SEU 知识融合a

2019年11月27日 13:25

- **一、问题分析**
- 知识图谱的异构
  - 从知识图谱构建的角度看
    - 早期知识工程的理想是构建一个统一的知识库
    - 人类知识体系复杂
    - 不同人对某些知识有主观看法
    - 知识会随时间自然演化
    - 同一领域有不同组织构建自己的知识库
    - 交叉领域中的交叉知识往往是独立构建
    - 知识图谱构建优先考虑重用现有的知识
  - **从知识图谱的应用看**
    - 不同领域需要进行交互
    - 系统需要处理来自不同领域的知识
  - **结论**

    知识具有共享性的同时，还需要兼顾自治性和动态性，知识的构建过程和应用场景决定了知识异构是一种自然现象，不能完全消除。

  - **知识异构的两个层次**
    - **语言层次**：语法、逻辑、表达能力不匹配
      - 语法异构

        采用不同的描述语言

        RDF OWL JSON XML
      - 逻辑异构

        逻辑表述不匹配

        disjointWith、 A NOT B AND B NOT A
      - 元语异构

        元语的语义有差异

        Class 在OWL DL和OWL FULL
      - 表达能力异构

        不同语言表达能力的差异

        owl：Thing， owl： Nothing
    - **模型层异构：**概念化、解释不匹配
      - 概念化异构

        “动物”划分为“哺乳动物”和“鸟“

        “动物”划分为“食肉动物”和“食草动物”
      - 解释不匹配

        同义术语：Car、Auto

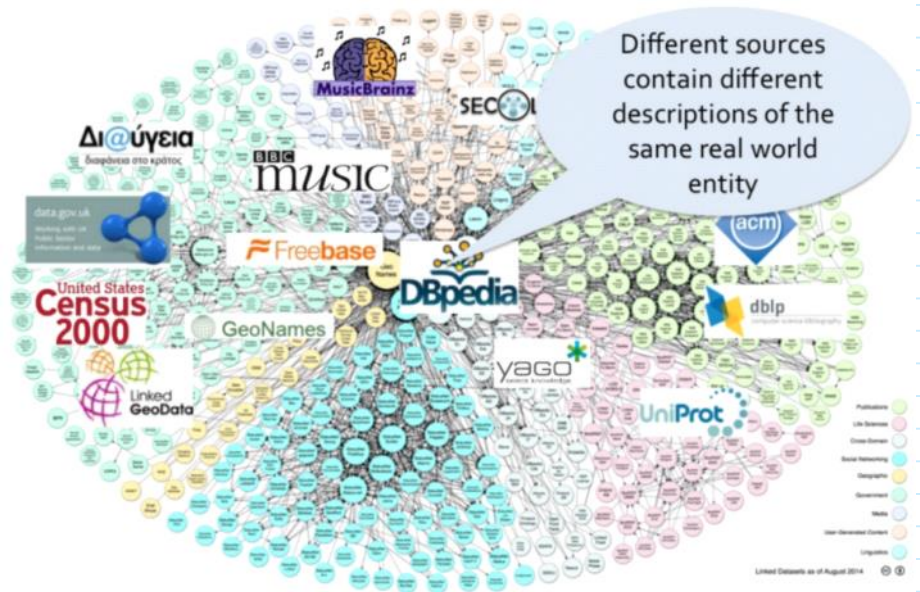
多义术语：Conductor（指挥家、半导体）

编码格式：FullName， FirstName+LastName

- **知识图谱中的数据特点**
  - 知识图谱VS关系数据库
    - 优势
      - 包含语义信息，可以进行 一定的推理
      - 形式更灵活，可扩展性好
    - 不足
      - 天生缺乏直接的有效处理工具，大规模图谱数据处理常常借助于数据库技术
      - 知识图谱不能代替数据库，二者各有所长

- DBpedia Ontology： 685个概念，2795个属性， 4233000个实例
- YAGO： 10000000个实例（人物，组织，城市等）
- GeoNames： 10000000地理实例
- 结论
  - 知识图谱包含一定的抽象层次知识和大量的实例层事实
  - 知识图谱中的知识融合要分别考虑两个层次的融合问题

- **为什么需要知识融合**
  - **数据清洗**
    - 构建的知识图谱可能存在异构
    - 知识融合是知识图谱应用的重要预处理步骤
  - **数据集成**
    - 需要同时利用或融合多个不同来源的知识图谱
    - 不同源的知识图谱可能存在重叠的知识

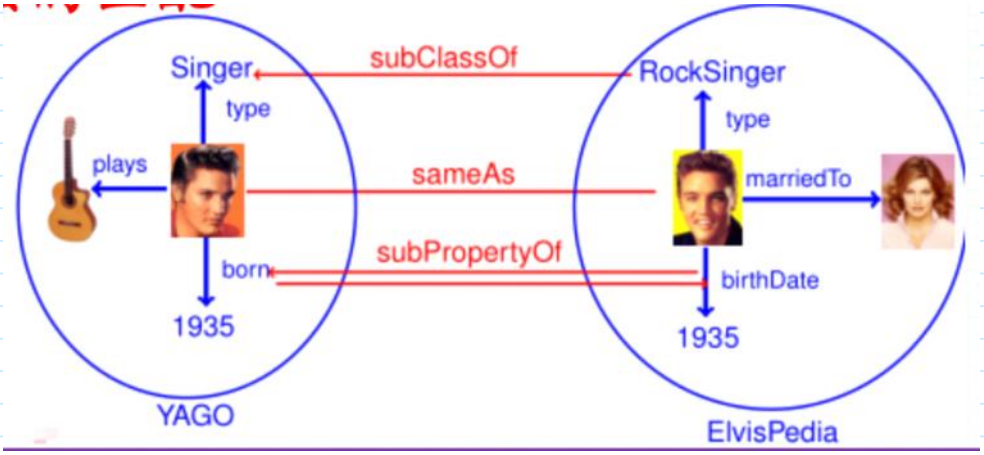


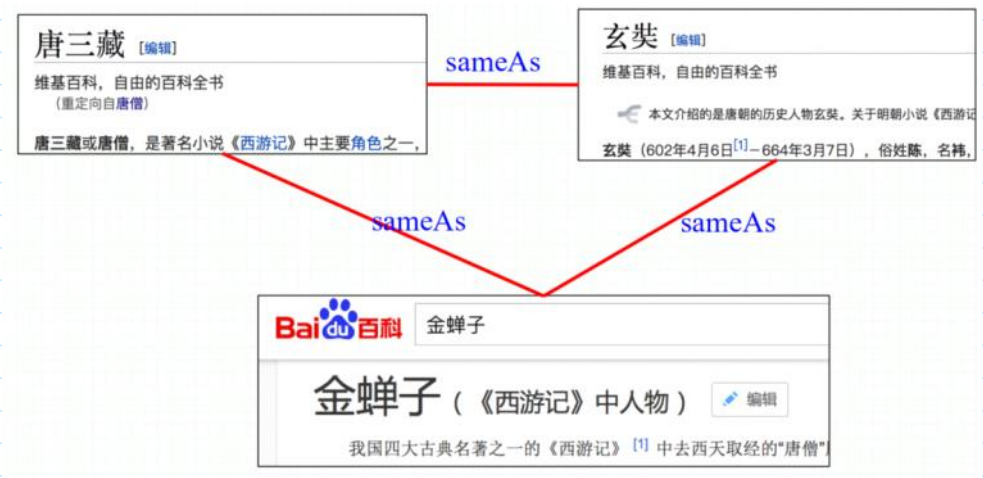
需要大量重用现有的知识

- **知识融合的目标**

  合并多个知识图谱

  - 解决本体层的匹配
    - 等价类/子类
    - 等价属性/子属性
  - 解决实例层的匹配
    - 等价实例





实例匹配的例子

- 复杂映射
  - 语义映射
    - 复杂概念的上下文
    - 属性的上下文
  - 函数的映射
    - 计算函数
    - 合并函数
    - 复杂函数

映射的势：1:1 N:M

- **二、解决思路**
- **抽象层次的融合解决方法-本体匹配（Ontology Match）**

- 本体匹配框架

预处理	解析、数据清洗、构造基础数据
匹配计算	构造匹配线索、相似度计算
后处理	匹配结构抽取、匹配调谐

- **本体匹配器**
  - 基础匹配器
    - **字符串匹配器**
      - **编辑距离**

定义 5.4 给定一个字符串操作集合 Op 和一个代价函数 w，对于任意一对字符串 s 和 t，存在将 s 转换为 t 的操作序列集合，两字符串间的编辑距离 $\delta(s,t)$ 是将 s 转换为 t 的最小操作序列的代价和：

$$\delta(s,t)=min\sum_{i=1}^{n}w_{op_i},\quad 且op_n(...op_1(s))=t$$

☐ Levenshtein distance（最小编辑距离），目的是用最少的编辑操作将一个字符串转成另一个，如下：

$$'Lvensshtain' \xrightarrow{插入\ 'e'} 'Levensshtain'$$

$$'Levensshtain' \xrightarrow{删除\ 's'} 'Levenshtain'$$

$$'Levenshtain' \xrightarrow{替换\ 'a'\ \to 'e'} 'Levenshtein'$$

上述将‘Lvensshtain’转换成‘Levenshtein’，总共的操作3次，编辑距离也就是3。

☐ Wagner and Fisher distance是Levenshtein的一个扩展，将这个模型中编辑操作的代价赋予了不同的权重，如下：

$$\begin{cases} D(0,0) = 0 \\ D(i,0) = D(i-1,0) + del[x(i)] & 1<i\leqslant N \\ D(0,j) = D(0,j-1) + del[y(j)] & 1<j\leqslant M \end{cases}$$

$$D(i,j)=\min\begin{cases} D(i-1,j) + del[x(i)] \\ D(i,j-1) + ins[y[j]] \\ D(i-1,j-1) + sub[x(i),y(j)] \end{cases}$$

其中，del 和 ins 以及 sub 分别是删除和插入以及替换的代价

      - **汉明距离**

一种常用来比较两个字符串的直接方法是汉明距离，它计算两个字符中字符出现位置的不同。

定义 5.1 对于给定的任意两个字符串 s 和 t，它们的汉明距离相似度定义为：

$$\delta(s,t)=1-\frac{(\sum_{i=1}^{min(|s|,|t|)}s[i]\neq t[i])+||s|-|t||}{max(|s|,|t|)}$$

- **子串相似度**

  还可进一步精确度量两字符串包含共同部分的比例，即子串相似度：

  定义 5.3 子串相似度度量任意两个字符串 s 和 t 间的相似度 δ，令 x 为 s 和 t 的最大共同子串，则它们的子串相似度为：$\delta(s,t)=\frac{2|x|}{|s|+|t|}$。

- **Dice系数**

  用于度量两个集合的相似性，因为可以把字符串理解为一种集合，因此Dice距离也会用于度量字符串的相似性，Dice系数定义如下：

  $$sim_{Dice}(s,t)=\frac{2|S\cap T|}{|S|+|T|}$$

  以Lvensshtain和Levenshtein为例，两者相似度为 2*9/(11+11)= 0.82

- **Jaccard系数**

  适合处理短文本的相似度，定义如下

  $$sim_{Jaccard}(s,t)=\frac{|S\cap T|}{|S\cup T|}$$

  可以看出与Dice系数的定义比较相似

- **N-Gram**

  将文本转换为集合，除了可以用符号分割单词以外，还可以考虑用N-Gram分割单词，用N-Gram分割句子等来构成集合，计算相似度

- **匹配计算基础-文本匹配器**
  - **TF/IDF**

    用来评估某个字或者某个词对一个文档的重要程度

    $$tf_{i,j}=\frac{n_{i,j}}{\sum_k n_{k,j}} \qquad idf_i=\log\frac{|D|}{1+|\{j:t_i\in d_j\}|}$$

    $$sim_{TF-IDF}=tf_{i,j}\times idf_i$$

    比如某个语料库中有五万篇文章，含有健康的有两万篇，现在有一篇文章，拥有1000词，“健康”出现30次，则

    sim_{TF-IDF} = 30/1000 * log(50000/(20000+1)) = 0.012

  - **文本匹配器的原理**
    - 思想：将文档变为向量形式，通过向量相似度实现文本匹配
    - 本体中的概念和属性往往有大量相关的文本信息标签，注释，描述等等

- 将待匹配的对象的相关文本组织为文档形式，在转化为文档向量
- 实际应用中非常有效，能解决很多实际应用场景

□ 虚拟文档

- 思想
- 三种虚拟文档
- 虚拟文档例子

向量 D_i 中的权重 d_{ij} 采用 *TF-IDF* 方法来进行计算。令 n_i 为含有词条 t_i 的文档数目，$tf(t_i)$为 t_i 在 D_i 中出现的频率，这里的频率计算考虑的是语义描述文档中词项所带的权重，即 $tf(t_i)=p_i$。计算公式为：

$$d_{ij} = TF * IDF,\ TF = \frac{p_i}{\sum p_x},\ IDF = \log(\frac{N}{n_i}) \tag{3-18}$$

$$Sim(D_i, D_j) = \cos\theta = \frac{\sum_{k=1}^{n} d_{ik} \times d_{jk}}{\sqrt{\sum_{k=1}^{n} d_{ik}^2 \times \sum_{k=1}^{n} d_{jk}^2}}$$

- **匹配计算基础-结构匹配器**

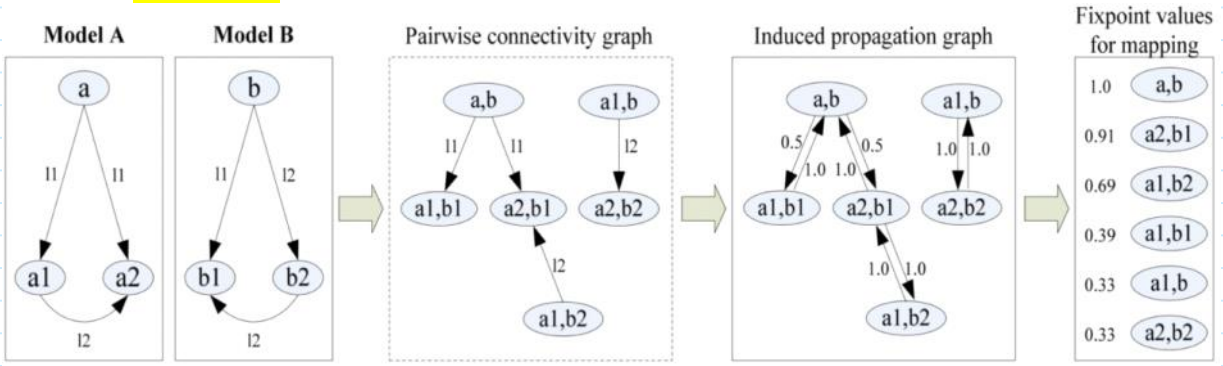
□ 结构匹配器的原理

- 思想：利用本体结构信息来弥补文本信息不足的情况
- 本体中的概念和属性往往有大量相关的其他概念和属性，组成一种图的结构
- 结构匹配器不采用图匹配技术，后者代价太高且效果不好
- 采用相似度传播思想的方法更好

□ 结构匹配器

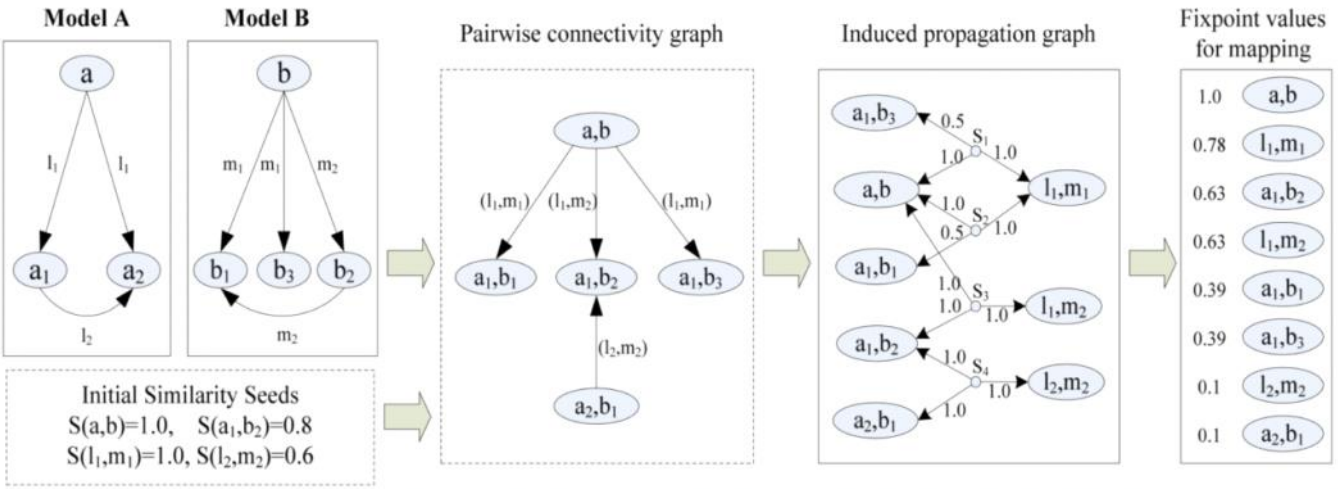
- 间接的结构匹配器
  在文档匹配器中考虑结构信息，如邻居、上下义、属性等
- 直接的结构匹配器
  图匹配复杂度高，无法直接使用
  相似度传播模型的各种辩题很有效

- **相似度传播**



相似度传播模型缺点：不能处理边

改进的适用于本体的相似度传播模型



改进的适用于本体相似度传播的模型

定义 3-2. (三元组相似度传播强条件). 给定本体中的任意两个三元组 t_i=<s_i,p_i,o_i>和 t_j=<s_j,p_j,o_j>，S_s、S_p 和 S_o 分别表示三个对应位置上的元素相似度。当且仅当 t_i 和 t_j 均满足下列三个条件时，它们之间才存在相似度传播：

（1）S_s、S_p 和 S_o 中至少有 2 个的相似度值大于阀值 θ；

（2）如果 t_i 中包含本体元语，则 t_j 的对应位置也必须是相同的本体元语；

（3）t_i 或 t_j 中包含的元语不能超过 2 个。

$$s^{i+1}(x,y) = s^{i}(x,y) + \sum_{\substack{<a_u,p_u,x>\in A \\ <b_u,q_u,y>\in B}} s^{i}(a_u,b_u)\cdot s^{i}(p_u,q_u)\cdot w_{sp} + \sum_{\substack{<x,p_v,a_v>\in A \\ <y,q_v,b_v>\in B}} s^{i}(a_v,b_v)\cdot s^{i}(p_v,q_v)\cdot w_{po} + \sum_{\substack{<a_t,x,c_t>\in A \\ <b_t,y,d_t>\in B}} s^{i}(a_t,b_t)\cdot s^{i}(c_t,d_t)\cdot w_{so}$$

- 结构匹配器的效果
  - Lily， Falcon， RiMoM都不约而同的利用了相似度传播的结构匹配技术，且效果表现良好
  - 结构匹配对解决文本信息量少，文本信息不规范，文本编码不一致，跨语言的匹配中有效果
  - 本体匹配需要采用结构匹配器，归结原因还是本体自身的质量不够好
  - 结构匹配器是解决若信息场景的有效办法

- **匹配计算基础-知识表示学习**
  - 原理

    利用机器学习中的表示学习技术，将知识图谱中的实体和关

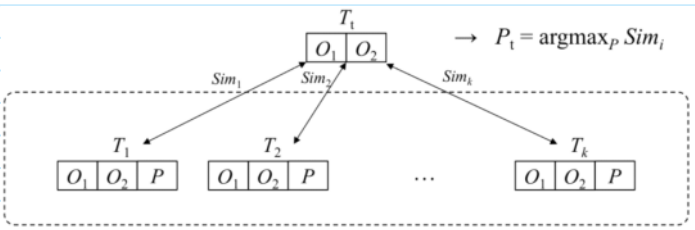
系都映射低维空间向量，直接用数学表达式来计算各个实体之间的相似度。

这类方法不依赖任何的文本信息，获取到的都是数据的深度特征。

- 匹配结果抽取目标
  - 不同匹配的结果通常表示为一个相似矩阵，如何从中得到最终的匹配结果是一个重要的后处理步骤。
- 匹配结果抽取方法
  - 稳定混婚姻算法
  - 其他启发式算法

- **匹配协调**
  - 匹配协调问题
    - 大量的系统参数
    - 调参过于依赖人工经验
    - 复杂的可选模块及参数：策略调谐
  - 匹配调谐解决思路
    - 基于本体相似度
    - 基于历史匹配参数
  - 解决技术
    - 相似度计算
    - 蚁群算法
    - 机器学习
    - 深度学习、强化学习

  - 基于本体相似度的本体匹配自动调谐方法



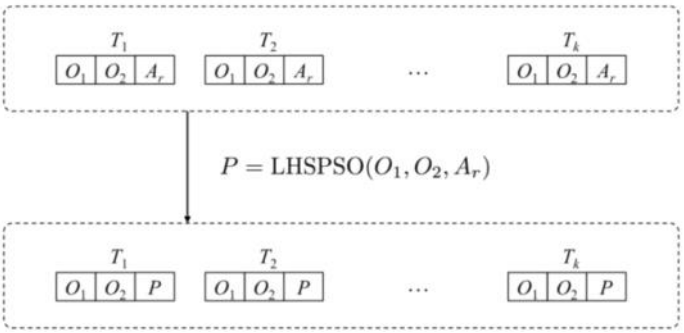
❖ 相似度的度量（本体 / 匹配任务）

❖ 历史记录的构建 / 查阅

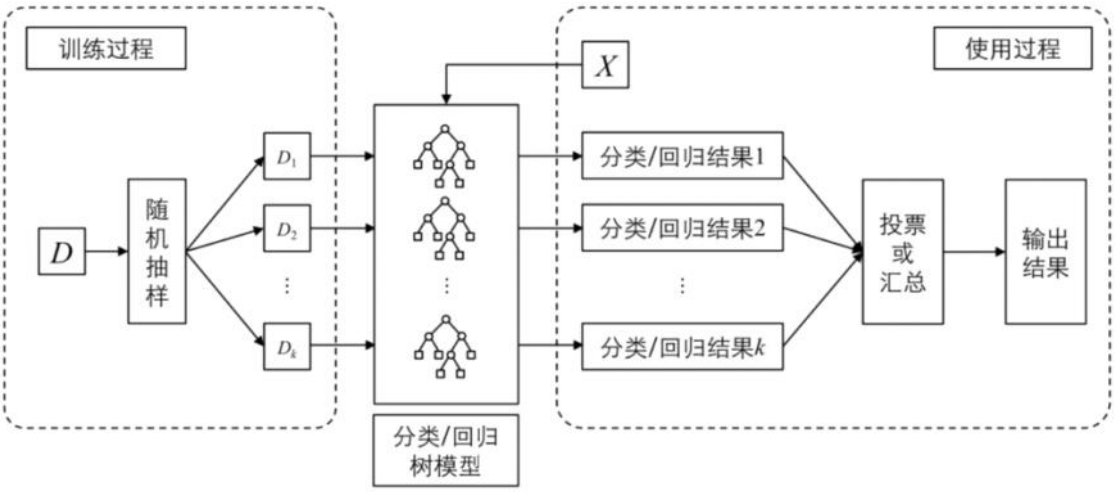
匹配任务相似度

$$\overline{S}(T_a, T_b) = \max\left(\frac{S(O_{1a}, O_{1b}) + S(O_{2a}, O_{2b})}{2}, \frac{S(O_{1a}, O_{2b}) + S(O_{2a}, O_{1b})}{2}\right)$$

构建历史记录库



- 基于ERT机器学习模型的本体匹配自动调谐方法
- ERT机器学习模型：随机森林模型



通用本体匹配系统Lily

- ✓ 基于语义子图的通用本体匹配框架（2006-2010）
  特色：精确描述语义、高精确度、高召回率
- ✓ 大规模本体匹配（2008-2013）
  特色：巧妙利用层次结构和匹配区域性，无需划分本体
- ✓ 弱信息本体匹配（2008-2013）
  特色：利用语义子图和相似度传播
- ✓ 本体匹配调试 (2008-2009)
  特色：首次提出并给出启发式解决之一
- ✓ 本体匹配调谐 (2012-至今)
  特色：基于机器学习实现匹配系统自动调谐，提升可用性
- ✓ 实例匹配 (2013-至今)
  特色：高效处理大规模实例匹配，并保证匹配精度
- ✓ 本体匹配系统Lily (2006-至今)

特色：高效处理大规模实例匹配，并保证匹配精度

✓ 本体匹配系统Lily (2006-至今)

模块：实验数据、预处理、匹配计算、后处理、用户平台、基础工具库

其他优秀的本体匹配库

-Falcon-AO 南京大学

-RIMOM 清华大学