金融知识图谱设计文档

1. 需求分析

1.1 背景

随着金融数据的大规模增长，传统数据模型已经不能满足现在的需求。知识图谱作为大规模语义网，可以将大规模数据以三元组的形式构建，其丰富的语义信息对金融风险预警，推荐系统都具有重要意义。

1.2 需求概述

通过提供的金融数据，构建一个大规模金融知识图谱，图谱满足跨语言要求，增量更新，金融风险预警等功能。

1. 系统用例建模

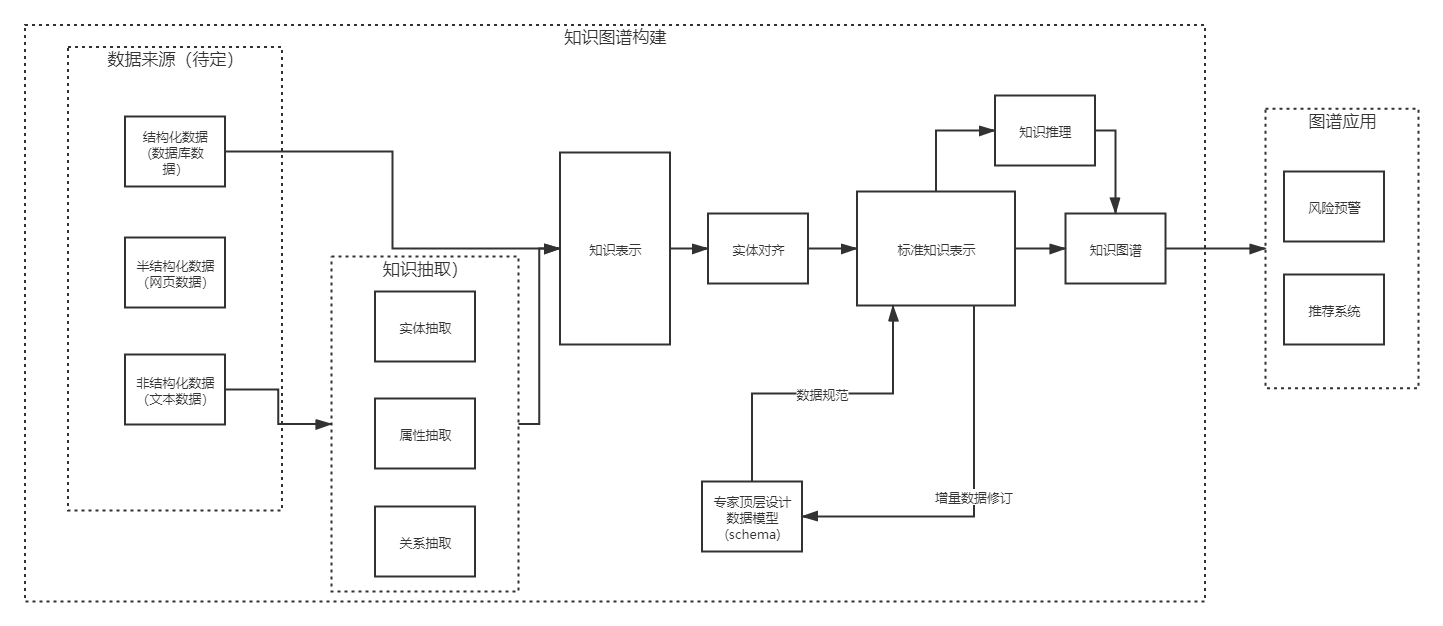
2.1 功能描述

图2-1：知识图谱构建流程

根据需求，对知识图谱的设计包含如上几个功能步骤。 数据类型未定,所以暂时将三种数据的清洗都书写，后续拿到数据再更新。

1. 数据清洗

针对结构化数据经过sql语句构建需要的数据模式。半结构化数据主要是网页的百科数据，采用编写正则爬虫抽取。非结构数据基本为纯文本数据，需要经过知识抽取（实体抽取，关系抽取）。（计划采用Baidu开源工具LAC完成）

1. 知识融合

知识融合为将上述三个不同的知识来源进行实体对齐，知识消融。以统一的表示形式进行表示存储。（方法待定，通过相似度计算，聚合，聚类等技术来实现）

1. 数据模型设计

作为领域知识图谱，需要由专家设计数据模式，定义实体类型，关系类型，属性类型。（暂定）

1. 知识推理

图谱构建中会存在实体关系缺失，为进一步完善图谱的完整性，需要进一步进行图谱推理，表现为图谱补全。（预计采用基于GCN和基于翻译距离的方法）

1. 增量数据导入

对于增量数据的导入和首次导入相同，需要经过数据验证等环节。

1. 知识图谱应用

风险预警，主要根据图谱的推理，发现冲突。对金融市场出现的风险及时提醒。

推荐系统，传统推荐算法以协同过滤算法为先。知识图谱有大量的先验知识，作为推荐系统的工作分为三大类，其一是基于图谱的嵌入翻译距离（Trans），其二是基于连接的方法，其三是根据基于结合关系的传播（GNN）。

1. 系统设计

3.1 总体设计

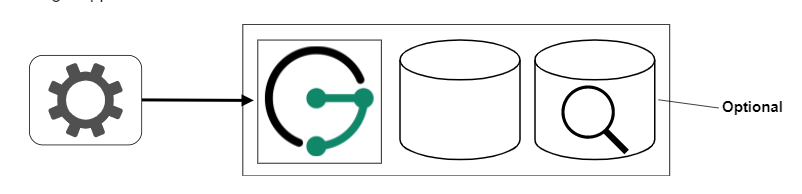
3.1.1 开发语言

Java1.8,python3.7

3.1.2 数据库

知识图谱的存储主要是以RDF格式或者图数据库。本次设计使用完全开源的图数据库janusgraph，搭配hbase和elasticsearch。

测试部署如下



3.1.3 IDE选择

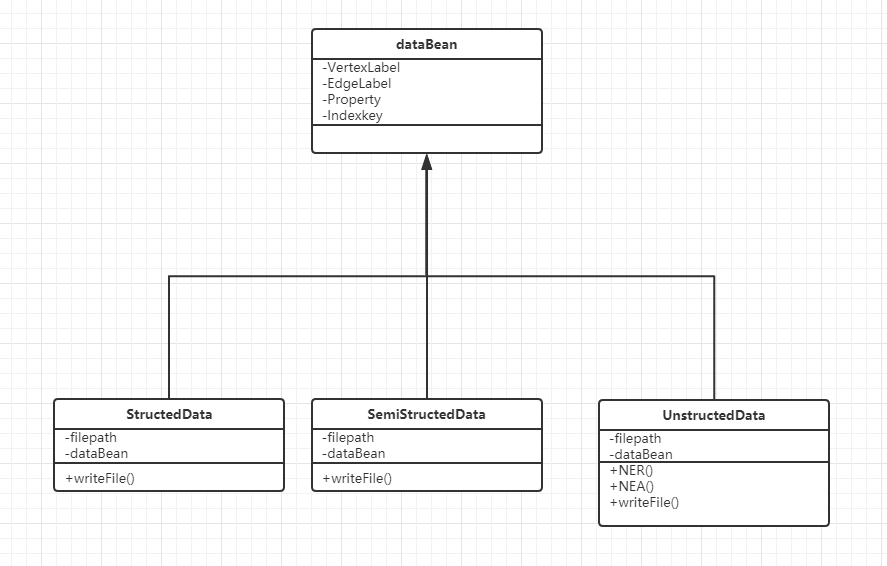
Idea+vscode

1. 模块详细设计

4.1 数据清洗

dataBean

|  |  |
| --- | --- |
| 数据类型 | 说明 |
| VertexLabel | 节点标签，包含类别名 |
| EdgeLabel | 边标签，包含类别名和多元关系标识 |
| PropertyKey | 包含属性名，数据类型 |
| IndexKey | 包含索引名，索引方式，Property |



4.2 知识融合

待补充

4.3 数据导入

使用java编写，通过导入janusgraph server形式导入。

|  |  |
| --- | --- |
| Package | 包说明 |
| Generator | 根据json生成数据csv文件和schema,json，以及数据映射dataMaping.json |
| Importer | 根据上述生成的三个文件导入数据 |
| Schema | 导入schema |

Generator Json数据格式解析

Demo:

{

  "VertexTypes": [

    {

      "name": "User",

      "columns": {

        "name": {"dataType":"String", "dataSubType":"name", "composit":true}

      },

      "row": 150

    },

  ],

  "EdgeTypes": [

    {

      "name": "Follows",

            "multiplicity": "SIMPLE",

            "selfRef": false,

      "relations": [

        {"left": "User", "right": "User", "row": 10000, "selfRef":false,  "supernode":{"vertices":1, "edges":50}  }

      ]

    },

  ]

}

Schema.json

{

  "propertyKeys" : [ {

    "name" : "node\_id",

    "dataType" : "Integer",

    "cardinality" : "SINGLE"

  } ],

  "vertexLabels" : [ {

    "name" : "User",

    "partition" : false,

    "useStatic" : false

  } ],

  "edgeLabels" : [ {

    "name" : "Follows",

    "multiplicity" : "SIMPLE",

    "signatures" : null,

    "unidirected" : false

  }],

  "vertexIndexes" : [ {

    "name" : "node\_id\_comp",

    "propertyKeys" : [ "node\_id" ],

    "composite" : true,

    "unique" : true

  }],

  "edgeIndexes" : [ {

    "name" : "Date\_comp",

    "propertyKeys" : [ "Date" ],

    "composite" : true,

    "unique" : false

  } ],

  "vertexCentricIndexes" : [ ]

}

Datamapper.json

{

  "vertexMap" : {

    "Tweet.csv" : {

      "[VertexLabel]" : "Tweet",

      "text" : "text",

      "node\_id" : "node\_id"

    },

  },

  "edgeMap" : {

    "Tweet\_Mentions\_User\_edges.csv" : {

      "[edge\_left]" : {

        "Left" : "Tweet.node\_id"

      },

      "[EdgeLabel]" : "Mentions",

      "[edge\_right]" : {

        "Right" : "User.node\_id"

      }

    }

  }

}

具体操作见github

4.4 知识推理

待补充

4.5 图谱应用

待补充