知识图谱入坑笔记

梁子

插入时间戳。

Table of Contents

[1. 知识图谱本身](#org5e5c08e)

[1.1. 在知识图谱之前，有待了解的一些基本知识](#org2d3432d)

[1.1.1. 实体（entity）、本体（ontology）、与模式（schema）](#orge0ef9b3)

[1.1.2. RDF（Resource Description Framework） 资源描述框架](#org9f1caa0)

[1.1.3. RDFS ——RDF Schema](#org44bd326)

[1.1.4. OWL Ontology of Web Language](#org2804b6d)

[1.1.5. SPARQL： RDF的Query Language](#org34f5ecb)

[1.2. 知识图谱相关数据集归纳](#orga56c4bf)

[1.2.1. DBpedia](#orga283dd8)

[1.2.2. Yago](#org6e70977)

[1.2.3. wikidata](#org7c80792)

[1.2.4. ConceptNet](#orge03425e)

[1.2.5. Microsoft Concept Graph](#org338cd4d)

[1.2.6. 特定的中文知识图谱](#orgc91c443)

[2. 将知识图谱用于\*\*\*](#orgcda524e)

# 知识图谱本身

## 在知识图谱之前，有待了解的一些基本知识

### 实体（entity）、本体（ontology）、与模式（schema）

这三个词汇有点像是从哲学中舶来味道，本质上都是对存在的一种抽象。下面依次简单介绍之。

实体是一种对象。它本身更像是一种静态的东西，但是却从动态（谓词）的结果中反应其特点。一个实体本质上就是在各个属性维度上取了一些值（实体）的集合。 可以用JAVA等编程语言中的对象或者数据库中每一个表里的一条来进行理解。实体就是实例（instance）。

本体是对实体的特点和行为的的抽象。（另一个定义：本体是对概念和关系的形式化表述）。同样用面向对象理解，class的定义就是对应object的本体。

模式（schema）是数据库名词。一个数据库（也就是表的集合）就是一个schema。

下文介绍的RDF主要是在实体层面，RDFS主要是在schema层面。

### RDF（Resource Description Framework） 资源描述框架

参考链接：

1. [知乎-语义网络基础笔记1](https://zhuanlan.zhihu.com/p/57202859)
2. [知乎-讲故事一样的介绍](https://zhuanlan.zhihu.com/p/31726910)
3. 基本要素

资源描述框架是干什么的？

描述资源的！

资源是什么？

资源是对象！对，资源就是对象。

所以，RDF本质上类似于面向对象编程语言中对对象的封装，也就是， 使用各种属性和对应的属性值来描述一个实体 。

因此，一个RDF本质上包括：

* + 资源（R）。也就是对象。
  + 属性（Realtion)。描述了资源之间的关系。

但是，光有这两点显然还是不够的，资源提供了一个个的节点，而属性提供了节点的连接，所以应该最后还有一个描述的操作，被称作声明。

RDF中存储的就是声明。声明主要是包括了（资源，属性，值）的一个三元组。而此处的“值”，当属性为“谓词”的时候，为一个资源对象。 但是，值未必一定是实体，也可能是构成实体的“原子值”（当然这也是一种特殊的实体），原子值在绝大多数时候是字符串）。

那么，整个知识图谱只剩下了最后一步，如何查询（寻找）这些资源或者属性？

辨识的方式是基于URL进行的。可 以网络中的URL进行理解。所以，一个声明（一个三元组）本质上就是存储了三个URL的列表。

一般而言，URL按类型还可以继续分为：

* 1. IRI（international Resource Identifies），类似于一般意义上的index；
  2. blank nodes，空白；
  3. literals。原子值中最大的一种。

1. 声明也是一种对象，图也是一种资源

刚才说过，一个实体可以有多个属性，这样不同的实体就可以彼此连接起来，形成一张巨大而稀疏的图，这和知识图谱具有了结构上的一致。 一般而言，这张图是有向的，从主体（subject）借助关系（谓词，predicate）指向客体（object）。这样的图往往只适合用三元组 进行表达，而非邻接矩阵等东西。因为两个节点之间可以有1000条边。

一般而言，在知识图谱中，资源用圆圈表示，谓词用方框表示.

那么，对于这样的一个图——也就是一个声明的集合——一个三元组的集合，能否把它也看做一个实体呢？

首先，单个的一个声明是可以看作一个实体的，它有三个属性（主体、谓词、客体）。 一张图其实也可以看作一个实体，这里就有了一些meta的味道。

1. RDF常用语法
   1. Turtle .ttl文件

下图展示了一个基本的三元组：

<http://www.semanticwebprimer.org/ontology/apartments.ttl#BaronWayBuilding>

<http://dbpedia.org/ontology/location>

<http://dbpedia.org/resource/Amsterdam>.

可以看出：

* + 1. 每一行都是URL。
    2. 最后以句号结束。

下面展示一个带原子值的：

<http://www.semanticwebprimer.org/ontology/apartments.ttl#BaronWayAppartment>

<http://www.semanticwebprimer.org/ontology/apartments.ttl#hasNumberOfBedrooms>

"3"^^<http://www.w3.org/2001/XMLSchema#integer>.

可以发现，原子值被放在了双引号内，后面有URL展示其数据类型。

在此基础上，主要诞生了以下语法：

* + 1. 缩写。类似于C++中的明明空间的概念，可以用一个简段的符号把上面例子中的重复部分做映射，使用命名空间的方式进行定义；
    2. 省略主体。如果主体相同，声明之间可以用分号隔开，省略主语；
    3. 省略主体和为词。如果主体和谓词相同，声明之间可以用逗号隔开，省略主体和谓词；
    4. 为声明或图定义URL。使用大括号获得一个声明集合的对象化。

详情可以参考这篇笔记： [知乎-语义网络基础笔记1](https://zhuanlan.zhihu.com/p/57202859)

* 1. XML语言

例子如下：

<?xml version="1.0" encoding="utf-8"?>

<rdf:RDF xmlns:dbpedia-owl="http://dbpedia.org/ontology/"

xmlns:dbpedia="http://dbpedia.org/resource/"

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:swp="http://www.semanticwebprimer.org/ontology/apartments.ttl#">

<rdf:Description

rdf:about="http://www.semanticwebprimer.org/ontology/apartments.ttl#BaronWayAppartment">

<swp:hasNumberOfBedrooms

rdf:datatype="http:www.w3.org/2001/XMLSchema#integer">

3

<swp:hasNumberOfBedrooms>

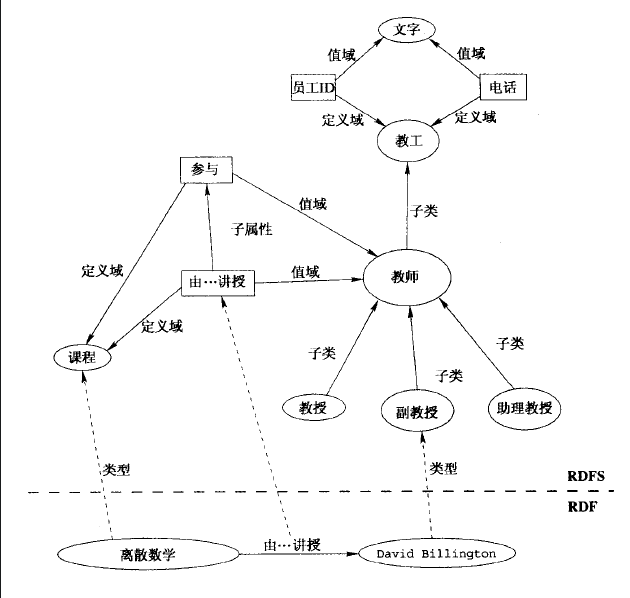
</rdf:Description>

### RDFS ——RDF Schema

1. 与RDF的区别

如果说RDF是在对对象进行的操作的话，那么RDFS就是在对类进行操作。这就是二者的区别。

可以简单用下图作为区别式的理解：



1. 基本语法

现以[这篇笔记](https://zhuanlan.zhihu.com/p/32122644)为例展示其基本语法。

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix : <http://www.kg.com/ontology/> .

### 这里我们用词汇rdfs:Class定义了“人”和“地点”这两个类。

:Person rdf:type rdfs:Class.

:Place rdf:type rdfs:Class.

### rdfs当中不区分数据属性和对象属性，词汇rdf:Property定义了属性，即RDF的“边”。

:chineseName rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:string .

:career rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:string .

:fullName rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:string .

:birthDate rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:date .

:height rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:int .

:weight rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:int .

:nationality rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range xsd:string .

:hasBirthPlace rdf:type rdf:Property;

rdfs:domain :Person;

rdfs:range :Place .

:address rdf:type rdf:Property;

rdfs:domain :Place;

rdfs:range xsd:string .

:coordinate rdf:type rdf:Property;

rdfs:domain :Place;

rdfs:range xsd:string .

上面我们定义了person和place两个东西——用三元组定义的。我们在定义他们的时候，他们同样也是实体，但在我们使用他们时，他们成了类。

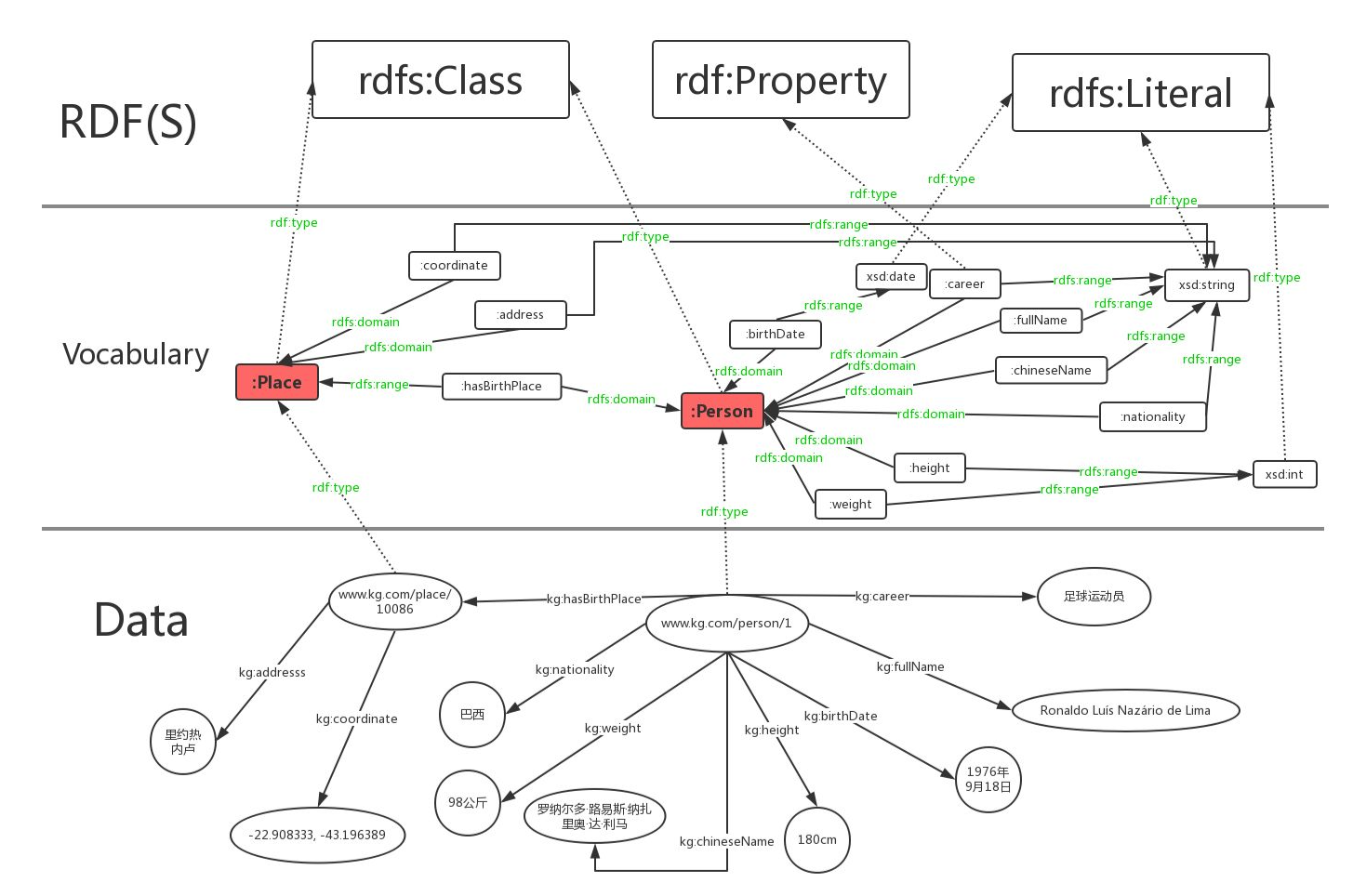
简单列举一下可能用到的一些约定：

* + 核心类：
    - rdfs：Resource，所有资源的类；
    - rdfs:Class,所有类的类
    - rdfs:domain, 属于哪个类
    - rdfs:range, 类型
    - rdfs:Literal，所有文字(字符串)的类。
    - rdf:Property，所有属性的类。
    - rdf:Statement，所有具体化声明的类。
  + 核心属性：
    - rdf:type，将一个资源关联到它的类。该资源被声明为该类的一个实例。
    - rdfs:subClassOf，将一个类关联到它的超类。一个类的所有实例都是它的超类的实例。注意，一个类可能是多个类的子类。
    - rdfs:subPropertyOf，将一个属性关联到它超属性中的一个。
    - rdf:subject，讲一个具体化声明关联到它的主语。
    - rdf:predicate，将一个具体化声明关联到它的谓语。
    - rdf:object，讲一个具体化属性关联到它的宾语。

个人观点：随着实体的复杂，一个实体可以属于无数个类，或者说，大量的有限多个类。跨领域很重要。所谓的层次结构，并不天然适合描述 实体。实体是复杂的，一个层次结构仅仅是一个角度上的投影。

另外，谈论到这里，可以挖掘到过去的一些符号主义的影子。什么叫符号？ Place 和 Person就是符号。因为他们本质上不过是string， 却同时有类的值和实体的类两种特征。

下图可以更明白地区分两种层次：

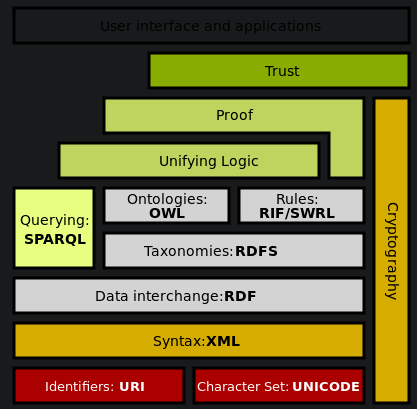


参考链接：

[知乎-语义网络基础笔记1](https://zhuanlan.zhihu.com/p/57202859)

[lalal](https://zhuanlan.zhihu.com/p/32122644)

现在可以呈上一个架构图，看看我们到底走了多远：



### OWL Ontology of Web Language

从schema走出来之后，我们看看另一种更为严格一点的封装方式，被称作OWL。 同样地，先给出一段示例代码：

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix : <http://www.kg.com/ontology/> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

### 这里我们用词汇owl:Class定义了“人”和“地点”这两个类。

:Person rdf:type owl:Class.

:Place rdf:type owl:Class.

### owl区分数据属性和对象属性（对象属性表示实体和实体之间的关系）。词汇owl:DatatypeProperty定义了数据属性，owl:ObjectProperty定义了对象属性。

:chineseName rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:string .

:career rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:string .

:fullName rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:string .

:birthDate rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:date .

:height rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:int .

:weight rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:int .

:nationality rdf:type owl:DatatypeProperty;

rdfs:domain :Person;

rdfs:range xsd:string .

:hasBirthPlace rdf:type owl:ObjectProperty;

rdfs:domain :Person;

rdfs:range :Place .

:address rdf:type owl:DatatypeProperty;

rdfs:domain :Place;

rdfs:range xsd:string .

:coordinate rdf:type owl:DatatypeProperty;

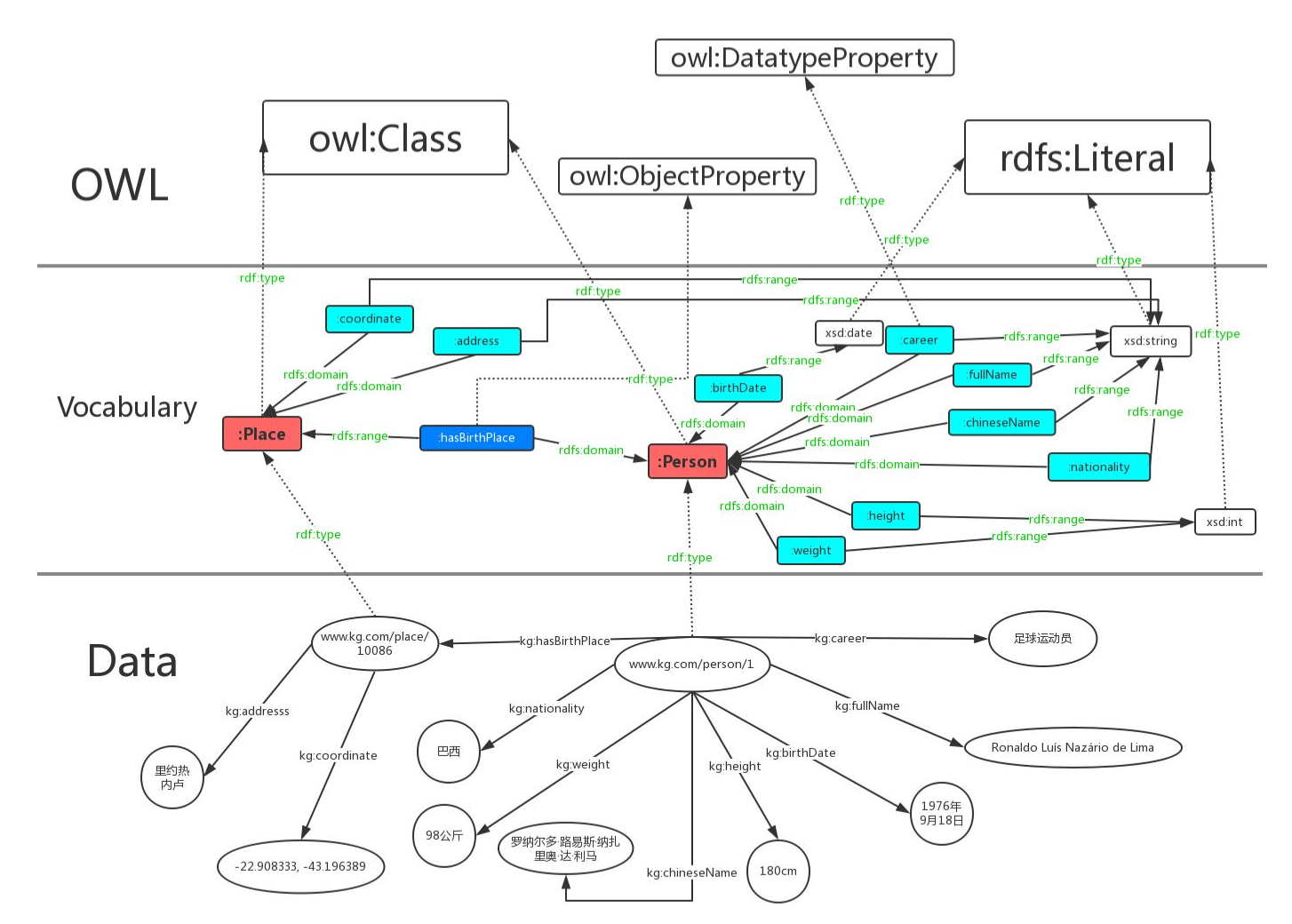
rdfs:domain :Place;

rdfs:range xsd:string .

从上面可以看出，该段代码同RDFS最大的区别在于对关系的定义。OWL细化了关系，最基本的，OWL将关系划分为两个层面：

1. 数据属性。 也就是结果是一个原子值的那些属性；
2. 对象属性。结果也是一个实体的属性。

如果我们也把他画成一个图的话：



1. OWL的推理与知识图谱补全

OWL的优势在于推理。主要包括以下推理：

* 1. 基于谓词逻辑的推理。 暂略，有时间专门写笔记填坑。
  2. 知识图谱补全。主要是根据关系之间的关系。比如，如果A是B的爸爸，那么B肯定是A的儿女。
  3. 本体映射。主要应用在知识图谱融合。比如，判断两个类是否是同一个，两个实体是否是同一个，连个属性是否是同一个等等。

所有的基于规则的推理，大约都是可以应用在这种三元组上的。

### SPARQL： RDF的Query Language

在做查询之前，先想一个一个问题，我们要查的是什么？

答案： 三元组。一个或一些三元组。

那么我们有什么可以用来去查？

答案：主体，客体，谓词。

所以，所做的查询大概是这样的一类事物：通过一系列条件构成一个查询的图模式，寻找所有可能的候选项，返回结果。

下面是最简单的只针对一个三元组的同时对三个要素都无任何要求的查询：

SELECT \* WHERE {

?s ?p ?o

}

复杂一点的，下列链接中给的例子，“周星驰出现了哪些电影？”

SELECT ?n WHERE {

?s rdf:type :Person.

?s :personName '周星驰'.

?s :hasActedIn ?o.

?o :movieTitle ?n

}

上面的一段代码，可以用这样的一个逻辑去思考：

1. s 满足 Person 这个类型要求；（第一句话也是一个三元组）
2. s 的名字是 “周星驰”；
3. s 在 o 里参与演出；
4. o 的名字是 n；
5. 返回n；

在4和5之间，有一步，是找到了所有的o的名字是n的集合，然后才是返回n这最后一步。

其中，如果我们再看一下代码，可以发现变量前面有一个问号，这是模板匹配相关的基本语法。我觉得还好，比shell中的变态程度低一点。

详情请见： [参考链接1](https://zhuanlan.zhihu.com/p/32703794)

## 知识图谱相关数据集归纳

本节首先介绍当前世界范围内知名的高质量大规模开放知识图谱，包括

* DBpedia
* Yago
* Wikidata
* BabelNet
* ConceptNet
* Microsoft Concept Graph
* 中文开放知识图谱平台 OpenKG。

### DBpedia

地址： <https://wiki.dbpedia.org>

github 主页：<https://github.com/dbpedia>

DBpedia 是一个大规模的多语言百科知识图谱，可视为是维基百科的结构化版本。 DBpedia 使用固定的模式对维基百科中的实体信息进行抽取，包括 abstract、infobox、category 和 page link 等信息。 DBpedia 目前拥有 127 种语言的超过两千八百万个实体与数亿个 RDF 三元组，并且作为链接数据的核心， 与许多其他数据集均存在实体映射关系。 DBpedia 支持数据集的完全下载。

### Yago

Yago 是一个整合了维基百科与 WordNet的大规模本体，它首先制定一些固定的规则对维基百科中每个实体的 infobox 进行抽取， 然后利用维基百科的category进行实体类别推断（Type Inference）获得了 大量的实体与概念之间的 IsA 关系（如：“Elvis Presley” IsA “American Rock Singers”）， 最后将维基百科的 category 与 WordNet 中的 Synset（一个 Synset 表示一个概念）进行映射， 从而利用了 WordNet 严格定义的 Taxonomy 完成大规模本体的构建。 随着时间的推移，Yago 的开发人员为该本体中的 RDF 三元组增加了时间与空间信息， 从而完成了 Yago2[98]的构建，又利用相同的方法对不同语言维基百科的进行抽取，完成了 Yago3[99]的构建。 目前，Yago 拥有 10 种语言约 459 万个实体，2400 万个 Facts，Yago 中 Facts的正确率约为 95%。Yago 支持数据集的完全下载。

### wikidata

Wikidata 是一个可以自由协作编辑的多语言百科知识库，它由维基媒体基金会发起， 期望将维基百科、维基文库、维基导游等项目中结构化知识进行抽取、存储、关联。 Wikidata 中的每个实体存在多个不同语言的标签，别名，描述，以及声明（statement）， 比如 Wikidata 会给出实体“London”的中文标签“伦敦”，中文描述“英国首都”以及图 3 给出了一个关于“London”的声明的具体例子。 “London”的一个声明由一个 claim 与一个 reference 组成， claim 包括property:“Population”、value:“8173900”以及一些 qualifiers（备注说明）组成， 而 reference 则表示一个 claim 的出处，可以为空值。 目前 Wikidata 目前支持超过 350 种语言，拥有近 2500 万个实体及超过 7000 万的声明[100]， 并且目前 Freebase 正在往 Wikidata 上进行迁移以进一步支持 Google 的语义搜索。Wikidata 支持数据集的完全下载。

### ConceptNet

ConceptNet 是一个大规模的多语言常识知识库，其本质为一个以自然语言的方式描述人类常识的大型语义网络。 ConceptNet 起源于一个众包项目 Open Mind Common Sense， 自 1999 年开始通过文本抽取、众包、融合现有知识库中的常识知识以及设计一些游戏从而不断获取常识知识。 ConceptNet 中共拥有 36 种固定的关系，如 IsA、UsedFor、CapableOf 等， 图 4 给出了一个具体的例子，从中可以更加清晰地了解 ConceptNet 的结构。 ConceptNet 目前拥有 304 个语言的版本，共有超过 390 万个概念，2800 万个声明（statements，即语义网络中边的数量）， 正确率约为 81%。另外，ConceptNet 目前支持数据集的完全下载。

### Microsoft Concept Graph

Microsoft Concept Graph 是一个大规模的英文 Taxonomy， 其中主要包含的是概念间以及实例（等同于上文中的实体）概念间的 IsA 关系， 其中并不区分 instanceOf 与 subclassOf 关系。 Microsoft Concept Graph 的前身是 Probase，它过自动化地抽取自数十亿网页与搜索引擎查询记录， 其中每一个 IsA 关系均附带一个概率值， 即该知识库中的每个 IsA 关系不是绝对的，而是存在一个成立的概率值以支持各种应用， 如短文本理解、基于 taxonomy 的关键词搜索和万维网表格理解等。 目前，Microsoft Concept Graph 拥有约 530 万个概念，1250 万个实例以及 8500 万个 IsA 关系（正确率约为 92.8%）。 关于数据集的使用，MicrosoftConcept Graph 目前支持 HTTP API 调用，而数据集的完全下载需要经过非商用的认证后才能完成。

### 特定的中文知识图谱

1. Zhishi.me

Zhishi.me 是第一份构建中文链接数据的工作， 与 DBpedia 类似，Zhishi.me 首先指定固定的抽取规则对百度百科、互动百科和中文维基百科中的实体信息进行抽取， 包括 abstract、infobox、category 等信息； 然后对源自不同百科的实体进行对齐，从而完成数据集的链接。 目前 Zhishi.me 中拥有约 1000 万个实体与一亿两千万个 RDF 三元组，所有数据可以通过在线 SPARQL Endpoint 查询得到。

1. Zhishi.schema

Zhishi.schema 是一个大规模的中文模式（Schema）知识库，其本质是一个语义网络， 其中包含三种概念间的关系，即equal、related与subClassOf关系。 Zhishi.schema抽取自社交站点的分类目录(Category Taxonomy)及标签云（Tag Cloud）， 目前拥有约40万的中文概念与150万RDF三元组，正确率约为84%，并支持数据集的完全下载。

1. XLore

XLore 是一个大型的中英文知识图谱， 它旨在从各种不同的中英文在线百科中抽取 RDF 三元组，并建立中英文实体间的跨语言链接。 目前，XLore 大约有 66 万个概念，5 万个属性，1000 万的实体，所有数据可以通过在线 SPARQL Endpoint 查询得到。

1. 中文开放知识图谱平台

从上面可以找到各种知识图谱相关的数据。 <http://openkg.cn/home>

# 将知识图谱用于\*\*\*