

图可视化解决方案 知识图谱

阿里巴巴集团
蚂蚁集团

2020.11.22

图可视化解决方案：知识图谱

01.摘要

知识图谱是一种揭示类型或实体之间关系的语义网络，其建立的具有语义处理能力与开放互联能力的知识库，可在安全风控、健康、智能问答等智能信息服务中产生应用价值。此篇白皮书侧重在图可视化领域，详细讨论知识图谱在建设过程中（如知识构建、知识编辑、知识计算、领域图谱等）遇到的诸多图可视化问题以及相应提出来的解决方案。

02.背景介绍

2.1 定义

知识图谱是一种基于图的数据结构构成的语义网络，它以结构化的形式描述客观世界中类型、实体及其关系。实体是客观世界中的事物，本体类型是对具有相同属性的事物的概括和抽象。2012年5月17日，Google 正式提出了知识图谱（Knowledge Graph）的概念，其为了优化搜索引擎返回的结果，增强用户搜索质量及体验。

如下图所示，这是在 google 上搜索钱学森夫人的搜索结果（图左），通过知识图谱技术，知识抽取，我们可以知道抽取为2个实体“钱学森”与“蒋英”，关系为“夫人”，可视化结果（图右）



左图

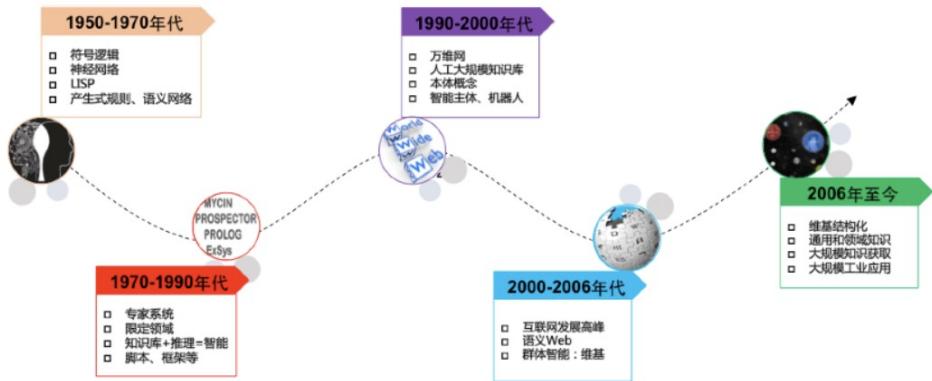


右图

2.2 发展历史

知识图谱的发展是人工智能重要分支知识工程在大数据环境中的成功应用^[1]

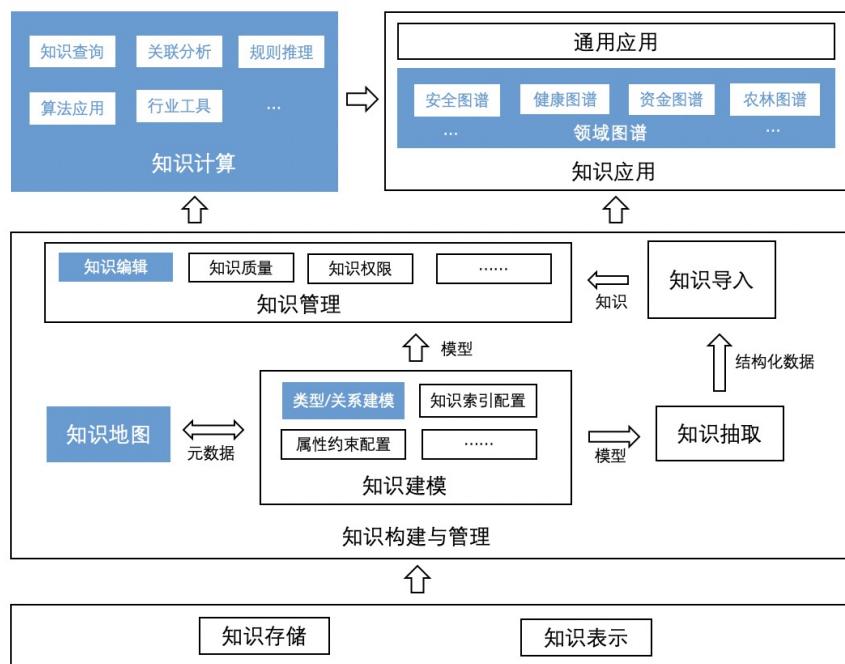
知识工程的五个标志性的阶段：前知识工程时期、专家系统时期、万维网 1.0 时期、群体智能时期及知识图谱时期。从 2006 年开始，大规模维基百科类富结构知识资源的出现和网络规模信息提取方法的进步，使得大规模知识获取方法取得了巨大进展。再到2012年google正式推出知识图谱概念，知识图谱已发展成为语义 Web、自然语言处理和机器学习等的交叉学科。



在当前下的大数据时代，知识图谱从互联网大数据中获得知识、同时又将知识提供给互联网，这是一个迭代的相互增强过程，可以实现从互联网信息服务到智能知识服务的跃迁。

2.3 图谱建设

知识图谱的建设包括知识表示、知识存储、知识建模、知识获取、知识管理、知识计算、知识应用等^[2]，整体架构如下（蓝色高亮部分为前端图可视化参与的阶段）：



由上图看出，图可视化在知识图谱的建设中参与了知识构建、知识编辑、知识计算、领域图谱等阶段：

- 知识构建：
 - 知识建模：建立知识图谱的数据模型和知识的表达方式，即构建Schema^[3]的过程；
 - 知识地图：业务域下知识概览即业务下的知识元数据中心；
- 知识编辑：是对知识进行增删改查的过程，方便业务对单个知识进行变更；
- 知识计算：也称分析推理，基于已构建的知识图谱进行能力输出的过程；
 - 知识查询：是对知识进行查询、分析的过程；包括单点查询和Gremlin查询等；

- 关联分析：输入两个或者多个实体，查看之间是否有边的链接；
- 规则推理：通过配置将一些经验规则化，来推导图谱中的关系，可用于做图谱挖掘；
- 领域图谱：安全图谱、农林图谱、资金图谱、健康图谱等。

03 问题分析

根据 2.3 总结的图谱建设架构图，我们将每个流程里的产品需求，细分拆解，讨论其中遇到的可视化问题，以及我们相应的解决方案，总结为下表：

图谱建设流程	产品功能	细分需求	问题	细分方案
知识表式		暂不在图可视化的讨论范围		
		暂不在图可视化的讨论范围		
		暂不在图可视化的讨论范围		
		暂不在图可视化的讨论范围		
知识构建	知识地图	看全大图	节点和边太多看不清	大图布局
		渲染快	白屏时间长	布局时间优化、动画时间优化
		快速定位	很难在大图中快速寻找探索出发点	搜索能力、核心节点、过滤能力
	知识建模	本体类型表达/编辑	节点和边多、定位难、难区分、承载信息有限	看清画布 看清节点/边
		本体关系表达/编辑		
	知识管理	实体/关系的编辑		
知识计算	知识查询	查询过程可持续	数据变化时画布不稳定	画布稳定
		查询过程可看清	数据过多时看不清	画布看清
		试错能力	若误操作，每次得重头再来	支持撤销重做
	关联分析	路径展示	不同路径展示区分	路径颜色、粗细
		路径筛选	路径太多，无法筛选路径	路径度数筛选
		数据联动	路径详情和图的展示割裂	路径联动高亮
	规则推理	图结构定义	代码形式去定义，不直观、使用成本高	图可视化规则推理
		规则定义		
知识应用	领域图谱	差异性展示分析	布局不灵活	自主布局

3.1 知识建模不直观且效率低

- 传统的知识建模依赖 Schema 管理，表达信息有限，且只能从上至下进行表达，不能同时进行横向和纵向的表达，也无法直接看到本体类型^[4]之间的关系、不直观；
- 传统的本体类型和本体关系^[5]的新建/编辑，是通过在图之外填写相关信息，导致了展示和编辑相互割裂、效率非常低，用户不能从 schema 的展示中直接发现操作点并且直接定位进行操作。

3.2 知识地图展示与性能问题

知识地图是业务域下的知识概览，涵盖了业务域内的所有知识，节点和边非常多，目前有两大痛点：

- 大图渲染慢：节点数据较多时图渲染的速度较慢、白屏时间较长；
- 大图很难看清：进入知识地图就展示所有的节点和边，用户很难看清全局；用户很难快速寻找关注点进行探索。

3.3 分析推理持续性分析存在问题

早期通用的知识分析止步于一次性的知识查询，用户只能根据一定的条件去检索到相应的实体和关系。但是用户基于知识图谱的分析推理其实并不是一次就能完成的，它是一个循序的不断探索的过程，并且需要在看清的基础上一步步发现，这就需要图可视化分析具有较高的可持续分析的能力。

3.4 图谱需要自主灵活的布局

不同领域的图谱之间，以及同一领域的图谱不同场景的分析之间，由于数据的不同，对布局都有着不同的要求。例如领域知识图谱是由领域数据构成的图谱，相对于通用知识图谱，更多的是结合了很多业务的具体场景。例如企业数据的图谱，它的投资关系是具备上下游的，有向分层布局更为合适；而一个团伙骗贷图谱，它用同心圆布局可能更为合适。如何优化布局，找到实际数据最适合的布局，是目前图分析中一个巨大的挑战。

04 解决方案

上述问题分析与解决方案相对应的索引表如下：

业务问题	可视化面临挑战	解决方案
知识构建	3.1 知识建模不直观且效率低	4.1 可视化知识建模
	3.2 知识地图展示与性能问题	4.2 知识地图方案
分析推理	3.3 分析推理持续性分析存在问题	4.3 可视分析推理
知识应用	3.4 图谱需要自主灵活的布局	4.4 自动布局方案

4.1 可视化知识建模

知识建模是指建立知识图谱的数据模型和采用什么样的方式来表达知识，即构建schema的过程，用户可以定义本体类型及他们之间的关系。

4.1.1 模型展示

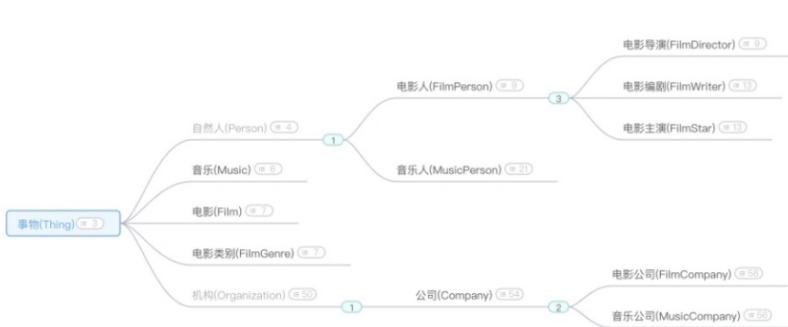
模型展示即对schema的展示。schema是知识结构的一种定义，即知识的元数据，包含了本体类型和本体关系。

4.1.1.1 看清全局

本体类型：学术界称为本体（Ontology），类似与java代码中定义Class，用来定一个实体的元数据信息，比如企业是一个本体类型，其包含企业名称、法人代表、注册地等属性。

本体关系：指本体类型之间的关系，如本体类型-企业和本体类型-自然人之间有一个实控人的关系。

- 图选型：使用AntV树图^[6]来进行本体类型的横向和纵向的信息表达；使用AntV关系图^[7]来进行本体关系的表达，同时通过一些交互手段：画布的缩放，拖拽；节点的折叠，展开，拖拽；节点搜索定位，帮助用户看清全局。



左图（本体类型）



右图（本体关系）

4.1.1.2 看清类型

- 节点样式：节点颜色由实体对应的本体类型决定，一种颜色代表一个类型；同时可以通过自定义节点：对排名topN的类型进行icon定制以增加实体类型识别度；



- 节点信息承载：tooltip、定制属性icon、右键菜单面板等；

4.1.1.2 看清关系

- 边样式：颜色由本体关系的类别决定，一种颜色代表一种关系类别：



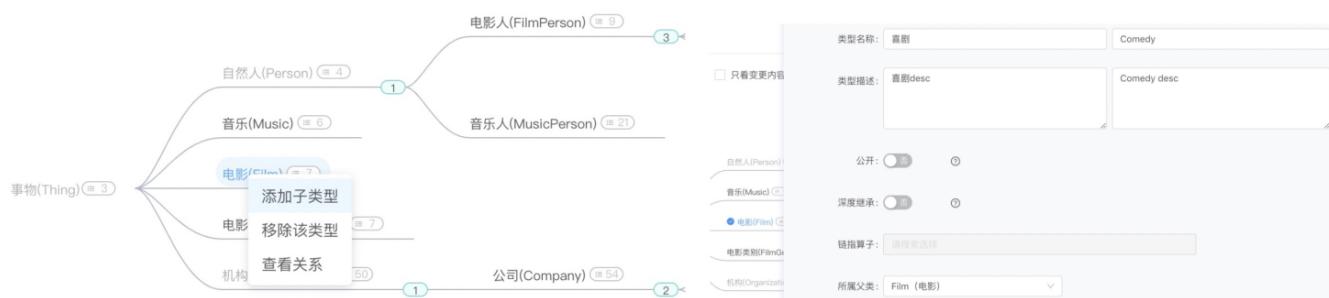
- 边信息承载：tooltip、右键菜单面板等；

4.1.2 模型schema编辑

模型编辑即对schema的编辑，包括本体类型的编辑和本体关系的编辑。

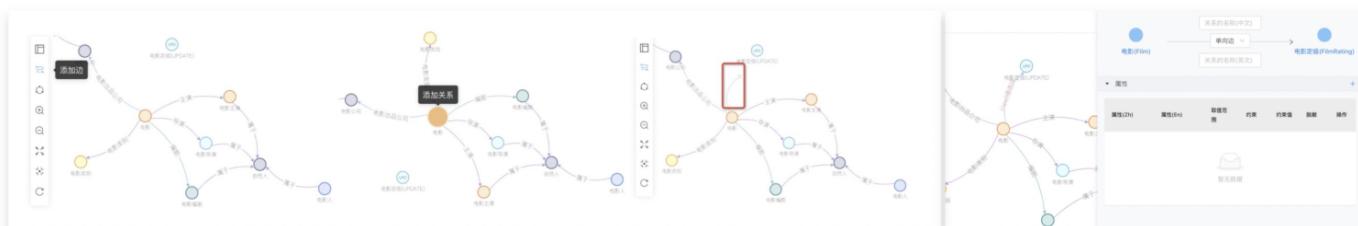
4.1.2.1 本体类型编辑

选择某一类型，右键菜单进行操作（添加子类型/修改类型/删除类型）；



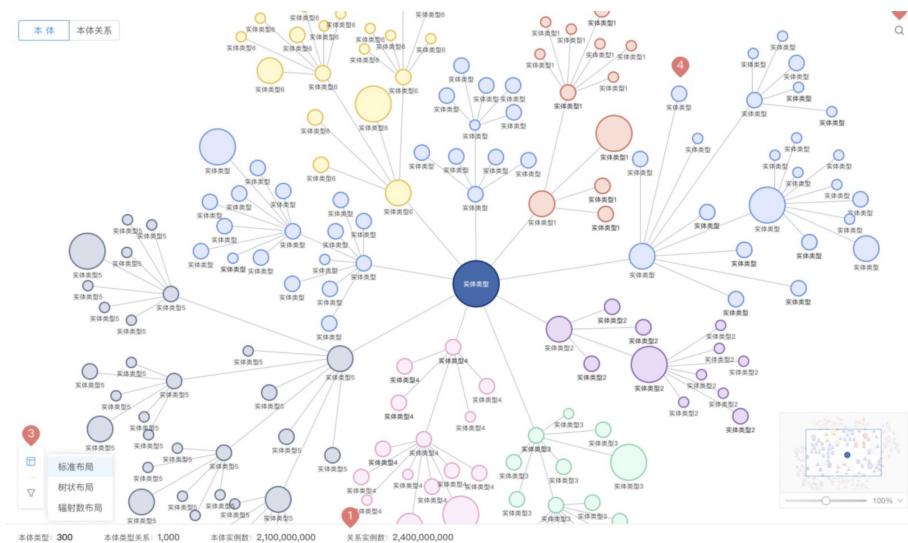
4.1.2.2 本体关系编辑

画布操作两个点之间拖拽连线形成边，并弹出右侧面板进行信息填写；选中节点右键进行本体关系的编辑/删除。



4.3 知识地图方案

知识地图的出发点是共建业务域下的一张图，它是业务域下的知识概览即业务下的知识元数据中心，除了支撑每个业务的领域图谱之外，也能做到跨领域图谱共享（业务公布的数据）。可以辅助知识建模，为知识运维（知识指标统计、连通度统计等，让知识质量白盒化）和知识应用提供了很好的数据支撑。



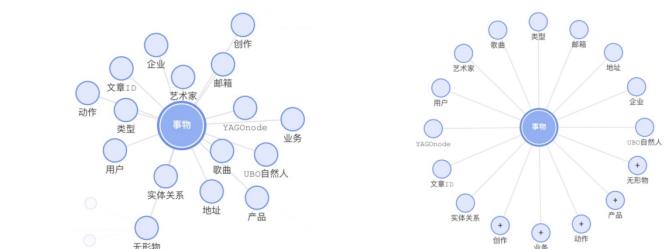
4.2.1 产品方案

- 大图渲染时效：缩短白屏时间；
- 能看清大图：节点和边非常多的情况下，帮助用户快速看清、快速寻找探索的出发点
 - 提供多种布局展示：从多种方式看清大图；
 - 提供节点搜索能力：通过搜索高亮聚焦到对应的节点，快速定位发现；
 - 定义核心节点：可根据业务规则定义大图中的核心节点，默认高亮聚焦到这些节点；
 - 提供筛选能力：可根据业务域项目等进行节点和边的筛选，过滤提取有效信息。

4.2.2 技术方案

4.2.2.1 大图布局

针对本体类型的大图，可以采用多种布局算法进行布局，默认折叠一些类型的子节点，从而从多个角度和方式帮助用户看清。



图一: d3-force

图二: G6辐射树dendrogram



图三: G6紧凑树compactBox

4.2.2.2 渲染时效

- 布局时间优化：减少不必要的节点位置x、y的计算（节点位置计算需要耗时）；
- 动画时间优化：做完节点的布局，原来所有节点都会默认开启一些动画，动画结束后才会渲染出视图。为提高渲染效率，可去掉一些不必要的动画。

4.3 可视分析推理

图谱的分析推理是基于实体和它们之间关系的探索，包含知识查询、关联分析、DSL等子方向，它的业务价值是辅助做策略分析，基于分析的结果去落地。

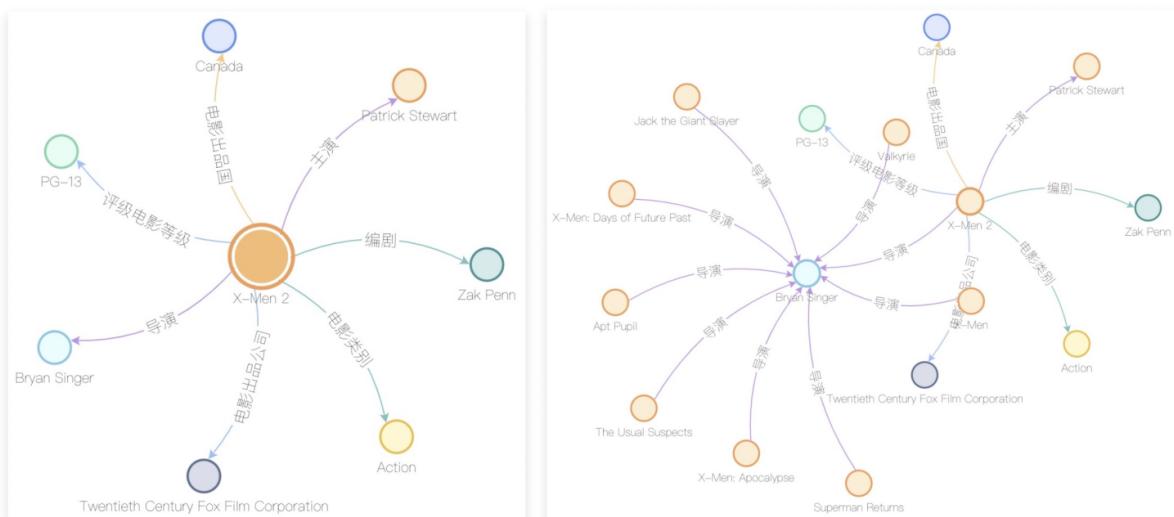
增强分析推理可持续分析的能力，可以从以下方面出发：

4.3.1 数据变化保持画布稳定

基于某个实体去展开它的N度关系、关系预测等，每次操作都会影起数据的变化。而画布的布局则是根据数据去计算其中节点位置的x、y得到的。如果不固定住已有布局，会造成以下问题：

- 如力导布局等每次布局出来的节点位置都是随机的，用户在画布中每次操作完都会重新去寻找之前关注的节点；
- 用户之前对节点的拖动失效；

基于解决以上问题，我们每次布局应该尽可能去实现增量布局，缓存已有节点的位置，以保持图整体布局的稳定性。



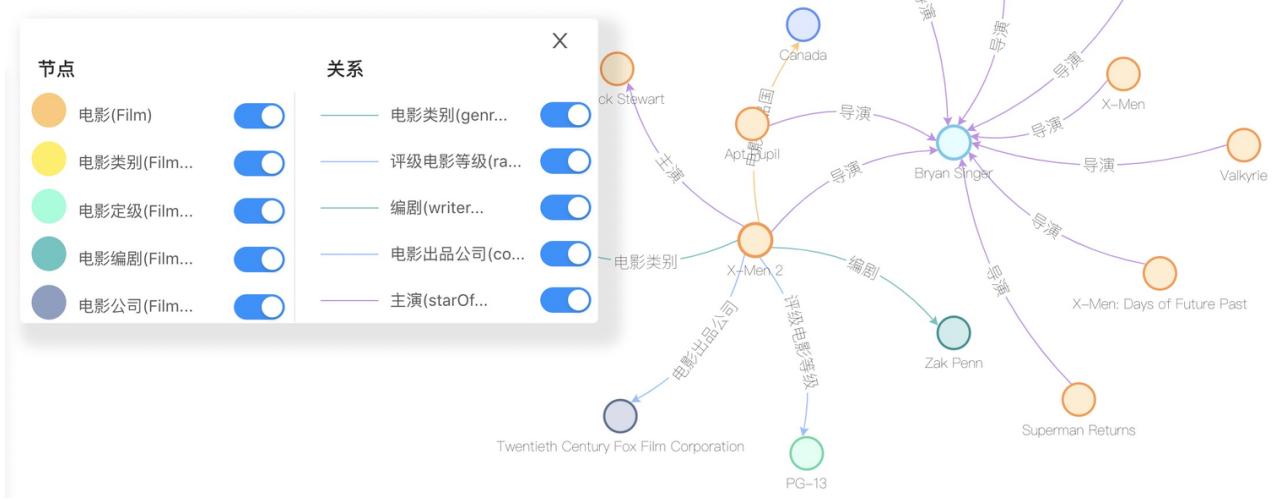
左图：由 X-Men2 扩散出的图

右图：由 Bryan Singer 扩散出的图

4.3.2 数据过多保持画布看清

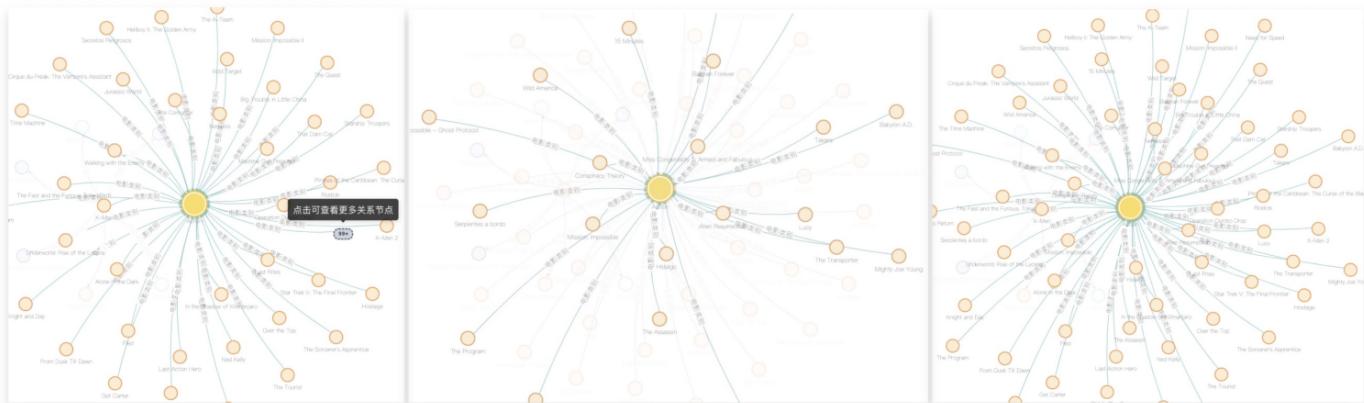
如果数据本身包含了很多的节点和边，或者某一次的查询展开操作，新增的子节点过多时，会比较乱、影响整个画布的查看。保持画布看清可以从以下角度进行处理：

4.3.2.1 支持节点和边的筛选过滤



4.3.2.2 扩展节点

在某节点展开一度关系后的新增节点超过一定数量时，超过数量的节点隐藏，点击扩展节点后展示隐藏的节点，扩展节点内展示被隐藏节点的数量（数量超过N时显示N+），点击查看全部节点；



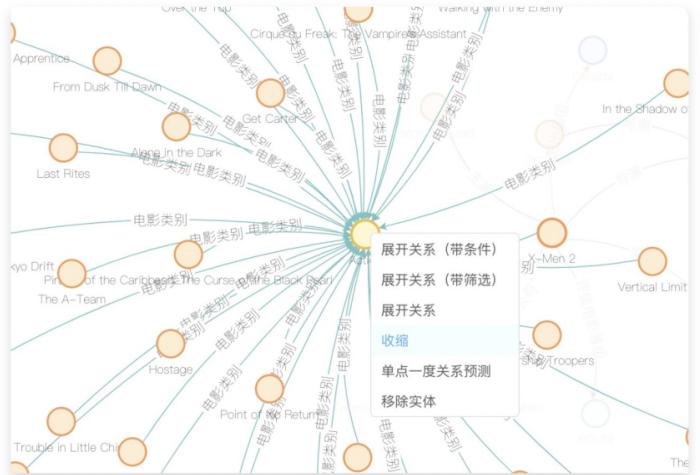
图一：点击查看更多前

图二：点击查看更多以后

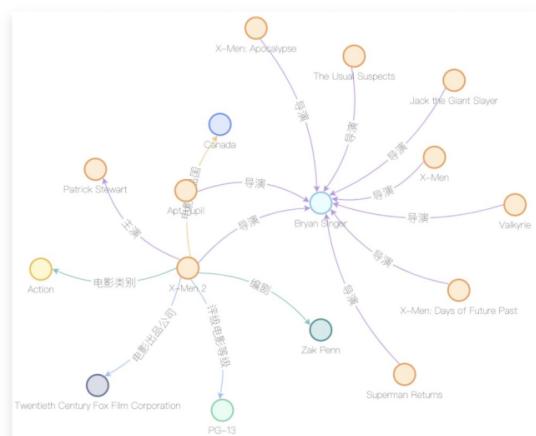
图三：取消当前高亮查看所有子节点

4.3.2.3 节点交互

收缩/展开节点（隐藏/显示某个节点的子节点）：可以在查看完该节点的这些子节点后选择“收缩”节点，将其子节点折叠收缩起来，从而还原一个相对干净的画布，帮助用户继续进行其他探索；



左图：节点收缩前

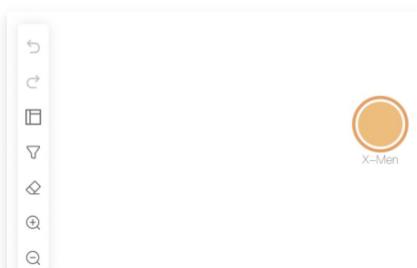


右图：节点收缩后

4.3.3 提供试错能力、支持撤销重做

用户在分析的过程中是连续性的，需要去探索，如果因为一次错误的探索操作就要一切从头再来，这样的效率是极低的。我们需要在平台层提供试错能力，可撤销、重做，提升业务的探索效率，从交互能力上让业务一步步的看清。

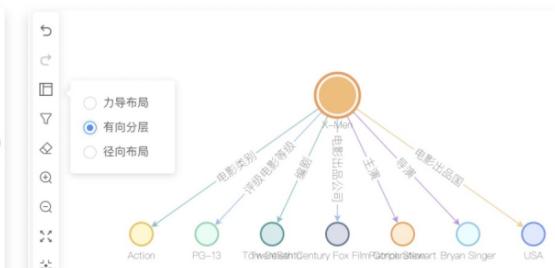
我们需要对数据变化和用户在画布上的其它交互操作，都提供试错能力。在图组件层把这些差异化抹平，让用户不需要感知某次操作是数据交互还是状态变更还是仅仅画布操作，从而留出更多的时间和精力去回归业务的本质。



图一：初始状态



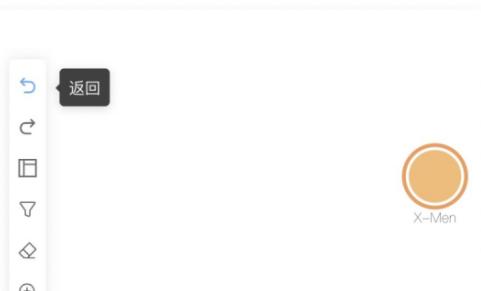
图二：展开一度关系



图三：切换布局



图四：返回



图五：返回



图六：撤销返回

4.3.3.1 产品方案

为了方便历史状态的记录，我们可以将用户交互事件分为以下3种类型：

- 只涉及画布和布局操作，无状态变更、无数据变更
 - 拖动画布
 - 缩放画布
 - 拖动节点
 - 切换布局
- 涉及状态变更
 - 点击画布
 - 单击/双击节点
 - 单击/双击边
 - 收缩/展开节点
- 涉及数据变化
 - 数据的整体更新
 - 单个节点的增/删/改
 - 单个边的增/删/改

4.3.3.2 技术方案

- step1：画布工具条露出撤销/重做（返回/撤销返回）的按钮；
- step2：图组件拦截过滤用户的交互及数据变化触发的事件，拦截以后提供状态管理（HistoryController）来存储历史状态；
- step3：HistoryController提供撤销重做的api来支持业务层的调用执行撤销/重做；
- step4：HistoryController每次状态变化，通过冒泡事件来通知业务层；

4.3.4 关联分析路径区分

关联分析用于分析两个实体（或两个实体组）之间的路径，关联分析可以帮助发现异常模式（例如环路刷单，循环持股）并反馈给用户。

路径结果 (2)

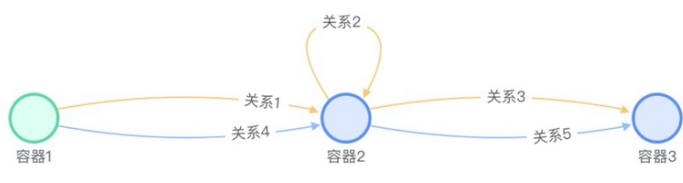
全部 展开全部:

▼ 路径1 3度

```
graph TD; A((容器1)) -- 关系1 --> B((容器2)); B -- 关系2 --> C((容器2)); C -- 关系3 --> D((容器3))
```

▶ 路径2 2 度

```
graph TD; A((容器1)) -- 关系4 --> B((容器2)); B -- 关系1 --> C((容器2)); C -- 关系3 --> D((容器3)); B -- 关系5 --> D
```



怎么看清关联分析的路径：

- 支持路径的排序、筛选、按组展示等（左图）；
- 路径渲染样式区分（根据不同路径定制对应的边的颜色、根据路径优先级定制路径对应的边的粗细）（右图）；
- 图中路径与图外的交互：鼠标hover左侧路径，右侧图中讲对应路径进行高亮等；

4.3.5 可视化规则推理

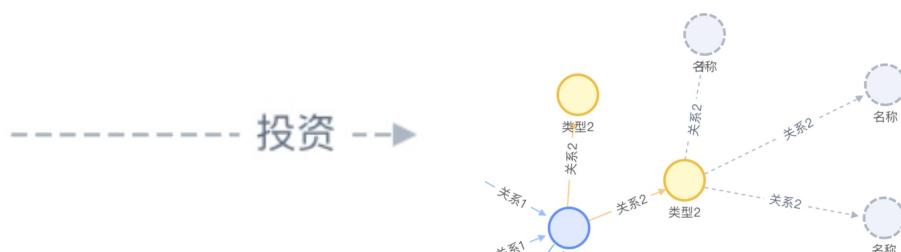
规则推理是指通过定义图结构、规则、action，输入一个实体id作为起点，运行输出子图。

4.3.5.1 DSL可视化编辑

目前DSL编辑大多是以代码框编写DSL代码的形式，这样就要求必须要理解DSL的语法，且图结构的定义和规则的定义也不够直观。以图可视化的形式进行DSL编辑，可以降低DSL学习成本，且在图中操作，用用户直接可以预览推理效果，更易理解。

4.3.5.2 推理结果展示

规则推理相对于普通的图分析，区别在于其中的边并不一定是我存在的，而是通过定义的规则，推导出来的联系。推导边的区分：推导边标识，定制虚拟边，区别为普通的边展示。



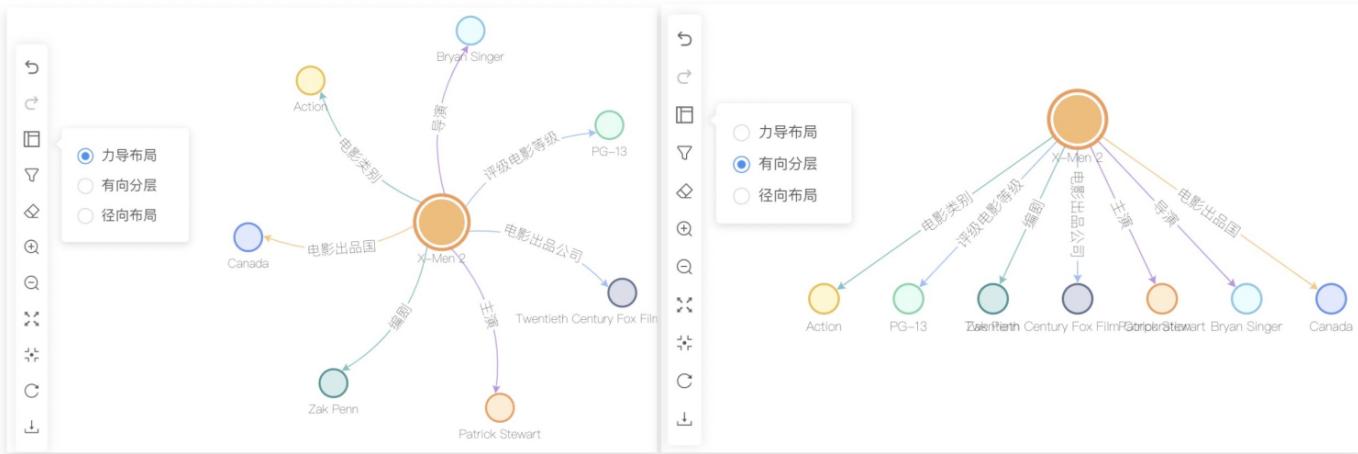
4.4 自动布局方案

内置丰富的布局，支持布局切换，满足不同场景下的布局需求。知识图谱中提供布局切换与智能布局两种方案。

4.4.1 产品方案

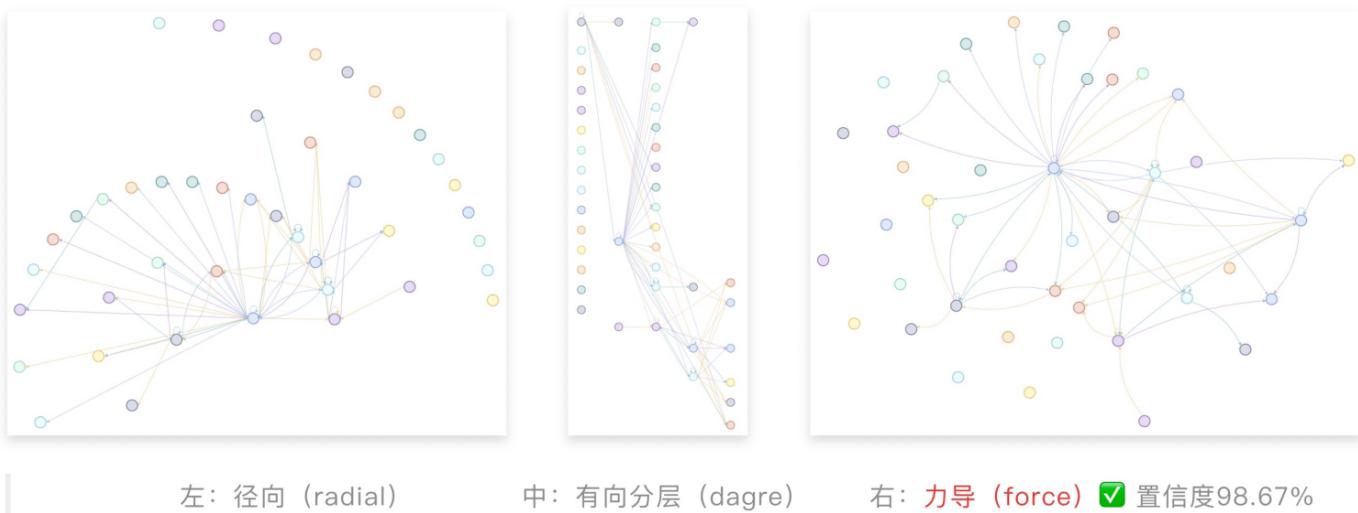
4.4.1.1 布局切换

用户能够根据不同的场景，自主切换布局方式。



4.4.1.2 智能布局

知识图谱能够根据数据特征，场景需求，智能化给用户推荐布局：



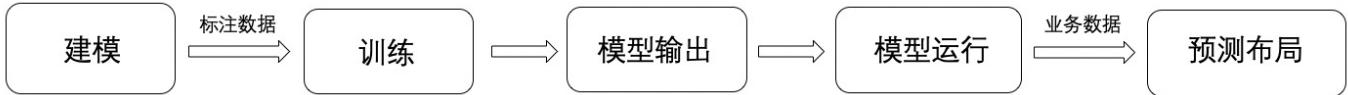
4.4.2 技术方案

4.4.2.1 布局切换

- 根据业务应用场景，人工预选一些相对适合的布局，形成布局配置，落到工具条；
- 工具条布局切换与传入图组件的布局参数进行联动，实现图布局实时变化。

4.4.2.1 智能布局

智能布局指在前端采用tensorflow.js、结合神经网络进行建模，通过大量的标注数据（标记布局分类）进行训练并输出模型，在业务场景中通过模型对真实的图数据进行预测，从而推荐出该数据最适合的布局分类的方法。



我们将图布局预测的能力封装到了 NPM 包 `@antv/vis-predict-engine` 中，通过 `predict` 方法来预测数据适合的布局。后续AntV/G6会内置这个可视化预测引擎，提供api给组件使用者来调用布局预测功能。基本用法如下：

```

1 import { GraphLayoutPredict } from '@antv/vis-predict-engine'
2 import G6 from '@antv/g6'
3 const data = { nodes: [], edges: [] }
4 const { predictLayout, confidence } = await GraphLayoutPredict.predict(data);
5 const graph = new G6.Graph({
6   layout: {
7     type: predictLayout
8   }
9 })

```

05 参考文献

- 【1】《2018知识图谱发展报告》
- 【2】《知识图谱标准化白皮书2019版》
- 【3】Schema：知识结构的一种定义，即知识的元数据，包含了本体类型和本体关系。
- 【4】本体类型：学术界称为本体（Ontology），类似与java代码中定义Class，用来定一个实体的元数据信息，比如企业是一个本体类型，其包含企业名称、法人代表、注册地等属性。
- 【5】本体关系：指本体类型之间的关系，如本体类型-企业和本体类型-自然人之间有一个实控人的关系。
- 【6】AntV树图：<https://g6.antv.vision/zh/examples/tree/compactBox>
- 【7】AntV关系图：<https://g6.antv.vision/zh/examples/net/forceDirected>