基础，完成从提及到知识的实例化转化功能，实现知识图谱的半自动构建。

### 类相关接口

#### 获取实体类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取某一图的所有实体类的集合。 |
| 方法 | HTTP GET /\_api/ckc/graph/{graphName}/ entity-class |
| 参数 | graphName：图的名称 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “classes”: [  {  “name”: “class name”,  “type”: “entity”,  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “parent”: “parent class name”,  “property”: “value”  }  },  {…}  ]  “status”: “graph classes retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取某一图的所有实体类的集合。 |
| 方法 | CKCApi.CKCGraph.getEntityClasses() |
| 参数 | 无 |
| 返回 | 所有该图的满足条件的实体或事件类的集合，  Collection<CCKEntityClass<String, Object>> |
| 异常 | CKCApiException |

#### 获取事件类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取某一图的所有事件类的集合。 |
| 方法 | HTTP GET /\_api/ckc/graph/{graphName}/ event-class |
| 参数 | graphName：图的名称 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “classes”: [  {  “name”: “class name”,  “type”: “event”,  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “property”: “value”  }  },  {…}  ]  “status”: “graph classes retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取某一图的所有事件类的集合。 |
| 方法 | CKCApi.CKCGraph.getEventClasses() |
| 参数 | 无 |
| 返回 | 所有该图的满足条件的实体或事件类的集合，  Collection<CKCEventClass<String, Object>> |
| 异常 | CKCApiException |

#### 获取关系类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取某一图的所有关系类的集合。 |
| 方法 | HTTP GET /\_api/ckc/graph/{graphName}/relation-classes |
| 参数 | graphName：图的名称 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relationClasses”: [  {  “name”: “relation class name”,  “from”: [“entity/event class”],  “to”: [“entity/event class”],  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “property”: “value”  }  }  ]  “status”: “graph relation classes retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取某一图的所有关系类的集合。 |
| 方法 | CKCApi.CKCGraph.getRelationClasses() |
| 参数 | 无 |
| 返回 | 所有该图的关系类的集合  Collection<CKCRelationClass<String, Object>> |
| 异常 | CKCApiException |

#### 添加实体类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 为某个图添加实体类，若底层集合不存在，则同时创建实体集合。 |
| 方法 | HTTP POST /\_api/ckc/graph/{graphName}/entity-class |
| 参数 | graphName：图的名称 |
| Body | {  “class”: “className”,  “classType”: “entity”,  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “parent”: “parent class name”,  “source”: “source name”,  “property”: “value”  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  “\_id”: “graph id”,  “name”: “graph name”,  “relationClasses”: [  {  “name”: “relation class name”,  “from”: [entity/event class name list],  “to”: [entity/event class name list],  “attributes”: {“attr1”: “value1”, …},  “property”: “value”  },  {…}  ],  “entityClasses”: [  {  “name”: “class name”,  “type”: “entity”,  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “parent”: “parent class name”,  “property”: “value”  }  },  {…}  ],  “eventClasses”: [  {  “name”: “class name”,  “type”: “event”,  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “property”: “value”  }  }  ]  }  “status”: “graph entity class added”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 为某个图添加实体类，若底层集合不存在，则同时创建实体集合。 |
| 方法 | CKCApi.CKCGraph.createEntityClass(  CKCEntityClass<String, Object> entityClass  ) |
| 参数 | entityClass – 实体类 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 添加事件类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 为某个图添加事件类，若底层集合不存在，则同时创建事件集合。 |
| 方法 | HTTP POST /\_api/ckc/graph/{graphName}/event-class |
| 参数 | graphName：图的名称 |
| Body | {  “name”: “class name”,  “type”: “event”,  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “property”: “value”  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC Graph Schema  }  “status”: “graph event class added”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 为某个图添加事件类，若底层集合不存在，则同时创建事件集合。 |
| 方法 | CKCApi.CKCGraph.createEventClass(  CKCEventClass<String, Object> eventClass  ) |
| 参数 | eventClass – 事件类 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 添加关系类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 为某个图添加关系类的定义，需包含关系两端的实体或事件类的集合。 |
| 方法 | HTTP POST /\_api/ckc/graph/{graphName}/relation-class |
| 参数 | graphName：图的名称 |
| Body | {  “name”: “relation class name”,  “from”: [“entity/event class”],  “to”: [“entity/event class”],  “attributes”: {“attr1”: “value1”, …},  “metadata”: {  “property”: “value”  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC graph schema  }  “status”: “graph relation class added”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 为某个图添加关系类的定义，需包含关系两端的实体或事件类的集合。 |
| 方法 | CKCApi.CKCGraph.createRelationClass(  CKCRelationClass<String, Object> relationClass  ) |
| 参数 | relationClass – 关系类 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 导入实体类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 为某个图批量导入已有的实体类定义，需按格式指定信息来源字段。 |
| 方法 | HTTP POST /\_api/ckc/graph/{graphName}/entity-classes |
| 参数 | graphName：图的名称 |
| Body | {  “source”: “entity class source url”,  “config”: {  “nameField”: “name field”,  “attributeFields”: [“attribute field”, …],  “metadataFields”: [“meta field”, …]  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC graph schema  }  “status”: “graph entity class imported”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 为某个图批量导入已有的实体类定义，需按格式指定信息来源字段。 |
| 方法 | CKCApi.CKCGraph.importEntityClasses(  String url,  Map<String, Object> config  ) |
| 参数 | url – 实体定义数据来源  config – 实体定义导入字段映射配置 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 导入事件类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 为某个图批量导入已有的事件类定义，需按格式指定信息来源字段。 |
| 方法 | HTTP POST /\_api/ckc/graph/{graphName}/event-classes |
| 参数 | graphName：图的名称 |
| Body | {  “source”: “event class source url”,  “config”: {  “nameField”: “name field”,  “attributeFields”: [“attribute field”, …],  “metadataFields”: [“meta field”, …]  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC graph schema  }  “status”: “graph event class imported”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 为某个图批量导入已有的事件类定义，需按格式指定信息来源字段。 |
| 方法 | CKCApi.CKCGraph.importEventClasses(  String url,  Map<String, Object> config  ) |
| 参数 | url – 事件定义数据来源  config – 事件定义导入字段映射配置 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 导入关系类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 为某个图批量导入已有的关系类定义，需按格式指定信息来源字段。 |
| 方法 | HTTP POST /\_api/ckc/graph/{graphName}/relation-classes |
| 参数 | graphName：图的名称 |
| Body | {  “source”: “entity class source url”,  “config”: {  “nameField”: “name field”,  “toField”: “point to field”,  “fromField”: “point from field”,  “attributeFields”: [“attribute field”, …],  “metadataFields”: [“meta field”, …]  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC graph schema  }  “status”: “graph relation class imported”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 为某个图批量导入已有的关系类定义，需按格式指定信息来源字段。 |
| 方法 | CKCApi.CKCGraph.importRelationClasses(  String url,  Map<String, Object> config  ) |
| 参数 | url – 关系定义数据来源  config – 关系定义导入字段映射配置 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 删除实体类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 删除在某个图中的实体类，可以选择同时删除对应的底层数据集合，类不能与其他类有关系定义，否则需先解除关系。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/entity-class/{className} |
| 参数 | graphName：图的名称  className：实体类的名称  ?dropCollection=true：同时删除实体的集合 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC Graph Schema  }  “status”: “graph entity class removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 删除在某个图中的实体类，可以选择同时删除对应的底层数据集合，类必须不与其他类有关系定义，否则需先解除关系。 |
| 方法 | CKCApi.CKCGraph.removeEntityClass(  String className,  boolean dropCollection  ) |
| 参数 | className – 实体类的名称  dropCollection - 同时删除实体类的集合 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 删除事件类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 删除在某个图中的事件类，可以选择同时删除对应的底层数据集合，类不能与其他类有关系定义，否则需先解除关系。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/event-class/{className} |
| 参数 | graphName：图的名称  className：事件类的名称  ?dropCollection=true：同时删除事件的集合 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC Graph Schema  }  “status”: “graph event class removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 删除在某个图中的事件类，可以选择同时删除对应的底层数据集合，类必须不与其他类有关系定义，否则需先解除关系。 |
| 方法 | CKCApi.CKCGraph.removeEventClass(  String className,  boolean dropCollection  ) |
| 参数 | className – 事件类的名称  dropCollection - 同时删除事件类的集合 |
| 返回 | 该图的schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 删除关系类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 删除在某个图中的一个关系类的定义，可以选择同时删除对应的底层数据集合，删除关系类不影响所连接的实体或事件类及其实例。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/relation-class/{relationClassName} |
| 参数 | graphName：图的名称  relationClassName：关系类的名称  ?dropCollection=true：同时删除关系的数据集合 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC Graph Schema  }  “status”: “graph relation class removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 删除在某个图中的一个关系类的定义，可以选择同时删除对应的底层数据集合，删除关系类不影响所连接的实体或事件类及其实例。 |
| 方法 | CKCApi.CKCGraph.removeRelationClass(  String relationClassName,  boolean dropCollection  ) |
| 参数 | relationClassName –关系类的名称  dropCollection - 同时删除关系的数据集合 |
| 返回 | 该图的更新schema信息  CKCGraphSchema |
| 异常 | CKCApiException |

### 图相关接口

#### 获取所有图schema

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 列表形式返回所有图的schema信息，包括实体和事件类及关系类。 |
| 方法 | HTTP GET /\_api/CKC/graph |
| 参数 | 无 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graphs”: [  {  CKC Graph Schema1  },  {  CKC Graph Schema2  }  ]  “status”: “all graphs listed”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 返回所有图的schema信息，包括实体或事件类及关系类。 |
| 方法 | CKCApi.getGraphs() |
| 参数 | 无 |
| 返回 | 所有图的schema信息和基本信息  Collection<CKCGraphSchema> |
| 异常 | CKCApiException |

#### 创建图

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 通过指定关系及实体或事件类来创建图结构，类对应于底层数据集合并包含元数据信息，新的类则创建空集合。 |
| 方法 | HTTP POST /\_api/CKC/graph |
| 参数 | 无 |
| Body | {  “data”: {  “name:”: “graph name”,  “relationClasses”: [  {  “name”: “relation class name”,  “from”: [entity/event class name list],  “to”: [entity/event class name list]  },  {…}  ]  “orphanClasses”: [ … ]  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  CKC Graph Schema  }  “status”: “graph created”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 通过指定关系及实体或事件类来创建图结构，类对应于底层数据集合，新的类则创建空集合。 |
| 方法 | CKCApi.createGraph(  String graphName,  Collection<CKCRelationClass> relationClasses,  Collection<String> orphanClasses  ) |
| 参数 | graphName - 图名称  relationClasses – 关系类的对象集合，包含所连接实体或事件  orphanClasses – 孤立类 |
| 返回 | 所创建的图的schema信息和基本信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 获取某一图schema

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 返回某一图的schema信息，包括实体或事件类及关系类。 |
| 方法 | HTTP GET /\_api/CKC/graph/{graphName} |
| 参数 | graphName：图的名称 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graph”: {  CKC Graph Schema  }  “status”: “graph found and returned”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 返回某一图的schema信息，包括实体或事件类及关系类。 |
| 方法 | CKCApi.CKCGraph.getSchema() |
| 参数 | 无 |
| 返回 | 图的schema信息和基本信息  CKCGraphSchema |
| 异常 | CKCApiException |

#### 删除图

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 根据某一图的名称删除图结构，可选择同时删除所含实体、事件和关系。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName} |
| 参数 | graphName：图的名称  ?dropCollections=true：同时删除图的实体、事件、关系集合 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “status”: “graph removed”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 根据某一图的名称删除图结构，可选择同时删除所含实体、事件和关系。 |
| 方法 | CKCApi.CKCGraph.drop(  boolean dropCollections  ) |
| 参数 | dropCollections – 是否同时删除图的实体、事件、关系集合 |
| 返回 | void |
| 异常 | CKCApiException |

## 知识图谱访问计算接口

**说明：**

根据知识图谱实例知识数据如实体、事件、关系等进行操作，设计针对集合和个体知识的增、删、改、查（CRUD）等基本管理与访问操作，支撑知识图谱数据的应用、计算、运营。

### 实例访问接口

#### 写入实体实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图的某个类下创建新实体实例，实体实例为JSON格式。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/class/{className} |
| 参数 | graphName：图的名称  className:实体类的名称 |
| Body | [  {Entity Instance}  ] |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “entity”: [“entity id”]  }  “status”: “entity created”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图的某个类下创建新实体实例，实体实例为JSON格式。 |
| 方法 | CKCApi.CKCGraph.createEntities(  String className,  Collection<CKCEntity> entities  ) |
| 参数 | className – 实体类的名称  entities – 实体实例对象集合 |
| 返回 | 所创建实体的id集合  Collection<String> |
| 异常 | CKCApiException |

#### 写入事件实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图的某个类下创建新事件实例，事件实例为JSON格式。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/class/{className} |
| 参数 | graphName：图的名称  className:事件类的名称 |
| Body | [  {Event Instance}  ] |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “event”: [“event id”]  }  “status”: “event created”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图的某个类下创建新事件实例，事件实例为JSON格式。 |
| 方法 | CKCApi.CKCGraph.createEvents(  String className,  Collection<CKCEvent> events  ) |
| 参数 | className – 事件类的名称  events– 事件实例对象集合 |
| 返回 | 所创建事件的id集合  Collection<String> |
| 异常 | CKCApiException |

#### 写入关系实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图的某个关系类下创建新关系实例，关系实例为包含两端实体实例ID的 JSON格式。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/relationClass/{relationClassName} |
| 参数 | graphName：图的名称  relationClassName:关系类的名称 |
| Body | {  “\_from”: “entity id1”,  “\_to”: “entity id2”,  “attribute”: “value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relation”: “relation id”  }  “status”: “relation created”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图的某个关系类下创建新关系实例，关系实例为包含两端实体实例ID的 JSON格式。 |
| 方法 | CKCApi.CKCGraph.createRelation (  String className,  CKCRelation relation  ) |
| 参数 | className – 关系类的名称  relation – 关系实例对象 |
| 返回 | 所创建关系实例的id  String |
| 异常 | CKCApiException |

#### 获取实体实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的某个类下的一个实体实例。 |
| 方法 | HTTP GET /\_api/CKC/graph/{graphName}/class/{className}/{entityId} |
| 参数 | graphName：图的名称  className：实体类的名称  entityId：实体实例的ID |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “entity”: {  “\_id”: “entity id”,  “attribute”: “value”  }  “status”: “entity retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的某个类下的一个实体实例。 |
| 方法 | CKCApi.CKCGraph.getEntity(  String className,  String entityId  ) |
| 参数 | className – 实体类的名称  entityId – 实体实例的ID |
| 返回 | 实体实例对象  CKCEntity |
| 异常 | CKCApiException |

#### 获取事件实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的某个类下的一个事件实例。 |
| 方法 | HTTP GET /\_api/CKC/graph/{graphName}/class/{className}/{eventId} |
| 参数 | graphName：图的名称  className：事件类的名称  eventId：事件实例的ID |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “event”: {  “\_id”: “event id”,  “attribute”: “value”  }  “status”: “event retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的某个类下的一个事件实例。 |
| 方法 | CKCApi.CKCGraph.getEvent(  String className,  String eventId  ) |
| 参数 | className – 事件类的名称  eventId – 事件实例的ID |
| 返回 | 事件实例对象  CKCEvent |
| 异常 | CKCApiException |

#### 获取关系实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图的某个关系类下获取关系实例信息。 |
| 方法 | HTTP GET /\_api/CKC/graph/{graphName}/relationClass/{relationClassName}/{relationId} |
| 参数 | graphName：图的名称  relationClassName:关系类的名称  relationId：关系实例的ID |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relation”: {  “\_id”: “relation id”,  “\_from”: “entity id1”,  “\_to”: “entity id2”,  “attribute”: “value”  }  }  “status”: “relation retrieved”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图的某个关系类下获取关系实例信息。 |
| 方法 | CKCApi.CKCGraph.getRelation (  String className,  String relationId  ) |
| 参数 | className – 关系类的名称  relationId – 关系实例的ID |
| 返回 | 关系实例对象  CKCRelation |
| 异常 | CKCApiException |

#### 获取实体实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中获取满足一定条件的实体实例集合。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/entities |
| 参数 | graphName：图的名称 |
| Body | {  “classFilter”: [“class name”],  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “entities”: [  CKC Entity  ]  }  “status”: “entities retrieved”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中获取满足一定条件的实体实例集合。 |
| 方法 | CKCApi.CKCGraph.getEntities(  CKCEntityFilter filter  ) |
| 参数 | filter – 筛选实体集合的条件配置 |
| 返回 | 实体实例对象  Collection<CKCEntity> |
| 异常 | CKCApiException |

#### 获取事件实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中获取满足一定条件的事件实例集合。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/events |
| 参数 | graphName：图的名称 |
| Body | {  “classFilter”: [“class name”],  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “events”: [  CKC Event  ]  }  “status”: “events retrieved”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中获取满足一定条件的事件实例集合。 |
| 方法 | CKCApi.CKCGraph.getEvents(  CKCEventFilter filter  ) |
| 参数 | filter – 筛选事件集合的条件配置 |
| 返回 | 事件实例对象  Collection<CKCEvent> |
| 异常 | CKCApiException |

#### 获取关系实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中获取满足一定条件的关系实例集合。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/relations |
| 参数 | graphName：图的名称 |
| Body | {  “classFilter”: [“class name”],  “toClassFilter”: [],  “fromClassFilter”: []  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relations”: [  CKC Relation  ]  }  “status”: “relations retrieved”  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中获取满足一定条件的关系实例集合。 |
| 方法 | CKCApi.CKCGraph.getRelations(  CKCRelationFilter filter  ) |
| 参数 | filter – 筛选关系集合的条件配置 |
| 返回 | 关系实例对象  Collection<CKCRelation> |
| 异常 | CKCApiException |

#### 更新实体实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 更新图的某个类下的一个实体实例，需给出要更新的属性和值。 |
| 方法 | HTTP PATCH /\_api/CKC/graph/{graphName}/class/{className}/{entityId} |
| 参数 | graphName：图的名称  className：实体类的名称  entityId：实体实例的ID |
| Body | {  “attribute”: “new value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “entity”: {  “\_id”: “entity id”,  “attribute”: “new value”  }  “status”: “entity updated”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 更新图的某个类下的一个实体实例，需给出要更新的属性和值。 |
| 方法 | CKCApi.CKCGraph.updateEntity(  String className,  String entityId,  Map<String, Object> attrToUpdate  ) |
| 参数 | className – 实体类的名称  entityId – 实体实例的ID  attrToUpdate – 待更新属性和值 |
| 返回 | 实体实例对象  CKCEntity |
| 异常 | CKCApiException |

#### 更新事件实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 更新图的某个类下的一个事件实例，需给出要更新的属性和值。 |
| 方法 | HTTP PATCH /\_api/CKC/graph/{graphName}/class/{className}/{eventId} |
| 参数 | graphName：图的名称  className：事件类的名称  eventId：事件实例的ID |
| Body | {  “attribute”: “new value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “event”: {  “\_id”: “event id”,  “attribute”: “new value”  }  “status”: “event updated”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 更新图的某个类下的一个事件实例，需给出要更新的属性和值。 |
| 方法 | CKCApi.CKCGraph.updateEvent(  String className,  String eventId,  Map<String, Object> attrToUpdate  ) |
| 参数 | className – 事件类的名称  eventId – 事件实例的ID  attrToUpdate – 待更新属性和值 |
| 返回 | 事件实例对象  CKCEvent |
| 异常 | CKCApiException |

#### 更新关系实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 更新图的某个类下的一个关系实例，需给出要更新的属性和值。 |
| 方法 | HTTP PATCH /\_api/CKC/graph/{graphName}/relationClass/{relationClassName}/{relationId} |
| 参数 | graphName：图的名称  relationClassName:关系类的名称  relationId：关系实例的ID |
| Body | {  “attribute”: “new value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relation”: {  “\_id”: “relation id”,  “\_from”: “entity id1”,  “\_to”: “entity id2”,  “attribute”: “new value”  }  “status”: “relation updated”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 更新图的某个类下的一个关系实例，需给出要更新的属性和值。 |
| 方法 | CKCApi.CKCGraph.updateRelation(  String relationClassName,  String relationId,  Map<String, Object> attrToUpdate  ) |
| 参数 | relationClassName – 关系类的名称  relationId – 关系实例的ID  attrToUpdate – 待更新属性和值 |
| 返回 | 关系实例对象  CKCRelation |
| 异常 | CKCApiException |

#### 更新实体实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中更新满足一定条件的实体实例集合。 |
| 方法 | HTTP PATCH /\_api/CKC/graph/{graphName}/entities |
| 参数 | graphName：图的名称 |
| Body | {  “filter”: {  “classFilter”: [“class name”],  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  },  “attribute”: “new value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “entities”: [  {  “\_id”: “entity id”,  “attribute”: “new value”  }  ]  “status”: “entities updated”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中更新满足一定条件的实体实例集合。 |
| 方法 | CKCApi.CKCGraph.updateEntities(  CKCEntityFilter filter,  Map<String, Object> attrToUpdate  ) |
| 参数 | filter – 筛选实体集合的条件配置  attrToUpdate - 待更新属性和值 |
| 返回 | 实体实例对象集合  Collection<CKCEntity> |
| 异常 | CKCApiException |

#### 更新事件实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中更新满足一定条件的事件实例集合。 |
| 方法 | HTTP PATCH /\_api/CKC/graph/{graphName}/events |
| 参数 | graphName：图的名称 |
| Body | {  “filter”: {  “classFilter”: [“class name”],  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  },  “attribute”: “new value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “events”: [  {  “\_id”: “event id”,  “attribute”: “new value”  }  ]  “status”: “events updated”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中更新满足一定条件的事件实例集合。 |
| 方法 | CKCApi.CKCGraph.updateEvents(  CKCEventFilter filter,  Map<String, Object> attrToUpdate  ) |
| 参数 | filter – 筛选事件集合的条件配置  attrToUpdate - 待更新属性和值 |
| 返回 | 事件实例对象集合  Collection<CKCEvent> |
| 异常 | CKCApiException |

#### 更新关系实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中更新满足一定条件的关系集合。 |
| 方法 | HTTP PATCH /\_api/CKC/graph/{graphName}/relations |
| 参数 | graphName：图的名称 |
| Body | {  “filter”: {  “classFilter”: [“class name”],  “toClassFilter”: [],  “fromClassFilter”: []  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  },  }  “attribute”: “new value”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relations”: [  {  “\_id”: “relation id”,  “\_from”: “entity id1”,  “\_to”: “entity id2”,  “attribute”: “new value”  }  ]  “status”: “relation updated”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中更新满足一定条件的关系集合。 |
| 方法 | CKCApi.CKCGraph.updateRelations(  CKCRelationFilter filter,  Map<String, Object> attrToUpdate  ) |
| 参数 | filter – 筛选关系集合的条件配置  attrToUpdate - 待更新属性和值 |
| 返回 | 关系对象集合  Collection<CKCRelation> |
| 异常 | CKCApiException |

#### 删除实体实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 删除图的某个类下的一个实体实例。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/class/{className}/{entityId} |
| 参数 | graphName：图的名称  className：实体类的名称  entityId：实体实例的ID |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “removed”: true,  “status”: “entity removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 删除图的某个类下的一个实体实例。 |
| 方法 | CKCApi.CKCGraph.removeEntity(  String className,  String entityId  ) |
| 参数 | className – 实体类的名称  entityId – 实体实例的ID |
| 返回 | 实体实例是否被删除  boolean |
| 异常 | CKCApiException |

#### 删除事件实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 删除图的某个类下的一个事件实例。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/class/{className}/{eventId} |
| 参数 | graphName：图的名称  className：事件类的名称  eventId：事件实例的ID |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “removed”: true,  “status”: “event removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 删除图的某个类下的一个事件实例。 |
| 方法 | CKCApi.CKCGraph.removeEvent(  String className,  String eventId  ) |
| 参数 | className – 事件类的名称  eventId – 事件实例的ID |
| 返回 | 事件实例是否被删除  boolean |
| 异常 | CKCApiException |

#### 删除关系实例

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 删除图的某个类下的一个关系实例。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/relationClass/{relationClassName}/{relationId} |
| 参数 | graphName：图的名称  relationClassName：关系类的名称  relationId：关系实例的ID |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “removed”: true,  “status”: “relation removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 删除图的某个类下的一个关系实例。 |
| 方法 | CKCApi.CKCGraph.removeRelation(  String relationClassName,  String relationId  ) |
| 参数 | relationClassName – 关系类的名称  relationId – 关系实例的ID |
| 返回 | 关系实例是否被删除  boolean |
| 异常 | CKCApiException |

#### 删除实体实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中删除满足一定条件的实体实例集合。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/entities |
| 参数 | graphName：图的名称 |
| Body | {  “classFilter”: [“class name”],  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “removed”: true,  “status”: “entities removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中删除满足一定条件的实体实例集合。 |
| 方法 | CKCApi.CKCGraph.deleteEntities(  CKCEntityFilter filter  ) |
| 参数 | filter – 筛选实体集合的条件配置 |
| 返回 | 实体实例是否被删除  boolean |
| 异常 | CKCApiException |

#### 删除事件实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中删除满足一定条件的事件实例集合。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/events |
| 参数 | graphName：图的名称 |
| Body | {  “classFilter”: [“class name”],  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “removed”: true,  “status”: “events removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中删除满足一定条件的事件实例集合。 |
| 方法 | CKCApi.CKCGraph.deleteEvents(  CKCEventFilter filter  ) |
| 参数 | filter – 筛选事件集合的条件配置 |
| 返回 | 事件实例是否被删除  boolean |
| 异常 | CKCApiException |

#### 删除关系实例集合（QL）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在图中删除满足一定条件的关系集合。 |
| 方法 | HTTP DELETE /\_api/CKC/graph/{graphName}/relations |
| 参数 | graphName：图的名称 |
| Body | {  “classFilter”: [“class name”],  “toClassFilter”: [],  “fromClassFilter”: []  “attributeFilter”: {  “attr1”: {“op”: “<>”, “value”: “100”},  “attr2”: {“op”: “like”, “value”: “something”}  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “removed”: true,  “status”: “relations removed”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在图中删除满足一定条件的关系集合。 |
| 方法 | CKCApi.CKCGraph.deleteRelations(  CKCRelationFilter filter  ) |
| 参数 | filter – 筛选关系集合的条件配置 |
| 返回 | 关系是否被删除  boolean |
| 异常 | CKCApiException |

### 实体链接接口

#### 实体消歧

* HTTP接口

|  |  |
| --- | --- |
| 功能 | 给定一个图和一个待消歧实体，返回图中与待消歧实体指代相同的实体。 |
| 方法 | HTTP POST /\_api/CKC/graph/entity\_disam |
| 参数 | 无 |
| Body | {  “graphName”: “graph name”,  “entity”: {  “attribute”: “value”  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “graphEntityId”: “entity id”,  “confidence”: 0.92  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 在某一图中，输入一系列实体提及，返回所链接上的实体ID。 |
| 方法 | CKCApi.CKCGraph.entityDisam(  String graphName,  Map<String, String> entityAttr  ) |
| 参数 | graphName – 图的名称  entityAttr – 实体属性字典 |
| 返回 | 链接到的实体ID及置信度  Map<String, Object> |
| 异常 | CKCApiException |

#### 实体融合

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 给定一个图中的实体id，和待融合的新实体，返回融合计算后的实体。 |
| 方法 | HTTP POST /\_api/CKC/graph/entity\_fusion |
| 参数 | 无 |
| Body | {  “graphName”: “graph name”,  “graphEntityId”: “entity id”,  “newEntity”: {  “attribute”: “value”  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “\_id”: “entity id”,  “attribute”: “value”  }  } |

* 编程接口

|  |  |
| --- | --- |
| 功能 | 给定一个图中的实体id，和待融合的新实体，返回融合计算后的实体。 |
| 方法 | CKCApi.CKCGraph.entityFusion(  String graphName,  String entityId  Map<String, String> entityAttr  ) |
| 参数 | graphName – 图的名称  entityId – 给定实体ID  entityAttr – 新实体属性字典 |
| 返回 | 融合后的实体  CKCEntity |
| 异常 | CKCApiException |

#### 事件链接

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 在某一图中，输入一系列事件提及，返回所链接上的事件ID。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/class/{className}/event-linking |
| 参数 | graphName：图的名称  className：类的名称 |
| Body | {  “mentions”: [  { MentionStandardFormat }  ],  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “eventIds”: [“id1”, ”None”, ”id2”, … ]  “status”: “event linking finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 在某一图中，输入一系列事件提及，返回所链接上的事件ID。 |
| 方法 | CKCApi.CKCGraph.eventLinking(  String className,  Collection<CKCEventMention> mentions  ) |
| 参数 | className – 实体类型名称  mentions – 实体提及集合 |
| 返回 | 链接到的事件ID集合  Collection<String> |
| 异常 | CKCApiException |

### 图算法接口

#### 图的遍历

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 遍历一个图，需声明起始实体实例、关系类的集合、方向、策略、深度范围等参数。 |
| 方法 | HTTP POST /\_api/CKC/traversal |
| 参数 | 无 |
| Body | {  “graphName”: “graph name”,  “startEntity”: “entity id”,  “direction”: “outbound”,  “minDepth”: 2,  “strategy”: “depthfirst”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “visited”: {  “entities”: [{entity1}, {entity2}],  “path”: [  {  “relations”: [{relation1}, {relation2}],  “entities”: [{entity1}, {entity2}]  },  {…}  ]  },  “status”: “traversal finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 遍历一个图，需声明起始实体实例、关系类的集合、方向、策略、深度范围等参数。 |
| 方法 | CKCApi.CKCGraph.traversal (  String graphName,  String startEntity,  String direction,  int minDepth,  String strategy  …  ) |
| 参数 | graphName – 图的名称  startEntity – 起始实体  direction – 遍历方向  minDepth – 最小深度  maxDepth – 最大深度  strategy – 遍历策略  … |
| 返回 | 遍历结果路径对象  CKCTraversalPath |
| 异常 | CKCApiException |

#### 路径检索（升级子图相关性排序）

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 给定一个图内的始末实体实例，和一个相关子图，返回所有可达的路径，并按子图相关性排序。可用于问答过程中答案归因，计算问题节点到答案节点的显式路径。 |
| 方法 | HTTP POST /\_api/CKC/paths |
| 参数 | 无 |
| Body | {  “graphName”: “graph name”,  “startEntityId”: “entity id1”,  “endEntityId”: “entity id2”,  “relatedSubGraph”: {  “entities”: [{entity1}, {entity2}, …]  “relations”: [  {  “id”: “relation id”,  “\_from”: “entity id1”,  “\_to”: “entity id2”,  “attribute”: “value”  },  {…}  ],  },  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “visited”: {  “entities”: [{start entity}, … , {end entity}],  “path”: [  {  “relations”: [{relation1}, {relation2}],  “entities”: [{entity1}, {entity2}],  “correlation”: 0.92  },  {…}  ]  },  “status”: “paths search finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 给定一个图内的始末实体实例，返回所有可达的路径。 |
| 方法 | CKCApi.CKCGraph.paths (  String startEntityId,  String endEntityId,  CKCSubGraph subGraph  ) |
| 参数 | startEntityId – 起始实体或事件实例的ID  endEntityId – 末尾实体或事件实例的ID  subGraph – 路径检索相关子图定义 |
| 返回 | 所有遍历结果路径对象  CKCTraversalPath |
| 异常 | CKCApiException |

#### 获取子图

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 根据图的某个实体或事件实例获取满足一定条件的子图，进而在领域知识图谱中，获取问题求解上下文相关的领域内知识脉络。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/class/{className}/{entityId}/subgraph |
| 参数 | graphName：图的名称  className：实体或事件类的名称  entityId：中心实体或事件实例  ?hopNum=5 ：子图包含的中心实体关系层数（默认为2） |
| Body | {  “entityFilter”: {  “classFilter”: [],  “attributeFilter”: {“attr1”: {“op”: “”, “value”: “”}, …}  },  “eventClassFilter”: {},  “relationClassFilter”: {}  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “subgraph”: {  “entities”: [{entity1}, {entity2}, …]  “relations”: [  {  “id”: “relation id”,  “\_from”: “entity id1”,  “\_to”: “entity id2”,  “attribute”: “value”  },  {…}  ],  },  “status”: “subgraph retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 根据图的某个实体或事件实例获取一定范围内的子图。 |
| 方法 | CKCApi.CKCGraph.subGraph (  String className,  String centerEntityId,  int hopNum,  CKCSubGraphFilter subGraphFilter  ) |
| 参数 | className - 实体或事件类的名称  centerEntityId– 中心实体或事件实例的ID  hopNum - 子图包含的中心实体关系层数（默认为2）  subGraphFilter – 子图筛选条件配置 |
| 返回 | 结果子图对象  CKCSubGraph |
| 异常 | CKCApiException |

#### 计算实例的关系数

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 计算图的某个实体或事件实例的双向关系数量。 |
| 方法 | HTTP GET /\_api/CKC/graph/{graphName}/class/{className}/{entityId}/degree |
| 参数 | graphName：图的名称  className：实体或事件类的名称  entityId：中心实体或事件实例 |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “degree”: {  “inDegree”: 5  “outDegree”: 3  },  “status”: “entity degree retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 计算图的某个实体或事件实例的双向关系数量。 |
| 方法 | CKCApi.CKCGraph.degrees (  String className,  String entityId  ) |
| 参数 | className - 实体或事件类的名称  entityId– 中心实体或事件实例的ID |
| 返回 | 实体或事件的双向关系数量  CKCDegree<String,int> |
| 异常 | CKCApiException |

#### 获取实例的邻居

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的某个实体或事件实例的所有邻居实例。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/class/{className}/{entityId}/neighbors |
| 参数 | graphName：图的名称  className：实体或事件类的名称  entityId：中心实体或事件实例 |
| Body | {  “direction”: “outbound/inbound/any”,  “entityFilter”: { },  “relationFilter”: { },  “minDepth”: 1,  “maxDepth”: 5  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “neighbors”: [“neighbor entity id”],  “status”: “entity neighbors retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的某个实体或事件实例的所有邻居实例。 |
| 方法 | CKCApi.CKCGraph.neighbors (  String className,  String entityId,  String direction,  Map<String, Object> entityFilter,  Map<String, Object> relationFilter,  int minDepth,  int maxDepth  ) |
| 参数 | className - 实体或事件类的名称  entityId – 中心实体或事件实例的ID  direction – 指明关系方向  entityFilter – 实例筛选条件  relationFilter – 关系筛选条件  minDepth – 允许检索的最小关系层级  maxDepth – 允许检索的最大关系层级 |
| 返回 | 实体或事件实例的邻居实例列表  Collection<String> |
| 异常 | CKCApiException |

#### 获取实例之间的共同邻居

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的某两个实体或事件实例的共同邻居实例。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/common-neighbors |
| 参数 | graphName：图的名称 |
| Body | {  “entity1”: “entity id 1”,  “entity2”: “entity id 2”,  “param1”: {entity1 filter options},  “param2”: {entity2 filter options}  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “commonNeighbors”: [“neighbor entity id”],  “status”: “entity common neighbors retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的某两个实体或事件实例的共同邻居实例。 |
| 方法 | CKCApi.CKCGraph.commonNeighbors (  String entityId1,  String entityId2,  Map<String, Object> entityOptions1,  Map<String, Object> entityOptions2  ) |
| 参数 | entityId1 – 实体或事件实例1的ID  entityId2 - 实体或事件实例2的ID  entityOptions1 – 实体或事件实例1的检索条件  entityOptions2 – 实体或事件实例2的检索条件 |
| 返回 | 实体或事件实例的共同邻居实例列表  Collection<String> |
| 异常 | CKCApiException |

#### 获取实例之间的最短路径

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的某两个实体或事件实例之间的最短路径。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/shortest-paths |
| 参数 | graphName：图的名称 |
| Body | {  “startEntity”: “start entity id”,  “endEntity2”: “end entity id”,  “options”: {  “direction”: “outbound/inbound/any”,  “weight”: “a relation attribute name”, （不指定，距离取1）  “entityFilter”: {},  “relationFilter”: {}  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “shortestPath”: [  {  “entities”: [“entity id”],  “relations”: [{relation attributes}],  “distance”: distance number}  }  ],  “status”: “shortest paths search finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的某两个实体或事件实例之间的最短路径。 |
| 方法 | CKCApi.CKCGraph.shortestPaths (  String startEntity,  String endEntity,  Map<String, Object> options  ) |
| 参数 | startEntity – 起始实例ID  endEntity – 末尾实例ID  options – 配置参数 |
| 返回 | 实体或事件始末实例的最短路径集合  Collection<CKCShortestPath> |
| 异常 | CKCApiException |

#### 计算实例的接近中心性

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的多个实体或事件实例的接近中心性（closeness）。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/closeness |
| 参数 | graphName：图的名称 |
| Body | {  “entities”: [“entity id”]  “options”: {  “direction”: “outbound/inbound/any”,  “weight”: “a relation attribute name”, （不指定，距离取1）  “entityFilter”: {},  “relationFilter”: {}  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “closeness”: {  “entity id1”: closeness1,  “entity id2”: closeness2,  …  },  “status”: “entities closeness retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的多个实体或事件实例的接近中心性（closeness）。 |
| 方法 | CKCApi.CKCGraph.closeness (  Collection<String> entityIds,  Map<String, Object> options  ) |
| 参数 | entityIds–实例ID集合  options – 配置参数 |
| 返回 | 实体或事件实例的接近中心性集合  Map<String, float> |
| 异常 | CKCApiException |

#### 计算实例的中介中心性

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 获取图的多个实体或事件实例的中介中心性（betweenness）。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/betweenness |
| 参数 | graphName：图的名称 |
| Body | {  “entities”: [“entity id”]  “options”: {  “direction”: “outbound/inbound/any”,  “weight”: “a relation attribute name”（不指定，距离取1）  }  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “betweenness”: {  “entity id1”: betweenness1,  “entity id2”: betweenness2,  …  },  “status”: “entities betweenness retrieved”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 获取图的多个实体或事件实例的中介中心性（betweenness）。 |
| 方法 | CKCApi.CKCGraph.betweenness (  Collection<String> entityIds,  Map<String, Object> options  ) |
| 参数 | entityIds–实例ID集合  options – 配置参数 |
| 返回 | 实体或事件实例的中介中心性集合  Map<String, float> |
| 异常 | CKCApiException |

## 图表示计算与推理接口

### 图定义接口

#### 图定义

|  |  |
| --- | --- |
| 功能 | 定义计算图的结构。 |
| 方法 | CKCApi.ComputeGraph(  int numEntities,  Collection<Map<String, String>> relations = null,  Map<String, CKCTensor> entityFeat = null,  Map<String, CKCTensor> relationFeat = null  ) |
| 参数 | numEntities：定义图的实体/事件实例数量  relations：关系集合  entityFeat：实体/事件实例特征  relationFeat：关系实例特征 |
| 返回 | 计算图对象  ComputeGraph |
| 异常 | CKCApiException |

#### 子图定义

|  |  |
| --- | --- |
| 功能 | 定义计算子图的结构。 |
| 方法 | CKCApi.ComputeSubGraph(  int numEntities,  Collection<Map<String, String>> relations = null,  Map<String, CKCTensor> entityFeat = null,  Map<String, CKCTensor> relationFeat = null,  Map<String, String> reindex=None  ) |
| 参数 | numEntities：定义图的实体/事件实例数量  relations：关系集合  entityFeat：实体/事件实例特征  relationFeat：关系实例特征  reindex:将父图实例ID映射到子图实例ID的字典 |
| 返回 | 计算子图对象  ComputeSubGraph |
| 异常 | CKCApiException |

#### 异构图定义

|  |  |
| --- | --- |
| 功能 | 定义异构图的结构。 |
| 方法 | CKCApi.HeterGraph(  int numEntities,  Collection<Map<String, String>> relations = null,  Map<String, CKCTensor> entityFeat = null,  Map<String, CKCTensor> relationFeat = null,  Collection<Map<String, String>> entityTypes=None  ) |
| 参数 | numEntities：定义图的实体/事件实例数量  relations：关系集合  entityFeat：实体/事件实例特征  relationFeat：关系实例特征  entityTypes：每个实体/事件实例的类型 |
| 返回 | 计算异构图对象  CKCHeterGraph |
| 异常 | CKCApiException |

#### 图封装

|  |  |
| --- | --- |
| 功能 | 进行图封装，提供虚拟图结构。 |
| 方法 | CKCApi.ComputeGraphWrapper(  String name,  String place,  Map<String, CKCTensor> entityFeat = null,  Map<String, CKCTensor> relationFeat = null,  ) |
| 参数 | name：图名称  place：使用GPU还是CPU  entityFeat：实体/事件实例特征  relationFeat：关系实例特征 |
| 返回 | 计算图封装对象  ComputeGraphWrapper |
| 异常 | CKCApiException |

#### 异构图封装

|  |  |
| --- | --- |
| 功能 | 进行异构图封装，提供虚拟异构图结构。 |
| 方法 | CKCApi.ComputeHeterGraphWrapper(  String name,  String place,  Map<String, CKCTensor> entityFeat = null,  Map<String, CKCTensor> relationFeat = null,  ) |
| 参数 | name：图名称  place：使用GPU还是CPU  entityFeat：实体/事件实例特征  relationFeat：关系实例特征 |
| 返回 | 计算异构图封装对象  ComputeHeterGraphWrapper |
| 异常 | CKCApiException |

### 图数据导入接口

|  |  |
| --- | --- |
| 功能 | 提供多种来源的图数据导入与图定义功能。 |
| 方法 | CKCApi.ComputeGraphLoader (  String type,  String source,  boolean symmetryEdges = true,  boolean selfLoop = true,  ) |
| 参数 | type：图数据来源类型  source：图数据来源地址  symmetryEdges：允许对称边  selfLoop：允许自连接边 |
| 返回 | 包含已定义图结构的图导入对象  ComputeGraphLoader |
| 异常 | CKCApiException |

#### 特征导入

|  |  |
| --- | --- |
| 功能 | 从图导入对象中获取实体的特征矩阵。 |
| 方法 | CKCApi.ComputeGraphLoader.getEntityFeature(  String entityID  ) |
| 参数 | entityID – 查询实体ID，如果为None则返回所有实体的特征矩阵 |
| 返回 | 图节点的特征矩阵  CKCTensor |
| 异常 | CKCApiException |

#### 邻接矩阵导入

|  |  |
| --- | --- |
| 功能 | 从图导入对象中获取邻接矩阵。 |
| 方法 | CKCApi.ComputeGraphLoader.getAdjacency() |
| 参数 | 无 |
| 返回 | 图数据的邻接矩阵  CKCTensor |
| 异常 | CKCApiException |

### 图神经网络表征接口

#### 图卷积神经网络GCN

|  |  |
| --- | --- |
| 功能 | 提供图卷积神经网络算法支持构建图向量表示模型。 |
| 方法 | CKCApi.Layers.gcn (  ComputeGraphWrapper graphWrapper,  CKCTensor feature,  int hiddenSize,  String activation  ) |
| 参数 | graphWrapper：封装计算图  feature：图初始化特征  hiddenSize：网络隐藏层数  activation：网络输出激活函数 |
| 返回 | 图表示张量  CKCTensor |
| 异常 | CKCApiException |

#### 图注意力神经网络GAT

|  |  |
| --- | --- |
| 功能 | 提供图注意力神经网络算法支持构建图向量表示模型。 |
| 方法 | CKCApi.Layers.gat (  ComputeGraphWrapper graphWrapper,  CKCTensor feature,  int hiddenSize,  String activation,  int headNum,  float featDrop,  float attentionDrop  ) |
| 参数 | graphWrapper：封装计算图  feature：图初始化特征  hiddenSize：网络隐藏层数  activation：网络输出激活函数  headNum：注意力头数  featDrop：特征Dropout率  attentionDrop：注意力Dropout率 |
| 返回 | 图表示张量  CKCTensor |
| 异常 | CKCApiException |

#### 图向量表征模型

|  |  |
| --- | --- |
| 功能 | 图向量表征模型，提供图的向量表征和动态更新。 |
| 方法 | CKCApi.GraphEmbedding (  ComputeGraphWrapper graphWrapper,  ComputeGraphLayer aggregateLayer,  CKCTensor feature,  int embedSize) |
| 参数 | graphWrapper：封装计算图  aggregateLayer: 图聚集函数  feature：图特征  hiddenSize：向量表征维数 |
| 返回 | 图向量表征模型  GraphEmbedding |
| 异常 | CKCApiException |

#### 图节点向量表征获取

|  |  |
| --- | --- |
| 功能 | 返回当前图的实体向量表征。 |
| 方法 | CKCApi.GraphEmbedding.getEntityEmbedding(NodeId nodeId=Null) |
| 参数 | nodeId： 查询实体Id,如果为Null则返回所有实体的向量表征 |
| 返回 | 图的实体向量表征  CKCTensor |
| 异常 | CKCApiException |

#### 图关系向量表征获取

|  |  |
| --- | --- |
| 功能 | 返回当前图的关系向量表征。 |
| 方法 | CKCApi.GraphEmbedding.getRelationEmbedding(RelationId relationId=Null) |
| 参数 | relationId： 查询关系Id,如果为Null则返回所有关系的向量表征 |
| 返回 | 图的关系向量表征  CKCTensor |
| 异常 | CKCApiException |

#### 图表征局部动态更新

|  |  |
| --- | --- |
| 功能 | 根据局部图结构的改变进行局部表征更新 |
| 方法 | CKCApi.GraphEmbedding.localUpdate(  String type,  Relation relation,  ) |
| 参数 | type：更新类型，增加/修改/删除  relation：更新的关系，包含头尾实体id和关系类型 |
| 返回 | 无 |
| 异常 | CKCApiException |

#### 图表征全局更新

|  |  |
| --- | --- |
| 功能 | 根据图结构的改变进行全局表征更新 |
| 方法 | CKCApi.GraphEmbedding.globalUpdate() |
| 参数 | 无 |
| 返回 | 无 |
| 异常 | CKCApiException |

### 图推理应用接口

#### 实体或事件分类模型训练

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 创建并训练一个节点多标签分类模型。 |
| 方法 | CKCApi.NodeClassifier(  DataLoader trainingData,  GraphEmbedding graphEmbedding,  Boolean embeddingTrainable=False  ) |
| 参数 | trainingData：训练数据加载器  graphEmbedding：图表征模型  embeddingTrainable：图表征是否为可训练参数（图表征在训练结束后会改变） |
| 返回 | 节点多标签分类模型和对应ID |
| 异常 | CKCApiException |

#### 实体或事件分类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 以分类模型标识为参数，为一个新的实体或事件进行多标签分类。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/entity/{entityId}/multi-label |
| 参数 | graphName：计算图的名称  entityId：实体或事件的ID |
| Body | {  “task”: “task id” （多标签分类模型ID）  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “labels”: [  “class label 1”,  “class label 2”,  …  ],  “status”: “multi-label prediction finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 以分类模型标识为参数，为一个新的实体或事件进行多标签分类。 |
| 方法 | CKCApi.ComputeGraph.multiLabel(  String entityId  String taskId  ) |
| 参数 | entityId–实例ID  taskId – 实例分类任务模型ID |
| 返回 | 实体或事件实例的分类标签集合  Collection<String> |
| 异常 | CKCApiException |

#### 相似实体或事件搜索

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 利用图的特征向量表示，找出跟某个实例相似的实例集合。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/entity/{entityId}/similarity |
| 参数 | graphName：计算图的名称  entityId：实体或事件的ID |
| Body | {  “topK”: “2”  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “similarEntities”: [  {  “entity”: “entity id1”,  “similarity”: 0.99  },  {  “entity”: “entity id2”,  “similarity”: 0.92  }  ],  “status”: “similar entities search finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 利用图的特征向量表示，找出跟某个实例相似的实例集合。 |
| 方法 | CKCApi.ComputeGraph.similarity(  String entityId  int topK  ) |
| 参数 | entityId–实例ID  topK – 返回前K个最相似的实例 |
| 返回 | 前K个最相似实例集合  Collection<Map<String, Object>> |
| 异常 | CKCApiException |

#### 实体关系预测模型训练

|  |  |
| --- | --- |
| 功能 | 创建并训练一个实体关系预测模型。 |
| 方法 | CKCApi.RelationClassifier(  DataLoader trainingData,  GraphEmbedding graphEmbedding,  Boolean embeddingTrainable=False,  Boolean pathRequire=False  ) |
| 参数 | trainingData：训练数据加载器  graphEmbedding：图表征模型  embeddingTrainable：图表征是否为可训练参数（图表征在训练结束后会改变）  pathRequire：关系预测结果是否需要给出推理路径 |
| 返回 | 实体关系预测模型和对应ID |
| 异常 | CKCApiException |

#### 实体关系预测

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 以关系预测模型标识为参数，为一对新的实体或事件进行关系预测。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/relation-prediction |
| 参数 | graphName：计算图的名称 |
| Body | {  “entity1”: “entity id1”,  “entity2”: “entity id2”,  “task”: “task id” （关系预测模型ID）  “topK”: 2,  “pathRequire”: False  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “relationPrediction”: [  {  “relation”: “relation class1”,  “from”: “entity id1”,  “to”: “entity id2”,  “confidence”: 0.99,  “path”:None  },  {  “relation”: “relation class2”,  “from”: “entity id1”,  “to”: “entity id2”,  “confidence”: 0.92,  “path”:None  }  ],  “status”: “relation prediction finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 以关系预测模型标识为参数，为一对新的实体或事件进行关系预测。 |
| 方法 | CKCApi.ComputeGraph.relationPrediction(  String entityId1,  String entityId2,  String taskId,  int topK,  Boolean pathRequire=False  ) |
| 参数 | entityId–实例ID  taskId - 关系预测模型ID  topK – 返回前K个最可能的关系  pathRequire – 是否返回推理路径结果 |
| 返回 | 前K个最可能的关系集合  Collection<Map<String, Object>> |
| 异常 | CKCApiException |

#### 子图模式搜索

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 输入一个局部子图，在全局图中检索相似特征模式的其他子图。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/pattern-search |
| 参数 | graphName：计算图的名称 |
| Body | {  “pattern”: {  “entities”: [“entity id”],  “relations”: [{relation}]  },  “topK”: 2  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “patterSearch”: [  {  “entities”: [“entity id”],  “relations”: [{relation}],  “similarity”: 0.85  },  {  “entities”: [“entity id”],  “relations”: [{relation}],  “similarity”: 0.77  }  ],  “status”: “pattern search finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 输入一个局部子图，在全局图中检索相似特征模式的其他子图。 |
| 方法 | CKCApi.ComputeGraph.patternSearch(  CKCSubGraph pattern,  int topK  ) |
| 参数 | pattern – 模式子图  topK – 返回前K个最可能的关系 |
| 返回 | 前K个最相似的模式子图集合  Collection<CKCSubGraph> |
| 异常 | CKCApiException |

#### 图的聚类

* + HTTP接口

|  |  |
| --- | --- |
| 功能 | 根据图的向量表示以高内聚低耦合的标准将图分为若干个子图。 |
| 方法 | HTTP POST /\_api/CKC/graph/{graphName}/clusters |
| 参数 | graphName：计算图的名称 |
| Body | {  “clusterNum”: 3  } |
| 返回 | {  “code”: “response code”,  “message”: “response code message”,  “result”: {  “clusters”: [  {  “entities”: [“entity id”],  “relations”: [{relation}],  },  {  “entities”: [“entity id”],  “relations”: [{relation}],  }  ],  “status”: “graph clustering finished”  }  } |

* + 编程接口

|  |  |
| --- | --- |
| 功能 | 根据图的向量表示将图分为若干个子图。 |
| 方法 | CKCApi.ComputeGraph.clusters(  int clusterNum  ) |
| 参数 | clusterNum – 待聚类出的子图数量 |
| 返回 | 子图集合  Collection<ComputeSubGraph> |
| 异常 | CKCApiException |

#### 主动学习包装器

|  |  |
| --- | --- |
| 功能 | 使一个节点分类模型或关系预测模型成为可主动请求新数据的模型 |
| 方法 | CKCApi.ActiveLearner(  ComputeGraph basicLearner  ) |
| 参数 | basicLearner – 基本学习模型 |
| 返回 | 主动学习模型  ActiveLearner |
| 异常 | CKCApiException |

#### 主动学习模型接收新数据

|  |  |
| --- | --- |
| 功能 | 使一个节点分类模型或关系预测模型成为可主动请求新数据的模型 |
| 方法 | CKCApi.ActiveLearner.feed(  DataLoader trainingData  )· |
| 参数 | trainingData – 新的训练数据 |
| 返回 | 无 |
| 异常 | CKCApiException |

#### 主动学习模型请求新数据

|  |  |
| --- | --- |
| 功能 | 使一个节点分类模型或关系预测模型成为可主动请求新数据的模型 |
| 方法 | CKCApi.ActiveLearner.require(  int batchNum  ) |
| 参数 | batchNum – 请求标注数据的数量 |
| 返回 | 请求新标注数据  Collection<Data> |
| 异常 | CKCApiException |

#### 导出当前学习模型

|  |  |
| --- | --- |
| 功能 | 使一个节点分类模型或关系预测模型成为可主动请求新数据的模型 |
| 方法 | CKCApi.ActiveLearner.basicLearner() |
| 参数 | 无 |
| 返回 | 基本学习模型  ComputeGraph |
| 异常 | CKCApiException |

## 人机交互接口

智能问答应用基于以上各课题知识学习与计算接口的交互逻辑：

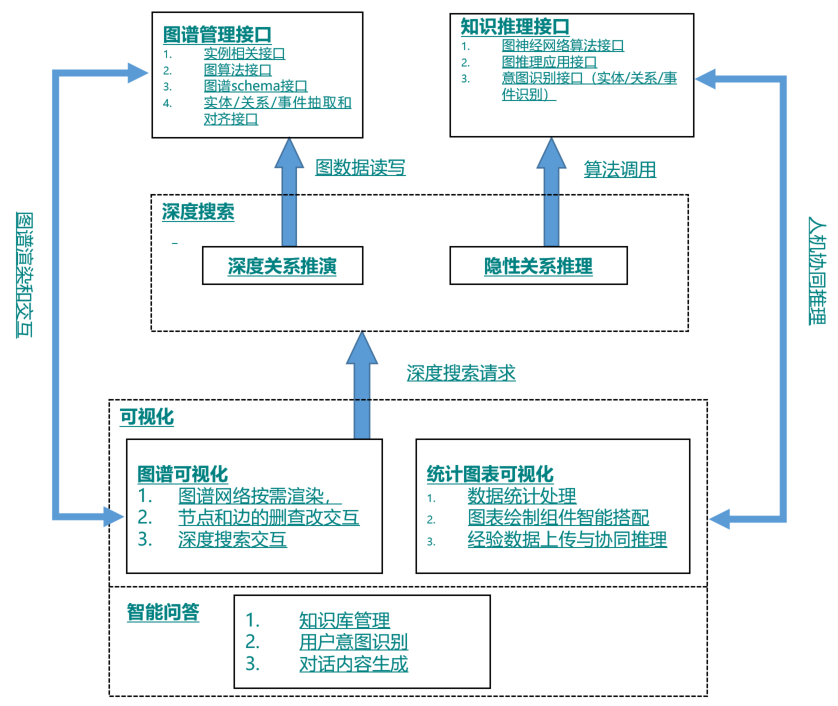


图2-1 智能问答人机交互与系统内部接口调用逻辑