法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院和讲者共同拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

第八讲：语义搜索

大纲

- 语义搜索简介
- 语义数据搜索
- 混合搜索
- 语义搜索的交互范式
- 实践展示：使用Elasticsearch实现简单语义数据检索

大纲

- 语义搜索简介
- 语义数据搜索
- 混合搜索
- 语义搜索的交互范式
- 实践展示：使用Elasticsearch实现简单语义数据检索

文档检索 vs. 数据检索

- 不同搜索模式之间的技术差异可以分为
  - 对用户需求的表示 (query model)
  - 对底层数据的表示 (data model)
  - 匹配方法 (matching technique)
- 信息检索(IR)支持对文档的检索**(document retrieval)**
  - 通过轻量级的语法模型(lightweight syntax-centric model)表示用户的检索需求和资源的内容，即目前占主导地位的关键词模式：词袋模型(bag-of-words)
  - 对主题搜索 (topic search)效果很好，即给定一个主题检索相关的文档
  - 但不能应对更加复杂的信息检索需求

文档检索 vs. 数据检索

- 数据库 (DB)和知识库专家系统 (Knowledge-based Expert System)可以提供更加精确的答案 **(data retrieval)**
  - 使用表达能力更强的模型来表示用户的需求
  - 利用数据之间内在的结构和语义关联
  - 允许复杂的查询
  - 返回精确匹配查询的具体答案

语义模型

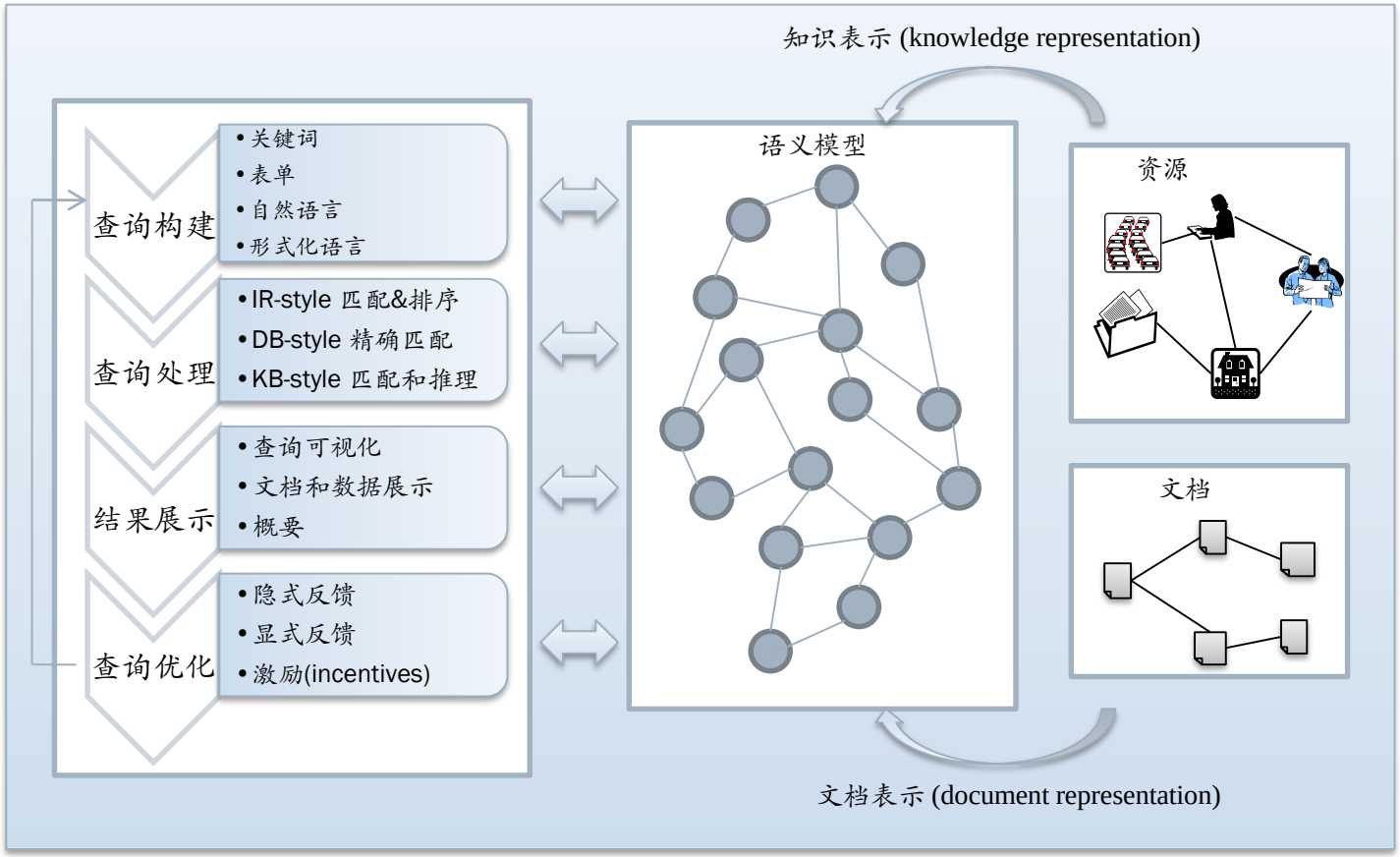
- 语义关注的是能用于搜索的资源的**含义 (meaning)**。
- 这些含义是通过语义模型构建的
- **语言学模型 (Linguistic model)**
  - 对词语级别的关系建模
  - 分类系统 (taxonomies), 同义词库 (thesauri)
- **概念模型 (Conceptual model)**
  - 对论域 (universe of discourse)中的语法元素(syntactic element)的关系建模
  - 解析(Interpretation)：从语法元素到论域的映射
- 表达能力 (expressivity)
  - 语言和建模结构的数量
- 形式化 (formality)
  - 解析过程是可计算的(computable)

语义搜索分类

- DB和KB系统属于重量级语义搜索系统
  - 对语义**显式的和形式化**的建模，例如
  - ER图
  - RDF(S)和OWL中的知识模型 (knowledge model)
  - 主要为语义的**数据检索系统**

- 基于语义的IR系统属于轻量级的语义搜索系统
  - **轻量级的语义模型**，例如分类系统或者辞典
  - 语义数据 (RDF) 嵌入文档或者与文档关联
  - 是基于语义的**文档检索系统**

语义搜索 – 流程图


知识表示 (knowledge representation)
查询构建
• 关键词
• 表单
• 自然语言
• 形式化语言
查询处理
• IR-style 匹配&排序
• DB-style 精确匹配
• KB-style 匹配和推理
结果展示
• 查询可视化
• 文档和数据展示
• 概要
查询优化
• 隐式反馈
• 显式反馈
• 激励(incentives)
语义模型
资源
文档
文档表示 (document representation)

搜索模式趋向一致

- 趋势：结构化和语义数据的可用性越来越高
- 数据Web搜索
  - 在更大的Web场景中检索语义数据和推理
  - 利用Web上的RDF数据和OWL本体
  - 采用传统上应用于IR领域的，扩展性较好的方法，来处理Web数据的质量问题，和与长文本描述相关的数据元素。
- 文档Web搜索
  - 数据库和语义搜索技术被应用到IR系统中，以便在搜索过程中结合运用日益增加的，高度结构化和表达能力强的数据。
- 趋同：
  - 在搜索中用到的数据
  - 在搜索中用到的技术

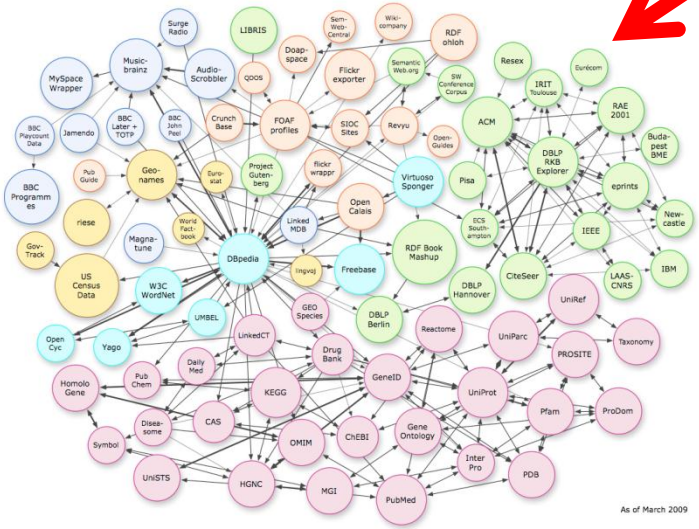
大纲

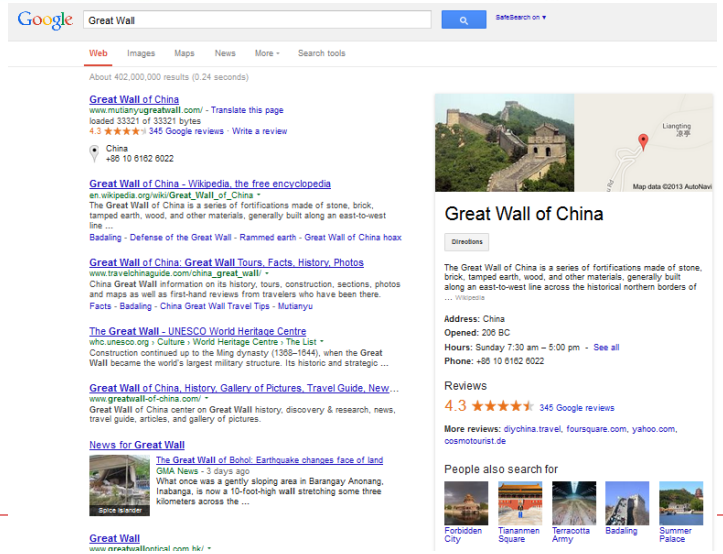
- 语义搜索简介
- 语义数据搜索
- 混合搜索
- 语义搜索的交互范式
- 实践展示：使用Elasticsearch实现简单语义数据检索

语义Web——数据Web

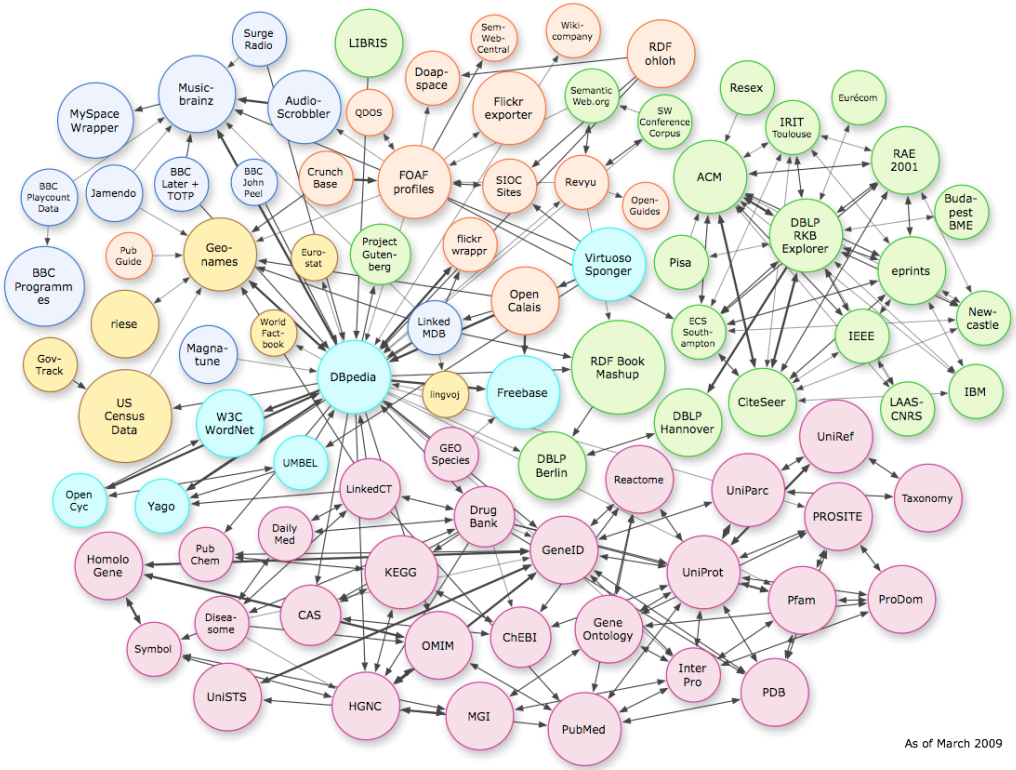
- 数据以结构化的形式发布和链接在一起
- 数据的含义和关系在形式化的模型中有详细说明：
  - 不同的场景(actors)有一个公共的术语词汇表
  - 精确的术语含义的说明
- 语义是基于标准化的逻辑语言，从而确保明确的形式化解析
- W3C联盟完成语言和协议的标准化

利用链接数据进行搜索

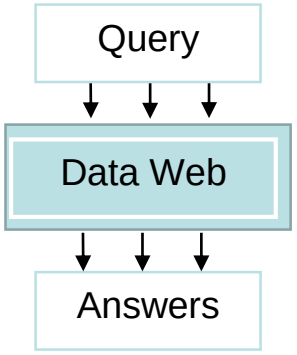




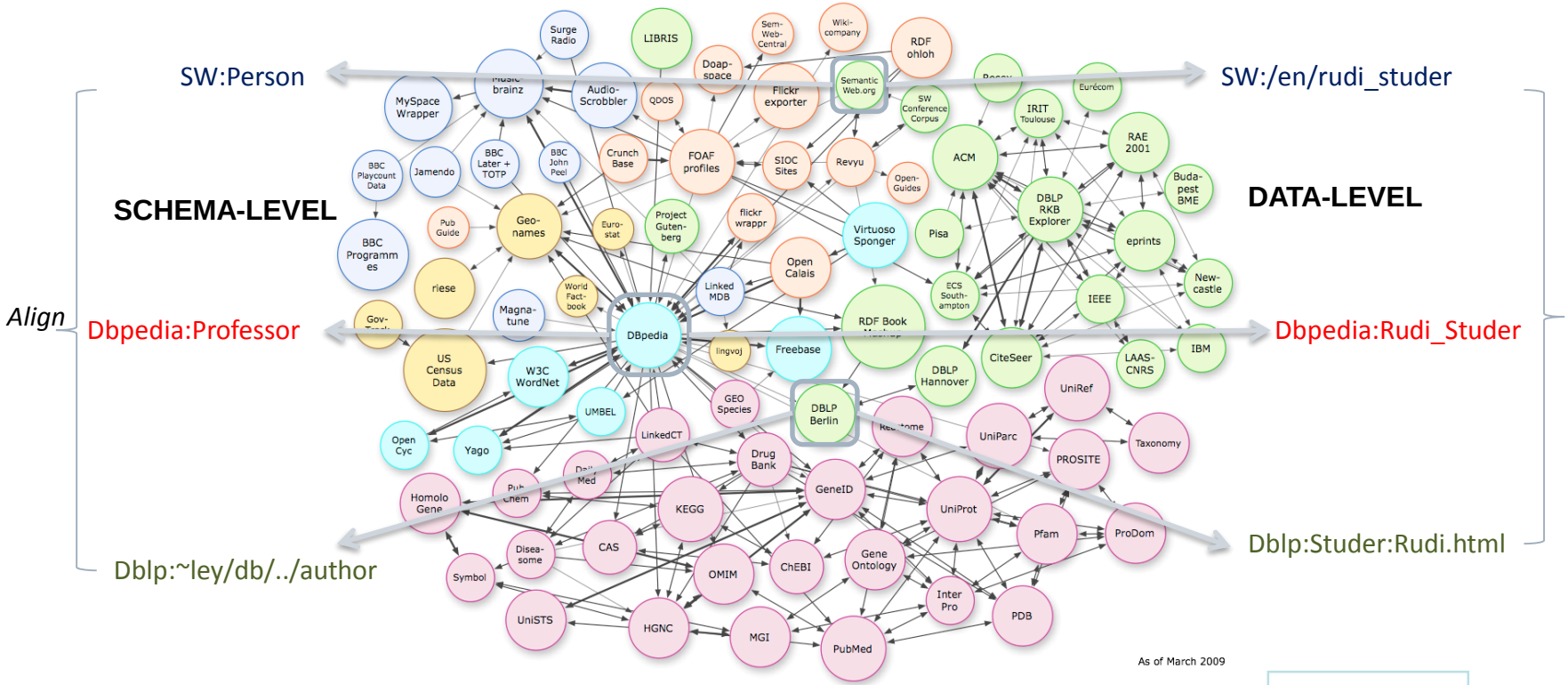
难点1—可扩展性



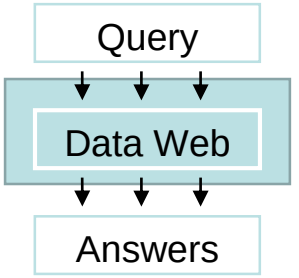
- 对链接数据的有效利用要求基础架构能扩展和应用在大规模和不断增长的内链数据上



难点2—异构性

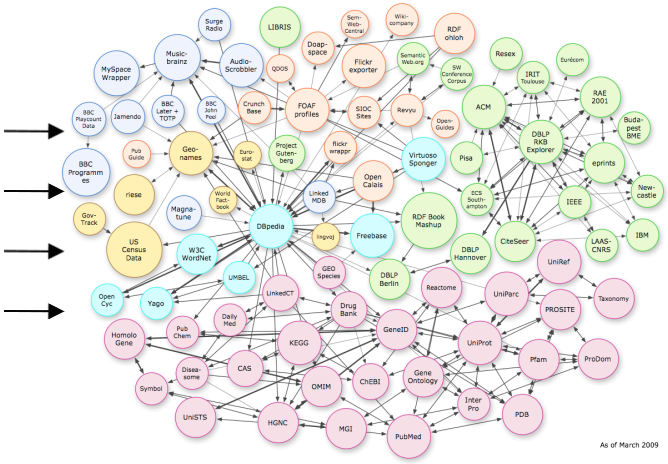


- 对数据Web的有效利用要求有效的机制：
  - 整合数据源（补充RDF链接）
  - 从不同数据源中找到与查询相关的数据
  - 合并来自不同数据源的查询结果



难点3—不确定性

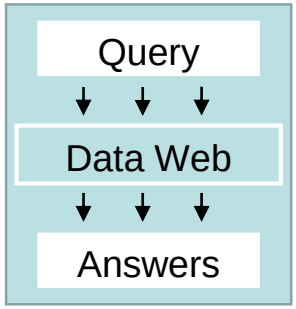
"*Find action films directed by some Hong Kong film director and starring Chinese martial actors*"



- 用户需求的表示不完整
  - 用户不能完全地描述需求
  - 用户事先不能准确地了解自己的需求

对链接数据的有效利用需要有效的机制：

- 以不精确的方式匹配需求和数据
- 并对结果进行排序
- 足够灵活以应对条件的变化！



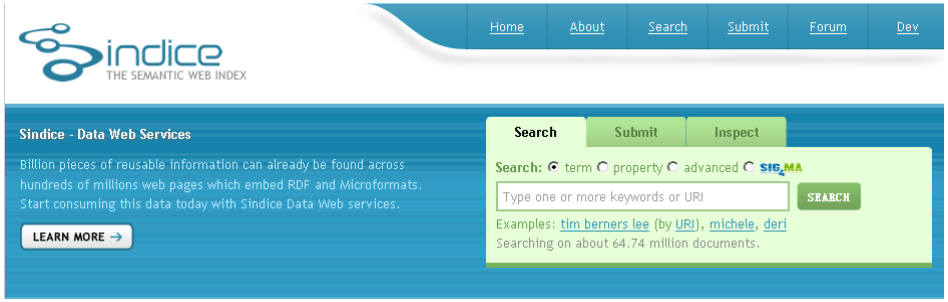
最佳实践

语义Web搜索引擎

- 本体搜索: Swoogle, Waston

最佳实践

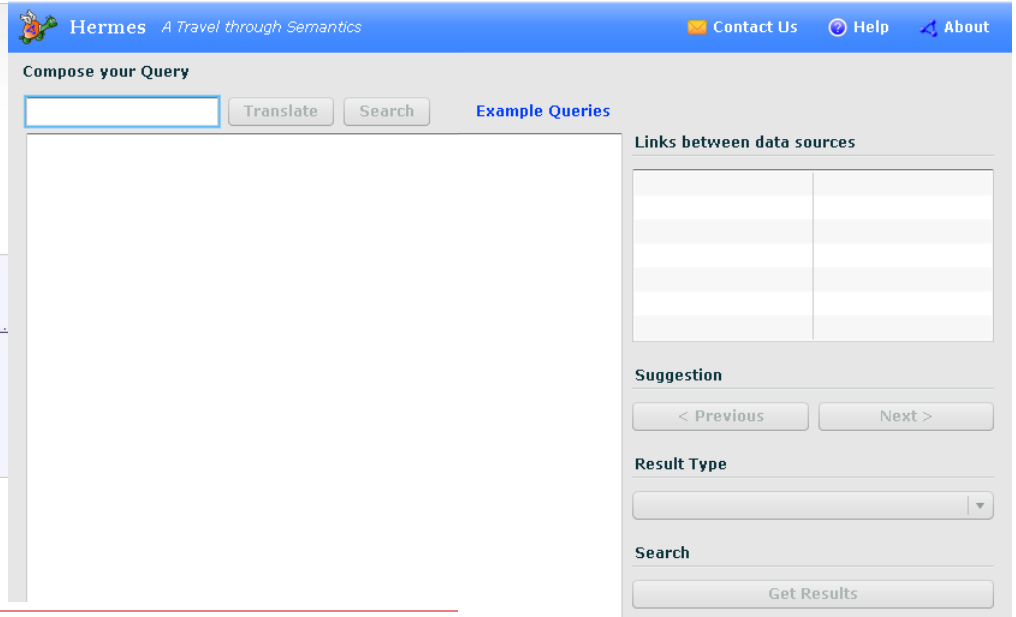
- 语义Web搜索引擎
  - Gateways: Swoogle, Waston
  - 实体搜索: Sigma on Sindice, FalconS



最佳实践

语义Web搜索

- Gateways: Swoogle, Waston
- 实体搜索： Sigma on Sindice, FalconS
- 数据Web搜索：SWSE， Hermes (SearchWebDB)



最佳实践

三元组存储

- 基于IR: Sindice, FalconS…
  单一数据结构和查询算法，*针对文本数据进行排序检索来优化*
- 高度可压缩，可访问
- 排序是组成部分
- 不能处理简单的select, joins等操作

最佳实践

- 三元组存储
  - 基于IR: Sindice, Falcons
  - 基于DB: Oracle的RDF扩展，DB2的SOR
    各种索引和查询算法，以适应各种*对结构化数据的复杂查询*
  - 空间开销和访问的局限性
  - 没有集成对检索结果的排序
  - 能够完成复杂的selects, joins,… (SQL, SPARQL)
  - 能应对高动态场景(许多插入/删除)

最佳实践

三元组存储

- 基于IR: Semplore…
- 基于DB: Oracle的RDF扩展, DB2 SOR…
- 原生存储 (Native stores): **Dataplore**, YARS, RDF-3x

- 高度可压缩，可访问
- 类似IR的检索排序
- 类似DB的selects和joins
- 可在亚秒级时间内在单台机器上完成对TB级数据的查询
- 高动态(许多插入/删除操作)

- 没有事务，恢复等

存储和索引 (Semplore)

- 重用IR索引来索引语义数据
- IR索引基于以下概念
  - 文档
  - 字段(field),例如，标题，摘要，正文…
  - 词语(terms)
  - Posting list和Position list

Table 1: Translating data into "Documents" in IR

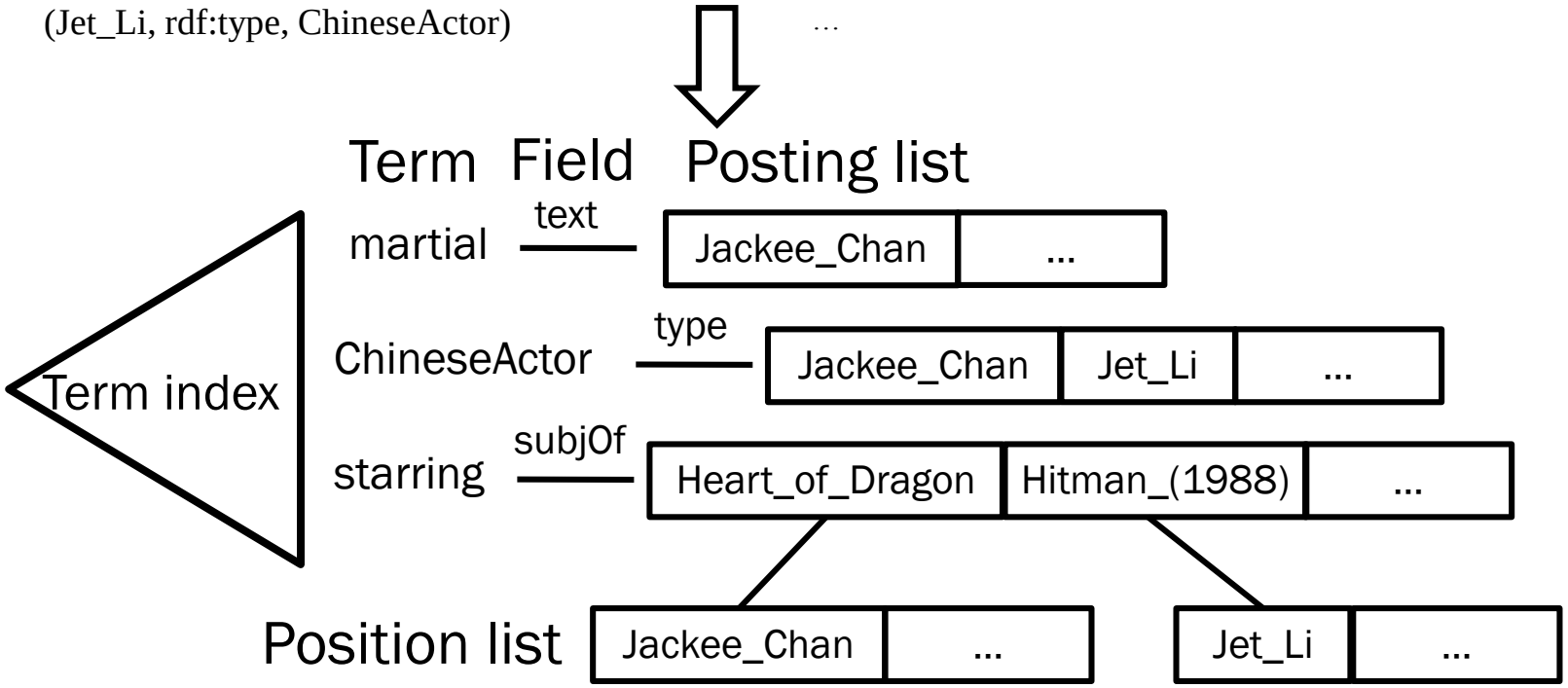
Document	Field	Term
concept C	text	tokens in textual properties
relation R		
individual i		$(i, R, ?)$ is a
	objOf	$(?, R, i)$ is a triple in data
	text	tokens in textual properties of i

核心想法：将RDF转换成具有fields和terms的虚拟文档

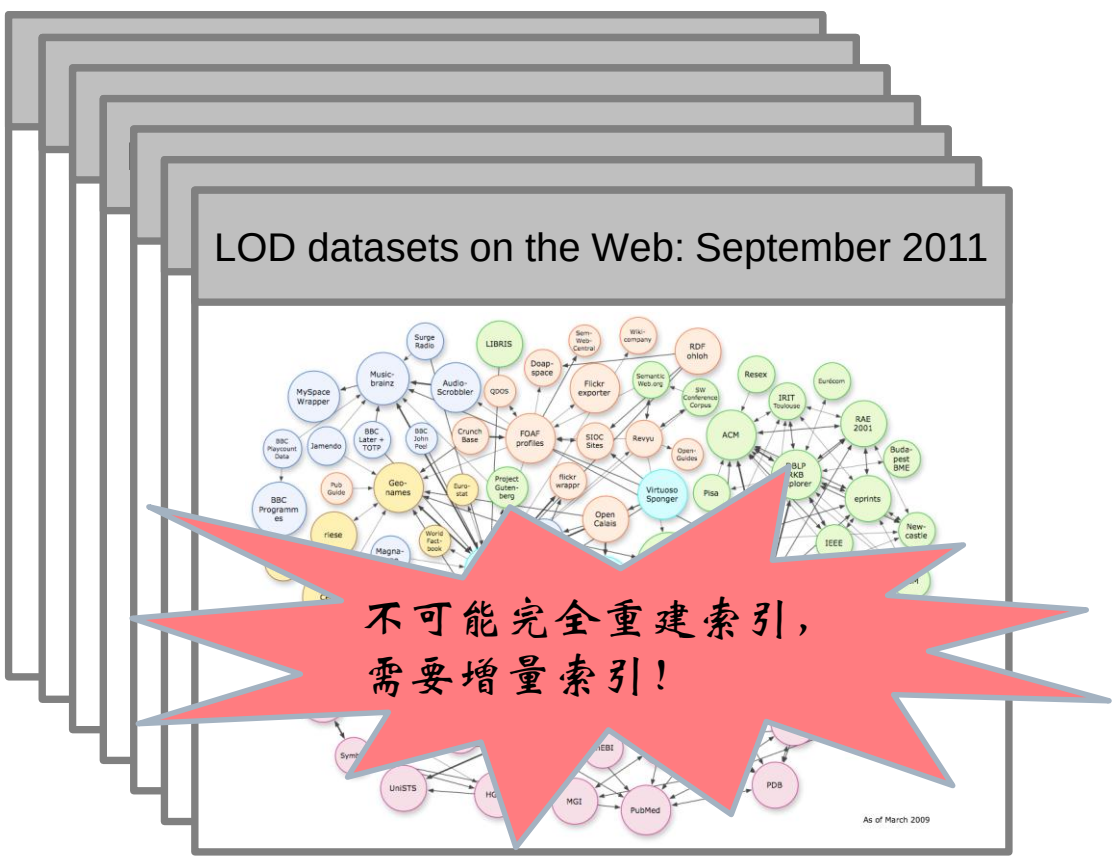
深入逻辑结构—索引虚拟文档

待索引的三元组

(Jackee_Chan, rdfs:comment, martial)
(Jackee_Chan, rdf:type, ChineseActor)
(Jet_Li, rdf:type, ChineseActor)
(Heart_of_Dragon, starring, Jackee_Chan)
(Hitman_(1988), starring, Jet_Li)
…

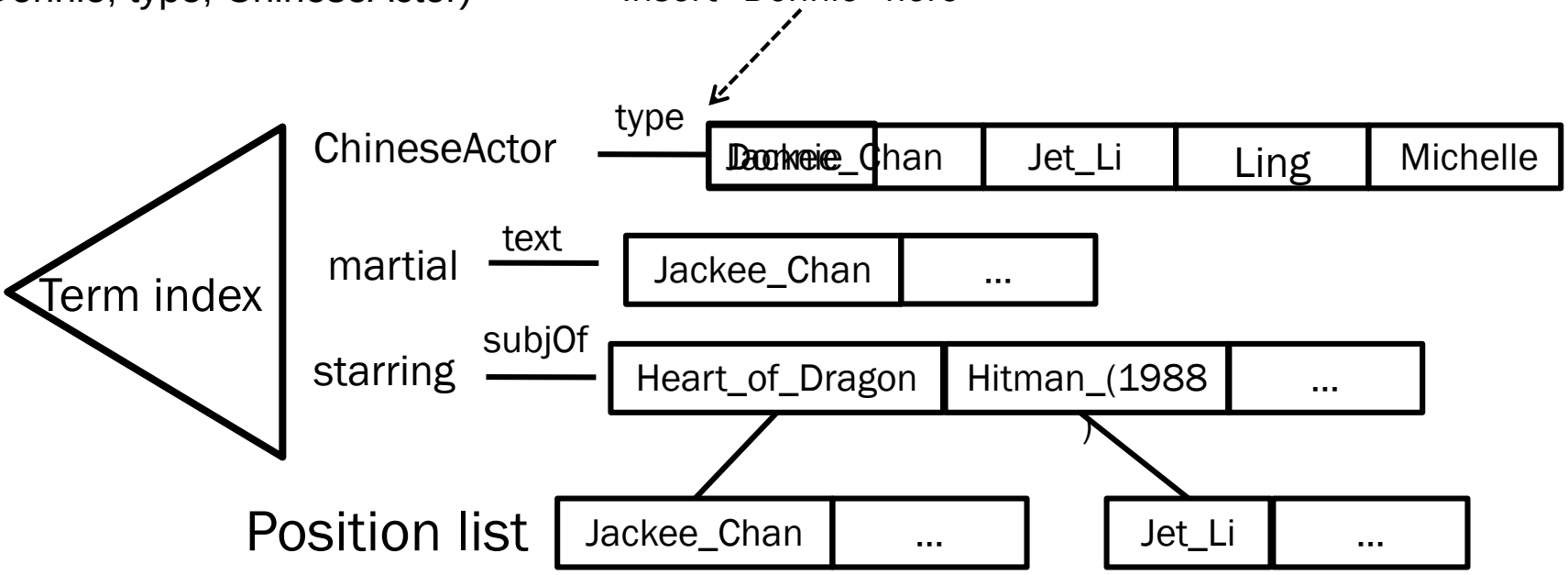


为什么增量索引?



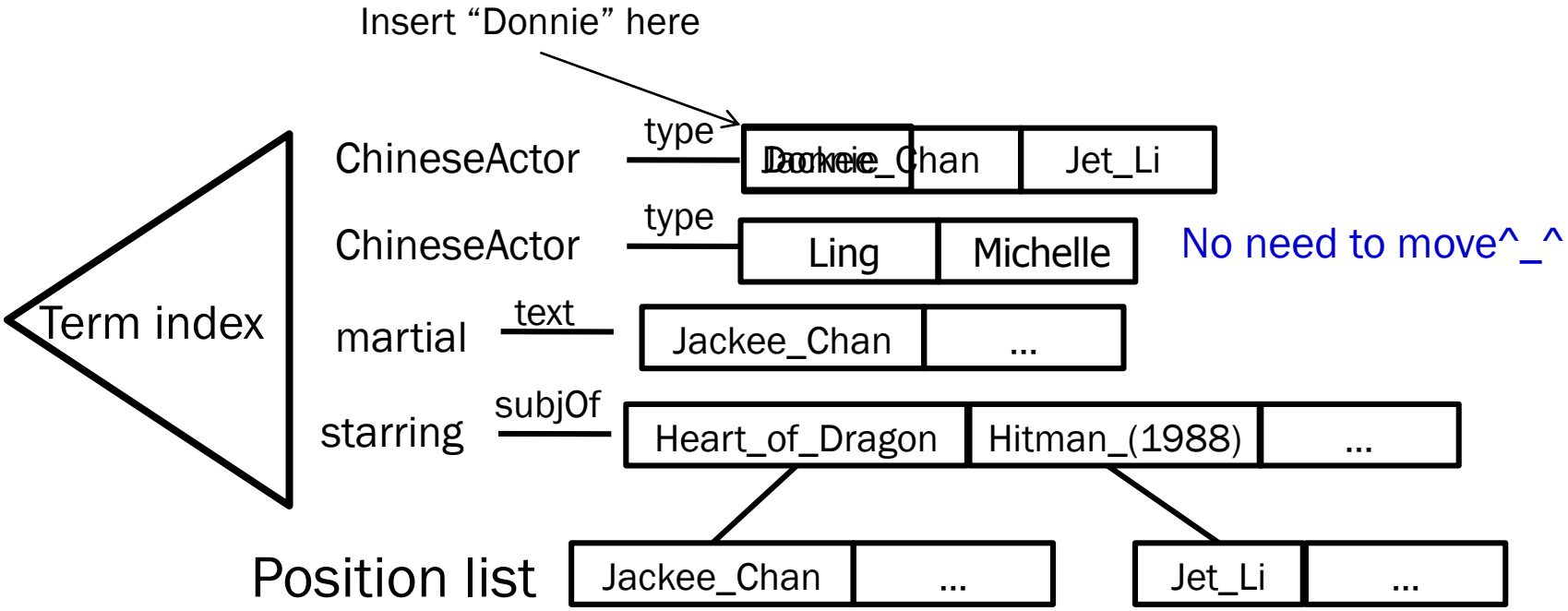
增量索引—处理当前索引

(Donnie, type, ChineseActor) ------> Insert “Donnie” here



移动大量元素非常耗时，所以我们该怎么做？

基于块的索引扩展



所以，块尺寸越小，所需的移动就越少

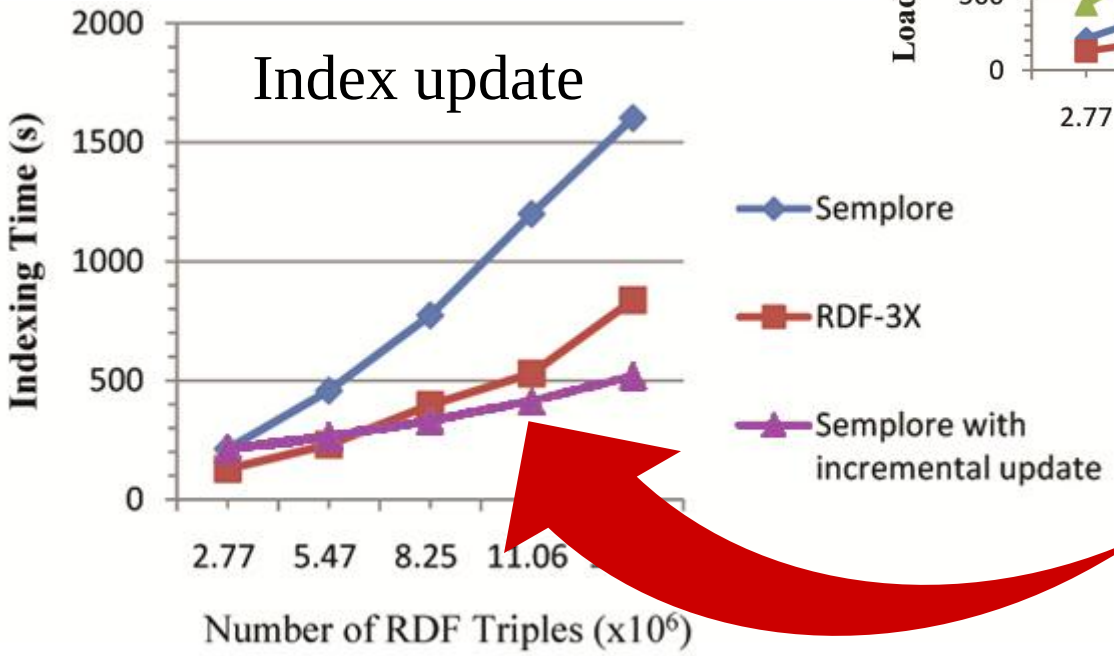
但是，块越小越好？

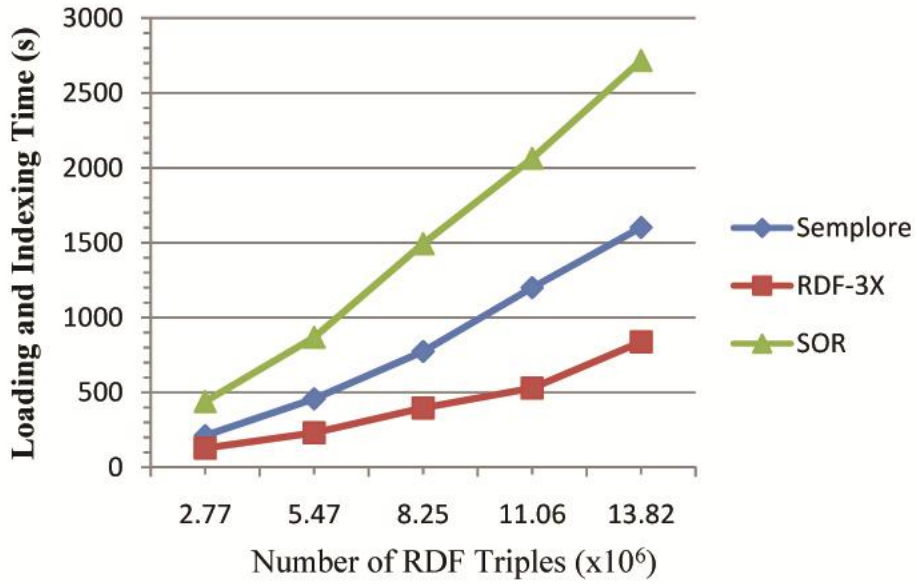
谨慎选择块的大小

- 搜索性能下降
  - 更多的块需要更多的随机访问，来定位这些块
- 索引大小的开销
  - 同时，更多的块需要更多的空间开销

索引构建 vs. 索引更新

- 因为增量索引，索引时间大幅减少





排序和索引

- 一个基本的数据结构: Ascending Integer Stream (AIS)
- 四种基本操作
  - 基础的检索: (f, t) ✔
  - 归并排序: m(S1, op, S2) ✔
  - 概念表达式计算(Concept Expression Evaluation): λ(C) ✔
    λ(AmericanFilm ∩ “war”)=
    m((type, AmericanFilm), ∩, (text, “war”))
  - 关系扩展(Relation Expansion): ⋈(S1, R, S2)

$$\{y \mid \exists x : x \in S_1 \wedge (x, R, y) \wedge y \in S_2\}$$ Needs Implementation

Dataplorer

A Scalable Search Engine for the Data Web

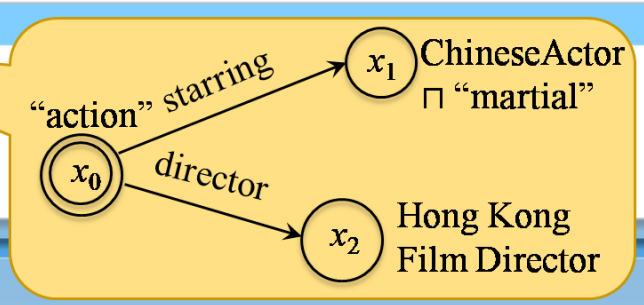
Query Tree

action | Search | New Query | Sample Queries

"action"

director :: Hong Kong film directors

starring :: Chinese actors "martial"



Results 1 - 10 of 50 (0.266 seconds)

Heart of Dragon

Select this item

"heart of dragon (long de xin) is a 1985 hong kong **action** film directed by sammo hung, assisted

http://dataplorer.apexlab.org/Heart_of_Dragon

Dragons Forever

Select this item

"dragons forever (\u98db\u9f8d\u731b\u5c06; fei lung maang jeung) is a 1988 **action** movie made

http://dataplorer.apexlab.org/Dragons_Forever

Wheels on Meals

Select this item

"wheels on meals (ku\u00e0ic\u0101n ch\u0113) is a 1984 hong kong **action** film directed by sammo

http://dataplorer.apexlab.org/Wheels_on_Meals

My Lucky Stars

Select this item

kong **action** comedy, directed by sammo hung. it is the second film in the lucky stars series.\n\nthe

http://dataplorer.apexlab.org/My_Lucky_Stars

Eastern Condors

Select this item

"eastern condors is a 1987 hong kong **action** film set in the aftermath of the vietnam war

http://dataplorer.apexlab.org/Eastern_Condors

Winners and Sinners

Select this item

\") is a 1983 hong kong **action** comedy film directed and co-written by sammo hung. it was the first

Navigator

Categories

- TOP Category(50)
- Cantonese language films(48)
- Hong Kong films(48)
- Martial arts films(35)
- Kung fu films(13)
- Action films(9)
- Action thriller films(7)
- films(4)
- films(4)
- films(4)

More

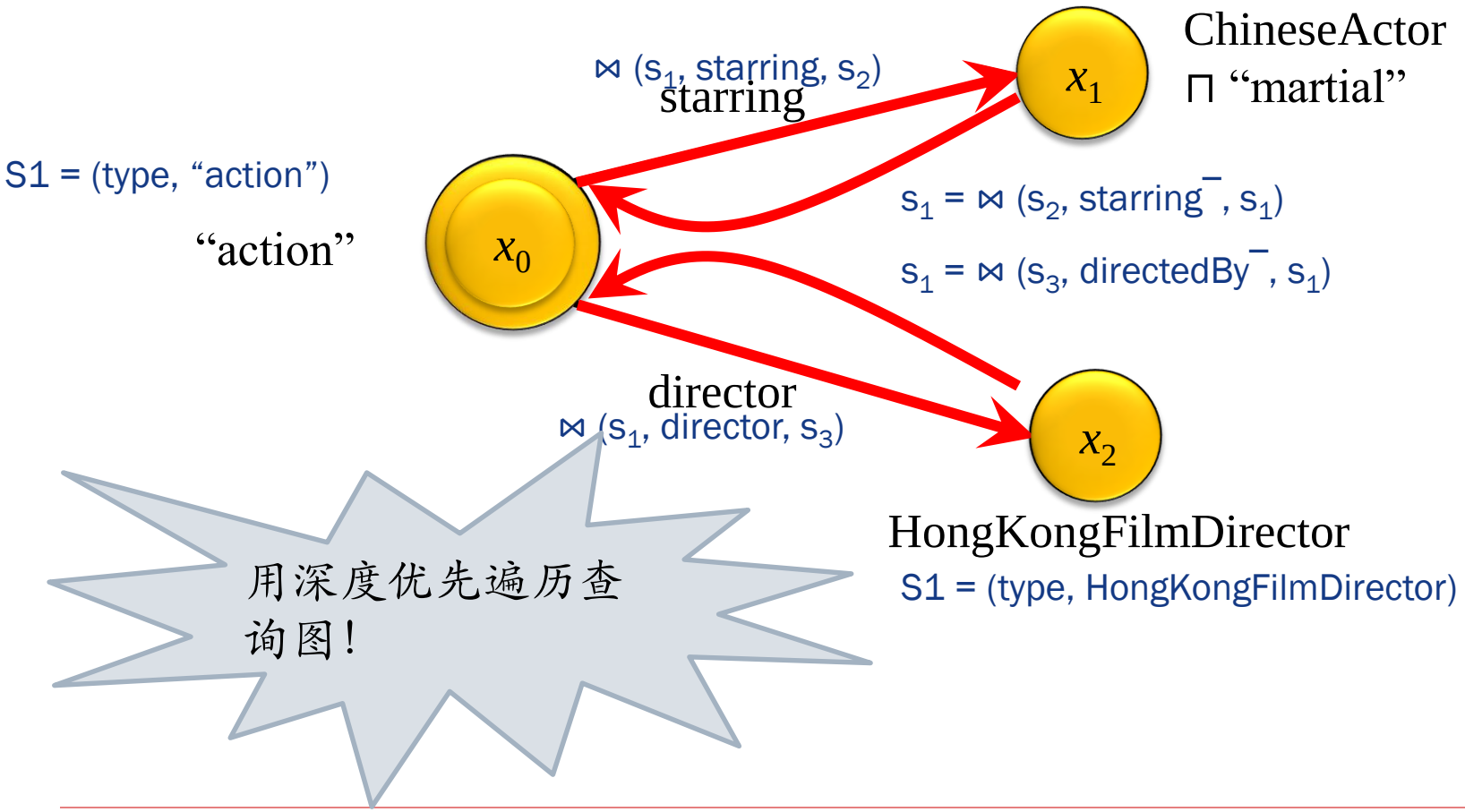
Related Information

- director(50)
- starring(50)
- language(48)
- writer(42)
- country(41)
- producer(36)
- music(21)
- cinematography(19)
- released(18)
- distributor(17)

More

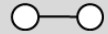
如何找到复杂查询的答案

例子



不同的查询种类(结构化查询)

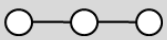
Single-atom queries (QC1)



$Q_{DBP}2$

```
PREFIX  res: <http://dbpedia../resource/>
PREFIX  prop: <http://dbpedia.org/property/>
SELECT ?x
{
  res:George_PC3A1l prop:academyawards ?x
}
```

Path queries (QC2)

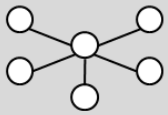


$Q_{LUBM}5$

```
PREFIX  lubm: <http://www.exam...#>
PREFIX  rdf: <http://www3.org/...rdf-syntax-ns#>
SELECT ?x
{
  ?x lubm:takesCourse  ?y .
  ?z lubm:teacherOf  ?y .
  ?z rdf:type  lubm:FullProfessor

}
```

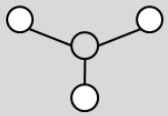
Star queries (QC3)



$Q_{LUBM}9$

```
PREFIX  lubm: <http://www.ex...#>
PREFIX  rdf: <http://www.w3.org...rdf-syntax-ns#>
SELECT ?x
{
   ?x lubm:takesCourse  ?y .
   ?b lubm:publicationAuthor  ?x .
   ?b lubm:name  "Publication7" .
   <...FullProfessor5> lubm:teacherOf ?y .
   ?x lubm:memberOf  ?z .
   ?a lubm:memberOf ?z .
   ?a lubm:telephone  "xxx-xxx-xxxx"
}
```

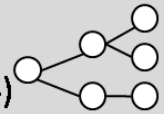
Entity query (QC4)



$Q_{DBP}6$

```
PREFIX  dbpedia: <http://...Category:>
PREFIX  prop: <http://dbpedia.org/property/>
PREFIX  rdf:   <http://www.w3.../rdf-syntax-ns#>
PREFIX  skos: <http://www.w3.org.../skos/core#>
PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?x
{
    ?x    prop:population "25089"^^<xsd:Integer>.
    ?x    prop:area     "428"^^<xsd:Integer>
}
```

Tree-shaped queries (QC5)



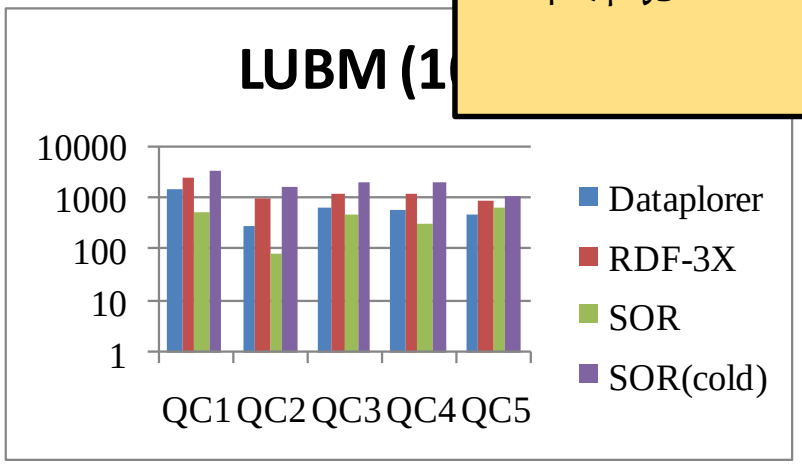
$Q_{DBP}15$

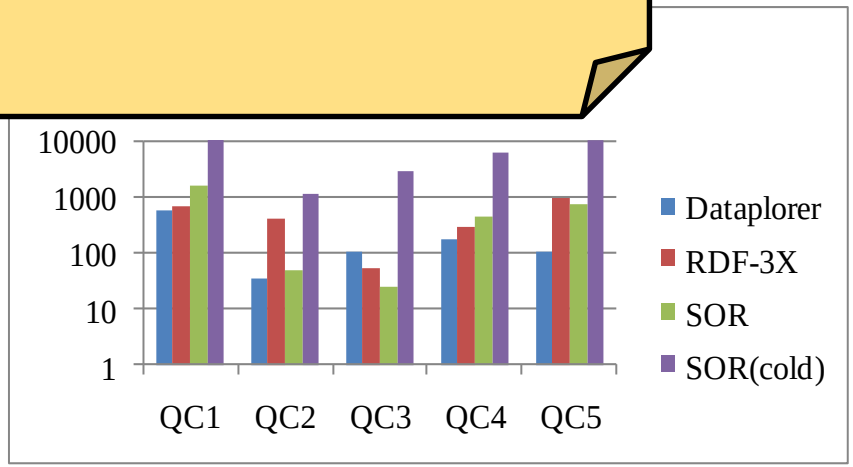
```
PREFIX  dbpedia: <http://...Category:>
PREFIX  prop: <http://dbpedia.org/property/>
PREFIX  rdf:   <http://www.w3.../rdf-syntax-ns#>
PREFIX  skos: <http://www.w3.org.../skos/core#>
SELECT ?x
{
   ?x prop:starring ?y.
   ?y skos:subject  dbpedia:Action_film_actors .
   ?a prop:starring  ?y.
   ?a prop:music <...resource/James_Wong> .
   ?x prop:director  ?z.
   ?z skos:subject dbpedia:H..._K..._film_directors
}
```

不同查询种类的响应时间(毫秒)

Query Class	LUBM (1000)				DBpedia			
	Dataplorer	RDF-3X	SOR	SOR (cold)	Dataplorer	RDF-3X	SOR	SOR (cold)
QC1	1,485	2,409	494	3,240	550	646	1,543	11,181
QC2	277							1,171
QC3	619							2,945
QC4	556							5,998
QC5	463							10,484

- 缓存可以大大地提高效率.
- 精心设计的查询功能的优化算法也可以缩短响应时间.
- 效率和查询表达式的复杂程度之间总是有一个折衷





排序原则

- 质量传播 (quality propagation): 一个元素的分数可以看成是其质量(quality)的度量，质量传播即通过更新这个分数同时反应该元素的相邻元素的质量。

- 例如： 当查找匹配关键字“战争”的美国总统的接班人时，肯尼迪应该排在前面，因为他的前任总统艾森豪威尔与“战争”紧密相关。

排序原则(续)

- **数量聚合：**除质量外，还考虑邻居的数量。因此，如果有更多的邻居，元素排名会更高

- **例如，**在查询“找到图灵奖获得者工作的机构”，CMU,UC伯克利和IBM是排在前3的机构，因为它们拥有最多的图灵奖获得者。

- **注意：排序方案需要满足单调性(monotonicity)**

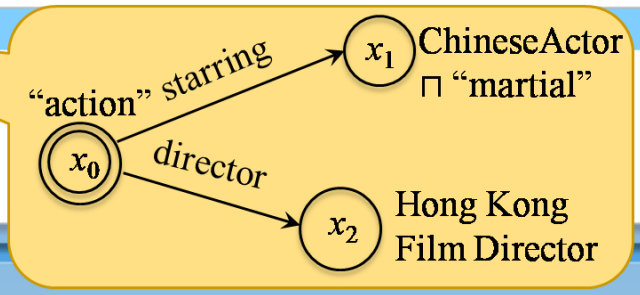
如何将排序紧密结合到基本操作中？

- 基本操作
  - 1. 基本检索 $b(f,t)$
    给定field f 和 term t，$b(f,t)$ 从倒排索引中检索出posting list，并输出一个Ascending Integer List (AIS)
  - 2. 归并排序 $m(S_1,\cap,S_2)$
    S_1 和 S_2 是两个AIS，$m(S_1,\cap,S_2)$ 计算S_1 and S_2 的交集
  - 3. 关系扩展 $u(S,R)$
    给定关系R和AIS S，$u(S,R)$ 计算集合 $\{o \mid \exists s: s \in S \wedge (s,R,o)\}$ 并将其作为AIS返回
- 广义操作 $S' = \{\langle d_1, 0.4\rangle, \langle d_3, 0.8\rangle, \langle d_4, 0.6\rangle, ...\}$
  - 每个AIS添加一个打分功能，形成Scored AIS（SAIS）
    例如，$S'[d_1] = 0.4$
  - 1. $b'(f,t)[d] = RSV(t,d)$
    RSV(t,d) in [0,1]，根据TF-IDF计算d w.r.t t的相关性
  - 2. $m'(S'_1,\cap,S'_2)[d] = S'_1[d] \cdot S'_2[d]$ S'$_1$ 和S'$_2$ 是两个SAIS
  - 3. $u'(S',R)[o] = 1 - \prod_{s:s\in S'\wedge(s,R,o)}(1 - S'[s])$ 根据概率论定义

Query Tree

action | Search | New Query | Sample Queries

"action"
director :: Hong Kong film directors ""
starring :: Chinese actors "martial"



Results 1 - 10 of 50 (0.266 seconds)

Heart of Dragon Select this item
"heart of dragon (long de xin) is a 1985 hong kong **action** film directed by sam hung, assisted
http://dataplorer.apexlab.org/Heart_of_Dragon

Dragons Forever Select this item
"dragons forever (\u98db\u9f8d\u731b\u e made
http://dataplorer.apexlab.org/

Wheels on Meals Select this item
"wheels on meals (ku\u00e0ic\u0101n mo
http://dataplorer.apexlab.org/Whe

My Lucky Stars Select this item
kong **action** comedy, directed by sammo hun the lm in y stars se n\nthe
http://dataplorer.apexlab.org/My_Lucky_Stars

Eastern Condors Select this item
"eastern condors is a 1987 hong kong **action** film set in the aftermath of the vietnam war
http://dataplorer.apexlab.org/Eastern_Condors

Winners and Sinners Select this item
\") is a 1983 hong kong **action** comedy film directed and co-written by sammo hung, it was the first

Heart of Dragon (a well-known action film) has the largest number of famous martial actors

Navigator

Categories

- TOP Category(50)
- Cantonese language films(48)
- Hong Kong films(48)
- Martial arts films(35)
- Kung fu films(13)
- Action films(9)
- Action thriller films(7)
- films(4)
- films(4)
- films(4)

More

Related Information

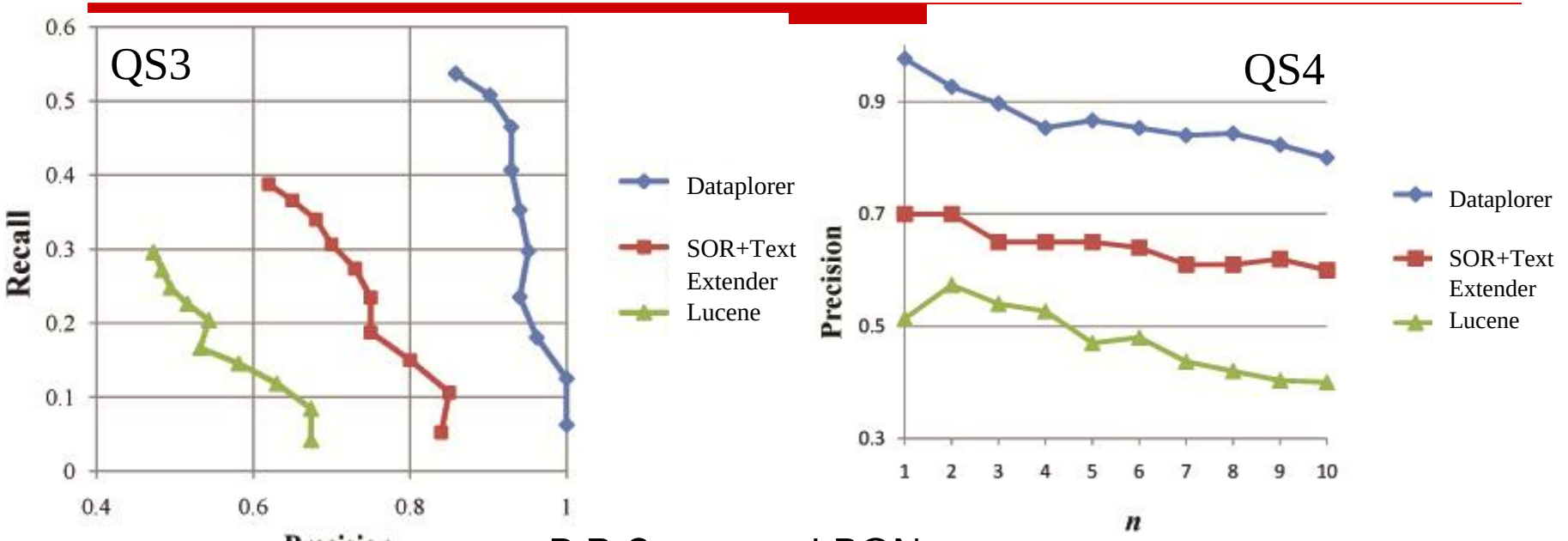
- director(50)
- starring(50)
- language(48)
- writer(42)
- country(41)
- producer(36)
- music(21)
- cinematography(19)
- released(18)
- distributor(17)

More

从DBpedia收集的混合的查询数据集

- QS1包含500个简单的关键字查询
- QS2包括1,242个具有类型约束的简单结构化查询
- QS3有287个具有一个关系的混合查询
- QS4根据10位用户提供的问题收集20个查询

在QS3和QS4上的实验结果



P-R Curve and P@N

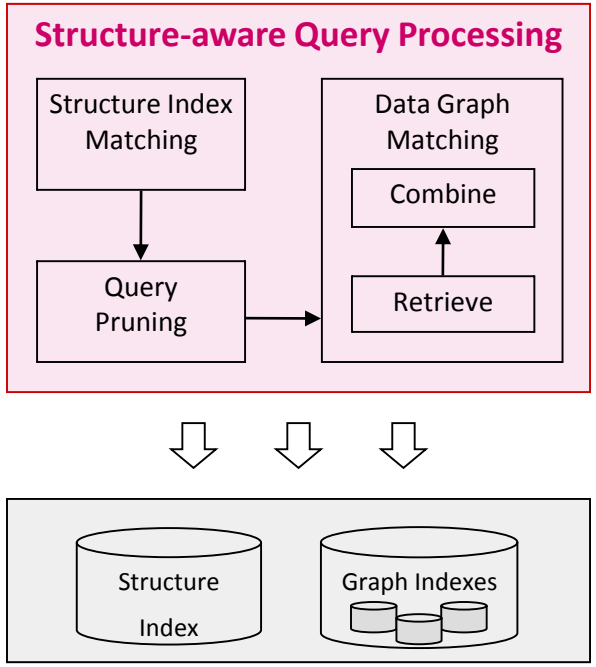
low → high

Lucene → SOR+Text Extender → Dataplorer

从数据的结构和排序原则中获得提升

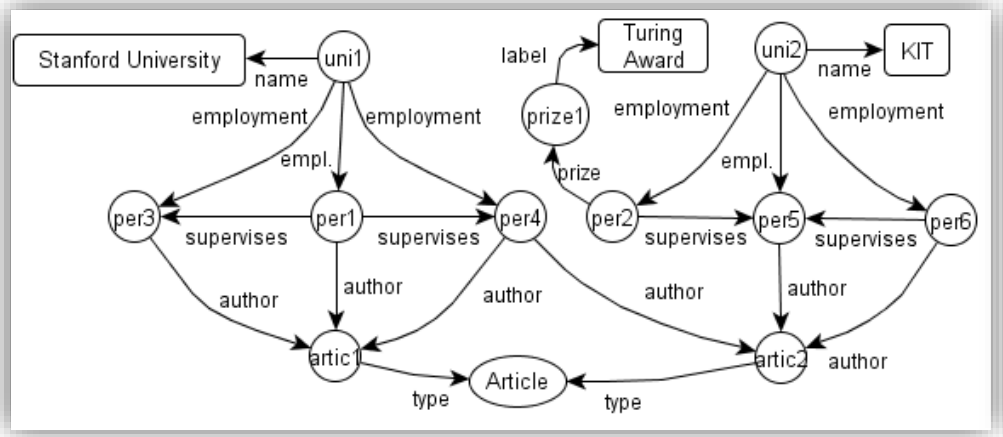
高效和可扩展的数据Web搜索

- 基于结构的分区和查询处理
- 基于结构的索引和分区
  - 将结构上相似的节点聚合到一起
  - 结构上相似的节点在硬盘上被连续的存储
- 结构感知(Structure-aware)的查询处理
  - 两阶段匹配
  - 1）只检索出匹配所查询的结构的数据
  - 2）通过剪枝(pruning)减少join和IO

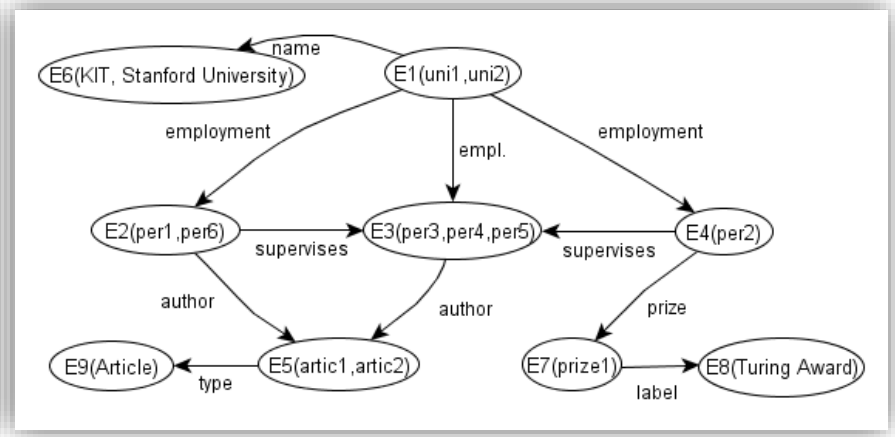


为图结构数据(RDF)建立结构化索引

数据图的一个例子 (e.g. RDF data)



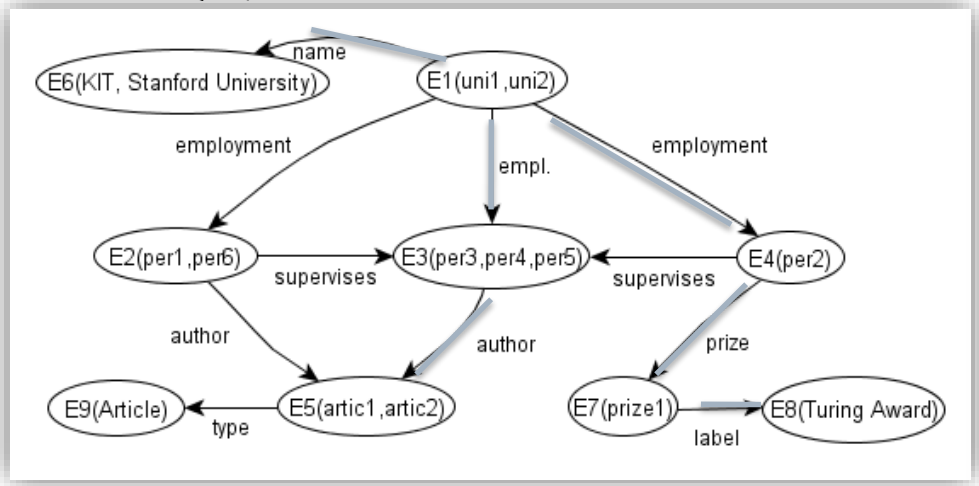
对应的**structure index**



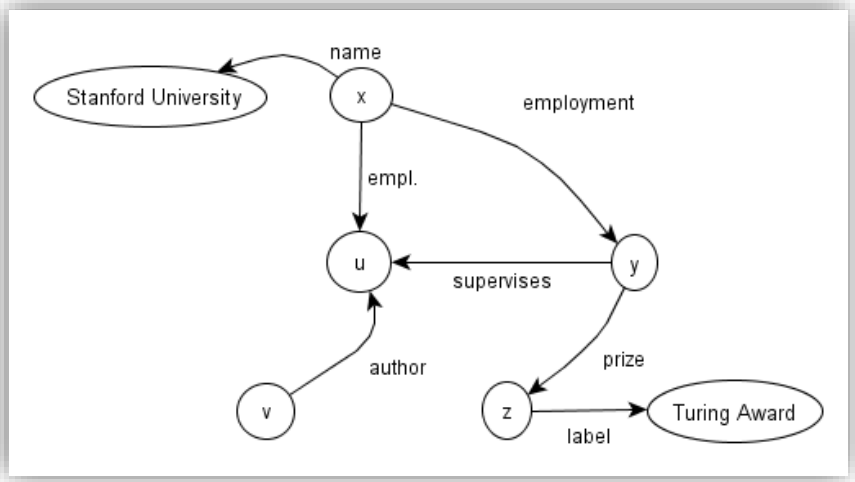
- 结构索引由称为扩展(extensions)的类和它们之间的关系组成
- 扩展中的资源具有相同的结构，即具有相同的(传入和传出)路径
- 结构索引的建立基于互模拟(bisimulation)
- 结构索引是更精细的结构描述(模式)

使用结构索引做结构匹配

结构索引

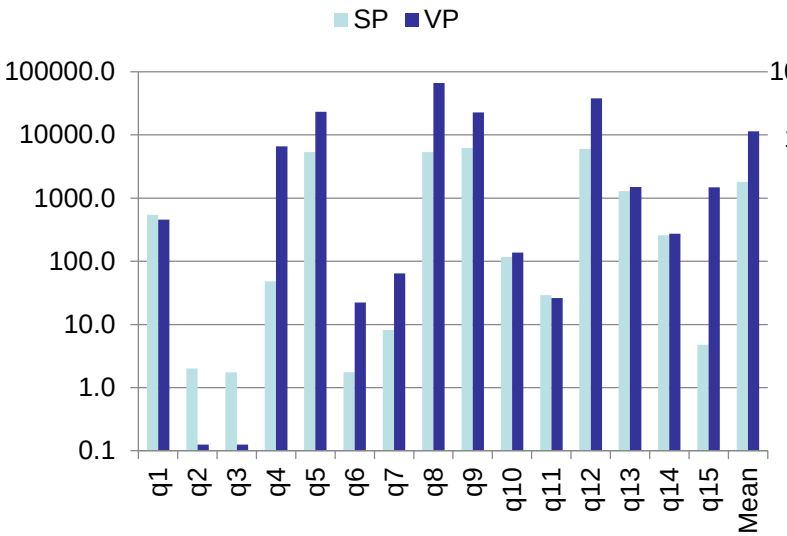


一个示例查询

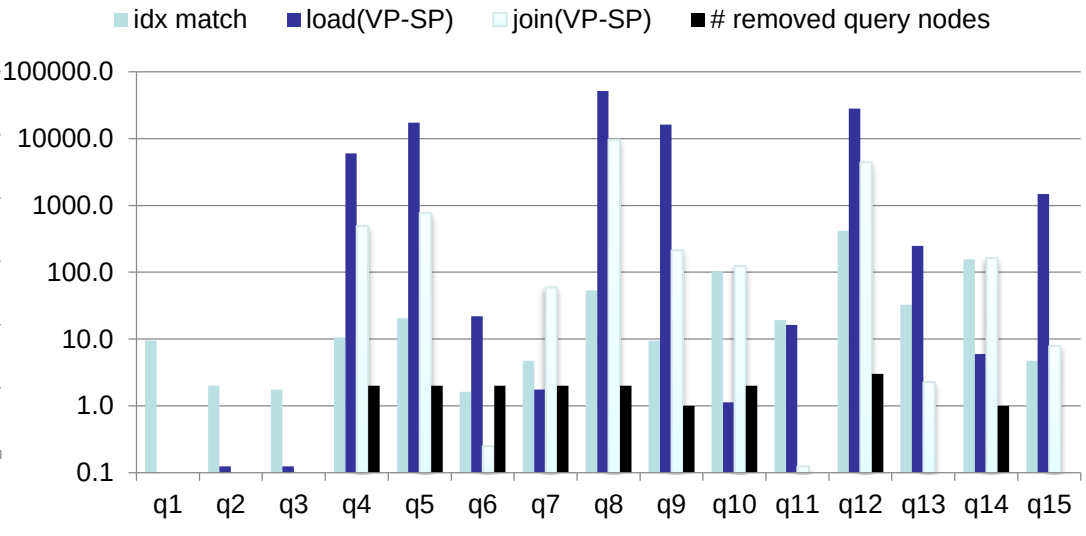


- 在答案空间里检索和join:用结构索引匹配查询
  - 产生一组所包含的数据元素**匹配查询中的结构**的结构索引
- 根据匹配的结构索引计算最终答案
  - **剪枝**仅包含非标识(non-distinguished)变量的的**树形查询部分**
  - 在资源空间检索和join:验证答案空间匹配中的元素是否也匹配具体的查询实体，即常量和标识(distinguished)变量
- **优点**：降低IO开销和union和join操作的次数

评估



Total time in ms on DBLP

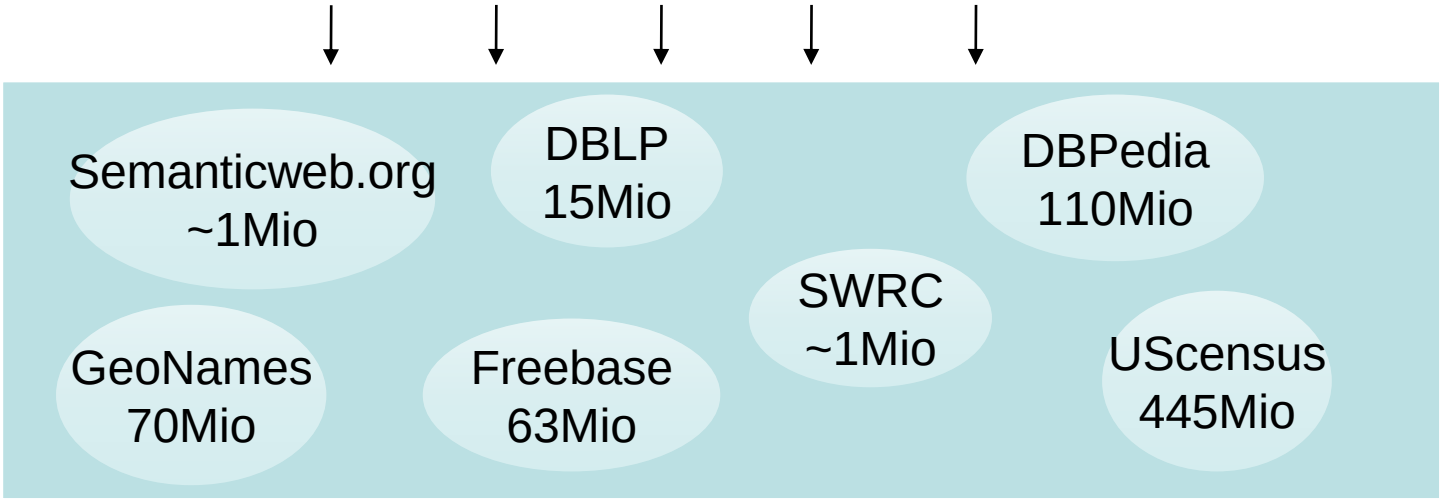


Time of separate steps in ms, #pruned query nodes

- 比较SP和垂直分区方式VP
- 对于复杂查询(4-15),SP要快8-9倍，对于简单查询,SP要稍慢一些
- 对于更复杂的查询，由索引图匹配引起的开销可能超过加载和加入的累积增益

在多数据源，多存储库的场景下搜索

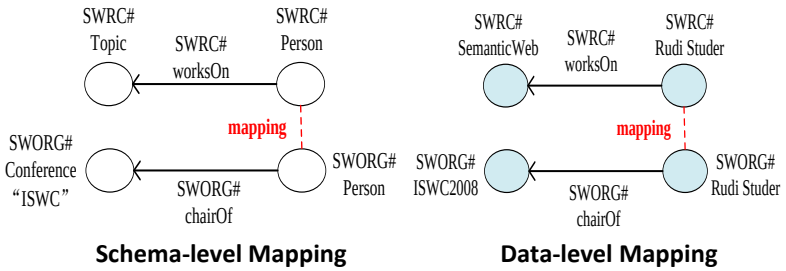
"Find articles from Turing Award winners at Stanford University"



- DBPedia 包含与Turing Awards winners相关的数据
- Freebase 包含Stanford and it's employees的数据
- FOAF profile 包含与people (from Stanford)相关的数据
- DBLP: 包含 参考文献数据 (of publication from Stanford)
- …

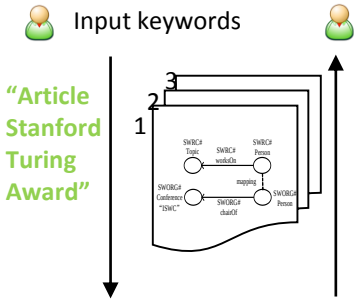
Hermes – Pay-as-you-go数据Web搜索框架

1. Integrate data sources (offline)

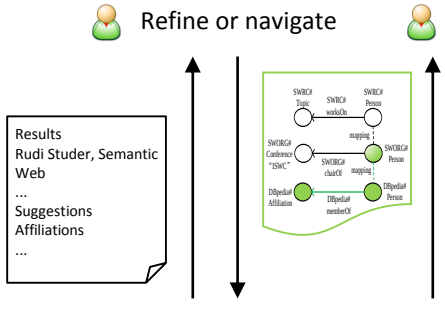


Schema-level Mapping **Data-level Mapping**

2. Understand user's need



3. Search and refine



Graph Data Processing

- Data Graph Summarization
- Element Label Extraction
- Graph Element Scoring
- Mapping Discovery
- Indexing

Keyword Translation

- Keyword Mapping
- Top-k Query Graph Search

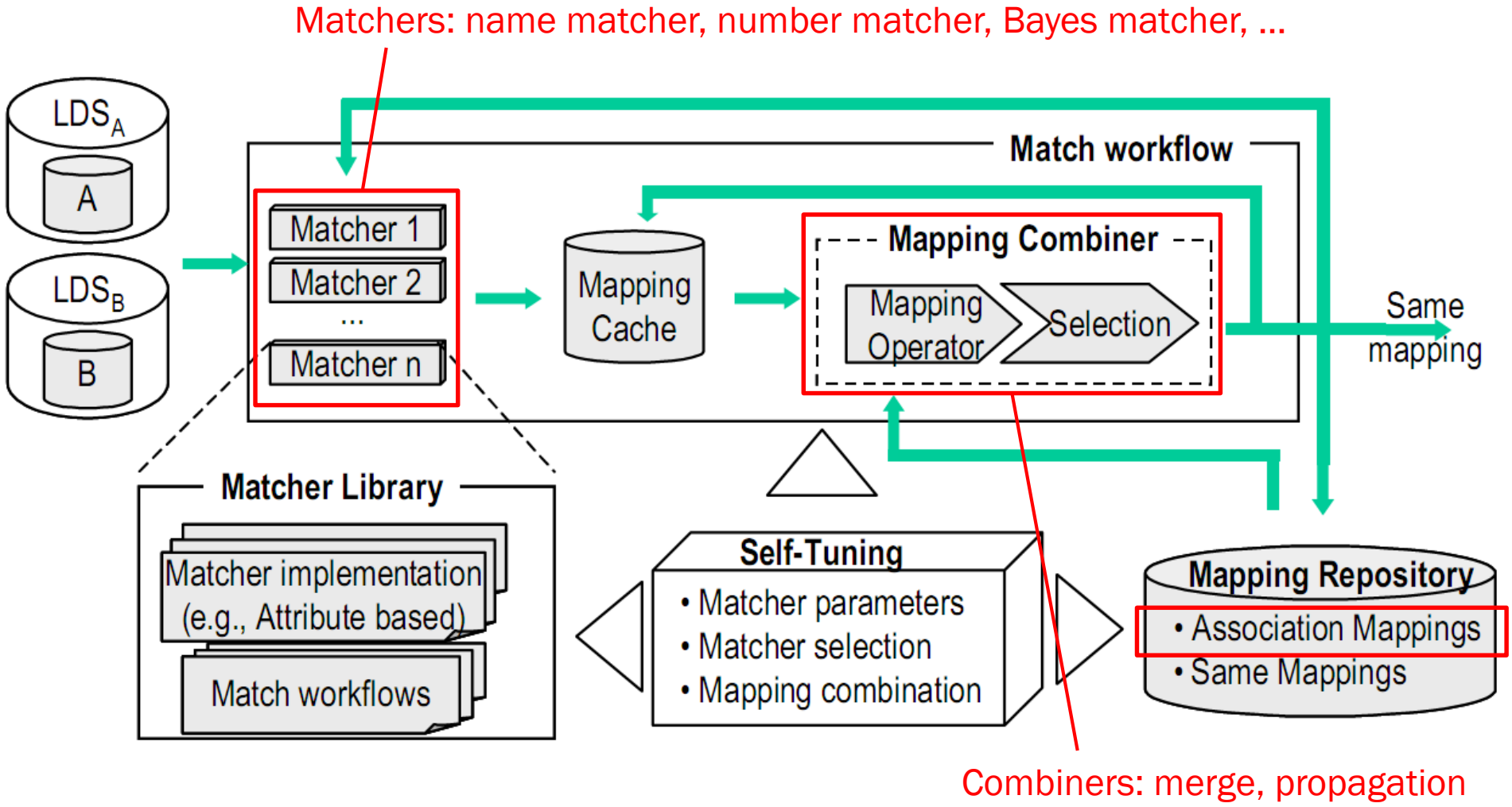
Distributed Query Processing

- Query Graph Decomposition
- Result Combination & Ranking
- Query Planning & Optimization
- Local Query Processing

Internal Indices

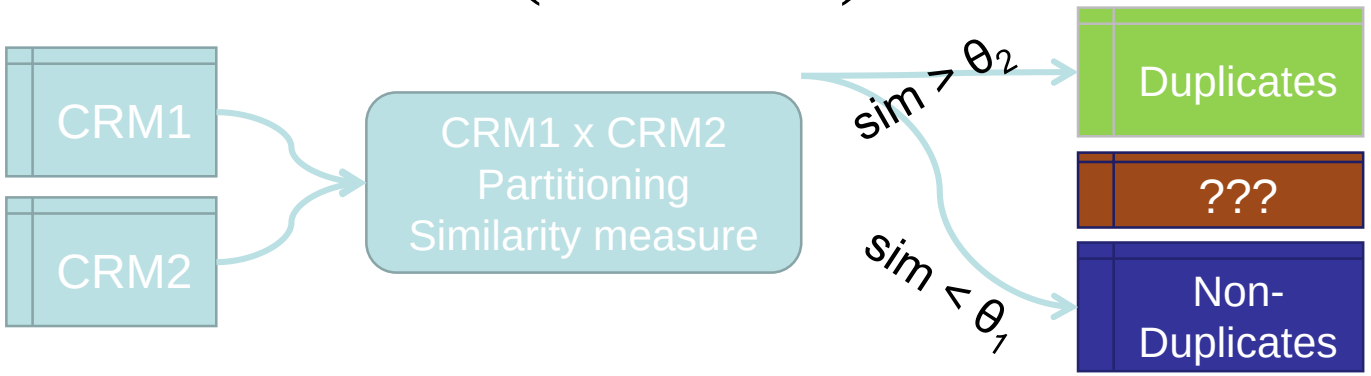
Keyword Index | Schema Index | Structure Index | Mapping Index | Graph Indices

知识融合工作流程

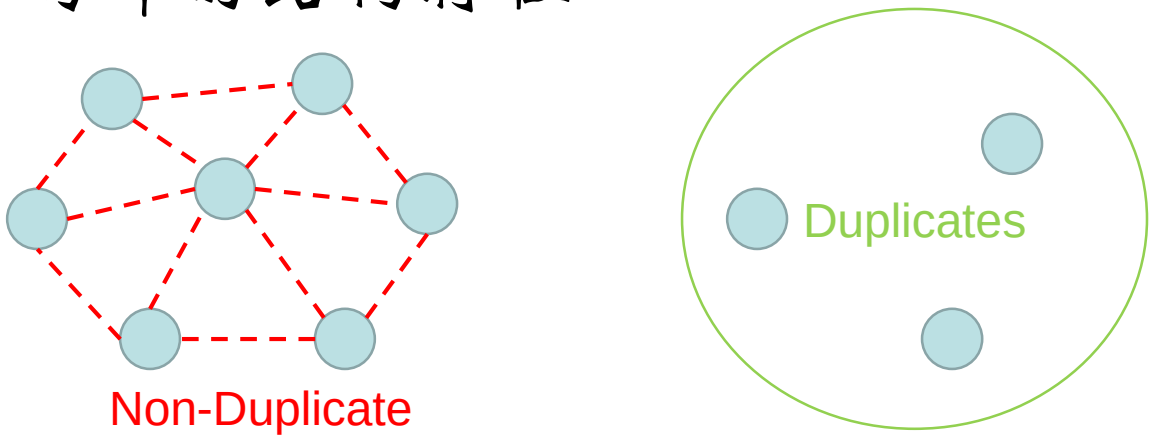

Matchers: name matcher, number matcher, Bayes matcher, ...
LDS_A
A
LDS_B
B
Match workflow
Matcher 1
Matcher 2
...
Matcher n
Mapping Cache
Mapping Combiner
Mapping Operator
Selection
Same mapping
Matcher Library
Matcher implementation (e.g., Attribute based)
Match workflows
Self-Tuning
• Matcher parameters
• Matcher selection
• Mapping combination
Mapping Repository
• Association Mappings
• Same Mappings
Combiners: merge, propagation

知识融合主流方法分类

- 基于全局的阈值(threshold)



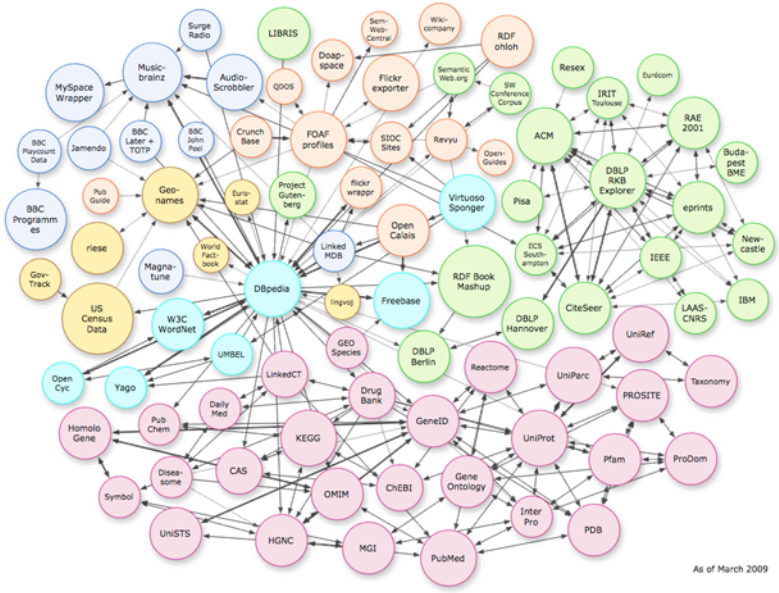
- 基于局部的结构特征



知识融合的主要挑战

- 效率
  - 大多数现有的研究集中于提高精确度和召回率，而忽视了效率
- 增量整合
  - 不断演变的数据和匹配结果
- 运行时整合
  - 整合查询结果集合

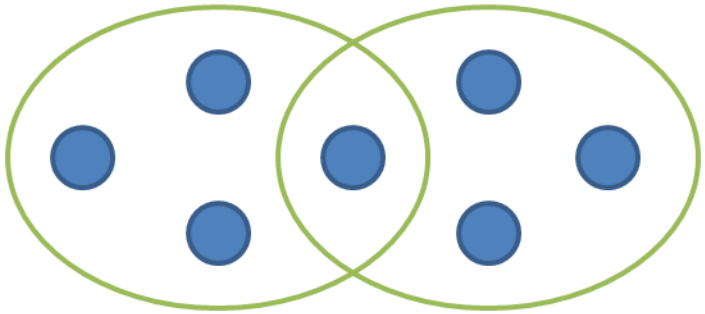
两阶段知识融合来解决效率瓶颈



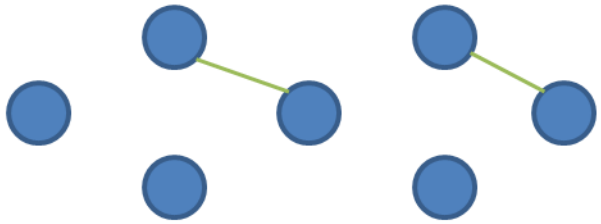
Index

URI	Features
<x>	A Little Help from My Friends
<y>	With A Little Help From My Friend
<z>	Harry Potter and the Philosopher's Stone
<u>	Harry Potter and the Chamber of Secrets
<v>	Harry Potter and the Prisoner of Azkaban
<w>	Harry Potter and the Goblet of Fire

Block

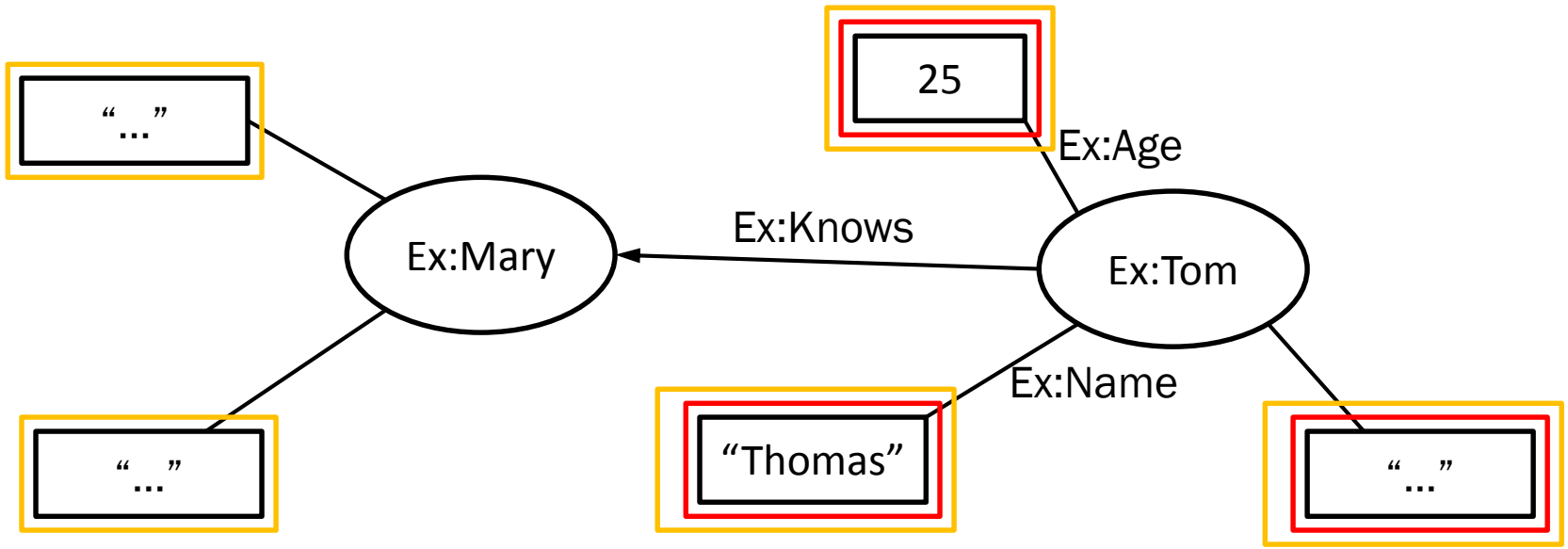


Cluster



索引 (Indexing)

- Extract literal attribute values of entities as their *basic features*
- Extract also their neighbors' basic features as their *extended features* (optional)



分块 (Blocking)

- 直观: 共享稀有特征 (*rare features*)的实体更可能是同一个实体
- 根据文档频率 (*document frequency*)来排序每个实体的特征
- 每个保留的倒排索引列表对应一个分块

$$
\begin{aligned}
w &= \{\underline{A}, B, C, K\} \\
x &= \{\underline{A, B}, E, F, K, L\} \\
y &= \{\underline{B}, K, L\} \\
z &= \{\underline{C}, K, L\}
\end{aligned}
\qquad
\begin{aligned}
A &: \{w, x\} \\
B &: \{x, y\} \\
C &: \{z\}
\end{aligned}
$$

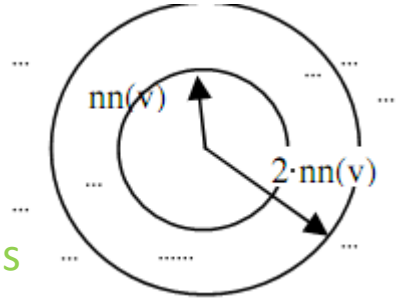
聚类 (Clustering)

- 紧致集合 Compact Set (CS)
  - 在一个CS中的实体互为最近邻
- 稀疏邻居 Sparse Neighborhood (SN)
  - The aggregated *neighborhood growth* values are low enough
- 基于CS & SN原则在每个分块中进行聚类
- 复杂度: O(n^3)

Entity	Name
x	A Little Help from My Friends
y	With A Little Help From My Friend
z	Harry Potter and the Philosopher's Stone
u	Harry Potter and the Chamber of Secrets
v	Harry Potter and the Prisoner of Azkaban
w	Harry Potter and the Goblet of Fire

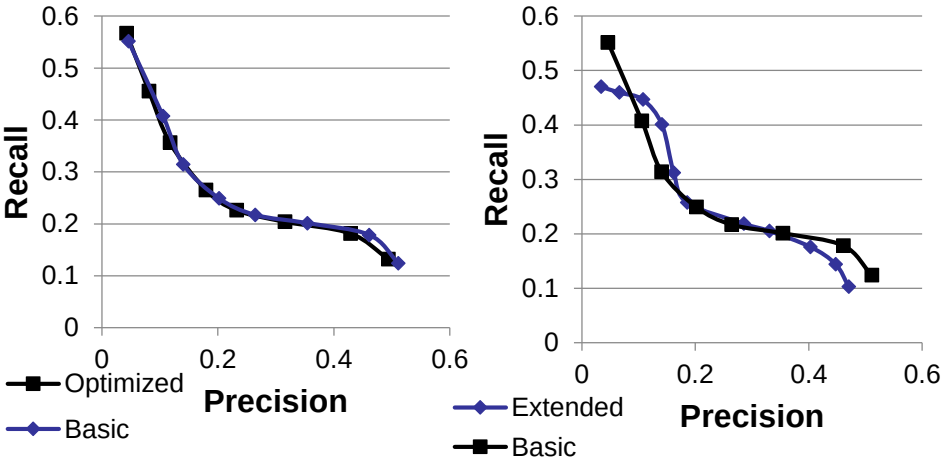
Duplicates

Non-Duplicates

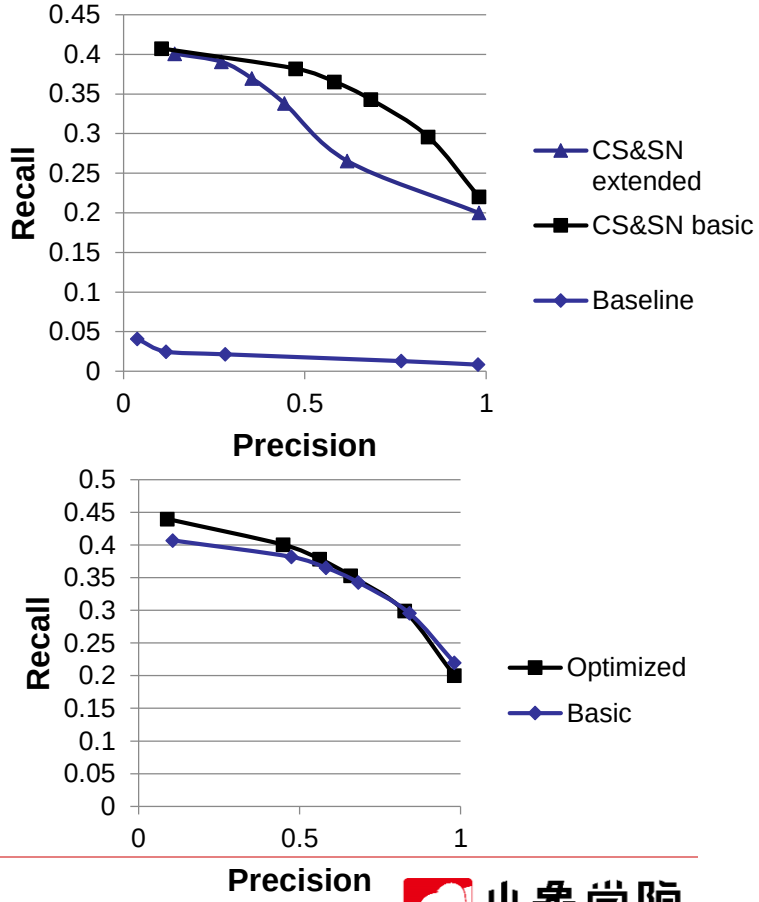


迭代式数据整合 – 分块和聚类的效果

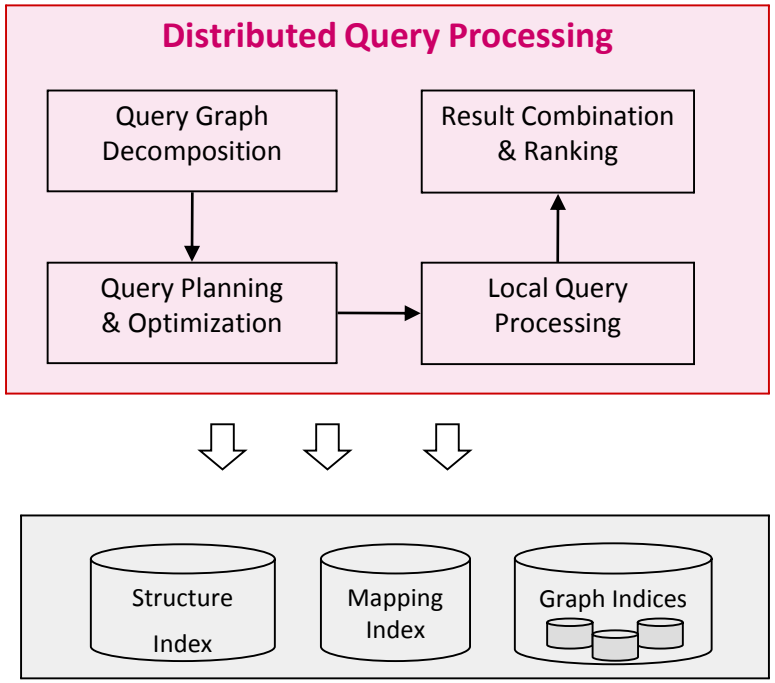
- 分块
- 扩展性: block 1,250,000 individuals in 10 min (vs. cluster 1,700 individuals in 10 min)
- 块大小: < 500 (聚类时间 < 16s)

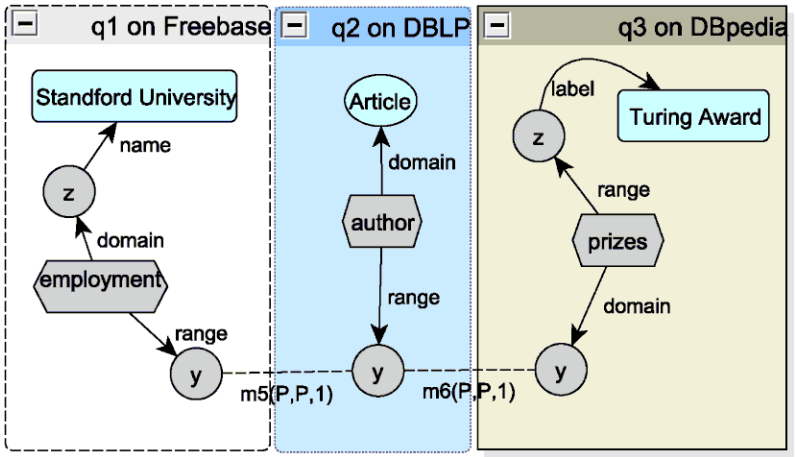


- 聚类
- 使用Jaccard相似性度量



联合查询处理





查询结果合并

- **Map join** 而不是 similarity join
- 利用预先计算的实体映射来获得两列“映射关系”

q1 on Freebase			m5		q2 on DBLP		m6		q3 on DBpedia		
name	**z**	**y**	**fb:Person**	**dblp:Person**	**y'**	**x**	**dblp:Person**	**dbp:Person**	**y''**	**z**	**label**
Stanford	uni1	per1	per1	per2	per2	pub1	per2	per3	per3	prize1	Turing Award
Stanford	uni2	per8	per8	per9	per9	pub2	per9	per7	per3	prize2	Turing Award

Mapping relations & intermediate query results

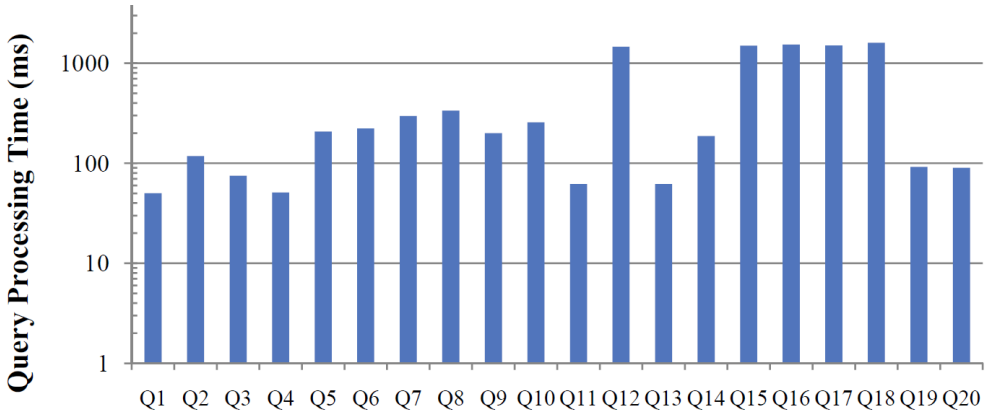
数据预处理

- 单台机器, DualCore 3Ghz, 8G RAM
- 数据集
  - 1.2 billion RDF三元组，2Mio 数据层面的映射和1,500 schema层面的映射

	Keyword Index	Structure Index		Mapping Index	Graph Index	
Data Set	Size(MB)	Time(s)	Size(MB)	Size(MB)	Time(s)	Size(MB)
SwetoDBLP	1,060	5,400	0.02	2.22	1522	655
DBpedia	2,630	36,000	75	42.7	14946	7,522
Freebase	1,590	10,800	29	48.4	8736	2,268
USCensus	980	21,600	0.01	4.1	63195	18,948
GeoNames	4,110	21,600	0.006	0	29714	3,431
SemanticWeb.org	7.35	30	4.2	1.2	53	5
SWRC	3.04	10	0.08	0.022	3	1.4

评估

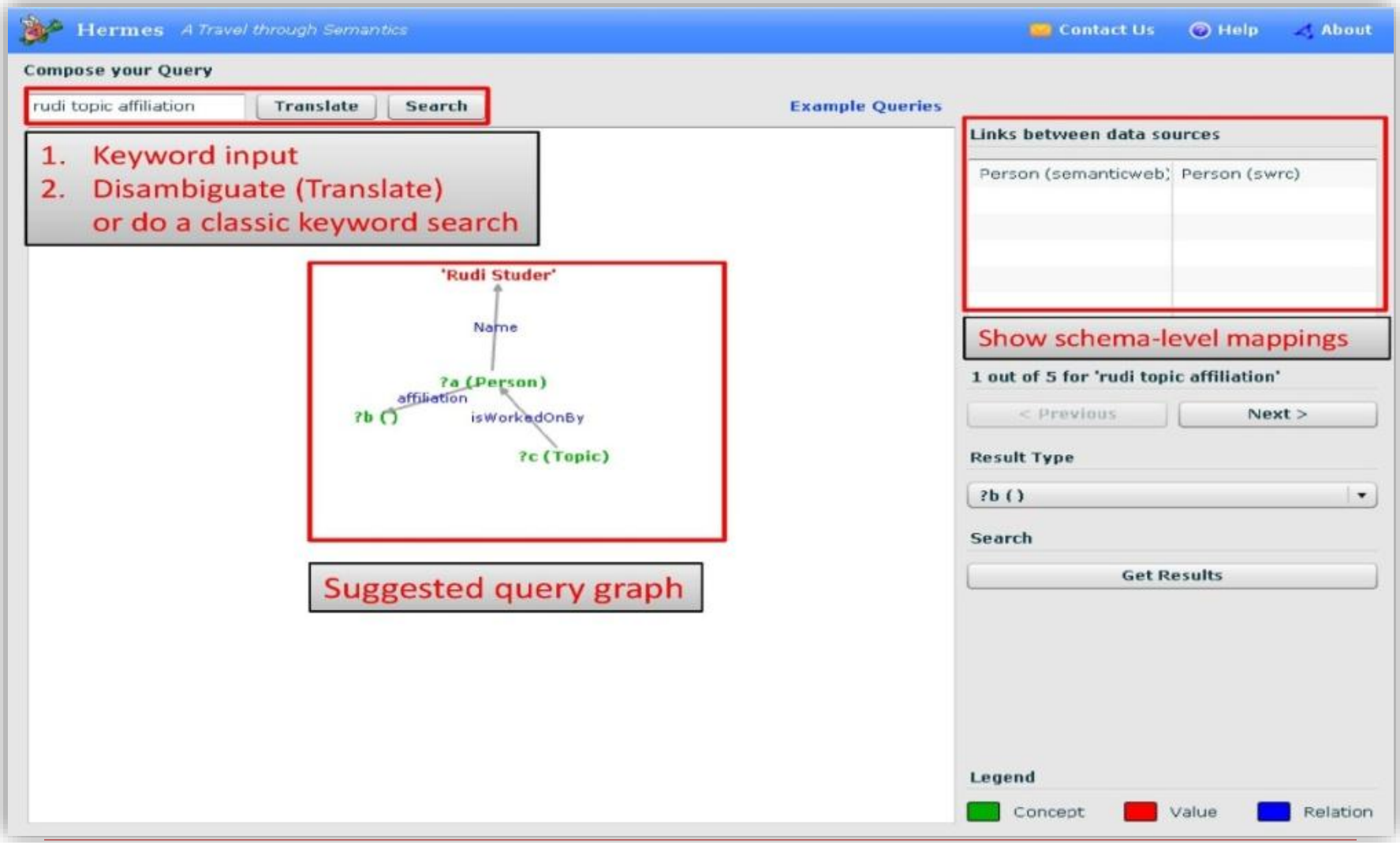
#	Keywords	Data Sources
Q1	project, ISWC, person	SWRC
Q2	Studer, publication	SWRC
Q3	undergraduate, topic	SWRC
Q4	Rudi, proceedings	SWRC
Q5	track, 323	Freebase
Q6	Pinocchio, film	Freebase
Q7	company, owner, "shopping center"	Freebase
Q8	restaurant, Berlin	Freebase
Q9	"the day after tomorrow", director	Freebase
Q10	film, Titanic	Freebase
Q11	ISWC2008,Studer, topic	SwetoDBLP, SW.org, SWRC
Q12	person, Shanghai, town	
Q13	Ronny Siebes, InProceeding	
Q14	tom, iswc2008, proceedings	
Q15	lake, citytown, wood	
Q16	person, town, village	
Q17	restaurant, german	Freebase, USCensus
Q18	album, town, mountain	Freebase, USCensus
Q19	Frank, publications	SwetoDBLP, SWRC
Q20	Markus, report	SwetoDBLP, SWRC



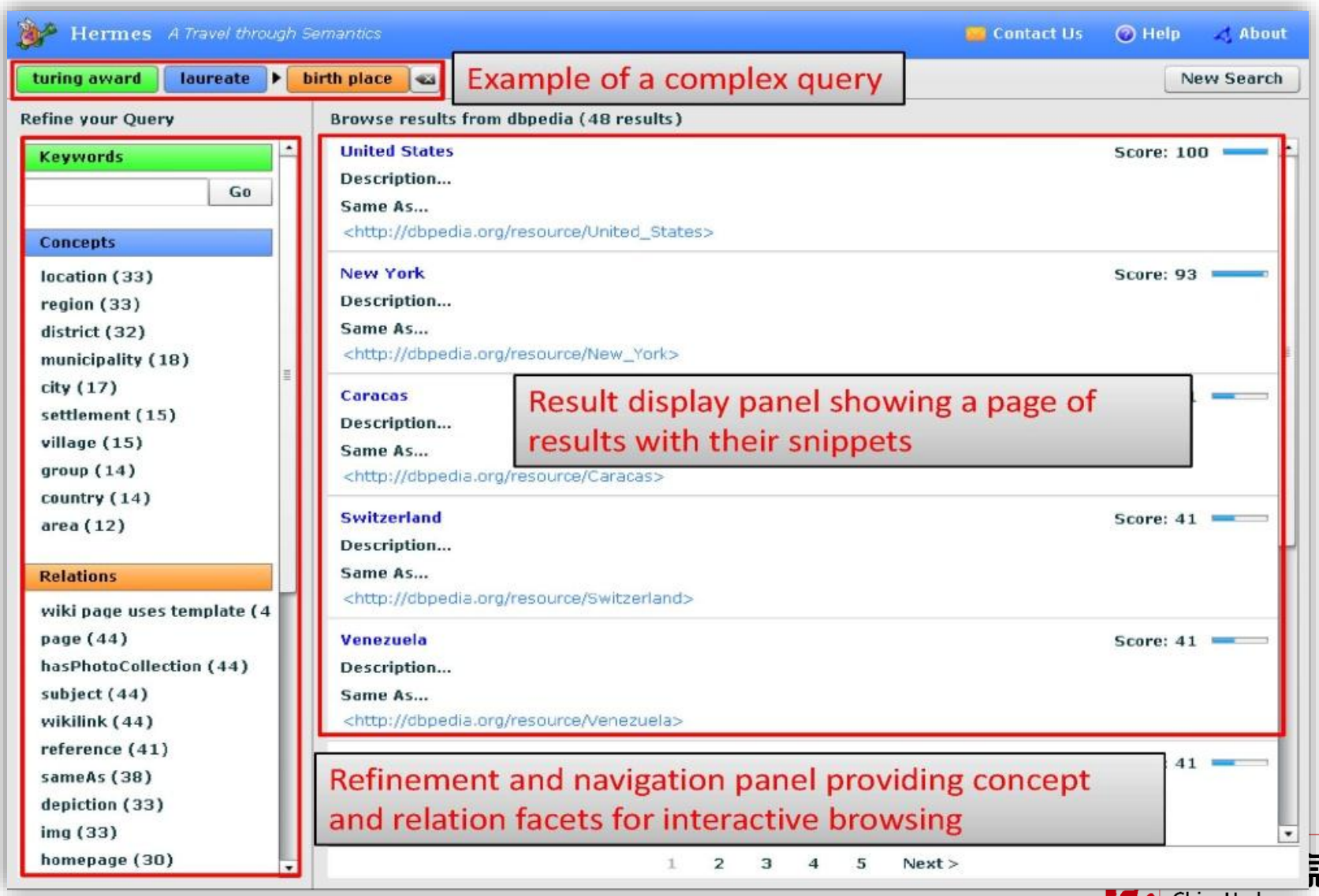
Queries	Local Join (ms)	Map Join (ms)	#Local Joins	#Map Joins
Q11	42.47	19.53	2	2
			9	4
			7	3
			83	82
			01,721	301,710
			2,064	2,063
Q17	1071.28	429.72	16,371	16,370
Q18	1179.17	419.83	23,394	23,385
Q19	70.6	21.4	8	3
Q20	62.47	27.53	874	873

- 多数据源查询处理比但数据源查询处理要多花一些时间
- Map join 相当于local join

查询翻译



查询和结果优化



结论

- 语义Web搜索有多种研究原型
- 有些应用IR技术以增强扩展性
- 复杂的查询也被诸如Hermes这样的搜索引擎支持，这些引擎结合基于倒排索引的数据访问，IR排序和复杂查询处理的数据库技术
- 可靠的答案，等价于 能扩展到数十亿级的三元组
- **数据质量**依然是一个问题
- 对**高质量映射**进行增量的，pay-as-you-go方式的计算和维护
- **集成IR和DB排序**以处理复杂查询

大纲

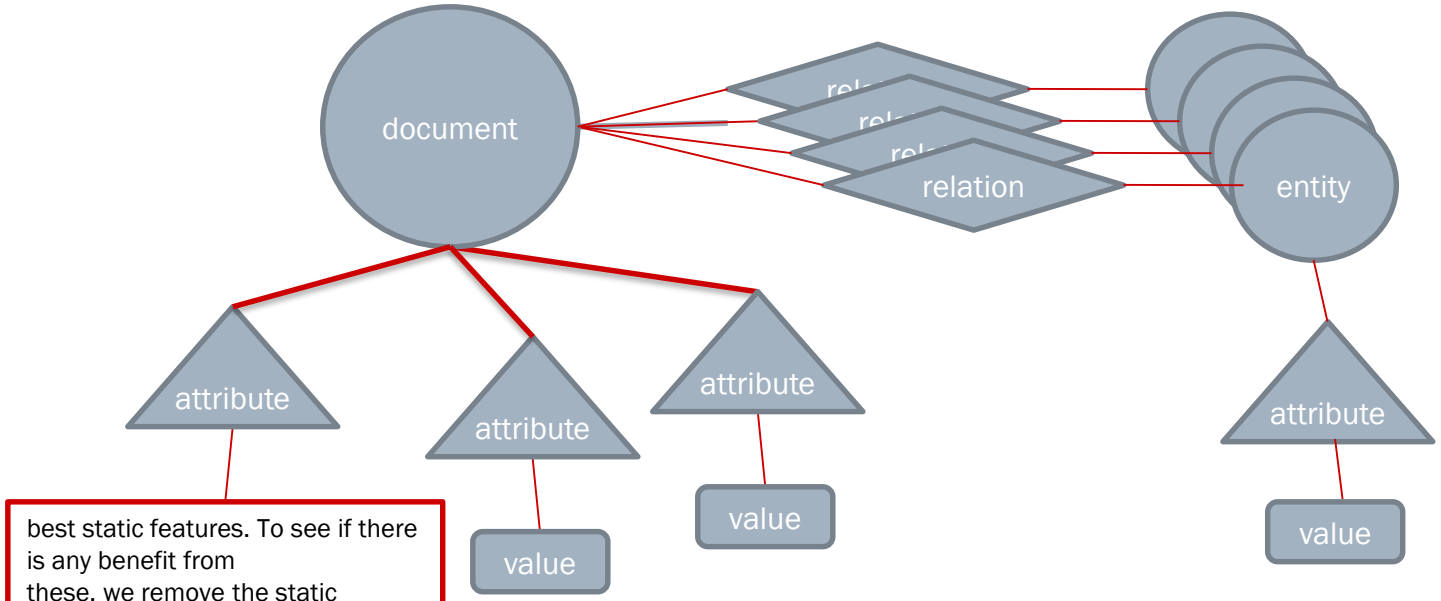
- 语义搜索简介
- 语义数据搜索
- 混合搜索
- 语义搜索的交互范式
- 实践展示：使用Elasticsearch实现简单语义数据检索

混合语义搜索系统

- “下一代语义搜索系统结合了一系列技术，从基于统计的IR排序方法，有效索引和查询处理的数据库方法，到推理的复杂推理技术等等”

- 混合的语义搜索系统
  - 结合文本，结构化和语义数据
  - 以整体的方式管理不同类型的资源
  - 支持结果为信息单元(文档，数据）的集成的检索

混合搜索 数据模型

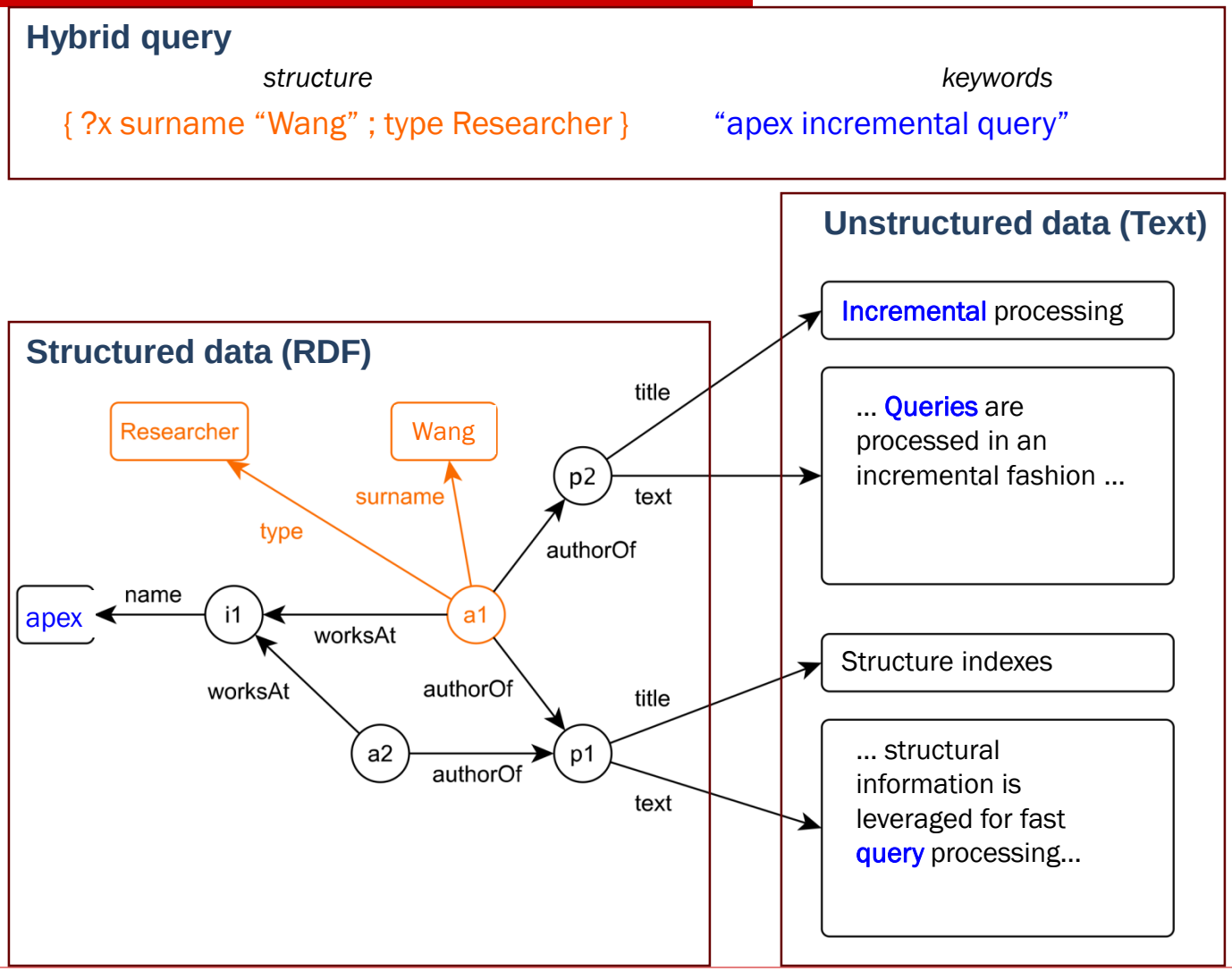


best static features. To see if there is any benefit from these, we remove the static features in the second tuning and use a uniform tuning of BM25F in the third. In the fourth and fifth tuning we add some elements were more 'risky': two additional static features (URLs, RE) and then the augmented Subject field that also contains the From line. In the event, the tuning gave 'URLs' a weight of zero, so the resulting ranking function uses only three static features.

异构资源如

- 文档中的**文本**
- 与“文本”关联的**结构化数据**
- 具有丰富语义的**本体**

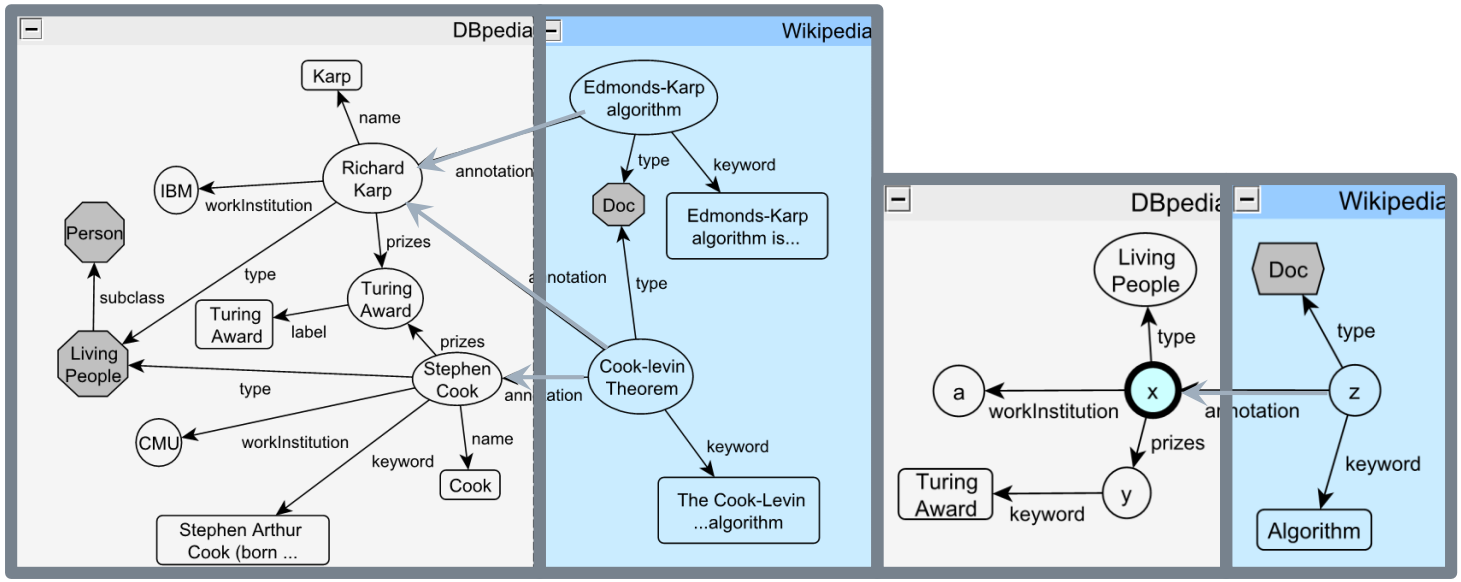
混合搜索 查询和数据模型



DB和IR的轻量级集成

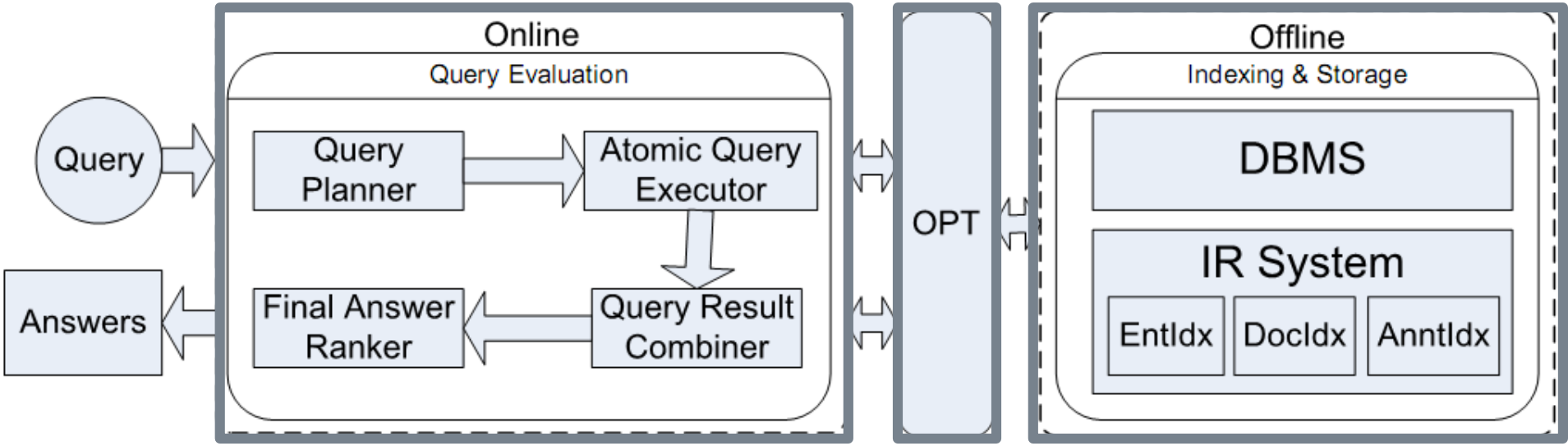
资源(查询)图

- 数据(查询)图 (概念，个体(或者变量)，关系，属性)
- 文档(查询)图 (文档概念，文档(或者变量)，超链接，关键词)
- 标注



示例资源图　　示例查询图

系统架构 (CE²)

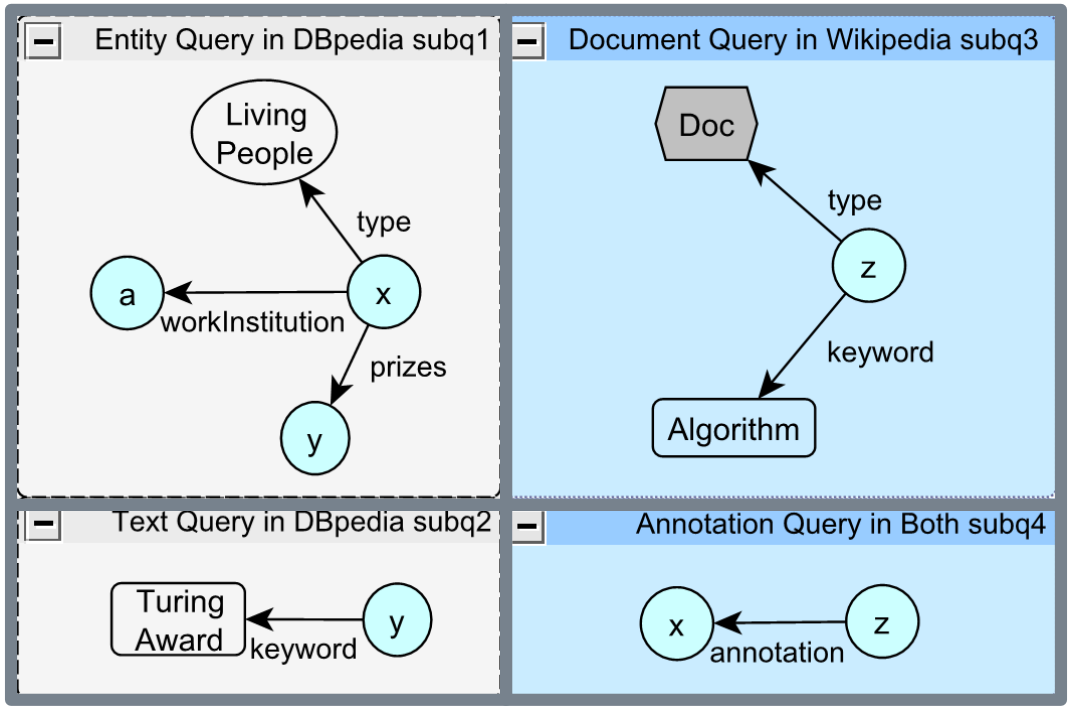


- 线下步骤
  - 数据图存储与DBMS中，除EntIdx中的三元组(个体，关键词，"xxx")外
  - Doc图存储在DocIdx中，注释存储在AnntIdx中
- 线上步骤
  - 将混合查询分解为一组原子查询(atomic queries) (步骤1)
  - 使用DB和IR引擎执行原子查询(步骤2)
  - 根据生成的查询树合并部分结果
  - 对最后的答案排序
- OPT(发生概率表)存储(部分)结果

查询分解和执行

原子查询类型

- 实体查询（由DBMS回答）
- 文本查询（由Entldx回答）
- 文档查询（由Docldx回答）
- 标注查询（由Anntldx回答）



通过分解得到的示例查询的子查询

OPT for $subq_1$

x	a	y	Row
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	1.0
(Stephen Cook, 1.0)	(CMU, 1.0)	(Turning Award, 1.0)	1.0

OPT for $subq_2$

y	Row
(Turing Award, 1.0)	1.0

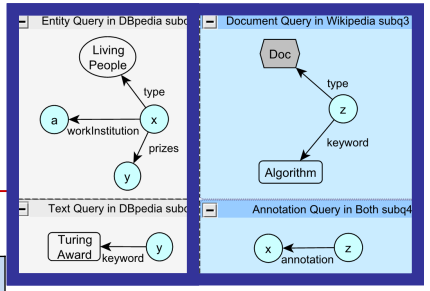
OPT for $subq_3$

z	Row
(Edmonds-Karp algorithm,0.9)	1.0
(Cook-levin Theorem,0.9)	1.0

OPT for $subq_4$

x	z	Row
(Richar Karp, 1.0)	(Edmonds-Karp algorithm,1.0)	1.0
(Richar Karp, 1.0)	(Cook-levin Theorem,1.0)	1.0
(Stephen Cook, 1.0)	(Cook-levin Theorem,1.0)	1.0

答案合并



OPT for $subq_1$

x	a	y	Row
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	1.0
(Stephen Cook, 1.0)	(CMU, 1.0)	(Turning Award, 1.0)	1.0

OPT for $subq_2$

y	Row
(Turing Award, 1.0)	1.0

OPT for $subq_5$

x	a	y	Row
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	1.0
(Stephen Cook, 1.0)	(CMU, 1.0)	(Turning Award, 1.0)	1.0

(a) Keyword Join on $subq_1$ and $subq_2$

OPT for $subq_5$

x	a	y	Row
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	1.0
(Stephen Cook, 1.0)	(CMU, 1.0)	(Turning Award, 1.0)	1.0

OPT for $subq_4$

x	z	Row
(Richar Karp, 1.0)	(Edmonds-Karp algorithm,1.0)	1.0
(Richar Karp, 1.0)	(Cook-levin Theorem,1.0)	1.0
(Stephen Cook, 1.0)	(Cook-levin Theorem,1.0)	1.0

OPT for $subq_3$

z	Row
(Edmonds-Karp algorithm,0.9)	1.0
(Cook-levin Theorem,0.9)	1.0

(b) Hybrid Join on $subq_5$ and $subq_3$ with $subq_4$

x	a	y	z	Row
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	(Edmonds-Karp algorithm,0.9)	1.0
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	(Cook-levin Theorem,0.9)	1.0
(Stephen Cook, 1.0)	(CMU, 1.0)	(Turning Award, 1.0)	(Cook-levin Theorem,0.9)	1.0

(c) Rank Projection by eliminating documents in *z*

x	a	y	Row
(Richar Karp, 1.0)	(IBM, 1.0)	(Turning Award, 1.0)	0.99
(Stephen Cook, 1.0)	(CMU, 1.0)	(Turning Award, 1.0)	0.9

(d) Rank Projection by eliminating other entities in *a* and *y*

x	Row
(Richar Karp, 1.0)	0.99
(Stephen Cook, 1.0)	0.9

示例查询的子查询的查询结果合并和排序

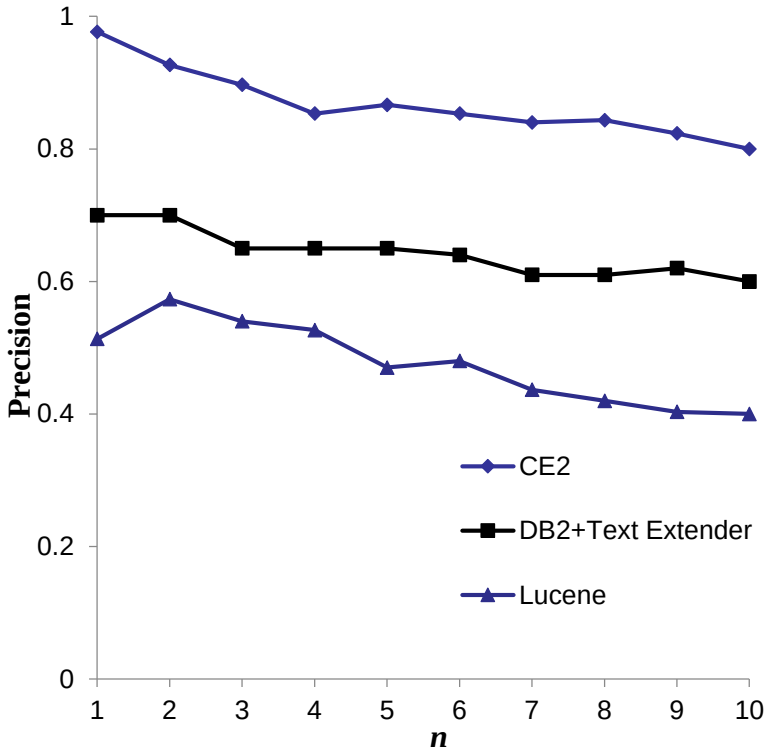
从DBpedia+Wikipedia中获取的混合查询数据集

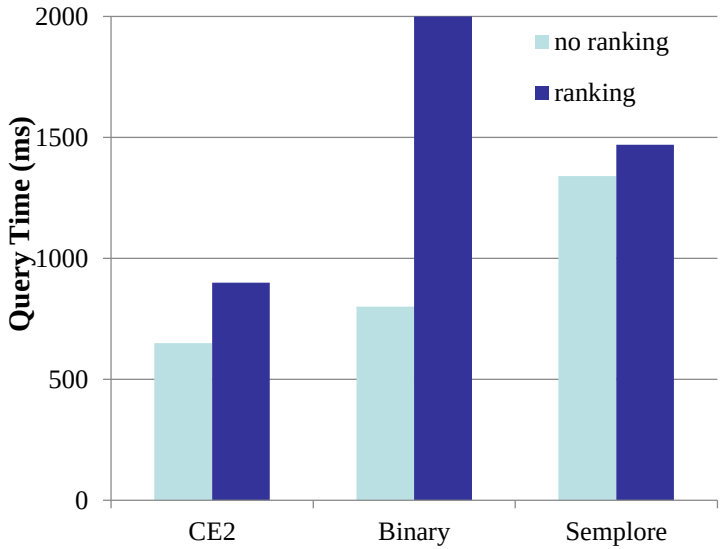
simple

complex

- QS1包含500个简单的关键字查询
- QS2包括1,242个具有类型约束的简单结构化查询
- QS3有287个具有一个关系的混合查询
- QS4根据10位用户提供的问题收集20个查询

初步结果





Effectiveness

Efficiency

- 有效性受益于数据结构和排名原则
- 效率来自利用DB&IR和综合排名支持

原生混合搜索系统 – 挑战

- 可扩展的混合存储模型
- 内置 (in-built)混合join处理
  - Top-k混合join
  - 模糊混合join
- 混合排序模型
  - IR排序
  - DB排序
  - IR和DB集成

结论

- 下一代的网络搜索是混合搜索，将内容单元提供给复杂的信息需求！
- 需要关注IR，DB和语义Web
- 一些初步的想法
- DB和IR的轻量级集成
- 原生混合搜索模型
  - 从头构建混合搜索
  - 定制索引和存储
  - 在匹配中紧密嵌入排序，定制查询评估

大纲

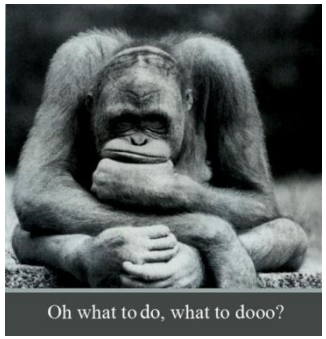
- 语义搜索简介
- 语义数据搜索
- 混合搜索
- **语义搜索的交互范式**
- 实践展示：使用Elasticsearch实现简单语义数据检索

可用性

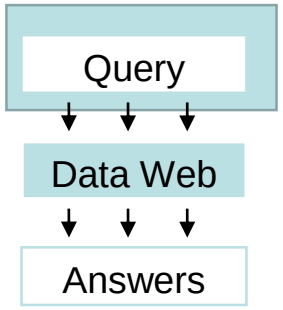
- 高效搜索数据Web需要用户知道
  - 网络数据源
  - 数据的模式
  - 包含的数据
  - 支持的结构化查询语言

```
PREFIX rdf: <http://www.w3.org/1
PREFIX foaf: <http://xmlns.com/fo

SELECT DISTINCT ?name
WHERE {
   ?x rdf:type foaf:Person .
   ?x foaf:name ?name
```



数据网络的有效利用需要一个简单的搜索范例，允许**外行用户以直观透明的方式查询和浏览数据网页！**



交互范式

- 自然语言接口
- 基于表单的查询接口
- 基于可视化的查询接口
- 基于关键词的查询接口
- 混合的查询接口
  - 结合自然语言，关键词，表单，分面(facets)和形式化查询
- 查询，数据和结果的可视化

动机 - 解读复杂的信息需求

- 语义搜索支持表现形式丰富的信息需求
- 如何捕捉用户的信息需求？
  - 多样化的查询，使用不同的语法 vs.
  - 有限但直观的查询
  - **表达能力至关重要!**
  - 支持用户以直观的方式指定信息需求也至关重要！
- 如何以一种直观的方式支持复杂的信息需求？

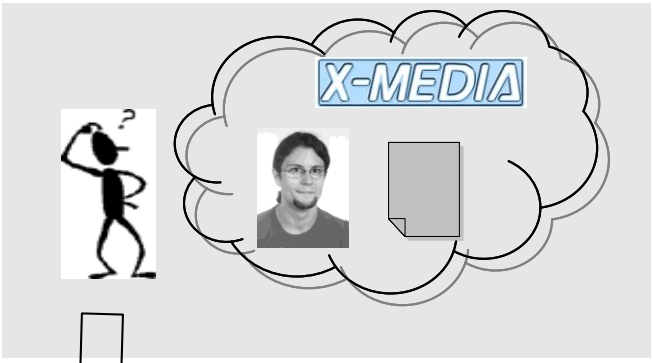
解读复杂信息需求-最佳方法

- 自然语言问题的翻译
  - 用户是否总是要指定精确的问题?
  - 模糊的信息需求?
- 将关键词翻译成结构化查询 (如XML、SQL等)
  - 如何使用复杂的结构
- 将关键词转化为基于本体的查询
  - 基于模板的机制缺乏灵活性
  - 具有两个以上关键词的查询需要大量的模板

一种基于本体的查询解释的通用方法

- 步骤
  - 将问题元素映射到本体元素
  - 探索本体元素找到连接
  - 从连接到处系统查询
- 假设
  - Ontology-Mental Correspondence
  - 定位信息需求

USER MENTAL MODEL



Query Specification

USER QUERY

„Philipp Cimiano X-Media publications“

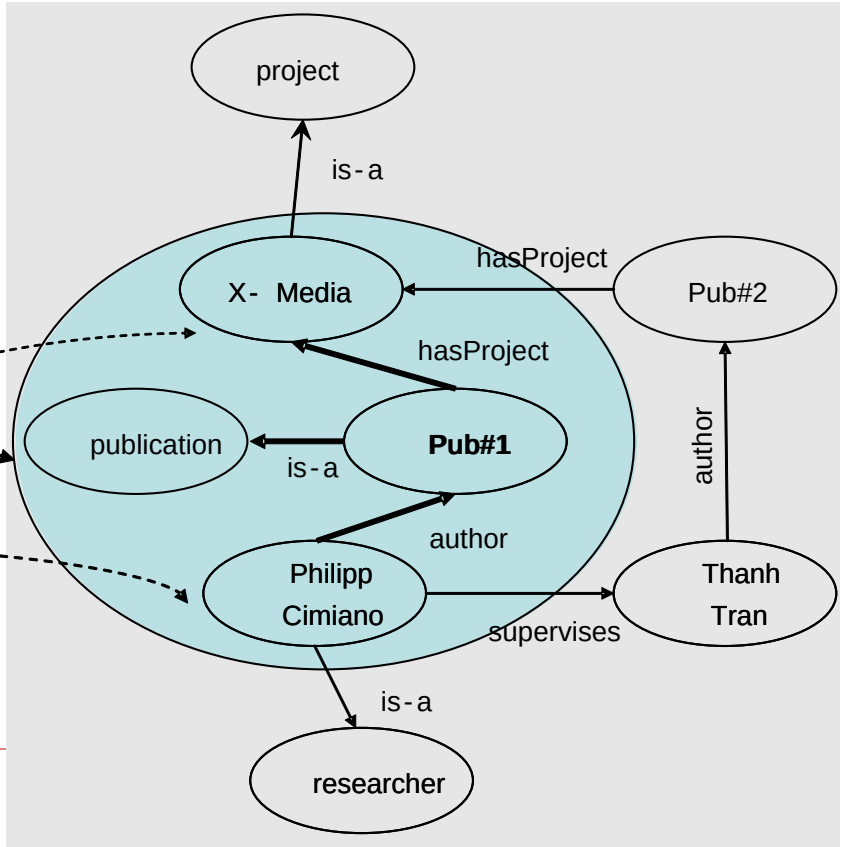
Query Interpretation

SYSTEM QUERY

?z <z , Philipp Cimiano>: author
<z , X-Media>: hasProject
<z , publication>: type

Query Processing

RESOURCE MODEL

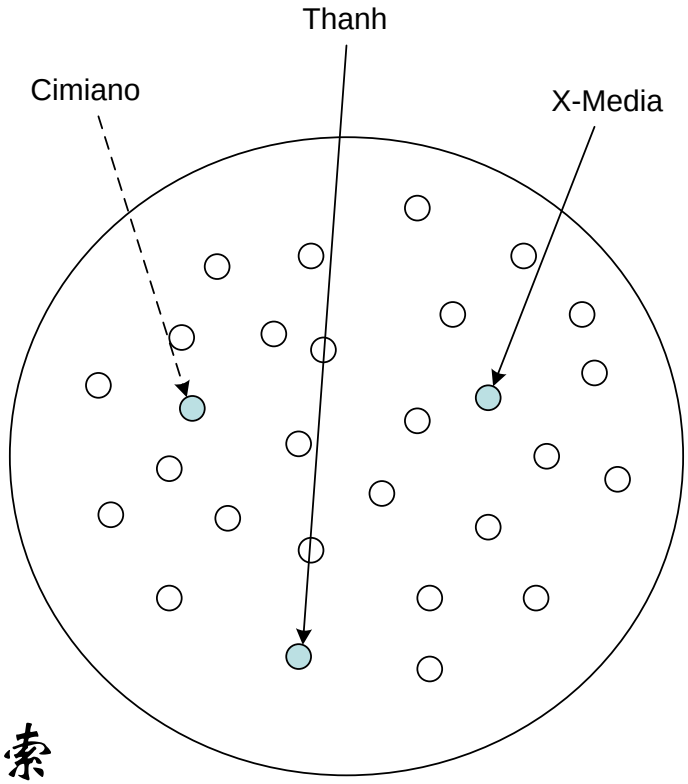


将关键词解释为合取查询

- 通用方法的一个实例
- 用户提问一组关键词
- 系统查询
  - Conjunction of terms的形式为 $x:C \quad \langle x,y\rangle:R$
  - 其中，C是一个concept，R是一个role，x和y是从一个变量名集合中选择的变量，或者从一个individuals集合中选择的individuals
- 系统资源模型
  - OWL DL知识库
  - 本体实体处于individuals集合，数据值，概念，数据范围，对象属性和数据属性的不相交的并集中
  - 实体之间的连接被术语公理和断言获取(概念，属性成员)

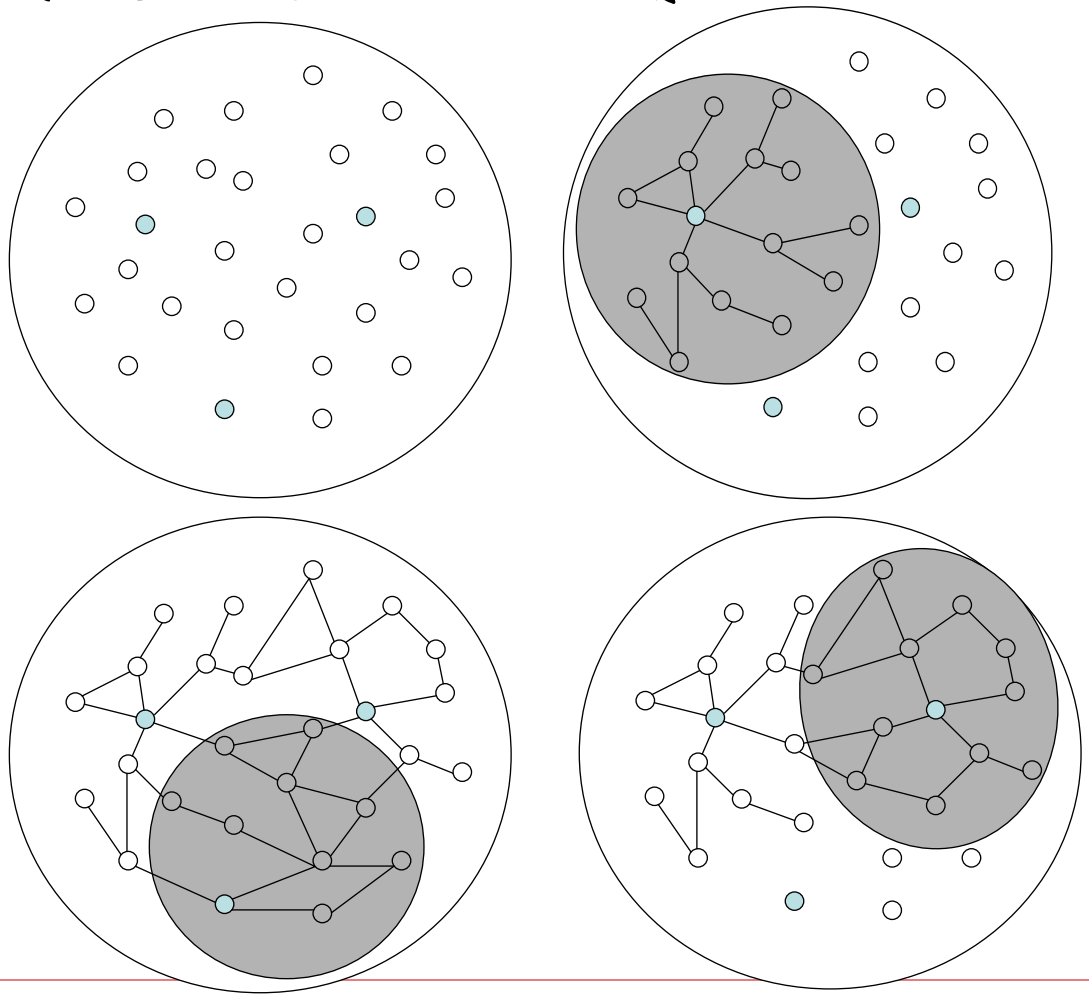
步骤1- 将关键词映射为本体实体

- 匹配
  - 关键词对应本体实体
- “健壮的”匹配函数
  - 句法的变体
  - 拼写的变体
- 实现了的匹配函数
  - 本体元素索引
  - 利用关键词对索引进行模糊搜索
  - 按照句法相似性返回本体实体



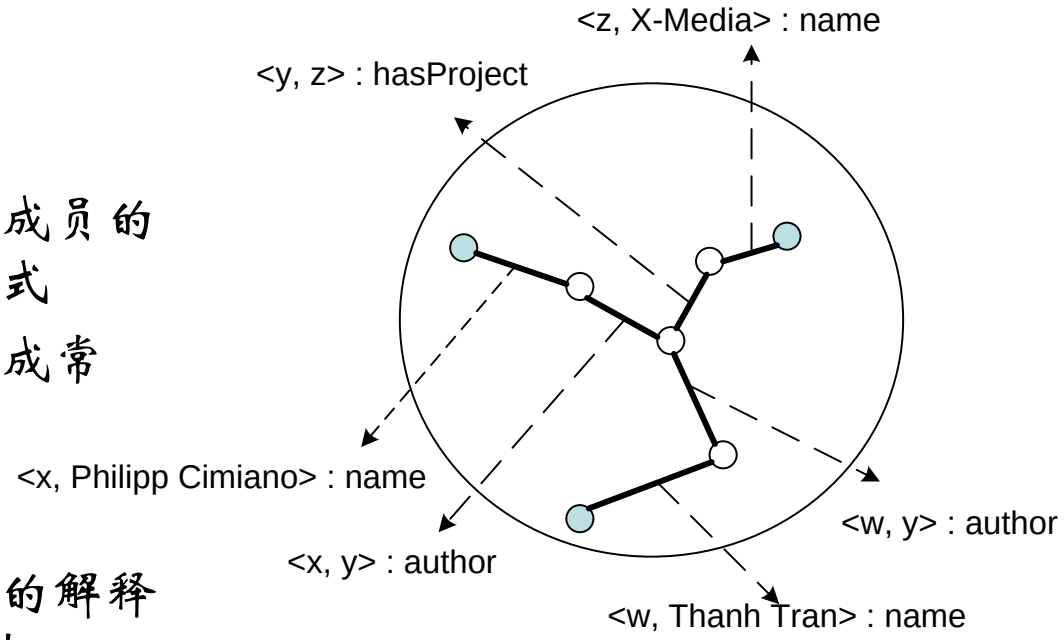
步骤2 - 发掘本体实体间的连接

- 基于元素递归遍历的KB探索



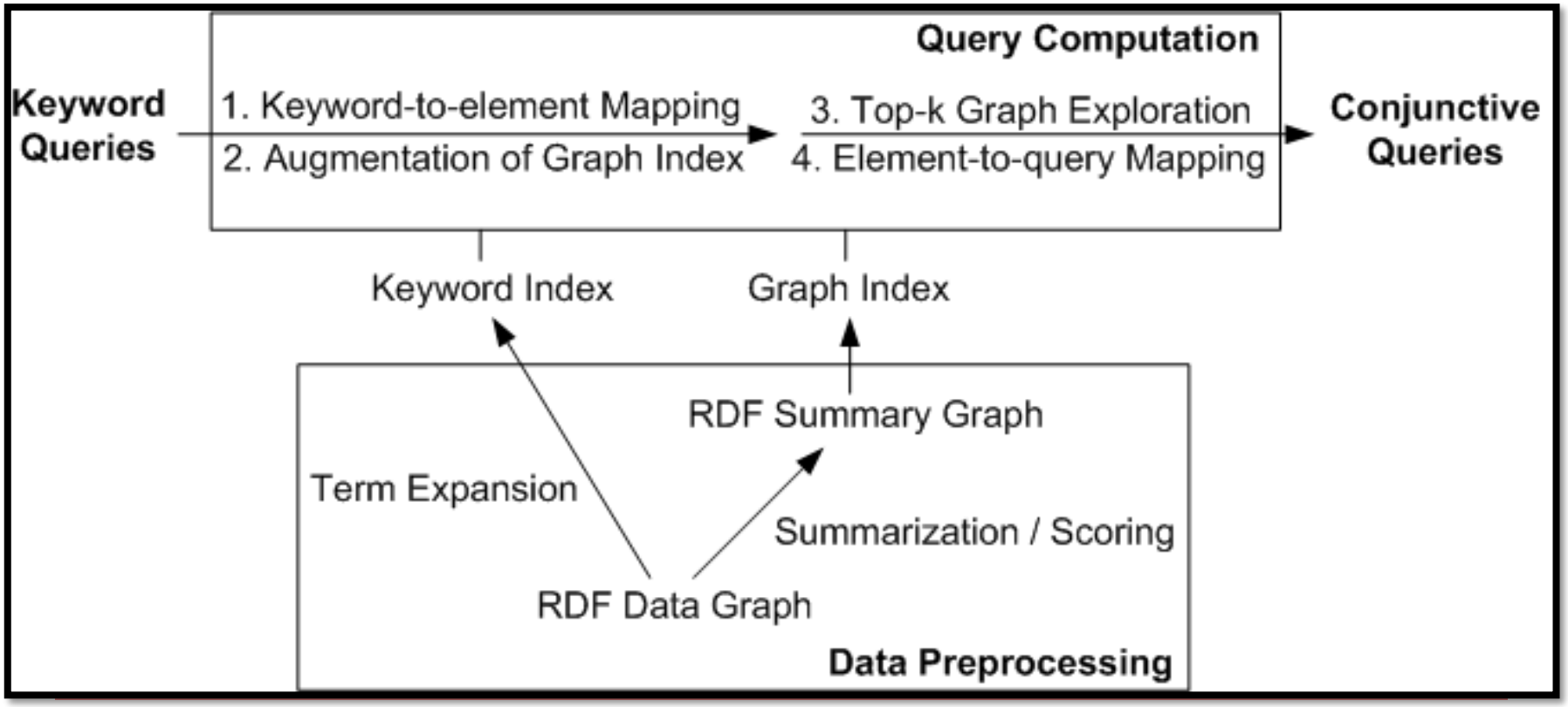
步骤3 - 从连接中导出DL合取查询

- 计算可能的子图
- 将连接映射到查询
  - 概念成员的连接和属性成员的连接映射到相应的表达式
  - 匹配查询元素的顶点变成常量，否则变量
- 对查询排序
  - 路径的长度越小，相应的解释(局部性假设)可能性越大
  - 连接图的最长路径的长度



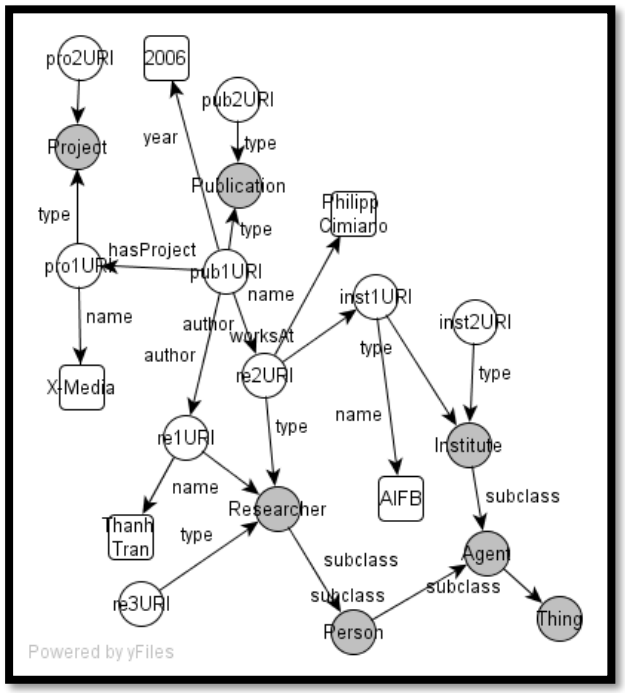
Top-k关键词查询 - 工作流程

- 线下：汇总，评分，术语扩展
- 线上：查询计算，查询处理

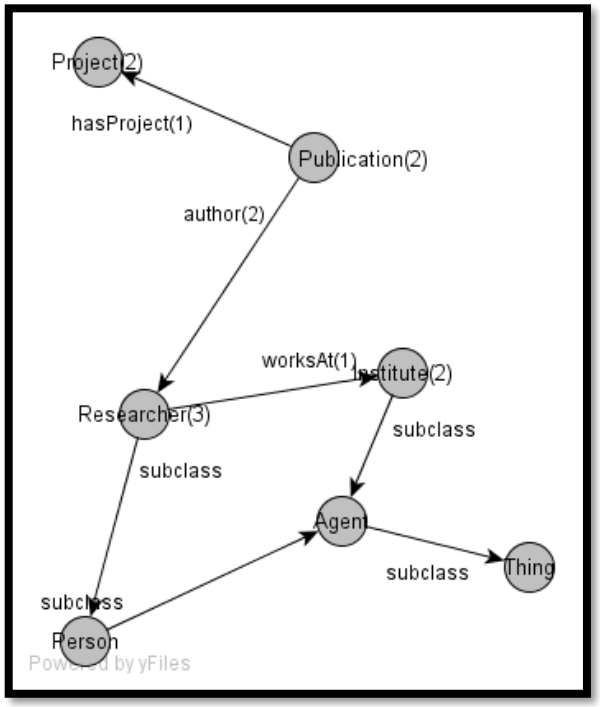


摘要图生成(Graph Summarization)

- 目标：保留足够的信息来计算查询所需的元素和结构，同时减少探索空间
- 摘要图(结构索引)捕获实体类之间的关系，从而保留原始数据图的结构信息



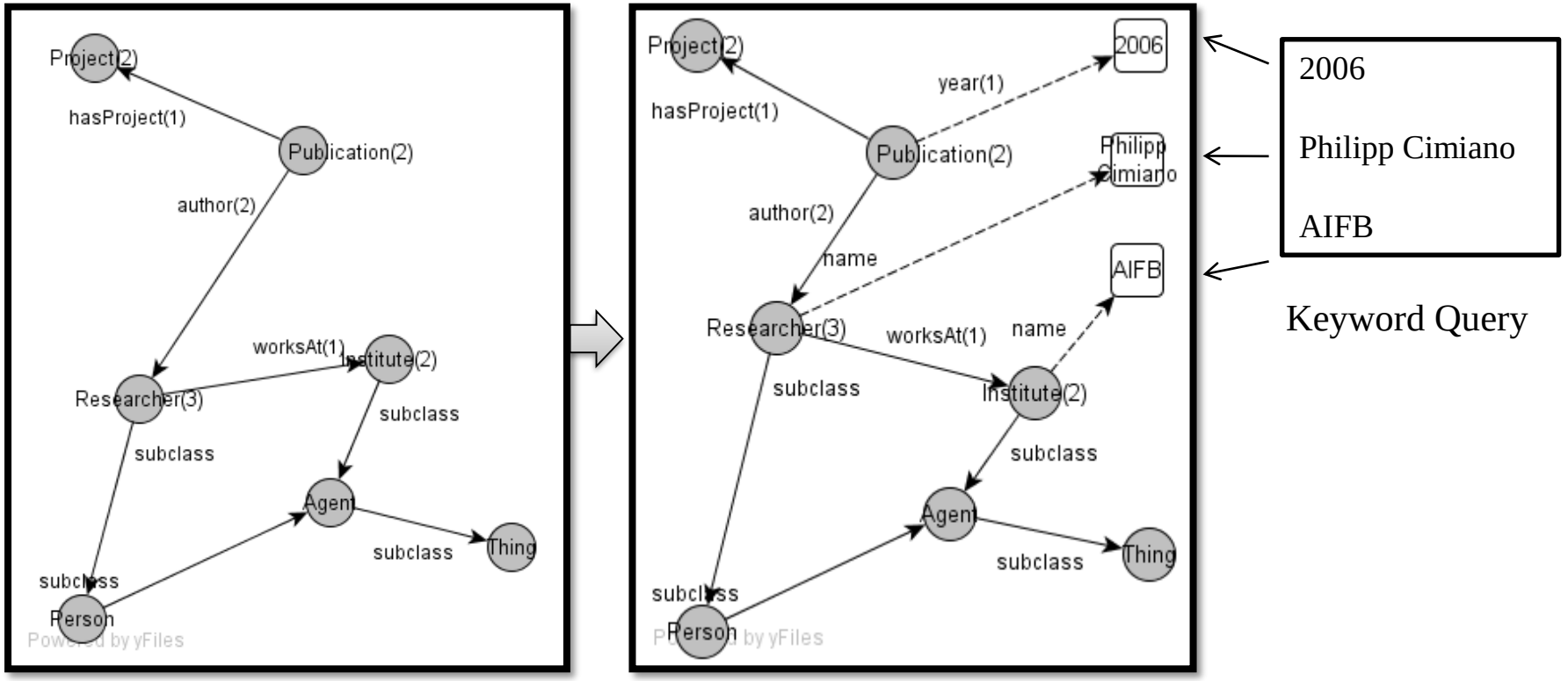
示例RDF图



摘要图

关键词映射和摘要图扩充

- 摘要图捕获查询结构的信息
- 在线扩充从关键词映射中获得的元素和分数
- 扩充的摘要图图包含查询元素的更多信息

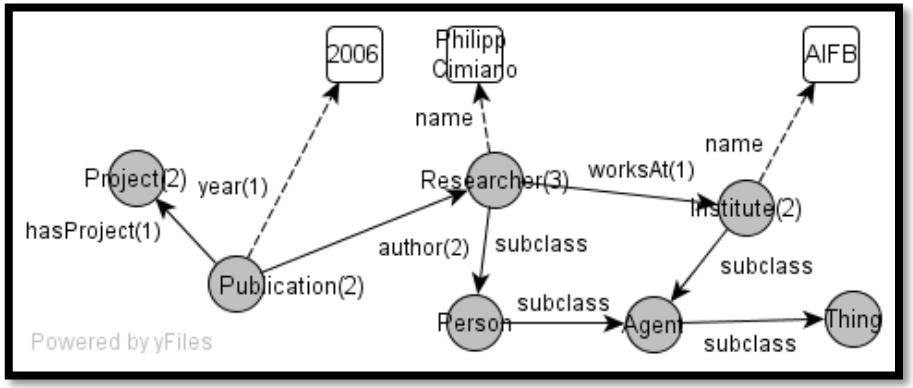


摘要图

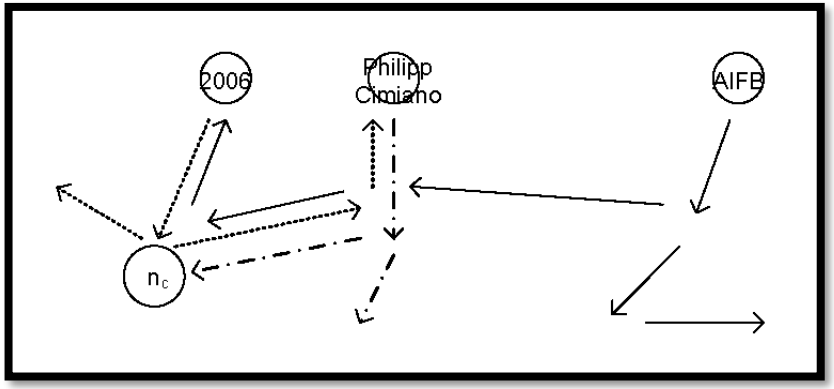
通过扩充的摘要图

Top-k图探索

- 从关键词元素N_k开始，对图进行**代价导向(cost-directed)的探索**
- 从N_k开始探索**所有可能的不同路径**
- 在每一步中，**从成本最低的队列中取cursor（“路径”）**进行探索
- 当找到**连接元素nc**时，
  - 从n_k到n_c的路径被合并以构建查询图
  - 调用Top-k将查询图添加到候选列表中
- 当候选列表的最高代价(排名为k的查询图的代价)被发现低于可用队列中尚待探索的路径所能达到的最低可能代价时，Top-k终止



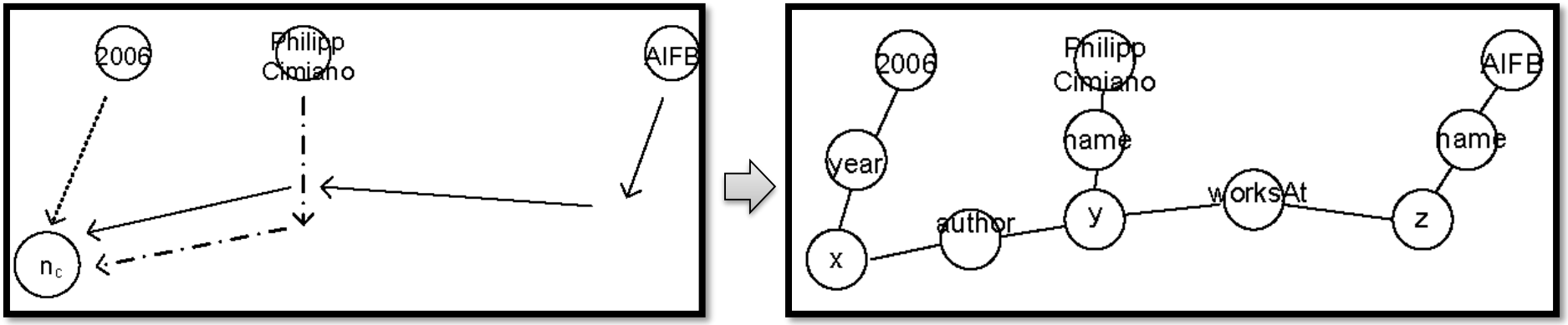
Augmented Summary Graph



Explored Paths

将查询图映射到合取查询

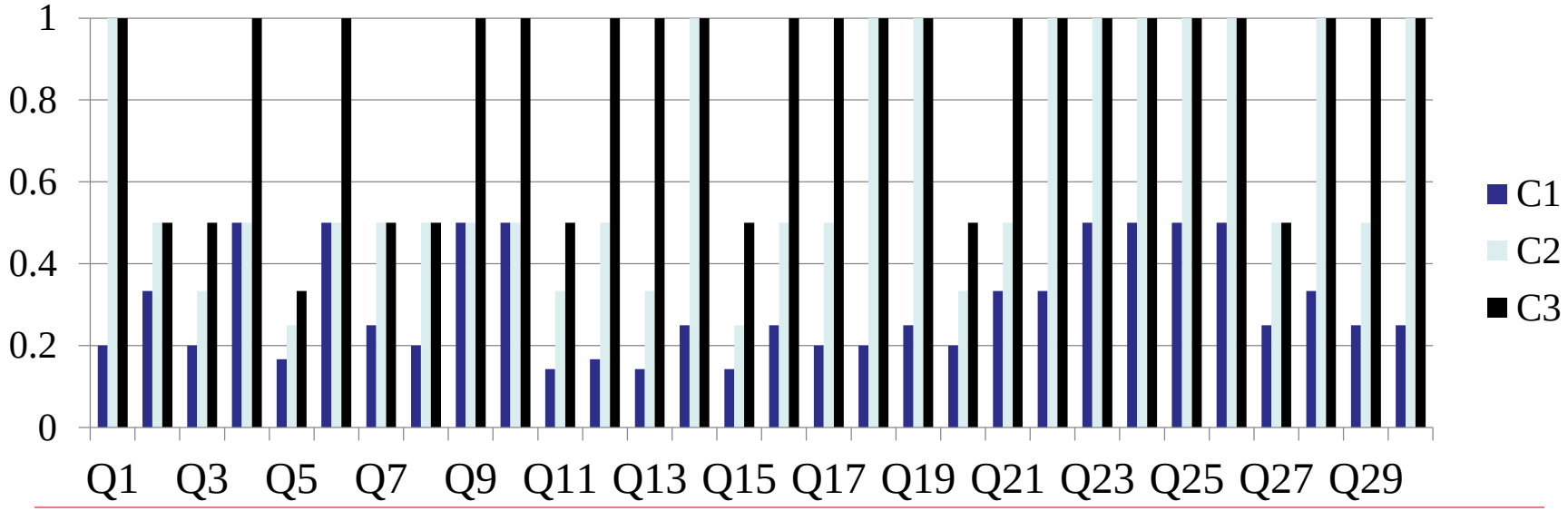
- 通过充分利用映射规则获得合取查询
  - 每个value vertex v_{vertex} → 一个常量
  - 每个class vertex c_{vertex} → 一个独特的变量
  - 每个A-edge $e(c_{vertex}, v_{vertex})$ → 一个查询谓词 $e[var(c_{vertex}), term(v_{vertex})]$
  - 每个 R-edge $e(c_{vertex1}, c_{vertex2})$ → 一个查询谓词 $e[var(c_{vertex1}), var(c_{vertex2})]$
- 将所有查询变量视为不同的
- 可以为用户提供特定的机制来选择可区分的变量
- 由用户选择的查询最终被翻译为查询引擎支持的查询形式(SPARQL)以检索查询答案



Query Graph　　Conjunctive Query

评估 – 效果

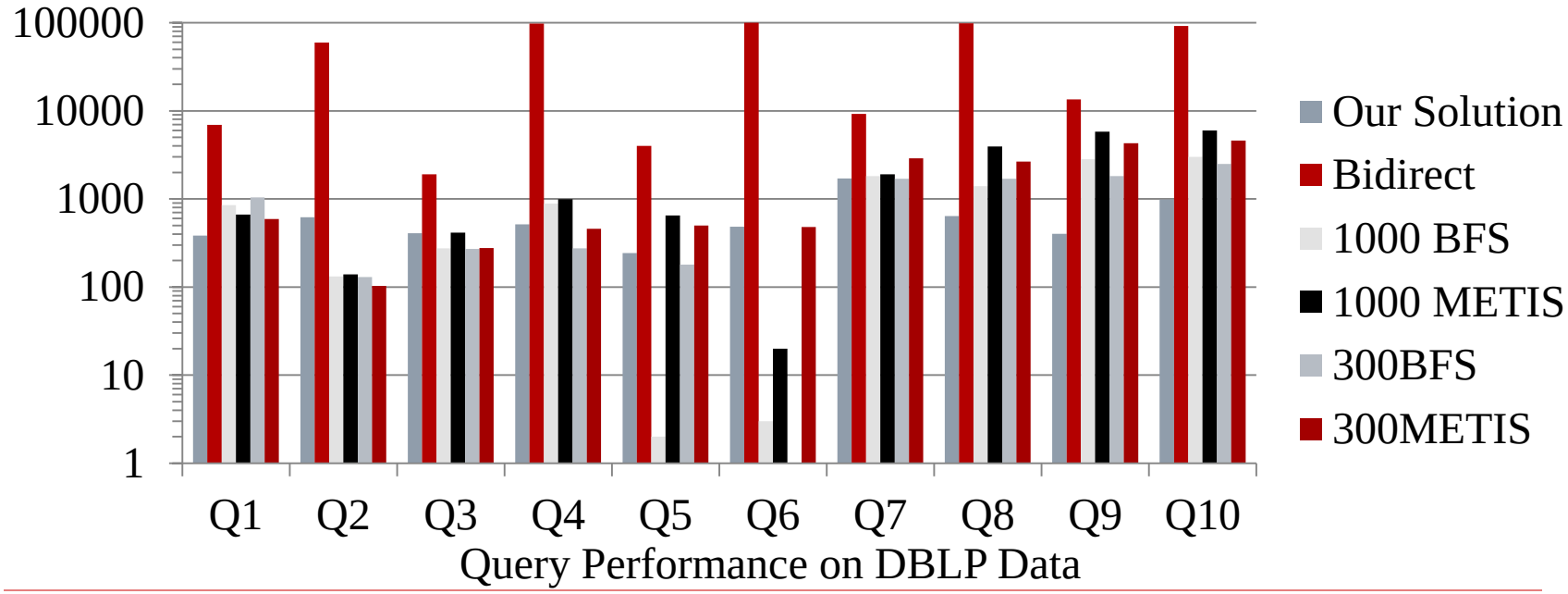
- 12位用户在DBLP上提供30个关键词查询，以及自然语言信息需求描述
- **Reciprocal Rank = 1/r**，其中r是正确查询的排序
- **如果查询符合信息需求**，则该查询是正确的
- 信息需求可以在大多数情况下被解释，特别是当**路径长度，匹配分数以及图表元素的流行性**被并入评分函数(C3)时



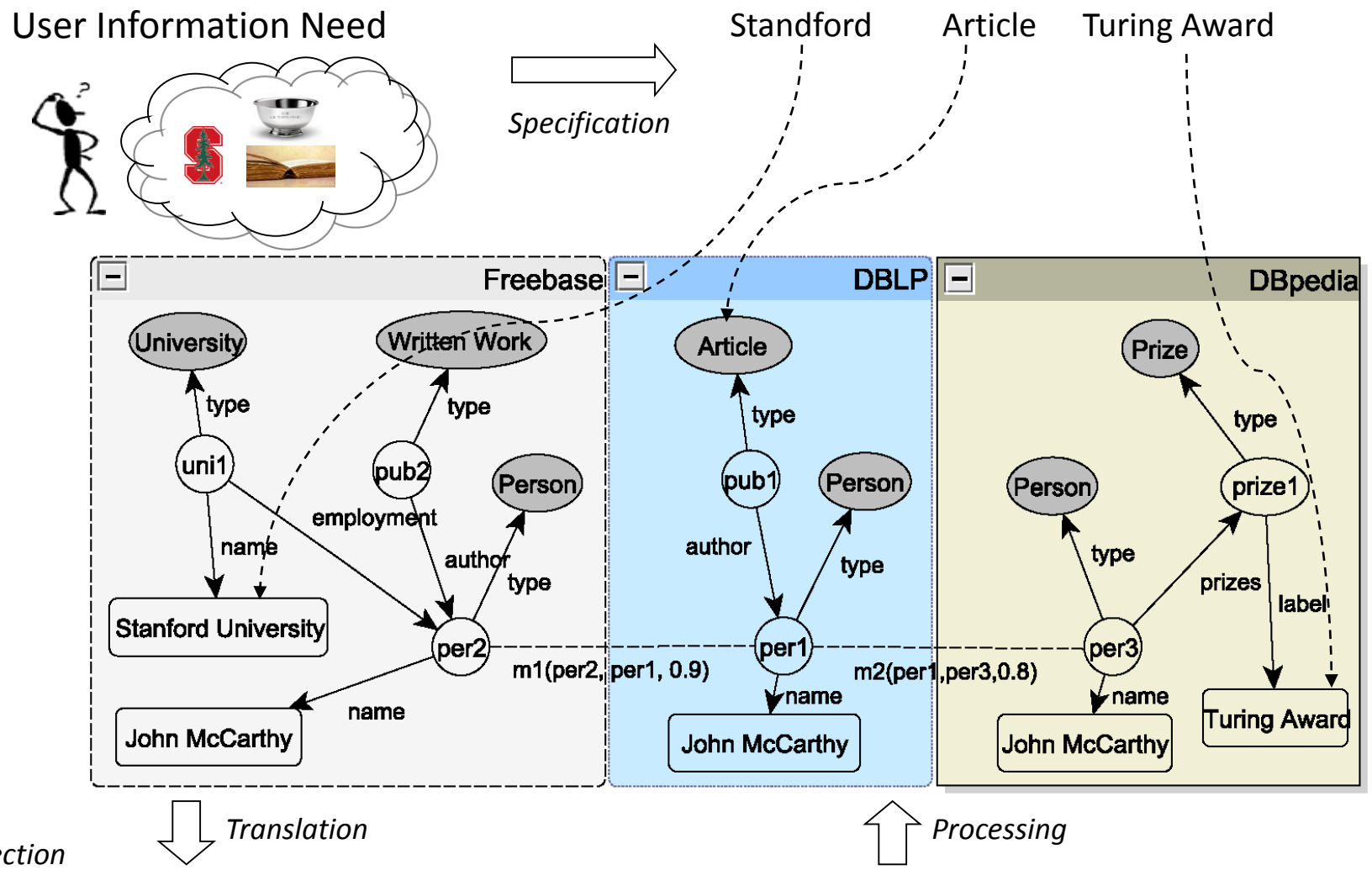
MRRs of different Scoring Functions on DBLP

评估 – 性能

- 与基于图索引的双向搜索和搜索的比较(1000 BFS，1000 METIS，300 BFS，300 METIS)
- 测量查询计算的时间+处理几个查询的时间，直到找到10个答案
- 比双向搜索更胜一筹，至少一个数量级
- 与基于索引的方法相比，表现相当好

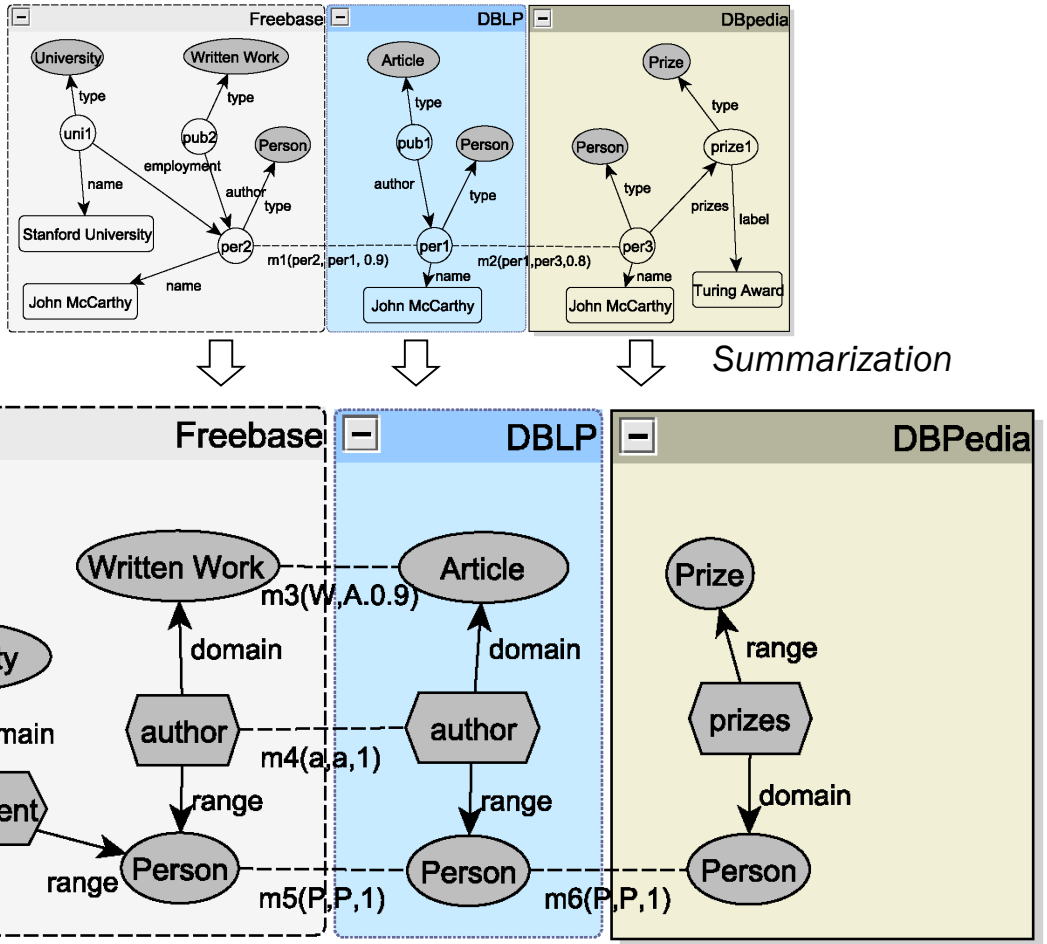


查询解析 – 理解用户的需求



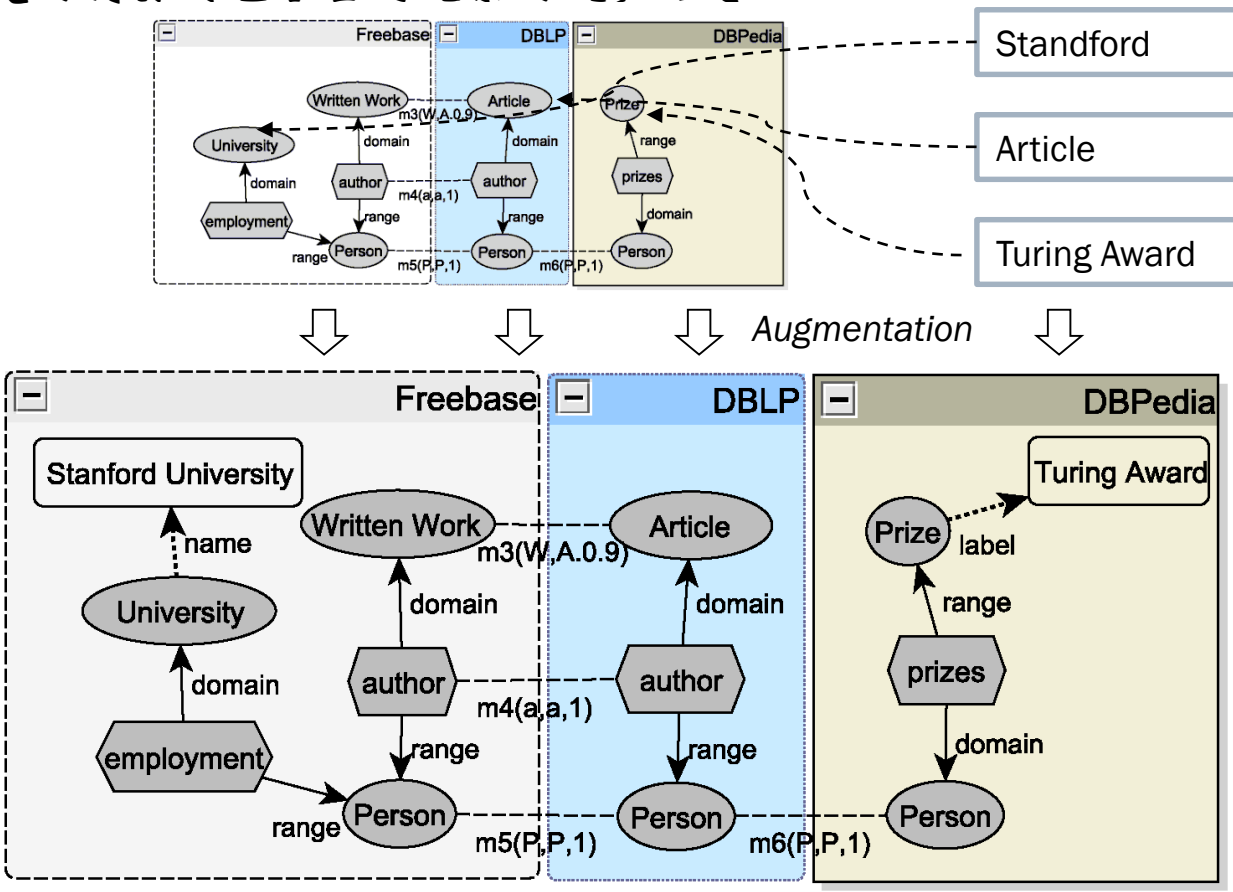
Web数据图摘要生成

- 目标：保留足够的信息来计算查询的元素和结构，同时减少搜索空间
- 模式图捕获实体类之间的关系，从而保留原始数据图的结构信息



关键词映射和图扩充

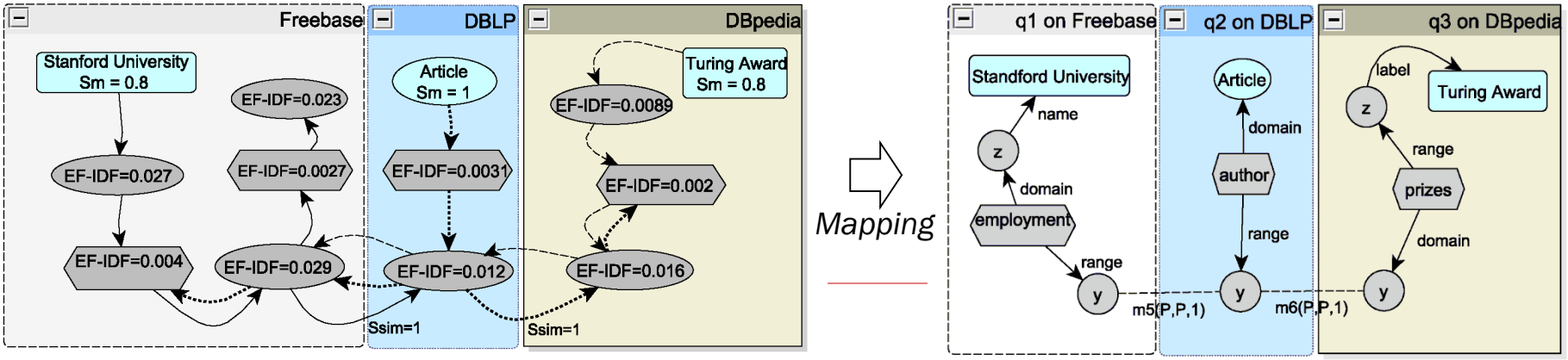
- 摘要图(模式图)捕获查询结构的信息
- 在线扩充从关键词映射中获得的元素和分数
- 扩充的摘要图包含查询元素的更多信息



Augmented Summary Graph

Top-k图计算

- 从关键词元素N_k开始，对图进行代价导向(cost-directed)的探索
- 探索从N_k开始的所有可能的不同路径
- 在每一步中，从代价最低的队列中取cursor（“路径”）进行探索
- 当找到连接元素N_c时，
  - 从N_k到N_c的路径被合并以构建查询图
  - 调用Top-k将查询图添加到候选列表中
- 当候选列表的最高成本(排名为k的查询图的成本)被发现低于可用队列中尚待探索的路径所能达到的最低可能成本时，Top-k终止
- 通过映射规则获得合取查询

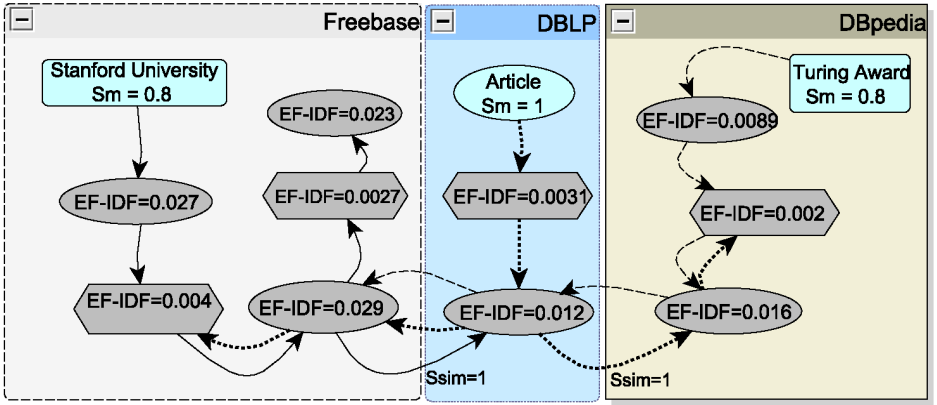


打分排序

关键词排序的打分

- 匹配度分数
- Popularity EF-IDF(n)
- 长度

$C_{g_q} = \sum_{p_i \text{ in } P} (\sum_{n \text{ in } p_i} C_n)$



- 在数据Web上的关键词搜索
  - 匹配度分数$S_m(n)$, popularity EF-IDF(n) 和长度
  - + **mapping score $S_{sim}(n)$**
  - + **data source coverage(g_{D_i})**

$C_{g_q} = \sum_{p_i \text{ in } P} \sum_{n \text{ in } p_i} 1 / (S_n * \text{coverage}(g_{D_i}))$ where

$S_n = S_{sim}(n)$ if n is a mapping element

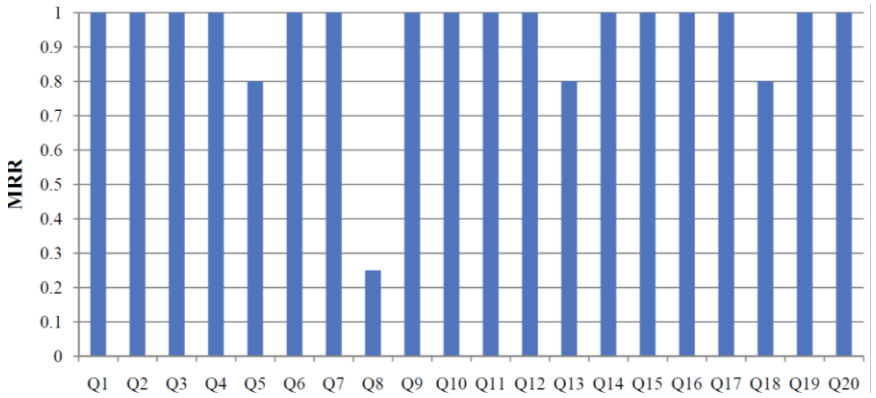
$= S_m(n)$ if n is a keyword element

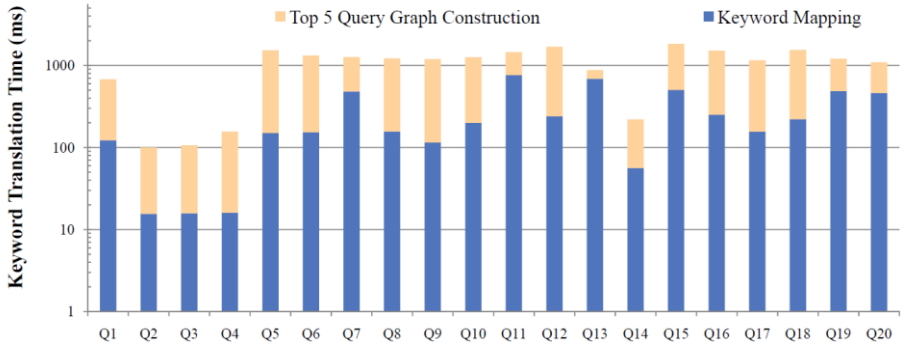
$=$ EF-IDF(n) if n is a data graph element

$=$ EF-IDF(n) * $S_m(n)$ if n is a data graph element and a keyword element

评估

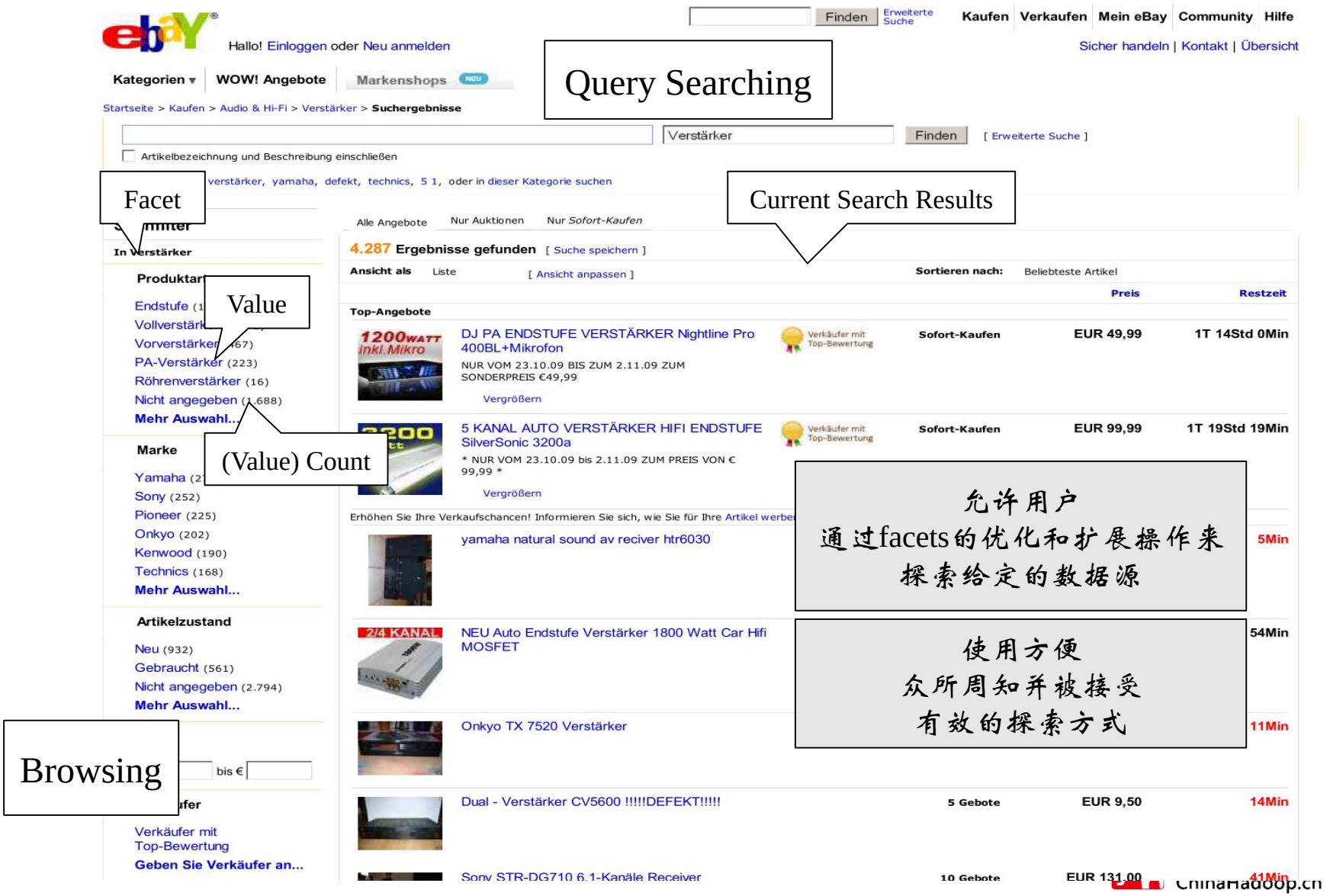
#	Keywords	Data Sources
Q1	project, ISWC, person	SWRC
Q2	Studer, publication	SWRC
Q3	undergraduate, topic	SWRC
Q4	Rudi, proceedings	SWRC
Q5	track, 323	Freebase
Q6	Pinocchio, film	Freebase
Q7	company, owner, "shopping center"	Freebase
Q8	restaurant, Berlin	Freebase
Q9	"the day after tomorrow", director	Freebase
Q10	film, Titanic	Freebase
Q11	ISWC2008,Studer, topic	SwetoDBLP, SW.org, SWRC
Q12	person, Shanghai, town	Freebase, USCensus
Q13	Ronny Siebes, InProceedings	SwetoDBLP, SWRC
Q14	tom, iswc2008, proceedings	SwetoDBLP, SW.org, SWRC
Q15	lake, citytown, wood	Freebase, USCensus
Q16	person, town, village	Freebase, USCensus
Q17	restaurant, german	Freebase, USCensus
Q18	album, town, mountain	Freebase, USCensus
Q19	Frank, publications	SwetoDBLP, SWRC
Q20	Markus, report	SwetoDBLP, SWRC





- 大部分都正确排序(即排在第一)
- 汇总图通常足够小，可以在内存中进行快速查询解析

分面搜索系统



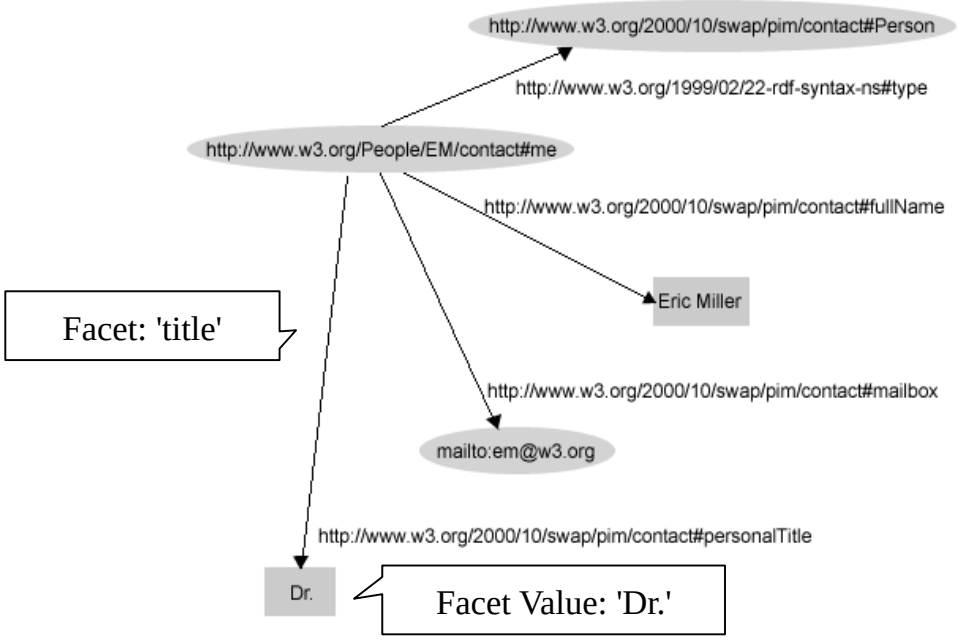
分面搜索系统

- 几乎没有预先定义的分面
  - 不支持动态分面
  - 不支持动态的值聚类
- 处理大型的，集成的，未知的数据集预定义的分面是不可能的！
- 只有少数最佳的分面和值是可见的或可访问的，基于计数的简单分面无效

使用语义数据进行分面搜索

- 如何在语义环境中定义分面？
  - 根据RDF-Triples定义分面和值
    - 分面可以被看作是一个在当前结果集中的资源的属性
    - 这些属性的object是分面的值

Facets & facet values are constructed from semantic model of data.



Dataplorer

A Scalable Search Engine for the Data Web

Query Tree

action | Search | New Query | Sample Queries

"action"

result-driven facet generation on demand

Navigator

Categories

- TOP Category(25244)
- Living people(2657)
- English language films(424)
- American films(339)
- Year of birth missing(334)
- Action films(324)
- Windows games(244)
- PlayStation games(203)
- American film actors(177)
- American military personnel of World

More

Related Information

- genre(1447)
- starring(1246)
- director(1147)
- publisher(1126)
- writer(919)
- producer(909)
- language(905)
- platforms(893)
- developer(860)
- modes(798)

More

Results 1 - 10 of 25361 (1.531 seconds)

Result display panel showing results with their snippets in multiple pages

Refinement and navigation panel providing concept and relation facets for interactive browsing

Action Boy

Select this item

"**action** boy, now known as kid **action**, is the sidekick of the well known captain **action**."@en .

http://dataplorer.apexlab.org/Ac

Action Force

Select this item

"**action** force was a range of cor d on **action** man

http://dataplorer.apexlab.org/Action_Force

Action This Day

"**action** this day refers to:\n\n* **action** this day (horse), ameri

http://dataplorer.apexlab.org/Action_This_Day

Action at a distance

"**action** at a distance may refer to:\n* **action** at a distance (pl

http://dataplorer.apexlab.org/Action_at_a_distance

Fields of Action

Select this item

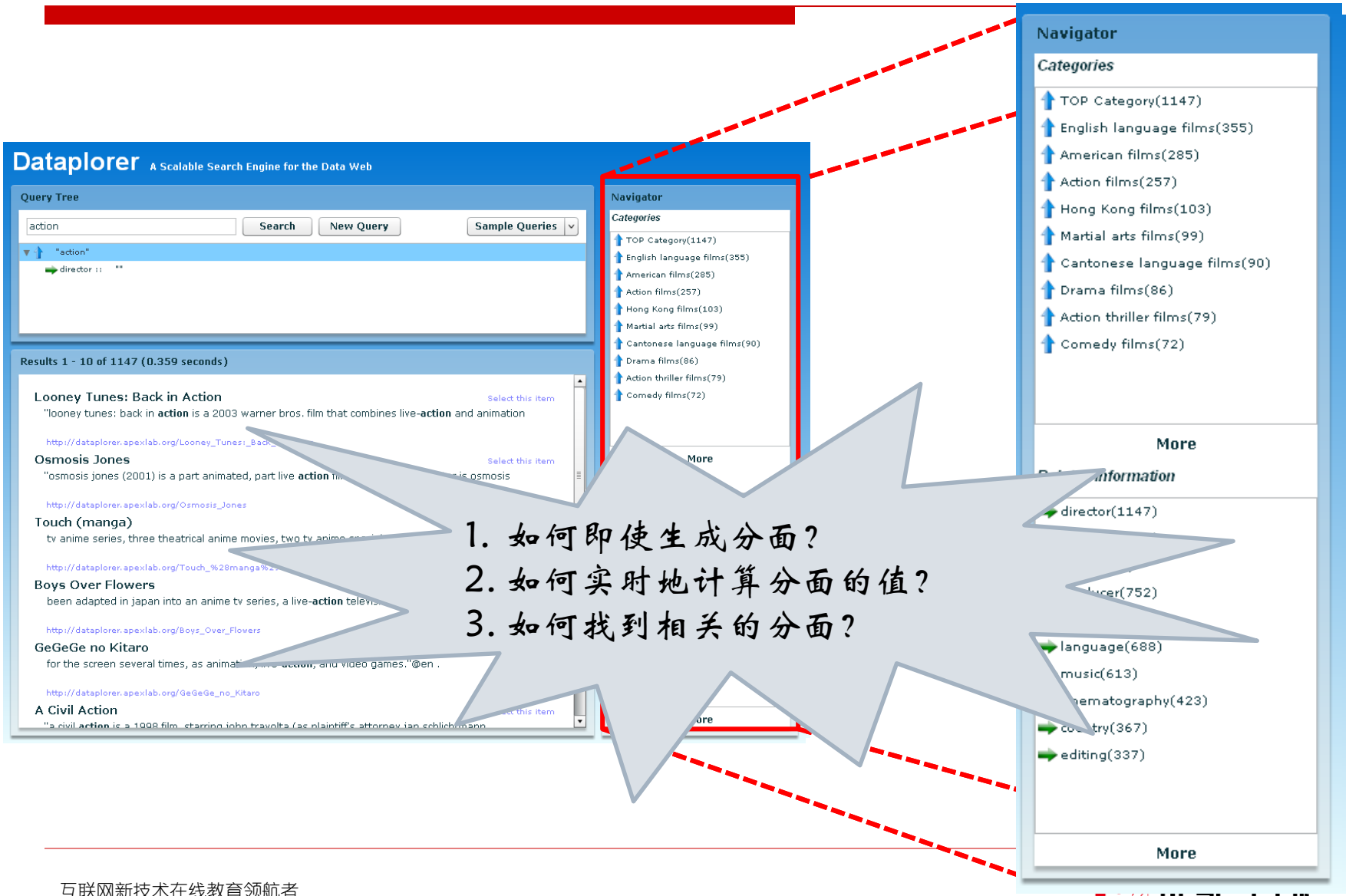
"fields of **action** is a two-player abstract strategy board game invented by sid sackson, which

http://dataplorer.apexlab.org/Fields_of_Action

Direct action#Nonviolent direct action

Select this item

分面浏览，简单还是困难？



Dataplorer

A Scalable Search Engine for the Data Web

Query Tree

action | Search | New Query | Sample Queries

"action"

director :: Hong Kong film directors ""

starring :: Chinese actors "martial"

Results 1 - 10 of 50 (0.266 seconds)

Heart of Dragon Select th

"heart of dragon (long de xin) is a 1985 hong kong **action** film directed by sammo hung, assist

http://dataplorer.apexlab.org/Heart_of_Dragon

Dragons Forever ct this item

"dragons forever (\u98db\u9f8d\u731b\u5c06; fei lung maang jeung) is a 19 ovie made

http://dataplorer.apexlab.org/Dragons_Forever

Wheels on Meals Select this item

"wheels on meals (ku\u00e0ic\u0101n ch\u0113) is a 19 hong kong **action** film directed by sammo

http://dataplorer.apexlab.org/Wheels_on_Meals

My Lucky Stars

kong **action** comedy, directed by sammo hung. it is the second film in the lucky stars series.\n\nthe

http://dataplorer.apexlab.org/My_Lucky_Stars

Eastern Condors Select this item

"eastern condors is a 1987 hong kong **action** film set in the aftermath of the vietnam war

http://dataplorer.apexlab.org/Eastern_Condors

Winners and Sinners Select this item

\") is a 1983 hong kong **action** comedy film directed and co-written by sammo hung. it was the first

D

D X_1 Type X_0

D X_1 Subj(Obj)_Of X_0

Navigator

Categories

- TOP Category(50)
- Cantonese language films(48)
- Hong Kong films(48)
- Martial arts films(35)
- Kung fu films(13)
- ilms(9)
- thriller films(7)
- 4)
- films(4)
- films(4)

More

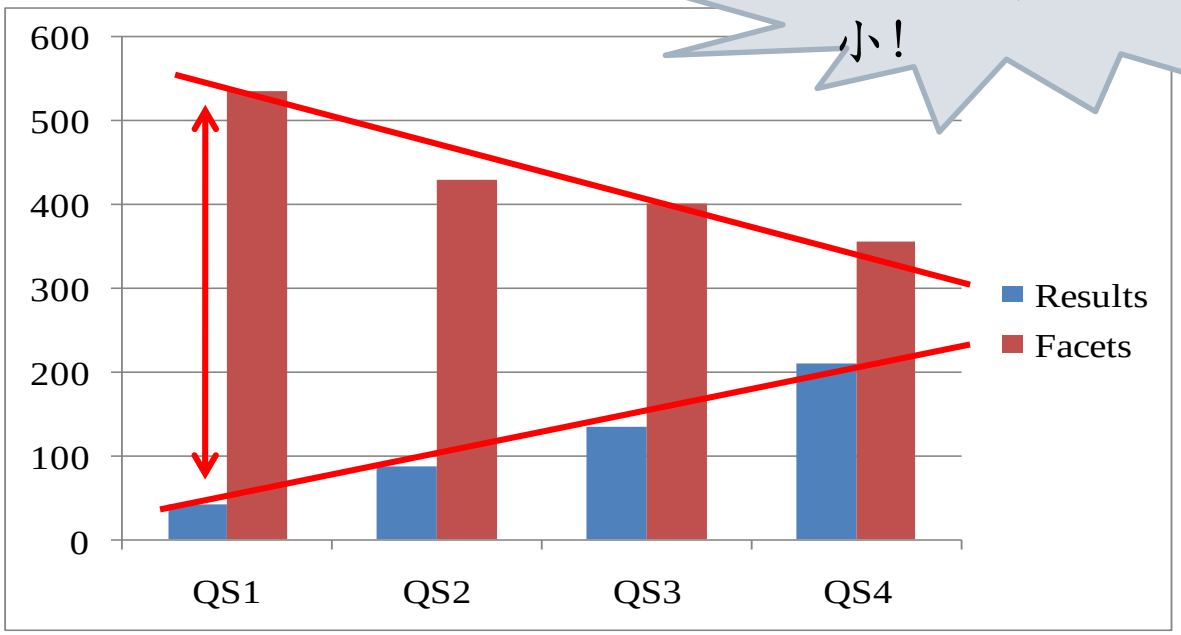
Related Information

- director(50)
- starring(50)
- language(48)
- writer(42)
- untry(41)
- cer(36)
- sic(21)
- cinematography(19)
- released(18)
- distributor(17)

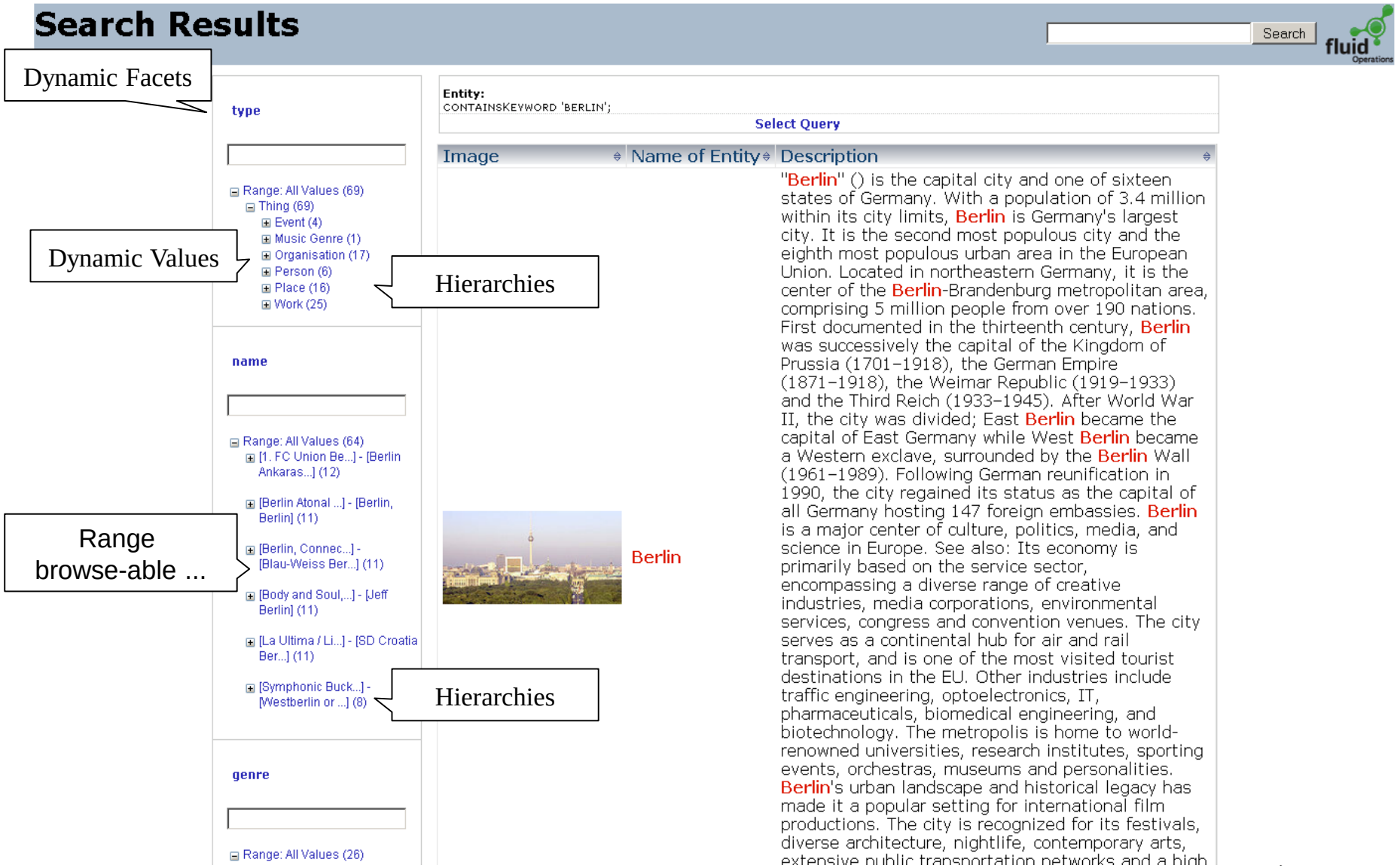
More

查询时间和相关的分面计算

Average Query Time (ms)	QS1	QS2	QS3	QS4
Results	41	88	134	210
Facets	535	429	401	356

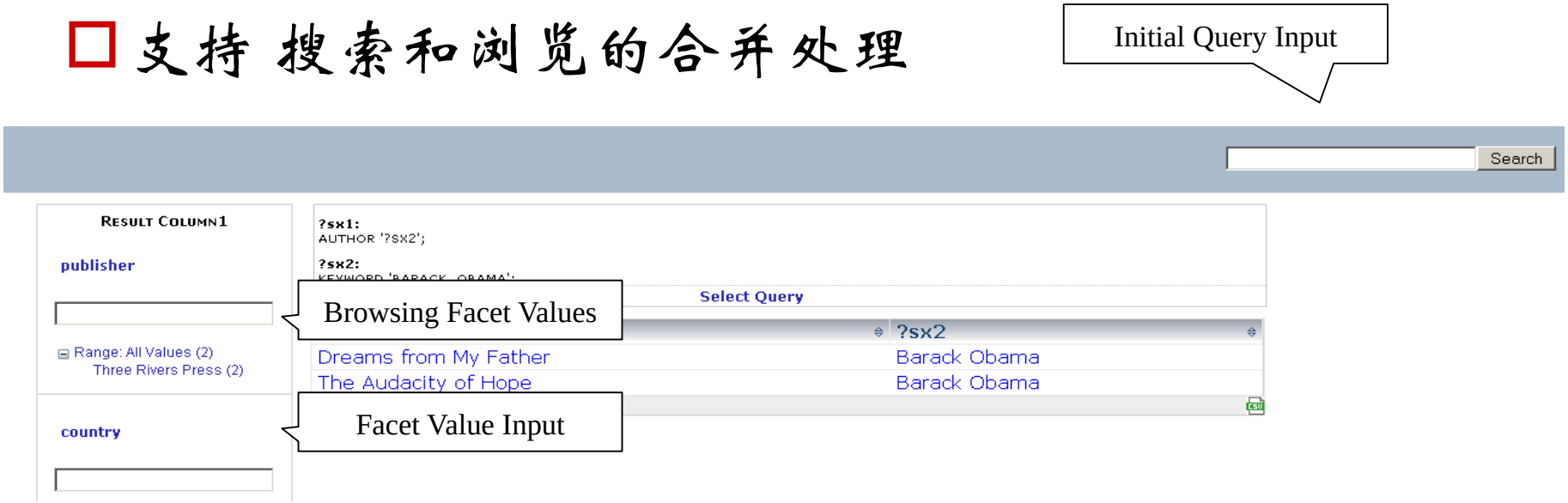


高级分面搜索特征


Search Results
Search
fluid Operations
Dynamic Facets
type
Range: All Values (69)
Thing (69)
Event (4)
Music Genre (1)
Organisation (17)
Person (6)
Place (16)
Work (25)
Dynamic Values
Hierarchies
name
Range: All Values (64)
[1. FC Union Be...] - [Berlin Ankaras...] (12)
[Berlin Atonal ...] - [Berlin, Berlin] (11)
[Berlin, Connec...] - [Blau-Weiss Ber...] (11)
[Body and Soul,...] - [Jeff Berlin] (11)
[La Ultima / Li...] - [SD Croatia Ber...] (11)
[Symphonic Buck...] - [Westberlin or ...] (8)
Range browse-able ...
Hierarchies
genre
Range: All Values (26)
Entity:
CONTAINSKEYWORD 'BERLIN';
Select Query
Image
Name of Entity
Description
Berlin
"Berlin" () is the capital city and one of sixteen states of Germany. With a population of 3.4 million within its city limits, Berlin is Germany's largest city. It is the second most populous city and the eighth most populous urban area in the European Union. Located in northeastern Germany, it is the center of the Berlin-Brandenburg metropolitan area, comprising 5 million people from over 190 nations. First documented in the thirteenth century, Berlin was successively the capital of the Kingdom of Prussia (1701–1918), the German Empire (1871–1918), the Weimar Republic (1919–1933) and the Third Reich (1933–1945). After World War II, the city was divided; East Berlin became the capital of East Germany while West Berlin became a Western exclave, surrounded by the Berlin Wall (1961–1989). Following German reunification in 1990, the city regained its status as the capital of all Germany hosting 147 foreign embassies. Berlin is a major center of culture, politics, media, and science in Europe. See also: Its economy is primarily based on the service sector, encompassing a diverse range of creative industries, media corporations, environmental services, congress and convention venues. The city serves as a continental hub for air and rail transport, and is one of the most visited tourist destinations in the EU. Other industries include traffic engineering, optoelectronics, IT, pharmaceuticals, biomedical engineering, and biotechnology. The metropolis is home to world-renowned universities, research institutes, sporting events, orchestras, museums and personalities. Berlin's urban landscape and historical legacy has made it a popular setting for international film productions. The city is recognized for its festivals, diverse architecture, nightlife, contemporary arts, extensive public transportation networks and a high

结合搜索和浏览

- 动机：用户的知识是不明确的
- 基本假设
  - 模糊知识将通过分面优化和扩展进入
  - 具体的知识将作为查询输入
- 支持 搜索和浏览的合并处理



动态的值分区

优点:

- 通过对值范围的分区支持有意义的浏览
- 支持模糊的用户知识

高级分面搜索特征

- 特定领域的方面，分面值和计数
  - 分面根据它们的起点分组
- 全面的浏览支持
  - 通过浏览可以达到每个分面的值;没有值被跳过
- 动态的分面和值的聚类
  - 对flat值聚类;对non-flat值使用类层次结构
  - 分面是基于当前的，outgoing属性
- 支持flat和non-flat的分面值

分面排序

- 可能有大量分面
  - 需要分面排序和分面隐藏
- 但是，目前的排序机制不支持我们的搜索过程
- 排序标准： 浏览能力
  - 分面应该以非常小的，相等的进度“引导”用户
  - 路径应该引向(单个)感兴趣的项
  - 在每个步骤：用户可以直观和明显(用最少的必需知识)的选择一个给定的聚类

结论

- 表达式(expressive)关键字查询
  - 基于本体的查询解析
  - Top-k关键字查询在汇总图上的解析
  - 使用映射信息扩展到多个数据源场景
- 动态分面计算w.r.t结果
  - 分面排序和值划分

语义搜索路线图

RDF triple store

Semantic search engine

Large scale semantic search engine

online

offline

Phase 1

Phase 2

Phase 3

参考文献

语义搜索(部分)

Journals

- Lightweight integration of IR and DB for scalable hybrid search with integrated ranking support, Journal of Web Semantics, 1st author，2011 9(4), IF=3.049, CCF B level
- Semplore: A scalable IR approach to search the Web of Data, Journal of Web Semantics, 1st author，2009 7(3), IF=3.049, CCF B level
- Hermes: Data Web search on a pay-as-you-go integration infrastructure, Journal of Web Semantics, 2nd author，2009 7(3), IF=3.049, CCF B level

Conferences

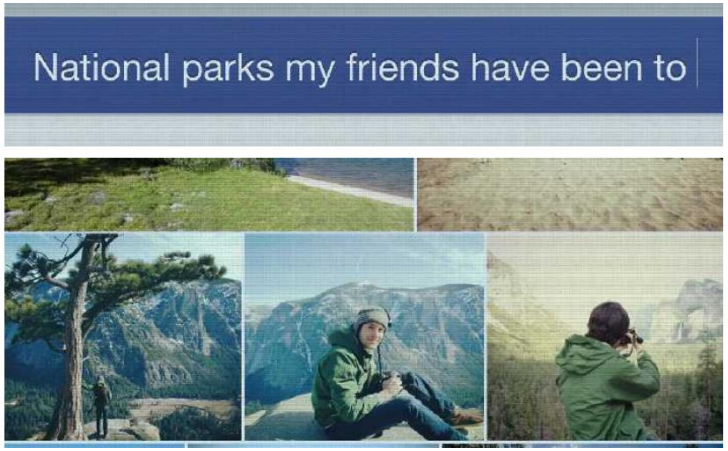
- Hermes: A Travel through Semantics on the Data Web, SIGMOD 2009, 1st author, CCF A level
- Dataplorer: A Scalable Search Engine for the Data Web, WWW 2009, 1st author，CCF B level
- CE2: Towards a Large Scale Hybrid Search Engine with Integrated Ranking Support, CIKM 2008, 1st author，CCF B level
- SearchWebDB: Searching the Billion Triples!, ISWC 2008, 1st author，CCF B level
- Q2Semantic: A Lightweight Keyword Interface to Semantic Search, ESWC 2008, 1st author, CCF C level
- Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data, ICDE 2009, 2nd author，CCF A level

大纲

- 语义搜索简介
- 语义数据搜索
- 混合搜索
- 语义搜索的交互范式
- 实践展示：使用Elasticsearch实现简单语义数据检索

相关应用 – Facebook Graph Search

- 用户可以直接使用自然语言搜索Facebook上公开的人，地点，照片等信息
- FGS能精确理解用户的自然语言
  - 例如，“附近朋友们常去的餐馆”，，“附近”会先定位用户的位置”,“朋友们”会检索用户的好友。通过将自然语言转化为特殊的查询语句，在图数据上查询。



FGS – 查询构建

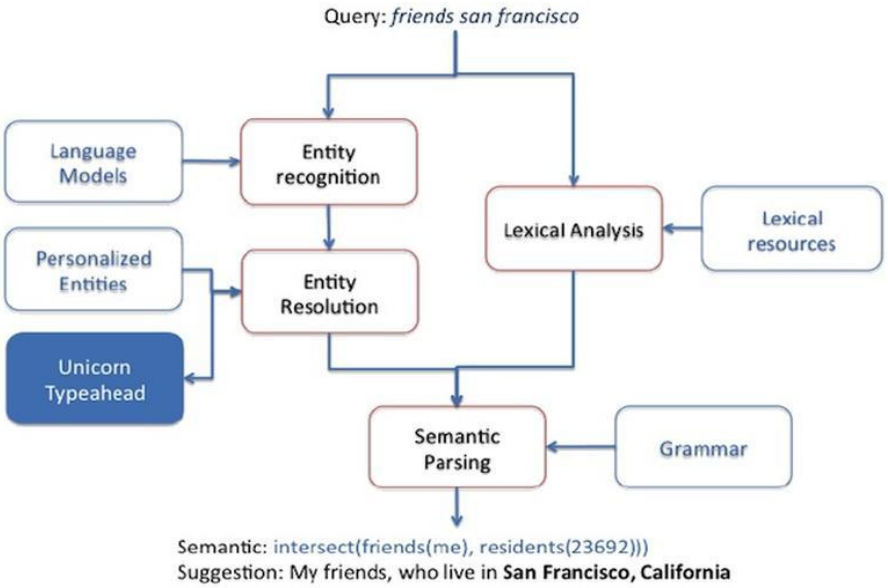
- FGS在处理用户查询时有三层查询语言
  - 用户输入的自然语言
  - 一种预先定义语法的语义语言
  - s-expression语法的语言，多个检索条件的组合，以便从倒排索引中检索出可能的结果

- 自然语言：friends san Francisco
- 语义语言：intersect(friend(me),residents(23692))
- S-expression:

  (and friend:767560056 city_to_user: 23692)

FGS – 查询构建

- 用户输入：friends san Francisco
  - 找到查询中出现的实体名字或者人名对应的对象
    - san Francisco -> 23692 (san Francisco的唯一id)
    - 如果有歧义的话,系统会根据一些准则确定，如拥有共同朋友最多的一个
  - 查询优化
    - 找到最匹配用户输入，且可构成有效查询的查询,显示给用户
    - 可按概率返回多个，让用户挑选
  - 用户点击了一个推荐的查询后，转化成内部的查询语言：
    - my friends who live in san Francisco -> intersect(friend(me), residents(23692)) -> (and friend:767560056 city_to_user: 23692)



实践展示

- 可以看到，Facebook Graph Search在生成查询的最后一步是将自然语言查询转化为多个查询条件的组合(s-expression)，而这些查询条件易于通过倒排索引检索
  - my friends who live in san Francisco ->
  - (and friend:767560056 city_to_user: 23692)
- 本实验：尝试使用Elasticsearch实现一个简单的语义数据检索引擎，可以完成FGS最后查询的基本功能。目前实现可以按照实体检索属性，按照多种属性条件的布尔组合检索实体等。实验使用的数据集为由三元组组成的人物属性数据集
- Demo: http://120.27.213.250:9000/

实践展示 - 功能

实体查询

- 用户输入实体名称，返回该实体的知识卡片

属性名称	属性值
birthPlace	上海
gender	男
children	姚沁蕾
职业	运动员 篮球运动员 其他 上海大鲨鱼队老板
birthDate	1980年9月12日
height	226
nationality	中国
subj	姚明
alumniOf	上海体育技术教育学院
民族	汉族

实践展示 - 功能

实体的属性查询

- 给定实体名和属性名，返回该属性对应的属性值

运动员 篮球运动员 其他 上海大鲨鱼队老板

多跳查询

- 可以输入多个属性，实现多跳查询

190

实践展示 - 功能

按照多种属性条件检索实体

- 属性条件之间：And，Or，Not
- 单个属性条件：=,>,<,>=,<=
- 例如：“职业：篮球运动员 or 职业：足球运动员 And Not 国籍:中国 And 身高>=180”

实体名称	查询链接
佩贾·斯托贾科维奇	佩贾·斯托贾科维奇
蒂姆·皮克特	蒂姆·皮克特
德里克·贝耶斯	德里克·贝耶斯
科里·马盖蒂	科里·马盖蒂
谢伦·科林斯	谢伦·科林斯

实现步骤 - 确定索引方式

- 原始数据集：人物属性数据，由三元组组成
  - 属性种类十几种，除height, weight外都是字符串类型

```
A.J.万德 affiliation 篮球
A.J.万德 description A.J.万德（A.J. Wynder），1964年出生，前美国篮球运动员。
A.J.万德 nationality 美国
A.J.万德 weight 82公斤
A.J.库克 birthDate 1978年7月22日
A.J.库克 birthPlace #加拿大安大略省奥沙瓦
```

- 依据多种属性条件检索实体 → 一个实体的所有属性及属性值为一个文档，而不是一个三元组一个文档 → 便于检索，且效率更高
- height, weight等属性须支持范围检索，存储类型为integer，单独作为字段存储
- 其它属性，都存储为keyword类型，且存储为nested object，即不作为单独的字段存储(种类较多)

实现步骤 - 数据格式转换

按照确定的索引方式转化为json文档

```
A.J.万德 affiliation 篮球
A.J.万德 description A.J.万德 (A.J. Wynder), 1964年出生，前美国篮球运动员。
A.J.万德 nationality 美国
A.J.万德 weight 82公斤
A.J.库克 birthDate 1978年7月22日
A.J.库克 birthPlace #加拿大安大略省奥沙瓦
```

```
{
    "subj":"A.J.万德",
    "weight":"82公斤",
    "height":None,
    "po":[{"pred":"affiliation","obj":"篮球"},
          {"pred":"description","obj":"A.J.万德 (A.J. Wynder) , 1964年出生，前美
国篮球运动员。"},
          {"pred":"nationality","obj":"美国"},
         ]
}
```

需要对数据做一些预处理

- 清理属性值中无关字符
- Height, weight等属性统一单位
- “职业”等属性属性值存在多个值的情况，切分成多个属性值对

实现步骤 - 导入Elasticsearch

- 为Elasticsearch新建index和type (mapping文件)
  - Mapping文件指定了文档中每个字段在ES中存储的数据类型和位置

```
#新建第一个index 和 type
curl -XPUT 'localhost:9200/demo?pretty' -H 'Content-Type: application/json' -d'
{
    "mappings": {
        "person": {
            "properties": {
                "subj": {"type": "keyword"},
                "height": {"type": "integer"},
                "weight": {"type": "integer"},
                "po":{
                    "type": "nested",
                    "properties":{
                        "pred":{"type":"keyword"},
                        "obj":{"type":"keyword"}
                    }
                }
            }
        }
    }
}
'
```

- 使用Elasticsearch的insert API将数据导入建立的index和type

属性同义词扩展

- 因为实验的数据集较小，包含的属性种类不多，因此可以人工增加一些同义的属性词
  - 每一行的第一个词是数据集中的属性，后面是添加的同义属性
  - 查询时如果碰到同义属性，映射到数据集中的属性

```
weight 重量 多重 体重
relatedTo 相关 有关
telephone 电话 号码 电话号 电话号码 手机 手机号 手机号码
birthDate 出生日期 出生时间 生日 时候出生 年出生
height 高度 海拔 多高 身高
sibling 兄弟 哥哥 姐姐 弟弟 妹妹 姐妹
workLocation 工作地点 在哪工作 在哪上班 上班地点
children 子女 孩子 女儿 儿子
年龄 几岁 多大
代表作品 代表作 著作 成就 作品
homeLocation 家庭住址 住哪 住在哪 住在什么
```

实现步骤 - 查询解析及构造

- 解析用户查询 → 生成ES查询 → 解析查询结果
- 1. 实体查询及属性查询
  - 解析查询中的实体名和属性名，以实体名为keyword检索实体，并解析出答案中属性名对应的属性值。
  - 例如查询“姚明”，构造类似如下查询来检索实体
  - 查询“姚明：身高”，先检索实体“姚明”，再获取结果中的“身高”属性的值

```
#按实体名字查询
curl -XGET 'localhost:9200/demo/person/_search?&pretty' -H 'Content-Type:application/json' -d'
{
    "query":{
        "bool":{
            "must":{
                "term":{"subj":"姚明"}
            }
        }
    }
}
'
```

实现步骤 - 查询解析

2. 多跳查询

- 根据实体和第一个属性查询对应的属性值
- 将此属性值作为下一步的实体，根据第二个属性接着查询属性值
- 如此循环，直至结束

实现：循环调用实体属性查询

```
def search_multihop_sp(entity, preds):
    '''
    args :
        entity:  entity name
        preds: list of predicate
    '''
    for p in preds:
        if not entity_in_KB(entity):
            return False
        card = search_single_entity(entity) #调用实体查询
        if not p in card:
            return False
        o = card[p]
        entity = o

    return o
```

实现步骤 - 查询解析

3. 依据多种属性条件检索实体

- 解析出每个属性条件及其中的操作，即(pred, op, obj)，生成这一部分对应的部分查询
- 例如："身高 >= 200"对应的部分查询为：

```
{
    "range":{
        "height":{
            "gte":200
        }
    }
},
```

- 检索除height, weight除外的其它存储在nested object中的属性，例如："Not 国籍：中国"：

```
{
    "nested":{                          指定查询nested object
        "path":"po",
        "query":{
            "bool":{
                "must":[
                    {"bool":{"must_not":{"term":{"po.obj":"中国"}}}},
                    {"term":{"po.pred":"nationality"}}
                ]
```

实现步骤 - 查询解析

3. 依据多种属性条件检索实体

- 依据属性条件间的And，Or关系合并每个属性条件对应的部分查询
- 例如，“职业：篮球运动员 And 身高>=200 And Not国籍:中国”:

```
"query": {
    "bool":{
        "must":[                          must表示And关系
            {
                "range":{
                    "height":{
                        "gte":200
                    }}},
            {
                "nested":{
                    "path":"po",
                    "query":{
                        "bool":{
                            "must":[
                                {"bool":{"must_not":{"term":{"po.obj":"中国"}}}},
                                {"term":{"po.pred":"nationality"}}
                            ]}}},
            {
                "nested":{
                    "path":"po",
                    "query":{
                        "bool":{
                            "must": [
                                {"term":{"po.obj":"篮球运动员"}},
                                {"term":{"po.pred":"职业"}}
                            ]}}}
```

实现步骤 - 用户查询分类

- 以上实现了几种不同种类的查询，对于用户的查询输入，先进行分类，判断属于哪一种查询，再调用对应的查询函数
- 分类步骤
  - 由于定义的查询格式比较简单，可以在将查询按照操作符分割后，判断第一个词是否是属性名，操作符为And,Or,Not,=,<,>等
  - 如果是，代表是依据属性条件检索实体
  - 否则，检索实体或实体的属性
  - 判断一个词是否是属性词可以通过查询ES完成

实现步骤 - 执行ES查询

- 在对用户查询分类后，基于每种查询问题的模板，填充解析时识别出来的实体名和属性名，生成Elasticsearch查询

```
def _search_single_subj(entity_name):       #查询单个实体
    query = json.dumps({"query": { "bool":{"filter":{"term" :{"subj" : entity_name}}}}}) #组装query
    response = requests.get("http://localhost:9200/demo/person/_search", data = query) #查询
    res = json.loads(response.content)
```

- 构造出用户查询对应的Elasticsearch查询后，执行该查询，并解析查询结果

实践 - 进阶

- 上述只是基于Elasticsearch实现了基本的查询功能，可通过一些扩展使其支持更复杂的查询

- 1. 别名检索
  - 用户在检索实体时，可能输入的不是知识库中存储的实体名，而是其别名或者简称等，例如”周杰伦”→”周董”
  - 解决方案
    - 知识库中每个实体在存储时，存储一个alias属性 (如果有)
    - 判断查询类型为检索实体时，除了根据查询语句检索实体名，还要检索包含属性为alias, 属性值为查询语句的实体
    - 推而广之，在所有需要检索实体的地方，除了检索实体名，同时检索alias为查询语句的实体
    - 前述实验中依据实体名唯一表示实体，此处可看出不恰当，应以id为唯一标识，而将实体名作为name属性的属性值，同alias同为属性更符合含义

实践 - 进阶

2. 概念检索

- 用户希望检索某一类型的实体，例如所有类型是“篮球运动员”的实体
- 解决方案：每个实体存储一个type属性。用户做概念检索时，将其当作属性为type的属性检索即可

- 推广，考虑类型的包含关系。例如，“运动员”包含“足球运动员”和“篮球运动员”。用户检索“运动员”时，将其拆分成“足球运动员”和“篮球运动员”单独检索再OR合并。
- 解决方案：为知识库中的type关系建立上下位关系。在解析查询时对上位属性进行查询扩展

实践 - 进阶

3. 属性值模糊检索或子串检索

- 在前述的实现中，属性值都是作为keyword类型存储，即不分词，只精确匹配。用户在检索时须指定具体属性和属性值
- 用户可能并不知道知识库中存在哪些属性。因此可将字符串类型属性值存储为String类型(分词处理)，此时用户可以只输入任意属性值，查询时将用户输入与知识库中的所有属性值做模糊匹配，按照相似度返回实体列表

4. 推广对不同类型属性值的操作

- 在前述实现中，所有属性，除了height和weight存储为integer类型单独存储，支持range search外，其它都看作Keyword类型存储在nested object中，只支持精确匹配
- 可将所有属性按照不同类型分别存在不同的nested object中，例如integer, Date, string, entity. 每种类型的属性支持不同的操作，例如integer, Date支持range search, entity支持多跳查询或逆关系查找

实践 - 进阶

5. 反向检索

- 知识库中可能只存在“姚明，女儿，姚沁蕾”,但用户查找“姚沁蕾:父亲”
- 解决方案：每个实体在存储时，对于其存在反向属性的属性，除了将其作为subject,存储其所有的(predicate, object)对，同时也将其作为object, 存储其所有的(reverse-predicate, object)对，此时原来的object作为反向关系的subject
  - 姚明:PO[(女儿，姚沁蕾)...]

    RSP[(父亲，姚沁蕾)...]
- 用户在查询SP时，例如“姚沁蕾：父亲”,除了将其作为原本的SP查询外，同时将其作为反向的PO查询，即检索RSP属性中拥有属性值对(父亲：姚沁蕾)的实体
- 同理，用户在查询PO时，同时也做反向的SP查询
- 可见前述在存储时将实体名存储为“subj”不恰当，应存储为“entity”；同时，不仅要存储PO属性对，还要存储SP属性对

实践 - 进阶

6. 属性path查询

- 除了支持依据实体查属性的多跳查询，也可以支持依据属性查实体的多跳查询，例如“女儿:星座:白羊座”的实体
- 解决方案：逆向搜索，先检索有最后一个属性和属性值的PO对的实体，即检索所有(星座，白羊座)实体
- 检索所有拥有属性”女儿“，且属性值在(星座，白羊座）实体中的实体

7. 在查询中嵌套path查询

- 除了多个PO属性条件检索实体，也可以在其中嵌套path查询。例如“国籍：中国 and 女儿：星座：白羊座”
- 解决方案：对于每个属性条件，可以是简单的PO，也可以PO path，也可以是反向SP

实践 - 进阶

8. 多元关系

- 除了二元关系，也可能是多元关系，例如，“2000年后在火箭队的篮球运动员”，这里不再是“职业:篮球运动员”查实体的二元关系
- 解决方案：对用户查询中的多元关系，同样是将其分解为多个二元关系(类Freebase CVT)的组合查询，上述即“职业：篮球运动员” And “player.member of: 火箭队”And “player.join_time: 2000”

9. 单位统一

- 知识库中的“身高”单位为cm，用户查询“身高>1.9m”
- 解决方案：对用户查询进行预处理，可以用正则表达式等方法识别出带单位的数值，日期等，转化为存储时的单位

谢谢大家！

本课程课件由OpenKG提供支持

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象
- 新浪微博：ChinaHadoop