寒假论文阅读总结 许茜

论文一：DBpedia P1-P3

论文二：zhishi.me P3-P5

论文三：大型 KB 开发运维 P6-P9

论文四：YAGO2 P9-P18

·论文一：DBpedia – A Crystallization Point for the Web of Data

1. 解决问题:在 wikipedia 中提取结构化信息,使信息可以在 web 上访问，并与其他数据源互连
2. 如何解决的：通用 infobox 抽取和基于映射的 infobox 抽取，使用全局唯一的标识符，可以将实体由标识符连接到 RDF 和其他数据源的数据。
3. 文章内容：介绍了 DBpedia 知识库的提取，与 web 上其他数据源互连的状态，简单概述了围绕 DBpedia 的应用程序。
4. 为什么推出 DBpedia：大多数 KnowledgeBase 仅涵盖特定领域，有较少的工程师维护需要很高成本。而 wiki 有很多人维护。
5. 目前进展到的阶段：实现了多领域、多语言。加上其他数据发布者设置的与 DBpedia 的 RDF 链接,围绕 DBpedia 的 RDF 三元组数量达 47 亿。
6. 项目贡献：

   a)开发了信息提取框架,可以将 wiki 的 infobox 映射到 KB 的本体中。可以根据 wiki 的文章及时更新。

   b）定义了 web-dereferenceable identifier（URI）

   c）项目包含部分从 DBpedia 指向其他 web 数据源的 RDF 链接，并支持其他数据源连入 DBpedia。

7. 未来工作:跨语言 infobox 知识融合(不同语言版本的侧重点不同)&wiki 文章扩充（通过其他数据源的图片、音频等扩充文章）&一致性检查（不同语言版本的 wiki 之间/wiki 与外部其他数据源之间）

8. DBpedia 体系结构&知识抽取框架

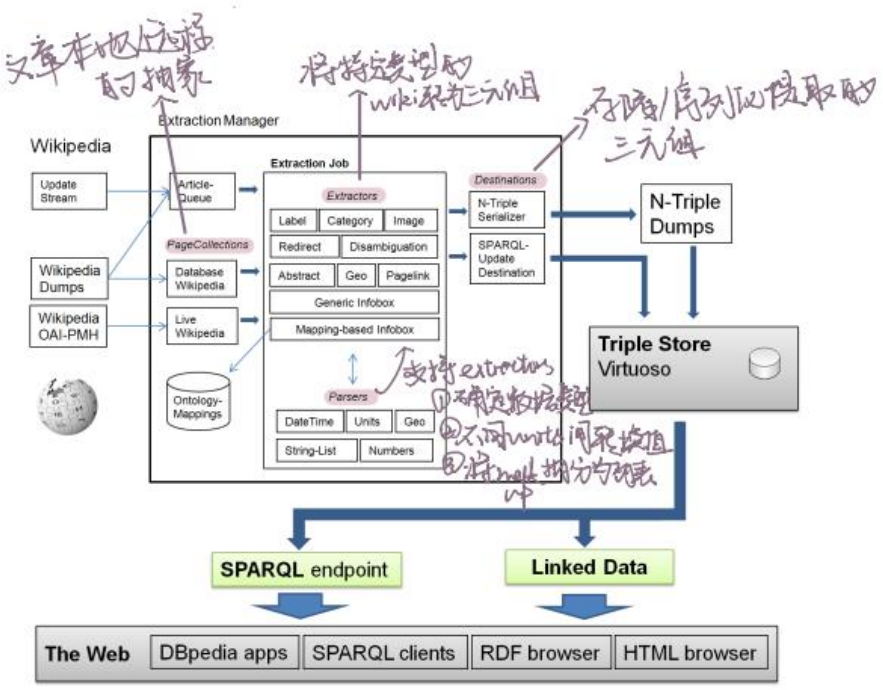

Figure 1. Overview of DBpedia components.

整个框架的核心是图中的 extraction manager，由 page collection, destinations, extractors, parsers 组成。Extraction manager 可以在 page collection 后将 wiki 的文章传递给 extractors，extractors 在 parsers 的支持下将特定类型的 wiki markup 转换成三元组，并将其输出传递到 destinations。框架包含 11 个 extractors，可分别处理 wikipedia 的标签、摘要、消歧、跨语言链接等等。

可实现两种工作流：基于 dump 的抽取和实时抽取。抽取的方法分为两种,一种针对广泛覆盖的通用infobox抽取(无法解析同义词属性名称)，一种是基于映射的 infobox 抽取（由于映射模板所需知识限制，本体的

扩展或许可利用众包方法。）

9. 实时抽取可以使得 DBpedia 随时根据 wikipedia 的更改而更新，抽取新的 RDF。这一步的实现是访问 wiki 实时提要，每当 wiki 的文章发生更改时，使用更新流抽取新的 RDF，再通过 Destination 删除目标三元组，存储新的三元组。

10. DBpedia 分类方案对比

    Wikipedia 的类别可以被很多编辑员协同更新，却不能形成有层次架构的主题，类与类之间关联太松散。YAGO 的分类 schema 层次架构很深，将 wiki 的类别映射到 wordnet synset 中创建得来。DBpedia Ontology 是根据英文版 wiki 中最常用的 infobox 模板手动创建。

11. Web 上为 DB 服务的不同访问机制

    关联数据参引、SPARQL 端点、可以在网页下载 DBpedia 的 RDF dumps、基于 Lucene 的索引对加权标签进行查找。

论文二：zhishi.me-Weaving

1. 解决问题：DBpedia 的中文语义数据方面所做工作较少
2. 如何解决：zhishi.me 从百度百科、互动百科、中文 wiki 抽取，并提出几种映射策略用来自动链接。
3. 为什么推出 zhishi.me: LOD 项目包含的中文知识非常稀疏，少有的中文知识数据集也是英文表示，无法用中文直接使用
4. 解决的构建 LOD 的两大挑战：管理不同数据源中的知识异构性；有效的发现实例之间的关系。

5. 构建纯中文 LOD 的突破：将查找和数据集成的操作可视化。

6. 文章内容：从三个中文百科中抽取结构化信息的过程；每两个百科之间构建关系进行整合；与现有的关联数据建立连接，将 CLOD 中的资源链接到 DBpedia 的资源；中文 LOD 的访问机制。

7. 下一步工作：（1）增加中文非百科数据源，如淘宝、豆瓣等，特定领域的知识库可以补充更准确的表述。（2）改进实例匹配策略（3）精炼抽取的属性，通过迭代，根据属性自动构建通用的本体。

8. 语义数据抽取：

   中文 wiki 的 dump 抽取采用类似 DBpedia 的抽取算法，百度百科和互动百科从 html 文件中抽取。抽取的 12 中类型的文章内容中，值得记录的有消歧、重定向等。消歧采用的办法是保留多个主题之间相同的主要术语，如将木星（神话）和木星岛都添加到木星的页面中，将具体含义放在括号里。重定向用于解决三个百科的同义问题，wiki 中包含简体中文和繁体中文的重定向，只需按规则将繁体转为简体即可。

   三个百科之间，百度百科的类别最多，涵盖了最广泛的主题，并且含有图片；互动百科的 infobox 属性数量高；中文 wiki 的摘要信息数量高，包含更多的链接。不同的数据源有着各自的特征，但表示主题的方式相似，可以进行整合。Zhishi.me 使用中文分词技术细化质量不是很高的类别，选择一些通用类别手动映射到 YAGO 类别。

9. 不同数据集的数据层映射

   数据集之间的映射通常分为两个级别，schema 级别的本体匹配和实

例匹配。Zhishi.me 基于 silk 框架的原理，即索引预匹配来降低时间复杂度。即便如此，仍具有很大的时间空间复杂度，利用分布式 MapReduce 框架，将所有资源按索引项排序，类似的资源能聚集在一起。

三种生成索引的策略：（1）使用原始的标签，具有高精度（2）清除标点(相同的实体可能由于标点符号有不同的标签)(3)扩展同义词，利用重定向获得高质量同义词关系，根据传递属性找到更多链接。

Zhishi.me 利用 wiki 的跨语言链接将 wiki 数据集合 DBpedia 链接起来，再通过传递属性将整个 zhishi.me 与 DBpedia 链接。

10. 访问机制

（1） 关联数据，zhishi.me 使用 IRI 替代 URI，为使 IRI 与 HTML 兼容，对中文字符转编码。

Table 4. IRI Patterns

Sources	IRI Patterns
Baidu Baike	http://zhishi.me/baidubaike/resource/[Label]
Hudong Baike	http://zhishi.me/hudongbaike/resource/[Label]
Wikipedia Chinese version	http://zhishi.me/zhwiki/resource/[Label]

（2） 查找服务，当用户不知道确切 IRI 时，使用查找服务访问

使用四种匹配策略构造索引：返回标签与用户查询完全匹配的所有资源、使用重定向得来的 sameAs 链接提供共同的引用、使用已知同义词提供更多资源、返回给定名称的多种含义对应的所有资源。

（3） SPARQL 端点查询。

论文三：Building，Maintaining，and Using Knowledge Bases：A Report from the Trenches

1．解决的问题

当前，对知识库从构建，维护到使用的端到端过程论文非常少。本文介绍了以下内容:

（1）在 Kosmix 和 WalmartLabs 构建，更新和管理大型知识库的过程；

（2）如何使用该知识库支持一系列应用程序；

（3）知识库团队如何组织管理，及开发中的经验教训。

2．两种不同的知识库

（1） 特定领域知识库（Domain-specific KBs)：特定领域的知识库仅捕捉相关领域的概念、实例和关系，例如 DBLP、Google Scholar、DBLife、echonest 和等。

（2） 全局知识库（Global KBs)：试图覆盖整个世界，包括谷歌知识图谱，YAGO，DBPedia 和维基百科 infobox 集合。

3．构建知识库

维基百科不是传统意义上的知识库，将其转换为知识库的关键步骤有:

（1） 从维基百科构建分类树（taxonomy tree)；

（2） 在分类法（taxonomy）基础上构建有向无环图（DAG)；

（3） 从维基百科中提取关系；

（4） 添加元数据（metadata)；

（5） 添加其他数据源；

3.1 构建分类树

（1） **爬取维基百科**：维护一个维基百科的内部镜像（in-house mirror），并监视维基百科网站以保持其更新。或者，可以在 download.wikimedia.org/enwiki 下载其 XML dump，但时效性略差，因为 dump 是两周更新一次。而本项目期望每天更新。

（2） **构建维基百科图表**：维基百科中有两种**主要**页面：**文章页面**和**分类页面**，解析 XML dump 来构造一个图，其中每个节点都代表一篇文章或一个分类，每条边都代表一个维基百科链接，从一个分类 X 到另一子分类 X，或从一个分类 X 到一篇分类 X。

（3） **构建分类树**：根据维基百科生成的**有向循环图**构造分类树，这里采用 **Edmonds 算法**。这是一种用于寻找有向图中最优分支的最大或者最小数的算法。该算法的一个实现是 **Tarjan**。

3.2 构建有向无环图 DAG

项目希望从维基百科构建一个主要分类树，同时保留其他支线的信息。但由于直接获取得维基百科生成的是有向循环图，所以需要按以下步骤构造 DAG：得到分类树 T 后，回到原始维基百科图 G，并给 T 的边赋予权重，然后通过运行 DFS 的多次迭代来删除 G 中的循环。在每个 DFS 迭代中，如果检测到一个回边，那么就有一个循环，通过删除最低权重的边来打破该循环。当 DFS 不再检测到任何循环时，停止。

3.3 从维基百科中抽取关系

不预先定义关系，也不试图提取关系实例。相反，我们**在概念实例之间抽取形式自由的关系实例**。例如，假设文章页面“Barack Obama”有一个名为“Family”的部分，其中提到文章页面“Bo（Dog)”。然后我们创建一个关系

实例<Barack Obama，Bo（Dog)，Family>，表示“Barack Obama”和“Bo（Dog)”之间有一个关系“Family”。

3.4 添加元数据

（1） **添加同义词和同音词**：同义词在**重定向**页面中捕获。同音词在**消歧义**页面捕获。以统一的方式将所有这些同义词和同音词添加到知识库中：对于每个同义词，在图中创建一个节点，然后通过标记为“alias”的边将其链接到主节点。对于每个消歧页面，在图中创建一个节点，然后通过标记为“同音词”的边将其链接到所有可能的解释节点。维基百科通常指定一个同音词解释为默认解释。

（2） **为每个节点添加元数据**：对于知识库中的每个节点，都分配一个 ID 和一个名称，对应维基百科页面的标题。然后向节点添加多种类型的元数据。其中包括：1）从维基百科和网络搜索结果中获得的一组主页；2）人物的推特账号；3）一组经常同时出现的其他概念和实例；3）该维基百科页面的访问量；4）一个介于 0-1 之间的 Web DF 分数，表示该节点在网页中被提及的频率。5）一个介于 0-1 之间的社交媒体 DF 分数，表示该节点在社交媒体中被提及的频率。

3.5 添加其他数据源

首先，通过提取数据实例和实例上的分类（如果有的话)，从数据源 S 中提取数据。

对于每个实例，抽取各种属性，包括名称、类别（例如旅行簿、电影等)、url、关键字（例如，与 S 中的此实例共同出现的一组关键字)、关系（例如，此实例与 S 中的其他实例关联的一组关系）和同义词。

如果分类 T 存在，则使用最新的匹配器，将其类别（概念）与现有 KB 中的匹配，然后清理并添加这样的匹配（例如，car = automobile）到一致性表。

4．维护知识库

主要包括两方面：定期更新知识库和手动管理以提高准确性或添加更多内容。

5．知识库应用

主要应用有：理解用户查询，深网搜索，语境内广告（in-context advertising)，社交媒体中的事件监控，产品搜索，快速且可扩展的访问等。

论文四：YAGO2：A Spatially and Temporally Enhanced Knowledge Base from Wikipedia

1．为什么推出 YAGO2：现有知识库大多**忽略了时间维度**。在**空间维度**上也有类似的问题，即其表示的范围不够大。

目标是让知识库将充分记录时间和空间，不仅知道一个事实是真的，而且知道它**何时何地**是真的。

贡献：与最初的 YAGO 相比，构建 YAGO2（并维护它）的方法是自上而下系统地设计的，目的是将面向实体关系的事实与空间和时间维度整合起来。

（1）一个可扩展的事实提取框架，可以利用文本中的信息框、列表、表、类别和常规模式并允许快速、轻松地指定新的提取规则；

（2）扩展了为捕获时间和空间而定制的知识表示模型，以及将时间和位置信息传递到所有相关事实的规则；

（3）一种新的 SPOTL（X）时空增强事实表示法，具有表达性和易于使用的查询功能。

2. YAGO

YAGO 知识库是从维基百科自动构建的。维基百科中的每一篇**文章**都成为知识库中的一个**实体**（例如，由于 Leonard Cohen 在维基百科中有一篇文章，Leonard Cohen 在 YAGO 中成为一个实体）。维基百科中的某些**分类**可以用来传递类型信息（例如，关于 Leonard Cohen 的文章属于**加拿大诗人类别**，因此他被分类为加拿大诗人）。YAGO 将此类型信息链接到 WordNet[2]的分类（例如，加拿大诗人成为 WordNet 词组诗人的一个子类）。

Linkage algorithm 的步骤如下：

（1） 对于页面的每个类别（category），它通过浅层名词短语解析来确定类别名称的头词（head word）。在**加拿大诗人**的例子中，头词是诗人。

（2） 检查头词**是否复数**。如果是这样，建议将类别作为类（class），并将项目实体作为实例。

这一过程有效地区分了**主题**范畴（如加拿大诗歌）和**概念**范畴，简单地说，只有可数名词才能以复数形式出现，**只有可数名词才能成为本体类**。该类通过选择 WordNet 中最常用的词头词义链接到 WordNet 分类法。

3. 可扩展提取结构

规则本身是 YAGO2 知识库的一部分。有不同类型的规则：

（1）事实规则：是 YAGO2 知识库的附加事实（additional facts）。它们是旧版 YAGO 代码包含的所有手动定义的异常和事实的声明性翻译（declarative translations）。这些定义包括所有关系的定义、它们的域和范围，以及构成文

本类型的 YAGO2 层次结构的类的定义（yagoInteger 等)。

（2) 蕴涵规则：蕴涵规则假设知识库中出现了某些事实，则应添加另一个事实。因此，蕴涵规则用于从现有知识中推断出新知识。

（3）替换规则：假设源文本的一部分与指定的正则表达式匹配，则应将其替换为特定字符串。这需要解释微格式、清理 HTML 标记和规范化数字。它还负责删除管理维基百科类别（如“要清理的文章”）和我们不想处理的文章（如标题为“比较…”的文章)，只需将此材料替换为空字符串。

（4）提取规则：假设源文本的一部分与指定的正则表达式匹配，则应生成一系列事实。这些规则主要适用于维基百科信息框（infobox）中的模式，也适用于维基百科类别、文章标题，甚至源代码中的其他常规元素，如标题、链接或引用。

4．赋予 YAGO 时间维度

·实体和时间

有些实体的存在从一个时间点开始，到另一个时间点结束。例如人从生到死，国家从建立到灭亡。有些实体的存在则是永恒的，例如音乐，科学理论或文学作品。没有必要考虑每个实体类型的时间跨度是否有意义，而是关注以下四种主题类型：

（1）**人**，由关系 wasBornOnDate 和 diedOnDate 划定其存在时间；

（2）**团体**，如音乐乐队、足球俱乐部、大学或公司，它们的存在时间由 wasCreatedOnDate 和 wasDestroyedOnDate 划定其存在时间；

（3）**文物**，如建筑物、绘画、书籍、音乐歌曲或相册，wasCreatedOnDate 和 wasDestroyedOnDate 划定其存在时间；

（4）**事件**，如战争、体育比赛，如奥运会或世界锦标赛，startedOnDate 和 endedOnDate 标定其存在时间。对仅持续一天的事件（如柏林墙倒塌）这类开

始日期和结束日期重合的事件，用 happenedOnDate 处理这些情况。

·事实与时间

（1） 若事实的时间信息是提取出来的

事件也可以有时间维度，例如，事实：BarackObama holdsPoliticalPosition PresidentOfTheUnitedStates 标志着奥巴马从当选到被另一位总统取而代之的时间段。为了获取知识，引入了两个新的关系：occursSince 和 occursUntil，每个关系都有一个具体的事实和一个 yagoDate 实例作为参数。

对于时间只持续一天的事实，使用简写符号 occursOnDate 表示。例如，对奥巴马就职美国总统这一事件，本应写作 occursSince2009-01-20 和 occursUntil2009-01-20，我们将其简写为 occursOnDate 2009-01-20

如果同一个事实发生一次以上，YAGO 将用不同的 id 表示它们。

（2） 若事实的时间信息是推断出来的

在某些情况下，由事实中出现的实体可以推断事实发生的时间。对这些情形的处理方法是，**使用一系列规则，将事实中作为主体或者客体出现的实体的存在时间，推广到事实的发生时间**。为避免出现涉及大量规则的情形，我们将关系分为若干主要类别：永久关系、创造关系、消灭关系。

5. 赋予 YAGO 空间维度

所有实体对象在空间中都有一个位置。对于 YAGO2，我们关注的是在地球空间中具有永久空间范围的实体，例如国家，山脉，河流。在最初的 YAGO 类型层次结构中，这样的实体没有公共的超类。因此，YAGO2 引入了新的类

yagoGeoEntity。它囊括了所有的地理实体。

5.1 获取地理实体

YAGO2 从两个来源获取地理实体。第一个来源是**维基百科**。维基百科包含了大量的城市、地区、山脉、河流、湖泊等，其中许多还带有相关的地理坐标。一个更丰富的免费地理数据来源是：GeoNames (http://www.GeoNames.org)

5.1.1 匹配位置

在处理维基百科文章时，我们尝试匹配各个地理实体。步骤如下：

（1）如果维基百科实体具有类型 yagoGeoEntity，并且恰好与 GeoNames 中的一个实体共享其名称，则将二者匹配。

（2）如果维基百科实体具有类型 yagoGeoEntity，并且在 GeoNames 中与多个实体共享其名称，并且已知维基百科实体的坐标，则将其与地理位置与其最接近的 GeoNames 实体匹配（如果其距离不超过 5km）。否则不进行匹配。

（3）最后，将所有不匹配的 GeoNames 实体作为新的单独实体，以及在 GeoNames 中给出的与其相关的所有事实，添加到 YAGO2 中。

5.1.2 匹配类

孤立的地理位置匹配不足以将 GeoNames 完全集成到 YAGO2 中。因为在 YAGO2 中，每个个体都需要归属于一个类别（type）。幸运的是，GeoNames 为每个地理位置分配了一个类（class），我们可以将其用作类别。为了避免重复类，必须将它们与现有的类相匹配。以前的工作是将所有的 GeoNames 类与 WordNet 类（YAGO2 类层次结构的主干）对齐，最著名的是 geoordnet[13]。然而，GeoWordNet 依靠人工管理来完成正确的匹配。当 GeoNames 或

WordNet 发生更改时，这种方法既费时又脆弱。

为了解决这个问题，设计了一个自动匹配算法。该算法只使用现成的数据，即 YAGO2 类层次结构，以及 YAGO2 类和 GeoNames 类的文本描述。自动匹配的工作原理如下。

（1）对于 GeoNames 中的每个类，我们标记出一组来自 YAGO2 的 WordNet 类集合，这些类与 GeoNames 类名称相同（包括同义的替代名）。

（2）如果没有这样的类，则对 GeoNames 类名进行一个**浅名词短语解析**（shallow noun phrase parsing），以确定**头名词**（head noun）（例如，“mine”代表“gold mine”）。我们在 YAGO2 中搜索名称中包含该头名词的类。

（3）从生成的 YAGO2 类中，删除那些不是 yagoGeoEntity 子类的类，因为我们知道 GeoNames 只包含地理类。

（4）如果只剩下一个类，则将这个类作为匹配类返回。

（5）如果仍然有多个类，则使用 gloss 分别描述 GeoNames 类和 YAGO 类。将 gloss 标记化（tokenized），并在 GeoNames 类的 gloss 和每个候选 gloss 之间计算它们词袋的 Jaccard 相似度。重合度最高的类将作为最佳匹配返回。

（6）如果 glosses 之间没有重叠，将返回 yagoGeoEntity 类，使匹配尽可能泛用。

5.2 指定位置

5.2.1 实体与位置

知识库中有以下类别实体的空间数据：

事件：如战役或体育比赛，发生于一个特定地点。由关系 happendIn 存储。

团体：或有活动场地的组织，如公司总部或大学校园。由关系 isLocatedIn 存储。

文物：目前保存于某特定地点。由关系 isLocatedIn 存储。

这些关系的语义各不相同，并没有对他们分别处理，而是定义了一个新的关系 placedIn 来统一处理这些实体。happenedIn 和 isLocatedIn 作为这个新关系的子属性。YAGO2 为每个实体创建 placedIn 事实，它们可以由知识库推断出来。

5.2.2 事实和地点

有些事实也有空间维度，例如 Leonard Cohen 出生于 1934 年，这一事实发生于蒙特利尔。但并非所有事实都有空间维度，例如 schemal 级别的事实，比如 subclassOf 或标识符关系（如 hasISBN)。我们引入关系 occurdIn，用来建立具象事实和地理实体间的联系。一共有以下三种情形，可推断出本体上有意义的位置。

永久关系：实体之间有直接联系的关系。使用下列两条蕴涵规则，第一条将实体的地点传递给事实，第二条规则，如果实体是一个地理实体 (geo-entity)，则传递其自身：

$$\frac{\begin{array}{c}\$id: \$s\ \$p\ \$o;\\ \$p\ type\ permanentRelation;\\ \$s\ placedIn\ \$l\end{array}}{\$id\ occursIn\ \$l}$$

$$\frac{\begin{array}{c}\$id: \$s\ \$p\ \$o;\\ \$p\ type\ permanentRelation;\\ \$s\ type\ yagoGeoEntity\end{array}}{\$id\ occursIn\ \$s}$$

空间约束关系（space-bound relations)：有些事实发生地点可由其主语或者宾语表示。对此，引入了两个新的类来描述这种关系，

relationLocatedByObject 和 relationLocatedBySubject，它们都是 yagoRelation 的子类。

串联关系（tandem relations）：有些关系是同时发生的，一个关系决定另一个关系的位置。例如，关系 wasBornOnDate 定义了对应 wasBornIn 的时间，后者定义了前者的位置。使用两个关系之间的关系 timeToLocation 来表示这种情况。第一个关系表明事件的时间，第二个关系表明事件的地点。

6. YAGO2 中的文本（上下文）数据

YAGO2 不仅包含事实和实体的时间和位置，还包含有关实体的元信息，包括来自维基百科的非本体数据以及多语言数据。

6.1 来自维基百科的非本体数据

对于每个实体，YAGO2 都包含上下文信息。上下文由我们的提取工具收集自维基百科。它们包括以下关系，以实体和字符串作为参数：

hasWikipediaAnchorText：将一个实体与一个字符串相连接，该字符串在实体文章中作为锚文本（anchor text）

hasWikipediaCategory：将一个实体与一个类别名相连接，该类别名用分类存放维基百科文章。

这些关系都是关系 hasContext 的子属性。

6.2 多语言信息

对于单个实体，我们从维基百科文章中**不同语言间链接**（interlanguage links）中提取不同语言的翻译，这使得 YAGO2 支持多语言查询。YAGO2 将非英语实体名称用**具象化事实**（reified facts）表示。

这种技术适用于 YAGO2 中的个体，但不适用于类，因为 YAGO2 的分类是

从英文的 WordNet 中提取的。为了填补这个空白，将 **Universal WordNet (UWN)** 集成到 YAGO2 中。UWN 将 WordNet 中的单词和词义映射到它们的正确翻译。

7 SPOTL(X) 表示

SPOTL(X)-视图模型

为了使浏览和查询更加方便，视图应展示一个扩展五元组，其中每个事实已经包含其关联的时间和空间信息。这种数据视图称为为 SPOTL 视图：由 Time 和 Location 扩展的 SPO 三元组（w: **SPO** triples augmented by **T**ime and **L**ocation)。基于第 4，5 节介绍的工作，我们的知识库现在包含用于基本事实的定义良好的时间和空间信息，这种情况简化了 SPOTL (X) 视图的构建。具体而言，它由以下虚拟关系组成：

R(Id,S,P,O) – 所有知识库中的(id,s,p,o)元组。

T(Id,TB,TE) – 所有与时间区间[tb,te] 相关联，且含有由 id 标识的事实的(id,tb,te)元组。

L(Id,LAT,LON) – 所有与地点<lat,lon> (经纬度) 相关联，且含有由 id 标识的事实的(id,lat,lon)元组.

X(Id,C) – 所有与上下文 c 相关联，且含有由 id 标识的事实的(id,c)元组。

基于上述模块，将 SPOTL(X)视图定义为：

$$\pi_{[R.Id,[TB,TE],<LAT,LON>,C]}(((R \rtimes_{[Id=Id]} T) \rtimes_{[Id=Id]} L) \rtimes_{[Id=Id]} X)$$

连接了来自 R 的事实，以及来自 T, L 和 C 的它们的关联信息。$\rtimes$表示外部连接，以避免丢失那些没有时空或上下文事实的元组，并再对应领域产生空值 NULL。

SPOTL(X) 查询

为了处理时间，空间，上下文等维度，让它们的隐含语义能被用户所访问，引入谓语。时间谓词是由 Allen[16]提出的一系列谓词的子集。空间谓词是能反映两地相对位置，可以测试两地间的地理距离是否小于一个给定的阈值。matches 谓词用于上下文维度，测试上下文是否匹配一个给定的关键词查询。

查询语句可以从每个维度中添加一个谓词到每个三元组模式里。模式可以最多包含 6 个参数。感兴趣的时间或者地点往往不是显式已知的，但与实体相关联。再 SPOTL(X)的查询界面中，时间和空间能通过一个关联实体隐含说明，提升了查询便利性。

8. 事实评估

仅有小部分数据可以通过对比 YAGO2 和先验事实评估准确性，即 GeoWordNet 将 GeoNames 类匹配到 WordNet 语法集的这部分。对于其他事实，仅能依靠人工进行评估。

9. 基于任务的评估

用两个示例性任务评估最新的可用数据和查询功能。

（1） 任务 1 用新的 YAGO2 特性来回答时间或者空间属性方面的问题。这个任务展示了 SPOTL(X)查询语言的简洁和实用性，以及用于回答关于时间，空间，上下文数据的复杂问题时的可用性。

（2） 在任务 2 中，新特性用于自然语言文本中命名实体的消歧义效果增强的任务。