浙江大学

本科生毕业设计
开题报告

学生姓名：___周自强_____

学生学号：3120101943

指导教师：陈华钧

年级与专业：大四 计算机科学与技术

所在学院：计算机学院

一、题目：基于大规模知识图谱的规则挖掘系统的实现

二、指导教师对开题报告、外文翻译和中期报告的具体要求：

指导教师（签名）_____

年　　月　　日

毕业设计开题报告、外文翻译和中期报告考核

导师对开题报告、外文翻译和中期报告评语及成绩评定：

成绩比例	开题报告 占（20%）	外文翻译 占（10%）	中期报告 占（10%）
分 值			

导师签名_____

年　　月　　日

答辩小组对开题报告、外文翻译和中期报告评语及成绩评定：

成绩比例	开题报告 占（20%）	外文翻译 占（10%）	中期报告 占（10%）
分 值			

开题报告答辩小组负责人（签名）_____

年　　月　　日

-

目录

本科毕业设计开题报告

1. 项目背景

最近几年，许多知识库例如 Cyc、YAGO、DBpedia、Freebase 兴起。大规模知识库在各种应用中例如自然语言问答、语义搜索引擎等等都有作用。这些知识库由数百万的世界实体以及它们之间的关系所组成，以有向图的形式存储，结点表示实体，连接表示实体间的关系。尽管这样的知识库包含成千上万的实体，它仍然是稀疏的，即它在实体之间缺失了大量的关系。在大规模知识库中存储的实体（entity），包括了人、国家、河流、城市、大学、电影、动物等等信息；还有许多事实（fact），例如：伦敦是英国的首都、每一个歌手都是人类等等。从知识库中可以知道，谁出生在哪里，一个演员演了哪几部电影，或者是一个城市在哪个国家。现今的知识库包括了成千上万的实体（entity），以及更多的事实（fact）。

然而知识库并不是完整的。这时候，我们需要从已有的知识库中进行规则挖掘，进而找寻两个实体之间的关系，使得知识图谱密集化。例如一条规则：livesIn(h, p) ^ marriedTo(h, w) => livesIn(w, p)，它表示如果两个人是夫妻关系，那么他们（通常）住在同一个城市。

挖掘出这样的规则，我们能够利用它做什么？首先，我们可以进行知识库的补全，例如，我们知道奥巴马住在哪里，并且知道他的妻子是米歇尔，那么，我们通过挖掘出的规则，就可以知道米歇尔住在哪里；第二，这些规则可以找出知识库中一些潜藏的错误信息，例如知识库中说奥巴马的妻子米歇尔居住在中国，我们就能够判断这条信息很有可能是错误的；第三，规则可用于推理。最后一点，这些规则可以使我们更好地理解整个知识库，我们可以发现使用相同语言国家之间贸易往来频繁、婚姻是一种对称的关系等等。

2. 目标和任务

目标：实现一个规则挖掘的系统，分析大规模知识图谱，进行规则挖掘，通过挖掘出的规则进行知识库的补全。

具体分为以下几个部分：

1. 以 AMIE 为基础，分析大规模知识图谱，在知识图谱中实现封闭规则挖掘。
2. 在 AMIE 的基础上编写代码，进一步完善规则挖掘算法的细节，并使用提供的知识库数据进行挖掘测试。
3. 提供对中文知识图谱规则挖掘的支持。
4. 由于 AMIE 对于封闭霍恩规则的挖掘已经有了很好的支持，我们的工作就是在 AMIE 的基础上进行拓展。在封闭规则挖掘的基础上，进一步进行非封闭的规则挖掘。

3. 可行性分析

大规模知识库的规则挖掘这一课题有一定难度，但基本可行，原因主要基于以下几个方面：

1. 在大规模知识图谱中，存在成千上万的事实（fact），从这些已有事实中我们可以找到一些规则，用来描述这些事实。而通过挖掘出来的规则，我们可以进一步完善知识库。因此，对知识库进行相应的规则的挖掘是可行的。
2. 关联规则是反映一个食物与其他事物之间的相互依存和关联性，是数据挖掘中的一个重要技术。现在有许多关于关联规则挖掘的算法，例如 AMIE 在不完整知识库中的关联规则挖掘，或者使用桥接实体以及随机游走算法来进行规则的挖掘。并且 AMIE 已有相应的代码来实现关联规则挖掘的部分功能。由此可以看出，在知识库中进行关联规则的挖掘并不是不可能的。
3. 规则挖掘的算法越来越成熟，实际应用也越来越广泛，越来越多的人在研究学习规则挖掘，并提供了许多高质量的开源代码，这对于我们实现关联规则挖掘有很大的帮助。
4. 网络上提供了许多知识库数据例如 YAGO、DBpedia 的数据集用于测试，这对于我们实现并调试规则挖掘算法提供了便利。

4. 初步技术方案和关键技术考虑

初步技术方案：

AMIE 是一个数据规则挖掘系统，它能够实现从一个知识库中提取出逻辑规则的置信度等信息，从而进行规则的挖掘。我们在 AMIE 的基础上进行规则挖掘系统的开发。

1. 首先我们要了解一些相关概念：

RDF 知识库：

知识库中的一个 fact 可以使用一个三元组来表示，形式<x, r, y>，其中 x 为主语（subject），r 为关系（relation or predicate），y 为宾语（object）。这里我们表示成 r(x, y)。

函数（function）：

在关系 r 中，对于每一个 subject，最多只有一个 object 阈值对应，这样的关系称为函数（function）。另外，我们使用标记 functionality，关系 r 的 functionality 是一个 0 到 1 的值，当 r 是一个 function 的时候值为 1。

$$\mathrm{fun}(\mathrm{r}) \coloneqq \frac{\#x: \exists y: r(x,y)}{\#(x,y): r(x,y)}$$

规则（rule）：

形式如$\mathrm{B1} \wedge \mathrm{B2} \wedge \ldots \wedge \mathrm{Bn} \Rightarrow \mathrm{r(x,y)}$，缩写为$\vec{B} \Rightarrow \mathrm{r(x,y)}$

其中 B1、B2…为 atom，r(x,y)为 head，$\mathrm{B1} \wedge \mathrm{B2} \wedge \ldots \wedge \mathrm{Bn}$为 body。

Support：

对于一个规则，需要在知识库中有对应的 fact 来支持。Support 表示在 head 中不同的 subject 与 object 对的数量。

$$\mathrm{supp}(\vec{B} \Rightarrow \mathrm{r(x,y)}) \coloneqq \#(\mathrm{x,y}): \exists z1, \ldots, zm: \vec{B} \wedge \mathrm{r(x,y)}$$

其中 z1,…zm 是除了 x、y 之外的变量。

Head Coverage：

由于 support 是一个绝对数字，在这里我们使用 head coverage 来表示一个相对值：

$$\mathrm{hc}\left(\vec{B} \Rightarrow \mathrm{r(x,y)}\right) \coloneqq \frac{\mathrm{supp}(\vec{B} \Rightarrow \mathrm{r(x,y)})}{size(r)}$$

其中 size(r):=#(x',y') : r(x',y')，表示关系 r 对应的 fact 的数量。

标准置信度（standard confidence）：

$$\mathrm{conf}\left(\vec{B} \Rightarrow \mathrm{r(x,y)}\right) \coloneqq \frac{\mathrm{supp}(\vec{B} \Rightarrow \mathrm{r(x,y)})}{\#(\mathrm{x,y}): \exists z1, \ldots, zm: \vec{B}}$$

PCA 置信度（partial completeness assumption confidence）

$$\mathrm{conf}_{pca}\left(\vec{B} \Rightarrow \mathrm{r(x,y)}\right) := \frac{\mathrm{supp}(\vec{B} \Rightarrow \mathrm{r(x,y)})}{\#(\mathrm{x,y}): \exists z1, \ldots, zm, y': \vec{B} \wedge \mathrm{r(x,y')}}$$

2. AMIE 规则挖掘算法

输入知识库 KB，阈值 minHC，规则的最大长度 maxLen，最小置信度 minConf。输出规则。

首先获取一个规则队列，初始包含所有的 head atom，且 size 为 1。

然后每次从队列中获取一个 rule，如果这个 rule 满足一定条件，则添加到输出队列；如果该 rule 的长度不超过最大长度 maxLen，则对该 rule 进行进一步完善，并将完善后的满足条件的 rule 添加到规则队列中。

伪代码：

```
function AMIE(KB K, minHC, maxLen, minConf)
    q = [r1(x, y), r2(x, y)…rm(x, y)]
    out = < >
    while ¬ q.isEmpty() do
        r = q.dequeue()
        if AcceptedForOutput(r, out, minConf) then
            out.add(r)
        end if
        if length(r) < maxLen then
            R(r) = Refine(r)
            for all rules rc∈R(r) do
                if hc(rc) ≥ minHC & rc ∉ q then
                    q.enqueue(rc)
                end if
            end for
        end if
    end while
    return out
end function
```

3. 进一步细化，对队列中的规则进一步处理，直到满足相应的输出条件：

对规则的细化过程主要包含下面三个 atom 的添加：

a. Dangling atom

往 rule 中添加一个新的 atom，该 atom 使用了一个新的变量，另一个变量为规则中的一个已有变量。

b. Instantiated atom

往 rule 中添加一个新的 atom，该 atom 一个变量为一个实体（entity），另一个变量为规则中的一个已有变量。

c. Closing atom

往 rule 中添加一个新的 atom，该 atom 使用的变量为规则中的已有变量。

4. 算法的优化，包括算法效率的优化，对中文的支持。
5. 增加非封闭规则挖掘的算法，挖掘出更多规则。
6. 获取已有的一些知识库，例如 Yago、DBpedia 的知识库，进行这些知识库进行规则挖掘，进行统计分析。

关键技术：

1. 设计方法查找满足条件的 relation，用以下形式表示（Projection Query）：

   SELECT r COUNT(H)

   WHERE H∧B1∧…∧Bn-1∧r(X,Y)

   SUCH THAT COUNT(H) ≥ k

   由于这些 query 在算法中会大量出现，因此对它的优化是一个很关键的问题。
2. 设计方法实现 Dangling atom、Instantiated atom、Closing atom 三种 atom 的查找。这是 closed Horn rule 挖掘的关键步骤，实现好了这几个方面，可以说基本上完成了算法的大部分工作。
3. 多线程速度优化。
4. 队列规则细化过程的优化，例如通过设置阈值限制规则的最大长度，或者在找到 Perfect rule（PCA confidence≥100%）时停止添加 atom，优化 Projection Query 等。
5. AMIE 不支持对于非封闭霍恩规则挖掘，因此要实现它，需要进一步查询文献，获取一些理论支持，寻找一些较好的算法。

5. 预期工作结果

预期工作结果：在 AMIE 的基础上实现一个规则挖掘的系统，分析大规模知识图

谱，进行规则挖掘，通过挖掘出的规则进行知识库的补全。

具体细节如下：

1. 以 AMIE 为基础，分析大规模知识图谱，在知识图谱中实现封闭规则挖掘。
2. 在 AMIE 的基础上编写代码，进一步完善规则挖掘算法的细节，并使用提供的知识库数据进行挖掘测试。
3. 提供对中文知识图谱规则挖掘的支持。
4. 由于 AMIE 对于封闭霍恩规则的挖掘已经有了很好的支持，我们的工作就是在 AMIE 的基础上进行拓展。在封闭规则挖掘的基础上，进一步进行非封闭的规则挖掘。

6. 进度计划

3 月 1 日~3 月 15 日：熟悉 AMIE，了解 closed Horn 规则挖掘算法。

3 月 16 日~3 月 31 日：熟悉 AMIE 代码，阅读相关论文，了解 AMIE 的整体架构与算法实现。

4 月 1 日~4 月 7 日：获取一些知识库，使用 AMIE 进行知识库中规则的挖掘，将结果进行统计对比。

4 月 8 日~3 月 15 日：在 AMIE 的基础上进行系统开发，实现 closed Horn 规则挖掘。

4 月 16 日~4 月 23 日：编写代码，进一步完善规则挖掘算法的细节，并使用提供的知识库数据进行挖掘测试。

4 月 24 日~4 月 30 日：优化算法，使用多种方法提高算法效率，进行规则挖掘，并进行结果分析与对比。

5 月 1 日~5 月 7 日：在实现基本的规则挖掘功能的基础上，提供系统对中文知识图谱规则挖掘的支持。

5 月 8 日~5 月 12 日：学习非封闭规则挖掘的相关知识，获取理论基础。

5 月 13 日~5 月 23 日：在实现 closed Horn 规则挖掘的基础上，进一步进行非封闭的规则挖掘，并进行数据测试与比较。

5 月 24 日~5 月 28 日：完善一些细节，修复各类 bug，完成毕业设计。

本科毕业设计外文翻译

使用桥接实体对知识库进行推理

Bhushan Kotnis
Indian Institute of Science
bkotnis@dese.iisc.ernet.in

Pradeep Bansal
Indian Institute of Science
pradeepb@ee.iisc.ernet.in

Partha Talukdar
Indian Institute of Science
ppt@serc.iisc.in

摘要

大规模知识库（例如 NELL，Yago，Freebase，等等）通常较为稀疏，即在实体间的大量合法关系丢失。最近的研究解决了这个问题。在知识图谱的结点不变的情况下，通过给知识图谱添加额外的从大型文本语料库中挖掘出的边，然后使用路径排序算法（Path Ranking Algorithm，PRA）来对这个扩展的图谱进行知识库推导。在这篇论文中，我们通过不仅仅向知识图谱中添加边，还添加桥接实体，两者都是从一个 500 000 000 大小的网络文本语料库中挖掘出来的。通过对真实世界的实际数据集合进行实验，我们证明了桥接实体在提高性能以及降低知识库推导中 PRA 算法的运行时间中的价值。

1 介绍

大规模知识库像 Freebase（Bollacker 等，2008），Yago（Suchanek 等，2007），NELL（Mitchell 等，2015）在各种应用中例如自然语言问答、语义搜索引擎等等都有作用。这些知识库由数百万的世界实体以及它们之间的关系所组成，以有向图的形式存储，结点表示实体，连接表示实体间的关系。尽管这样的知识库包含成千上万的实体，它仍然是稀疏的，即它在实体之间缺失了大量的关系（West 等，2014）。

在知识图谱中进行推导，找寻两个实体之间的关系，是一种使知识图谱密集化的一种方式。例如，从(Germany, playTournament, FIFA)和(FIFA, tournamentofSport, Soccer)，我们可以推出(Germany, playSport, Soccer)。路径排序算法（PRA）（Lao 和 Cohen，2010）通过在知识图谱上学习推导规则来进行推导。

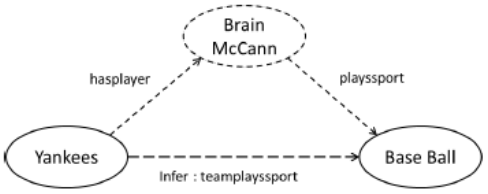

图 1：例子显示了桥接实体 Brian McCain，和两条边可以帮助 PRA 算法（Lao 和 Cohen，2010）推断出缺失的关系teamplayssport（Yankees, BaseBall）。原始知识图谱值包含两个结点Yankess 和Baseball，且没有边。

如果只是图谱是稀疏的，即在源实体与目标实体之间只有很少或者没有路径，那么 PRA 无法预测一个关系的存在性。为了解决这个缺陷，（Lao 等，2012）给知识图谱添加从一个外部语料库中获取的路径。添加的路径由从解析过从属外部语料库中得到的非词汇化的从属标签组成。为了增强添加的路径的表现性，（Gardner 等，2013）向知识图谱中添加谓语（表面关系），而不是非词汇化标签。谓词来自于超过 6 亿主语-谓语-宾语（Subject-Verb-Object，SVO）三元组。这些谓语以边的形式连接那些原本不连接的实体，于是增加了知识图谱的连接性，由此潜在地提高了 PRA 算法的性能。

然而，简单地添加这些边提高了稀疏性，降低了 PRA 中逻辑回归分类的判别能力。这个可以通过添加对表面关系聚类获取到的潜在关系来解决，而不是直接添加表面关系。这减少了稀疏型，并且已经被证明可以提高 PRA 推导能力（Gardner 等，2013），（Gardner 等，2014）。

在这篇文章中，我们提出一种方案，通过挖掘来自外部语料库的连接两个 SVO 三元组的名词短语来获取路径，以此扩展知识库。我们把这种名词短语叫做桥接实体，它桥接了两个知识库中的关系，形成了一条路径。这跟方案（Gardner 等，2013）与（Gardner 等，2014）不同，它通过从外部语料库中挖掘表面关系并向知识库结点中添加边。我们通过使用随需应变的方式在语料图中实现深度 DFS（深度优先搜索），在语料库中搜索这样的桥接实体。

我们将这个过程称作按需扩展（On-Demand Augmentation，ODA），因为搜索可以按照需求的方式在测试时间完成。相比之下，之前添加边或者嵌入知识库的方式（Gardner 等，2013），以及向量空间随机游走 PRA（Gardner 等，2014）是批处理程序。正如我们将要在第四部分看到的，由于有限的搜索空间，按需扩展比算法（Gardner 等，2013；Gardner 等，2014）更快。另外，由于边不是盲目添加的，按需扩展不增加稀疏性（致使性能下降的原因）。我们的实验表明，ODA 比（Gardner 等，2013）提供了更好的性能和与（Gardner 等，2014）几乎相同的预测能力，但是在这两种情况下，由于其在线和按需性质，增加了更快的运行时间与更大的灵活性的优势。代码以及结果可以从 https://github.com/malllabiisc/pra-oda 获取。

2 相关工作

使用表层关系和名词短语来提取出有意义的关系事实并不是一个新的想法（Hearst，1992）（Brin，1999）（Etzioni 等，2004）。然而，他们没有使用知识库来提高信息抽取。

第一次在（Lao 和 Cohen，2010）提出的路径排序算法（PRA）被使用在一个知识库推导（Lao 等，2011）。它被（Lao 等，2013）扩展，提高通过向知识库中添加从解析过的从属语料库中获取的语法信息。扩增知识库，使用从一个外部语料库中挖掘的表面关系与在表面关系中使用 PCA 获取的潜在的边标签，提高 PRA 推导，在（Gardner 等，2013）中探讨了。（Gardner 等，2014）采用了向量空间“软”映射，取代了表面关系到潜在关系的的硬映射。这允许了随机游走更频繁地遍历一条当前边类型的边。

尽管，像其他人一样，我们使用一个外部语料库来扩增知识库，在我们的方法中，关键的区别是，除了添加表面关系，我们也添加桥接实体，使我们能在知识库中添加新的路径。此外，改程序是有针对性的，因此，只有那些在推断关系中起到作用的路径会被添加。这样，用这种方式添加的路径的数量远远低于使用程序（Gardner 等，2013）添加的表面关系的数量。正如我们在第四节所见，这将导致更有效的算法和更快的运行时间。

3 方法

3.1 背景：路径排序算法（PRA）

本文首先简要概述了路径排序算法（PRA）（Lao 和 Cohen，2010）。PRA 使用路径特征的逻辑回归分类，预测给定的关系是否存在于一对实体之间。对于一对给定的实体 s 和 t，路径类型连接了 s 和 t，形成了向量特征。一个路径类型是一个有序的关系集合，具有相同有序关系但不同的中间或者末端实体的路径属于同一路径类型。例如，$s_1 \xrightarrow{v_0} x_1 \xrightarrow{v_1} t_1$和$s_2 \xrightarrow{v_0} x_2 \xrightarrow{v_1} t_2$属于路径类型$\xrightarrow{v_0} \xrightarrow{v_1}$。这个特征的值为 $P(s \to t;\pi)$，$P(s \to t;\pi)$是通过遍历路径π从 s 到 t 的可能性。PRA 通过在知识库中运行随机游走（RW）来近似得到这个可能性。令 $F=\{\pi_1, \pi_2, \dots, \pi_k\}$为所有路径类型的集合。为了预测实体 s 和 t 之间的关系 r 的存在性，逻辑回归分类输出一个分数值用来度量 s 和 t 之间存在关系 r 的置信度。它通过在训练阶段第一次分配权重给这项特征。分数值由一下公式给出：

$$S(s, t, r) = \sum_{\pi \in F} P(s \to t;\pi) \times \theta_\pi^r \qquad (1)$$

其中θ_π^r是权重，是在训练过程中，由逻辑回归分类学习关系 r 和路径类型π得到的。在测试阶段，由于目标不可用，PRA 收集候选目标进行随机游走，然后计算出向量特征，得到分数值。

3.2 PRA-SVO 和 PRA-VS

查询	候选答案	包括桥接实体（粗体表示）的路径
sportsteamPositionForSport(right handed pitcher, ?)	baseball	Right handed pitcher $\xrightarrow{\text{playsfor}}$ **Chicago Cubs** $\xrightarrow{\text{play}}$ baseball
riverFlowsThroughCity(Moselle, ?)	Koblenz	Mosell $\xrightarrow{\text{flows into}}$ **Rhine** $\xrightarrow{\text{meet at}}$ Koblenz
teamPlaysInLeague(Cleveland Indians, ?)	MLB	Cleveland Indians $\xrightarrow{\text{play}}$ **Detroit Tigers** $\xrightarrow{\text{blew}}$ MLB

表 1：使用 PRA-ODA 将包括桥接实体（粗体表示）的路径添加到知识库中。

PRA-SVO 和 PRA-VS 分别是（Gardner 等，2013）和（Gardner 等，2014）提出的系统，在知识库中添加了从一个大型主语-谓语-宾语（SVO）三元组语料库中挖掘的边。在这两个系统中，只有新的边被添加到固定的结点集合，这个添加过程是一个离线的批处理程序。相反，PRA-ODA，论文中提到的方法，还可以通过桥接实体扩展结点集，并按需进行扩展。

3.3 PRA 按需扩展（PRA-ODA）

训练：令 s 和 t 为任意两个知识库实体，$s^{(n)}$和$t^{(n)}$为他们对应的名词短语表示或别名。我们通过有限深度优先搜索（DFS）来搜索桥接实体$x_1, x_2, \dots, x_n$，从s^n开始，这样我们获取了一条路径s $\xrightarrow{ALIAS} s^{(n)} \xrightarrow{v_0} x_1 \xrightarrow{v_1} \dots \xrightarrow{v_{n-1}} x_n \xrightarrow{v_n} t^{(n)} \xrightarrow{ALIAS} t$，其中$\mathrm{v}_i$是在语料库图谱中出现的谓语。这些是在 n≤ $\mathrm{d}_{max} - 1$的条件下，d_{max}是 DFS 的最大深度。我们在知识库实体中添加“别名”边和它的名词表示。桥接实体的实用性如图 1 所示。

我们从一个来自 ClueWeb09（Callan 等，2009）的使用 MALT（Niver 等，2007）解析的超过 6 亿 SVO 三元组的语料库中挖掘桥接实体。我们使用 Mongo DB 来存储这些三元组，为一个邻接表。在训练期间，推断出的任何关系，源和其对应的目标实体是已知的。有限深度的 DFS 运行在所有深度小于d_{max}的 SVO 图中，使用主语实体的别名作为开始点。这样的别名在 NELL 和 Freebase 知识库中是可用的。如果路径的终端实体匹配任何目标实体的别名，DFS 就发现一条路径。我们选择使用别名来进行字符串匹配，是因为它容易通过简单地添加更多的别名来改变这种柔和度。这对于所有训练中的源目标对都是完成好的。表 1 中显示了添加路径的几个例子。

由于 SVO 图是通过解析从网络抓取到的 ClueWeb 语料库来获取的，因此它是嘈杂的。为了降低它的嘈杂度，我们添加 SVO 中发现的最频繁的 K 个路径类型，其中 K 是一个可调的参数。通过 SVO 路径类型，我们从 SVO 语料库中挖掘出有序谓语集合。有一种可能性是，从语料库中获取的桥接实体可能在知识库中存在。如果这个桥接实体匹配了任何别名，那么它被视作一个已存在的知识库实体的别名。如果不是，这个桥接实体被添加到知识库中作为新的实体。为了避免过度拟合，我们在训练集合中添加负面数据。另外，只有高质量的有表现力的桥接实体能够得到有意义且有判别力的路径。尽管桥接实体的质量依赖于语料库，低质量的桥接实体可以通过添加负面训练数据来过滤。低质量的桥接实体从正面和负面训练集合来连接源目标对，因此它被稀疏逻辑回归分类所去除。负面数据集是使用随机游走过程的封闭世界假设所产生的。

在知识库扩增之后，我们运行 PRA 算法的训练阶段来获取（路径）权重特征，它是由逻辑回归分类计算出的。

查询时间：对应于一个源实体和一个正在被预测的关系的目标实体集在查询（测试）时间内并不可用。我们使用包括在关系范围内的被预测为候选目标实体的所有实体。例如，关系是

KB 关系	PRA	PRA-SVO	PRA-VS	PRA-ODA
actorstarredinmovie	0.0	**1.0**	**1.0**	**1.0**
atheleteplaysforteam	1.0	1.0	1.0	1.0
citylocatedincountry	0.166	0.25	**1.0**	**1.0**
journalistwritesforpublication	1.0	1.0	1.0	1.0
riverflowsthroughcity	0.333	0.25	**1.0**	**1.0**
sportsteampositionforsport	1.0	1.0	1.0	1.0
stadiumlocatedincity	1.0	1.0	1.0	1.0
statehaslake	0.0	0.0	0.0	0.0
teamplaysinleague	1.0	1.0	1.0	1.0
writerwrotebook	1.0	1.0	1.0	1.0
平均（MRR）	0.649	0.75	**0.9**	**0.9**

表 2：NELL 中 10 个关系的 MRR 比较（值越大越好）。PRA-SVO，PRA-VS 是（Gardner 等，2013；Gardner 等，2014）提出的系统。PRA-ODA 是本文中提出的方法。与 PRA-SVO 上相比，PRA-ODA 的改善 $p < 0.007$ 有着统计学意义。

riverFlowsThroughCity，候选目标集将包括知识库中的城市实体。在训练期间，DFS 从源实体开始执行，但是这一次只限制路径为训练期间学习到的正权值。任何在这次搜索中找到的路径（连同桥接实体）被添加到知识库中，且 PRA 算法正运行于这个扩增的图谱中。

4 实验

我们使用（Gardner 等，2014）的作者提供的 PRA 算法的实现。在我们的实验中，我们使用相同的 10 个 NELL 关系数据，就像在（Gardner 等，2014）中。这个扩增在训练时期增加了 1086 条路径，在测试时期增加了 1430 条路径。

我们将 NELL 数据分为 60%训练数据，15%开发数据和 25%测试数据。d_{max}和最频繁路径 K 的值是通过在一个开发集上调整 4 个关系来获得的（athleteplaysforsport，actorstarredinmivie，citylocateincountry 和 journalistwritesforpulication）。超参数

时间（秒）	PRA	PRA-SVO	PRA-VS	PRA-ODA
训练	635.6	574.5	564.2	913.3
测试	354.3	322.0	301.2	436.7
批扩增	n/a	797	797	n/a
嵌入计算	n/a	n/a	812	n/a
总时间	989.9	1693.5	2474.4	1350

表 3：整个实验的运行时间比较（值越低越好）。PRA-SVO，PRA-VS 是（Gardner 等，2013；Gardner 等，2014）提出的系统。PRA-ODA 是本文中提出的方法。在两个系统 PRA-ODA 和 PRA-VS 中，PRA-ODA 快了 1.8 倍。

d_{max}=2，K=10 具有最高的 MRR，并给剩下的关系使用。在逻辑回归分类中的L_1和 L_2 正则化参数，我们使用与（Gardner 等，2013；Gardner 等，2014）相同的值，即 $L_1 = 0.005$，$L_2 = 1.0$。这是因为参数是稳健的，并能够很好地工作，即使是在知识库扩增的情况下。

我们比较（PRA-ODA）和 PRA 算法在 NELL 知识库上运行 的结果，NELL 知识库使用表面关系扩增（PRA-SVO）（Garnder 等，2013），和 PRA（PRA-VS）（Garnder 等，2014）向量空间随机游走时间。运行时间，即执行一整个 PRA-SVO 和 PRA-VS 实验，包括使用 SVO 边来扩增 NELL 知识库的时间。PRA-VS 运行时间也包括了生成嵌入的时间，用来执行向量空间随机游走。正如在表 2 和表 3 中所见，我们的方案，PRA-ODA，提供的性能相当于 PRA-VS，具有更快的运行时间（加快 1.8）。除了全部 SVO 扩增时间，PRA-VS 花费额外时间来从添加的谓语中生成嵌入（13 分钟）。我们注意到，在 PRA-SVO 和 PRA-VS 的批扩增，且以 PRA-VS 嵌入计算都是在特定评价集中的关系，这样就变成不能忽视的一个一次性线下成本。换句话说，这些成本可能会增加更多的关系（和它们的实例），包括训练和测试。在这样的设置下，PRA-ODA 运行时间会更加明显地增长。

该算法的另外一个优点就是，它可以运行在任何以 PRA 为基础的算法上，例如

PRA-SVO 和 PRA-VS。

5 结论

在这篇论文中，我们调查往知识库中添加路径的有用性，通过从外部语料库中挖掘桥接实体提高其连接性。虽然以前的知识库扩增方法只专注于使用挖掘出的表面谓语同时保持结点不变，我们扩展了这些方法，通过在线的方式添加桥接实体。我们使用一个 5 亿大小的大型网络文本语料库来挖掘这些添加的边和桥接实体。尽管在实际数据集中进行实验，我们证实了提出的方法不仅仅是相当或优于其他最先进的方法，而且更重要的是提供了比其他方案更快的整体运行速度。

致谢

这项工作由 Google 提供一定程度上的支持。

参考文献

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In Paolo Atzeni, Alberto Mendelzon, and Giansalvatore Mecca, editors, The World Wide Web and Databases, volume 1590 of Lecture Notes in Computer Science, pages 172–183. Springer Berlin Heidelberg.

J. Callan, M. Hoy, C. Yoo, and L. Zhao. 2009. Clueweb09 data set. boston.lti.cs.cmu.edu.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In Proceedings of the 13th International Conference on World Wide Web, WWW '04, pages 100–110, New York, NY, USA. ACM.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 833–838.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 397–406.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th Conference on Computational Linguistics - Volume 2, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination ofpath constrained random walks. Machine Learning, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in

a large scale knowledge base. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLPCoNLL '12, pages 1017–1026, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. Mitchell,W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In Proceedings of AAAI.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, G¨ulsen Eryigit, Sandra K¨ubler, Svetoslav
Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, 13(02):95–135.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In Proceedings of the 16th International Conference on World Wide Web, WWW'07, pages 697– 706, New York, NY, USA. ACM.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In Proceedings of the 23rd International Conference on World Wide Web, WWW '14, pages 515–526, New York, NY, USA. ACM.

外文原文

Knowledge Base Inference using Bridging Entities

Bhushan Kotnis
Indian Institute of Science
bkotnis@dese.iisc.ernet.in

Pradeep Bansal
Indian Institute of Science
pradeepb@ee.iisc.ernet.in

Partha Talukdar
Indian Institute of Science
ppt@serc.iisc.in

Abstract

Large-scale Knowledge Bases (such as NELL, Yago, Freebase, etc.) are often sparse, i.e., a large number of valid relations between existing entities are missing. Recent research have addressed this problem by augmenting the KB graph with additional edges mined from a large text corpus while keeping the set of nodes fixed, and then using the Path Ranking Algorithm (PRA) to perform KB inference over this augmented graph. In this paper, we extend this line of work by augmenting the KB graph not only with edges, but also with *bridging entities*, where both the edges and bridging entities are mined from a 500 million web text corpus. Through experiments on real-world datasets, we demonstrate the value of bridging entities in improving the performance and running time of PRA in the KB inference task.

1 Introduction

Large-scale knowledge bases (KB) like Freebase (Bollacker et al., 2008), Yago (Suchanek et al., 2007), NELL (Mitchell et al., 2015) can be useful in a variety of applications like natural language question answering, semantic search engines, etc. These knowledge bases consist of millions of real world entities and relationships between them which are stored in the form of a directed graph where links represent relations and nodes represent the entities. Although such KBs contain millions of entities, they are still very sparse, i.e., they are missing a large number of relations between existing entities (West et al., 2014).

Performing inference over the knowledge graph for predicting relations between two entities is one way of densifying the KB graph. For example, from *(Germany, playsinTournament, FIFA)* and *(FIFA, tournamentofSport, Soccer)*, we can infer *(Germany, playsSport, Soccer)*. The Path Ranking Algorithm (PRA) (Lao and Cohen, 2010), (Lao et al., 2011) performs such an inference by learning inference rules over the knowledge graph.

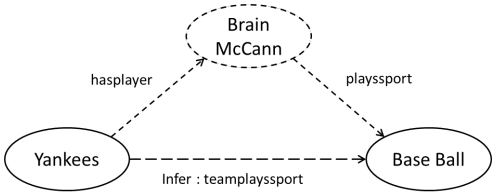

Figure 1: Example showing how addition of the *bridging entity*, *Brian McCain*, and the two edges incident on it can help the PRA algorithm (Lao and Cohen, 2010) to infer the initially missing relation instance *teamPlaysSport(Yankees, BaseBall)*. The original KB graph consisted only of two nodes, *Yankees* and *Baseball*, and no edges.

If the knowledge graph is sparse, i.e., if there are a very few or no paths between source and target entities, then PRA is unable to predict the existence of a relation. To address this shortcoming, (Lao et al., 2012) augmented the knowledge graph with paths obtained from an external corpus. The added paths consisted of unlexicalized dependency labels obtained from a dependency parsed external corpus. To improve the expressivity of the added paths, instead of the unlexicalized labels, (Gardner et al., 2013) augmented the KB graph with verbs (surface relations) from a corpus containing over 600 million Subject-Verb-Object (SVO) triples. These verbs act as edges that connect previously unconnected entities thereby increasing the connectivity of the KB graph which can potentially improve PRA performance.

However, naïvely adding these edges increases the feature sparsity which degrades the discriminative ability of the logistic regression classifier

used in PRA. This can be addressed by adding latent relations obtained by clustering the surface relations, instead of directly adding the surface relations. This reduces feature sparsity and has been shown to improve PRA inference (Gardner et al., 2013) , (Gardner et al., 2014).

In this article we propose a scheme for augmenting the KB using paths obtained by mining noun phrases that connect two SVO triples from an external corpus. We term these noun phrases as *bridging entities* since they bridge two KB relations to form a path. This is different from the scheme in (Gardner et al., 2013) and (Gardner et al., 2014), which adds edges between KB nodes by mining surface relations from an external corpus. We search for such bridging entities in the corpus by performing a limited depth DFS (depth first search) on the corpus graph in an *on-demand* fashion.

We term this procedure as **On-Demand Augmentation (ODA)**, because the search can be performed during test time in an on-demand manner. In contrast, the previous approaches of adding edges or embeddings to the KB (Gardner et al., 2013), and vector space random walk PRA (Gardner et al., 2014) are batch procedures. As we shall see in Section 4, due to a limited search space, on-demand augmentation is much faster compared to algorithms in (Gardner et al., 2013; Gardner et al., 2014). Furthermore, since edges are not added blindly, on-demand augmentation does not increase feature sparsity which is responsible for performance degradation. Our experiments suggest that ODA provides better performance than (Gardner et al., 2013) and nearly the same prediction performance as provided by (Gardner et al., 2014), but in both cases with the added advantage of faster running time and greater flexibility due to its online and on-demand nature. The code along with the results can be obtained at https://github.com/malllabiisc/pra-oda.

2 Related Work

Using surface level relations and noun phrases for extracting meaningful relational facts is not a new idea (Hearst, 1992),(Brin, 1999), (Etzioni et al., 2004). However, none of them make use of Knowledge Bases for improving information extraction.

The Path Ranking Algorithm (PRA) first proposed in (Lao and Cohen, 2010) was used for performing inference over a KB in (Lao et al., 2011). It was extended by (Lao et al., 2012), to improve the inference by augmenting the KB with syntactic information obtained from a dependency parsed corpus. Augmenting the KB for improving PRA inference using surface relations mined from an external corpus and using latent edge labels obtained by performing PCA on the surface relations was explored in (Gardner et al., 2013). Instead of hard mapping of surface relations to latent embeddings, (Gardner et al., 2014) perform a 'soft' mapping using vector space random walks. This allows the random walker to traverse an edge semantically similar to the current edge type more frequently than other edges.

Although, like others, we too use an external corpus to augment the KB, the crucial difference in our approach is that apart from adding surface relations, we also add bridging entities that enable us to create new paths in the KB. Furthermore, the procedure is targeted so that only paths that play a part in inferring the relations that are of interest are added. Thus, the number of paths added in this manner is much lower than the number of surface relations added using the procedure in (Gardner et al., 2013). As we shall see in Section 4, this results in a more effective algorithm with faster runtime.

3 Method

3.1 Background: Path Ranking Algorithm (PRA)

We first present a brief overview of the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010). The PRA uses paths as features for a logistic regression classifier which predicts if the given relation exists between a pair of entities. For a given pair of entities s and t, the *path type* connecting s to t form the feature vector. A *path types* π is an ordered set of relations. Paths with the same ordered relations but different intermediate or terminal entities belong to the same path type. For example, $s_1 \xrightarrow{v_0} x_1 \xrightarrow{v_1} t_1$ and $s_2 \xrightarrow{v_0} x_2 \xrightarrow{v_1} t_2$ belong to path type $\xrightarrow{v_0}\xrightarrow{v_1}$. The value of a feature, is taken to be $P(s \rightarrow t; \pi)$, where $P(s \rightarrow t; \pi)$ is the probability of reaching t from s by traversing paths of type π. PRA approximates these probabilities by running a random walk (RW) on the KB graph. Let $F = \{\pi_1, \pi_2, ..., \pi_k\}$ be the set of all path types. For predicting the existence of relation r between entities s and t, the logistic regression classifier outputs a score which is a measure of the

Query	Candidate Answer	Path added by PRA-ODA with bridging entity (in bold)
sportsteamPositionForSport(right handed pitcher, ?)	baseball	right handed pitcher $\overset{playsfor}{\longrightarrow}$ **Chicago Cubs** $\overset{play}{\longrightarrow}$ baseball
riverFlowsThroughCity(Moselle, ?)	Koblenz	Moselle $\overset{flows\ into}{\longrightarrow}$ **Rhine** $\overset{meet\ at}{\longrightarrow}$ Koblenz
teamPlaysInLeague(Cleveland Indians, ?)	MLB	Cleveland Indians $\overset{play}{\longrightarrow}$ **Detroit Tigers** $\overset{blew}{\longrightarrow}$ MLB

Table 1: Examples of paths involving bridging entities (marked in bold) added to the KB by PRA-ODA.

confidence that r exists between s and t. It does so by first assigning weights to the features in the training phase. The score is given by

$$S(s,t,r) = \sum_{\pi \in F} P(s \rightarrow t; \pi) \times \theta_{\pi}^{r} \qquad (1)$$

where θ_{π}^{r} is the weight learned by the logistic regression classifier during training specially for relation r and path type π. During the test phase, since targets are not available, the PRA gathers candidate targets by performing a random walk and then computes feature vectors and the score.

3.2 PRA-SVO and PRA-VS

PRA-SVO and PRA-VS are the systems proposed in (Gardner et al., 2013) and (Gardner et al., 2014) respectively, where the KB graph is augmented with edges mined from a large subject-verb-object (SVO) triple corpus. In these two systems, only new edges are added over the fixed set of nodes, and the augmentation happens in a batch, offline setting. In contrast, PRA-ODA, the method proposed in the paper, also expands the set of nodes through bridging entities, and performs the augmentation in an on-demand manner.

3.3 PRA On-Demand Augmentation (PRA-ODA)

Training: Let s and t be any two KB entities and let $s^{(n)}$ and $t^{(n)}$ be their corresponding noun phrase representations or aliases. We search for bridging entities $x_1, x_2, ..x_n$ by performing limited depth first search (DFS) starting with s^n such that we obtain a path $s \overset{ALIAS}{\longrightarrow} s^{(n)} \overset{v_0}{\longrightarrow} x_1 \overset{v_1}{\longrightarrow} ... \overset{v_{n-1}}{\longrightarrow} x_n \overset{v_n}{\longrightarrow} t^{(n)} \overset{ALIAS}{\longrightarrow} t$, where v_i are verbs present in the corpus graph. This is done for all $n \leq d_{max} - 1$, where d_{max} is the maximum depth of DFS. We add an 'ALIAS' edge between the KB entity and its noun phase representation. The usefulness of bridging entities is illustrated in Fig. 1.

We mine bridging entities from a corpus containing over 600 million SVO triples which were obtained from the ClueWeb09 corpus (Callan et al., 2009) parsed using the MALT parser (Nivre et al., 2007). We use Mongo DB to store the triples as an adjacency list. During training time, for any relation that is being inferred, both the source and its corresponding target entities are known. A limited depth DFS is performed for *all depths* less then d_{max} on the SVO graph with the aliases of subject entity acting as the starting points. Such aliases are available for the NELL and Freebase knowledge bases. The DFS is said to discover a path if the terminating entity of the path matches any alias of the target entity. We choose to use aliases to perform string match, since it is easy to change the softness of the match by simply adding more aliases. This is done for all training source-target pairs. A few examples of added paths are shown in Table 1.

The SVO graph is noisy since it is obtained by parsing the ClueWeb corpus which was obtained by scraping the web. To reduce noise, we add the top K most frequent discovered SVO path types, where K is a tunable parameter. By SVO path type we refer to a set of ordered verbs mined from the SVO corpus. There is a possibility that the bridging entities, extracted from the corpus, may be present in the KB. If the bridging entity matches any alias, then it is treated as an alias to an existing KB entity. If not, then the bridging entity is added to the KB as a new entity. To avoid overfitting we add negative data to the training set. Furthermore, only high quality expressive bridging entities result in meaningful and discriminative paths. Although the quality of bridging entities depend on the corpus, low quality bridging entities can be filtered out by adding negative training data. Low quality bridging entities connect source target pairs from both positive and negative training sets, and hence are eliminated by the sparse logistic regression classifier. The negative dataset is generated using the closed world assumption by performing a random walk.

After augmenting the KB, we run the training phase of the PRA algorithm to obtain the feature (path) weights computed by the logistic regression

KB Relations	PRA	PRA-SVO	PRA-VS	PRA-ODA
actorstarredinmovie	0.0	**1.0**	**1.0**	**1.0**
atheleteplaysforteam	1.0	1.0	1.0	1.0
citylocatedincountry	0.166	0.25	**1.0**	**1.0**
journalistwritesfor publication	1.0	1.0	1.0	1.0
riverflowsthroughcity	0.333	0.25	**1.0**	**1.0**
sportsteamposition forsport	1.0	1.0	1.0	1.0
stadiumlocatedincity	1.0	1.0	1.0	1.0
statehaslake	0.0	0.0	0.0	0.0
teamplaysinleague	1.0	1.0	1.0	1.0
writerwrotebook	1.0	1.0	1.0	1.0
Average (MRR)	0.649	0.75	**0.9**	**0.9**

Table 2: Comparison of Mean Reciprocal Rank (MRR) metric for 10 relations from NELL (higher is better). PRA-SVO, PRA-VS are the systems proposed in (Gardner et al., 2013; Gardner et al., 2014). PRA-ODA is the approach proposed in this paper. Improvements in PRA-ODA over PRA-SVO is statistically significant with $p < 0.007$, with PRA-SVO as null hypothesis.

classifier.

Query Time: The set of target entities corresponding to a source entity and the relation being predicted is not available during query (test) time. We use all the entities included in the range of the relation being predicted as candidate target entities. For example, if the relation is *riverFlowsThroughCity*, the candidate target set would include entities in the KB that are *cities*. The DFS is now performed starting from source entities as during training, but this time only restricting to paths with positive weights learned during training. Any path (along with bridging entities) found during this search are added to the KB, and the PRA algorithm is now run over this augmented graph.

4 Experiments

We used the implementation of PRA provided by the authors of (Gardner et al., 2014). For our experiments, we used the same 10 NELL relation data as used in (Gardner et al., 2014). The augmentation resulted in the addition of 1086 paths during training and 1430 paths during test time.

We split the NELL data into 60% training data, 15 % development data and 25% test data. Values for d_{max}, and K, the most frequent paths, were obtained by tuning on a development set for 4 relations (athleteplaysforsport,actorstarredinmovie,citylocatedincountry and journalistwritesforpublication). The hyperparameter values $d_{max} = 2$, $K = 10$ reported the highest MRR and were used for the rest of the relations. For the L_1 and L_2 regularization parameters in the logistic regression classifier, we used the same values as used in (Gardner et al., 2013; Gardner et al., 2014), viz., $L_1 = 0.005$, and $L_2 = 1.0$. This is because the parameters were reported to be robust, and seemed to work well even when the knowledge base was augmented.

Timings (seconds)	PRA	PRA-SVO	PRA-VS	PRA-ODA
Training	635.6	574.5	564.2	913.3
Test	354.3	322.0	301.2	436.7
Batch augmentation	n/a	797	797	n/a
Embedding computation	n/a	n/a	812	n/a
Total Time	989.9	1693.5	2474.4	1350

Table 3: Runtime comparison for the entire experiment (lower is better). PRA-SVO, PRA-VS are the systems proposed in (Gardner et al., 2013; Gardner et al., 2014). PRA-ODA is the approach proposed in this paper. Between the two top performing systems, i.e., PRA-ODA and PRA-VS, PRA-ODA is faster by a factor of 1.8.

We compare the results (PRA-ODA) with the PRA algorithm executed on the NELL KB, NELL KB augmented with surface relations (PRA-SVO) (Gardner et al., 2013) and vector space random walk PRA (PRA-VS) (Gardner et al., 2014). The run times, i.e, the time taken to perform an entire experiment for PRA-SVO and PRA-VS includes the time taken to augment NELL KB with SVO edges. The PRA-VS runtime also includes the time taken for generating embeddings to perform the vector space random walk. As can be seen from Table 2 and Table 3, our scheme, PRA-ODA, provides performance equivalent to PRA-VS with faster running time (speed up of 1.8). In addition to the time taken for the full SVO augmentation, PRA-VS takes additional time to generate embeddings (13 minutes) from the added verbs. We note that the batch augmentation in case of PRA-SVO and PRA-VS, and embedding computation in case of PRA-VS are all specific to the relations in the evaluation set, and hence can't be ignored as a one-time offline cost. In other words, these costs are likely to increase as more relations (and their instances) are included during training and testing. Runtime gains with PRA-ODA are likely to be even more pronounced in such settings.

An additional advantage of the proposed algorithm is that it can also be run on the top of any PRA based algorithm such as the PRA-SVO and PRA-VS.

5 Conclusion

In this paper, we investigated the usefulness of adding paths to a Knowledge Base for improving its connectivity by mining *bridging entities* from an external corpus. While previous KB augmentation methods focused only on augmentation using mined surface verbs while keeping the node set fixed, we extended these approaches by also adding *bridging entities* in an online fashion. We used a large corpus of 500 million web text corpus to mine these additional edges and bridging entities. Through experiments on real-world datasets, we demonstrate that the proposed approach is not only comparable or better than other state-of-the-art baselines, but more importantly provides faster overall runtime compared with the alternatives.

Acknowledgment

This work is supported in part by a gift from Google.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In Paolo Atzeni, Alberto Mendelzon, and Giansalvatore Mecca, editors, *The World Wide Web and Databases*, volume 1590 of *Lecture Notes in Computer Science*, pages 172–183. Springer Berlin Heidelberg.

J. Callan, M. Hoy, C. Yoo, and L. Zhao. 2009. Clueweb09 data set. boston.lti.cs.cmu.edu.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA. ACM.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 833–838.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 397–406.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination ofpath constrained random walks. *Machine Learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1017–1026, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA. ACM.