

DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning

黄友艺

摘要

在过去的方法中，Path-Ranking Algorithm (PRA)被认为是一种有潜力的推理路径学习方法。然而，PRA在完全离散的空间中操作，这导致在知识图谱中评估和比较相似实体和关系变得困难。为了克服这些局限性，本文作为首篇将强化学习应用于知识图谱推理领域的论文，提出了基于蒙特卡洛的DeepPath方法 [21]，通过在知识图谱向量空间中进行推理，设计并使用策略梯度训练和新颖的奖励函数来引导代理学习关系路径。本篇复现工作的内容包括将官方TensorFlow实现转换为PyTorch，尝试采用PPO、Actor-Critic方法替换原论文中的蒙特卡洛强化学习方法，并在数据集NELL-995上对代码增加实验批处理和可视化，以期提高训练效率和模型性能。

关键词：知识图谱；多跳关系推理；强化学习；路径学习；Actor-Critic

1 引言

随着深度学习技术在各领域的快速发展 [10, 12, 13]，对于大规模知识图谱（Knowledge Graphs，简称KGs）中进行推理的需求日益增长。本研究旨在探讨学习在KGs中进行多跳关系推理的问题，即学习在一个实体与另一个实体之间通过多个中间关系进行推理的过程。具体而言，作者关注的是一种全新的强化学习框架，即DeepPath，该框架使用了一种基于策略的代理，其连续状态基于知识图谱嵌入。通过在KG向量空间中采样最有前途的关系，代理能够扩展其路径，实现对知识图谱中的推理。

与之前的工作不同，作者的方法包含一个综合考虑准确性、多样性和效率的奖励函数。通过在训练过程中引入奖励函数，能够更全面地引导代理学习关系路径，从而提高推理的质量和效果。在实验中，通过在Freebase和Never-Ending Language Learning数据集上的对比实验证明，作者的方法相较于基于路径排名的算法和知识图谱嵌入方法取得了更优越的性能。

处理复杂的自然语言处理问题仍然是一个挑战。为了应对这一挑战，作者将研究重点放在多跳推理任务上，即在大规模知识图谱中学习明确的推理公式。例如，给定知识图谱中的关系，如Neymar效力于巴塞罗那，而巴塞罗那在联赛中，本文的方法旨在让机器能够学习以下公式： $\text{playerPlaysForTeam}(P,T) \wedge \text{teamPlaysInLeague}(T,L) \rightarrow \text{playerPlaysInLeague}(P,L)$ 。通过在测试时插入学到的公式，系统应该能够自动推断两个实体之间的缺失关联。

在过去的工作中，Path-Ranking Algorithm (PRA) [14, 15]曾被认为是在大规模知识图谱中学习推理路径的有希望方法。然而，由于PRA在完全离散的空间中操作，难以评估和比较知

识图谱中的相似实体和关系，因此存在局限性。因此，本研究提出了一种新颖的、可控的多跳推理方法，将路径学习过程框架化为强化学习。与PRA不同，我们使用基于翻译 [2] 的知识图谱嵌入方法来编码强化学习代理的连续状态，该代理在知识图谱的矢量空间中进行推理。我们的代理通过采样关系来扩展其路径，实现了逐步推理的过程。为了更好地引导强化学习代理学习关系路径，我们采用了带有准确性、多样性和效率综合考虑的策略梯度训练 [16]。在实证实验中，我们展示了我们的方法在Freebase和Never-Ending Language Learning [3] 数据集上的优越性能，相较于PRA和嵌入方法。

本篇复现的论文的三项主要贡献体现在以下三个方面：首先，本篇提出的DeepPath是首次考虑在知识图谱中学习关系路径的强化学习方法；其次，我们的学习方法使用了一个复杂的奖励函数，同时考虑了准确性、效率和路径多样性，提供了在路径查找过程中更好的控制和灵活性；最后，我们展示了我们的方法能够扩展到大规模知识图谱，并在两个任务中优于PRA和知识图谱嵌入方法。

2 相关工作

在路径查找和知识图谱推理领域，已经涌现出多种方法和模型。以下是一些与我们的研究相关的工作：

2.1 基于路径排名的算法（PRA）

Path-Ranking Algorithm（PRA） [14]是主要的路径查找方法之一，采用随机游走和重启策略进行多跳推理。Gardner等人 [7, 8]对PRA进行了修改，计算了在矢量空间中的特征相似性。Wang和Cohen [19]引入了一种整合背景KG和文本的递归随机游走方法，该方法同时对逻辑程序进行结构学习，并从文本中提取信息。然而，PRA的一个潜在瓶颈是，连接大量公式的超级节点会创建巨大的扇出区域，显著减慢推理速度并影响准确性。

2.2 卷积神经网络（CNN）方法

Toutanova等人 [18]提出了一种基于卷积神经网络的多跳推理解决方案。他们建立了一个基于词汇化依赖路径的CNN模型，但由于解析错误而导致的错误传播问题。Guu等人 [9]使用知识图谱嵌入来回答路径查询。Zeng等人 [22]描述了一个用于关系抽取的CNN模型，但它并未明确对关系路径进行建模。

2.3 循环神经网络（RNN）方法

Neelakantan等人 [17]提出了一种用于知识库完成（KBC）中建模关系路径的循环神经网络模型。然而，该方法训练了太多独立模型，因此不具有可扩展性。与PRA不同，我们的方法在连续空间中进行推理，并通过在奖励函数中引入各种标准，使我们的强化学习（RL）框架在路径查找过程中具有更好的控制和灵活性。

2.4 神经符号机（NSM）方法

Liang等人 [5]提出的神经符号机是最近关于知识图谱推理的工作之一，同样应用了强化学习。然而，NSM与我们的工作有不同的侧重点。NSM学习组合可以回答自然语言问题的程序，而我们的RL模型尝试通过对现有知识图谱三元组进行推理，向知识图谱（KG）中添加新事实。因此，NSM的动作空间是一组预定义的标记，而我们的目标是寻找推理路径，因此动作空间是知识图谱中的关系空间。

此外，Johnson等人 [11]也应用了类似的框架于视觉推理任务。

综上所述，已有多种方法尝试解决知识图谱推理的问题 [6, 17]，但每种方法都有其优劣势。在我们的工作中，我们提出了一种基于强化学习的框架，通过在连续空间中进行推理，结合奖励函数的多标准，使我们的方法在路径查找过程中更具控制性和灵活性。

3 本文方法

3.1 本文方法概述

本文的工作首先将知识图谱环境描述为马尔科夫决策过程的环境中，将路径查找问题转为了一个顺序决策问题，于是可以设计出一个强化学习模型，通过与环境交互，选择有希望的推理路径。在模型的训练过程中，本文借鉴了AlphaGo的训练方法，为了解决初期难以收敛的问题，首先引入带有随机性质的双向广度优先搜索得到的路径作为专家知识进行预训练。最后，为了更高效地搜索知识图谱中的路径，本文针对每个实体对增加了双向路径，通过增加双向查找的方法，提高模型对实体对之间关系路径的发现效率。

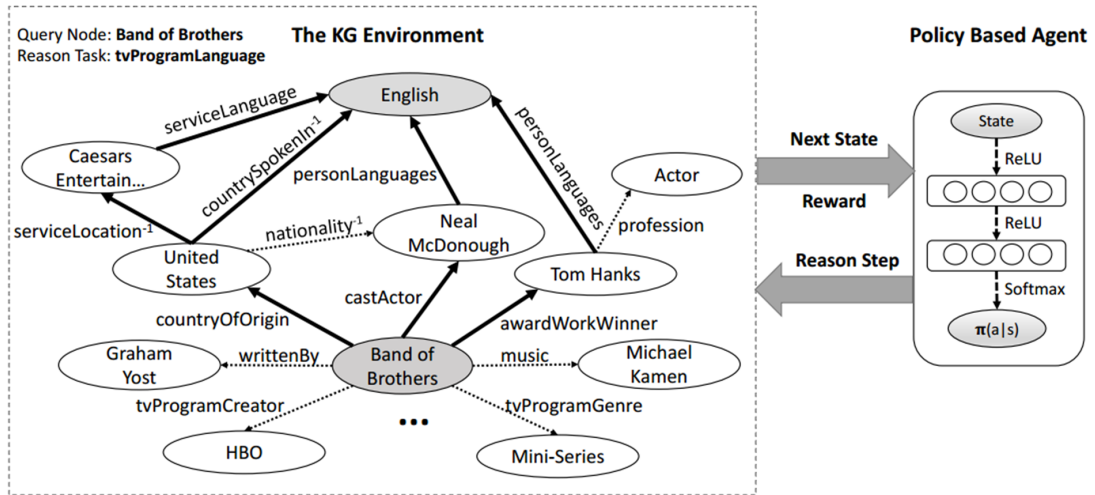


图 1. RL模型概述

如图所示，左图表示由MDP建模的KG环境。虚线箭头(部分)表示KG中现有的关系链接，粗体箭头表示RL代理找到的推理路径。 s^{-1} 为关系的倒数。右图是策略网络代理的结构。在每一步中，通过与环境的交互，智能体学习选择一个关系链接来扩展推理路径。

3.2 强化学习建模

在这一部分，我们将描述基于强化学习的多跳关系推理框架。关系推理的具体任务是找到实体对之间可靠的预测路径。我们将路径查找问题形式化为一个顺序决策问题，可以通过强化学习代理解决。首先，我们描述了基于知识图谱（KG）设计的环境以及基于策略的强化学习代理。通过与环境的交互，代理学习选择有前途的推理路径。接着，我们描述了强化学习模型的训练过程。

强化学习模型主要使用策略网络 $\pi_{\theta}(s, a) = p(a|s; \theta)$ ，将状态向量映射到随机策略。原论文中采用随机梯度下降法更新神经网络参数 θ 。

与深度Q网络(DQN) [16]相比，基于策略的强化学习方法更适合我们的知识图场景。一个原因是，对于KG中的寻径问题，由于关系图的复杂性，动作空间可能非常大。这可能导致DQN的收敛性差。此外，与DQN等基于值的方法中常见的贪婪策略学习不同，策略网络能够学习随机策略，防止智能体卡在中间状态。在我们描述策略网络的结构之前，我们首先描述RL环境的内容(动作、状态、奖励)

状态 知识图谱中的实体和关系是离散的自然原子符号。由于Freebase [1]和NELL [4] 等现有实用的知识图谱通常具有大量的三元组。直接模拟所有状态下的符号原子是不可能的。为了捕获这些符号的语义信息，我们使用基于翻译的嵌入，如TransE [2]和TransH [20]来表示实体和关系。

在本次复现中，每个状态捕获智能体在知识图中的位置。在采取行动后，代理将从一个实体移动到另一个实体。这两者通过智能体刚刚采取的行动(关系)联系在一起。步骤 t 处的状态向量如下：

$$\mathbf{s}_t = (\mathbf{e}_t, \mathbf{e}_{target} - \mathbf{e}_t)$$

这里的 \mathbf{e}_t 是当前节点的嵌入向量， \mathbf{e}_{target} 是目标节点的嵌入向量。

因为在寻径过程中，推理关系的嵌入保持不变，不利于训练，所以代码中也没有在状态中加入推理关系。

动作 从源实体 e 开始，代理使用策略网络选择最有希望的关系，在每一步扩展其路径，直到到达目标实体 e 。为了保持策略网络输出维度的一致性，将动作空间定义为KG中的所有关系。

奖励 复现中采取奖励主要是手工构建了三种奖励，分别从全局正确性、路径长度以及路径多样性来考虑。

为了防止代理寻找到错误的路径，如果agent在一系列行动后到达目标，它将获得+1的正奖励。

$$r_{\text{GLOBAL}} = \begin{cases} +1, & \text{if the path reaches } e_{target} \\ -1, & \text{otherwise} \end{cases}$$

对于关系推理任务，我们观察到短路径往往比长路径提供更可靠的推理证据。较短的关系链也可以通过限制 \mathbf{RL} 与环境交互的长度来提高推理效率。效率奖励定义如下：

$$r_{\text{EFFICIENCY}} = \frac{1}{\text{length}(p)}$$

我们训练智能体使用每个关系的正样本来寻找路径。这些训练样本(资源;目标)在向量空间中具有类似的状态表示。代理倾向于寻找具有相似语法和语义的路径。这些路径通常包含冗余信息，因为其中一些路径可能是相关的。为了鼓励智能体找到不同的路径，我们使用余弦相似度定义了一个多样性奖励函数。

$$r_{\text{DIVERSITY}} = -\frac{1}{|F|} \sum_{i=1}^{|F|} \cos(\mathbf{p}, \mathbf{p}_i)$$

3.3 专家知识监督

由于策略网络输出层维度较高，初期难以收敛的问题。作者借鉴了AlphaGo的方法，使用有监督的策略网络进行训练。通过引入有监督信号，作者提高了模型在早期训练阶段的稳定性，从而更好地指导代理的学习过程。

监督策略学习对于每个关系，我们首先使用所有正样本（实体对）的子集来学习有监督的策略。

对于每个正样本（ $\mathbf{esource}, \mathbf{etarget}$ ），对于路径 \mathbf{p} ，使用蒙特卡洛策略梯度（REINFORCE 方法）来最大化期望的累积奖励。

3.4 专家知识监督

在强化学习领域，尤其是涉及策略网络的训练时，由于输出层的高维性可能导致训练初期的不稳定性。为了解决这一问题，本研究采用了专家知识监督学习的方法。具体而言，我们从知识图谱中采用随机中间节点的两端BFS找到实体之间的潜在推理路径。结合使用监督学习信号和强化学习（特别是REINFORCE算法），我们设计了一个损失函数 $J(\theta)$ ，旨在优化策略网络并稳定训练过程。损失函数定义如下(引自原论文 [21])：

$$J(\theta) = \mathbb{E}_{a \sim \pi(a|s_t; \theta)} \left[\sum_t R_{s_t, a_t} \right] = \sum_t \sum_{a \in A} \pi(a|s_t; \theta) R_{s_t, a_t} \quad (1)$$

其中， $\pi(a|s_t; \theta)$ 表示在状态 s_t 下采取行动 a 的策略概率。通过使用BFS确定的路径作为训练中的正样本，我们能够引导策略网络更有效地学习和发现有价值的行动策略。

此外，我们还定义了一个用于优化的价值函数 V 来估计策略在特定状态下的期望回报，它的梯度 ∇J 可以近似表示为(引自原论文 [21])：

$$\nabla J \approx \nabla_{\theta} \sum_t \log \pi(a = r_t | s_t; \theta) \quad (2)$$

其中 r_t 代表在时间步 t 下获得的奖励。综合以上方法，我们的模型能够利用专家知识在复杂的知识图谱中有效地学习推理路径，进而加强模型的推理能力和稳定性。

3.5 双向路径约束

最后，作者详细介绍了针对每个实体对进行双向查找的方法。通过双向查找，我们能够更全面地搜索知识图谱中的路径，并且描述了一个高效的路径约束搜索算法，该算法利用强化学习代理发现的路径进行关系推理。这一方法提高了对实体对之间关系路径的发现效率。

4 复现细节

4.1 与已有开源代码对比

在本研究的复现过程中，我们重点参考了位于GitHub上的开源项目xwhan/DeepPath¹。本节旨在比较原始代码和我们的实现，同时强调我们工作中的独创性和改进之处。

首先，需要指出的是，尽管我们的工作基于现有的开源代码，但在复现的过程中，我们引入了显著的创新和改进。具体来说，我们不仅转换了代码的框架，还针对现有问题提供了解决方案，对算法进行了改进，并在工程实现上作出了优化。

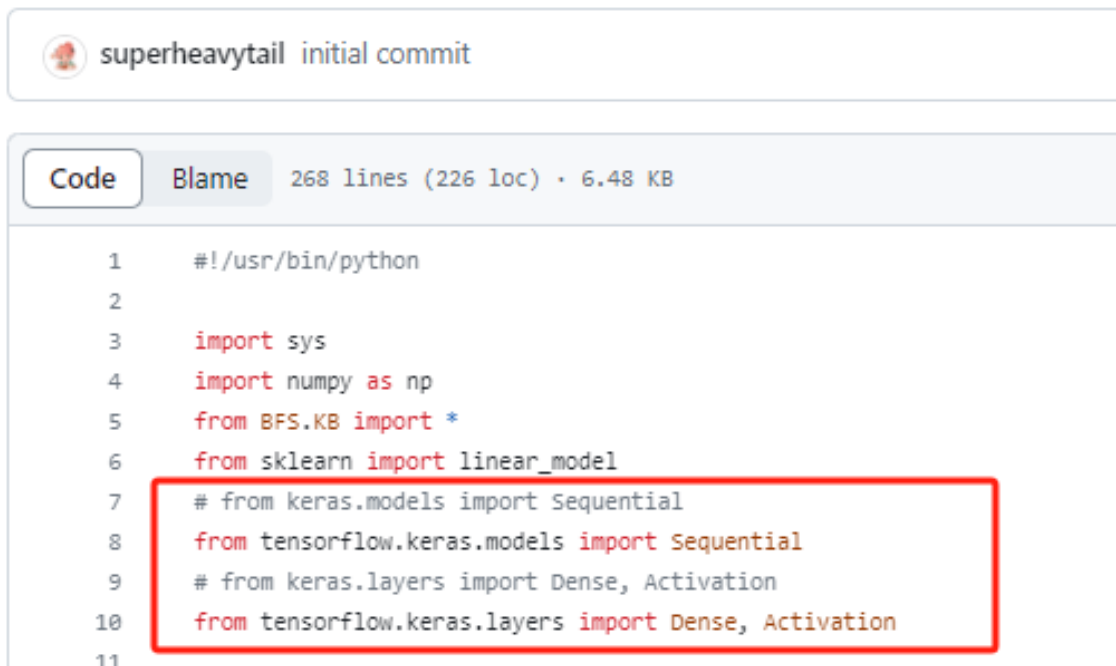
4.1.1 代码转化

原项目是基于TensorFlow 1.x 实现的，考虑到相关生态系统已不再维护，我们将其转换为了PyTorch实现。在这一过程中，我们参考了superheavyltail/deep-path-pytorch²的实现，但也发现了其中少部分保留TensorFlow 2语法的实现。

例如，在原实现中，使用Keras模块是基于TensorFlow和单独安装的Keras的情况。但在我们的PyTorch实现中，我们采用了torch.nn.Linear来替代Keras中的Dense层，从而更好地适应PyTorch框架。

¹xwhan/DeepPath:<https://github.com/xwhan/DeepPath>

²superheavyltail/deep-path-pytorch:<https://github.com/superheavyltail/deep-path-pytorch>



```
superheavytail initial commit

Code Blame 268 lines (226 loc) · 6.48 KB

1  #!/usr/bin/python
2
3  import sys
4  import numpy as np
5  from BFS.KB import *
6  from sklearn import linear_model
7  # from keras.models import Sequential
8  from tensorflow.keras.models import Sequential
9  # from keras.layers import Dense, Activation
10 from tensorflow.keras.layers import Dense, Activation
11
```

图 2. 原有的tensorflow依赖

Listing 1: 代码转化

```
1 # before
2 model.add(Dense(1, activation='sigmoid', input_dim=input_dim))
3 # after
4 self.linear = nn.Linear(input_dim, 1)
5 self.sigmoid = nn.Sigmoid()
```

此外，我们还发现现有的实现中均存在训练时‘Cannot find a path’的问题。通过深入分析和重写教师模型的生成逻辑，才得以成功解决此问题。

4.1.2 工程优化

我们对原代码进行了多方面的优化，以增强其实用性和性能。

硬编码修改 原代码中存在硬编码路径的问题，这限制了其不同操作系统上的可移植性。我们通过使用os库来基于项目路径动态拼接文件路径，从而实现了代码在Linux和Windows上的无缝运行。

批量处理 考虑到NELL-995数据集包含多种关系推理任务，我们增加了批量训练和推理程序。我们还对原代码中的一些性能瓶颈进行了优化，例如通过统一数据写入操作，依据任务间的独立关系使用进程池ProcessPoolExecutor来并行化处理，从而提高了整体性能。

可视分析 为了更好地监控和分析训练过程，我们引入了wandb库来可视化训练和测试中的关键指标，如损失函数、平均成功率、平均奖励和平均路径长度。

4.1.3 算法改进

我们主要尝试了两种不同的方法来优化强化学习中的蒙特卡洛方法。

1. 第一种方法是引入了基于agent-only的PPO-clip算法来优化模型损失函数。
2. 第二种方法则是引入了一个判别模型Critic，以评估图谱的状态价值，并计算优势函数来更新agent。

PPO-clip 在强化学习中，特别是在使用PPO（Proximal Policy Optimization）算法时，通常采用带剪辑的策略梯度方法来更新策略。PPO算法通过限制策略更新步骤的大小，以防止训练过程中的不稳定性，从而实现更稳健的学习。

PPO的目标损失函数定义如下：

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right]$$

其中：

- $L^{CLIP}(\theta)$ 是PPO的剪辑损失函数。
- t 表示时间步。
- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ 是策略比率，即在新策略下选择动作相对于旧策略的概率比。
- \hat{A}_t 是优势估计，衡量在状态 s_t 下执行动作 a_t 相比平均情况的额外价值。

损失函数的设计确保了在更新策略参数时，更新的步骤不会太大，从而使训练更加平稳和高效。下面的算法显示了如何计算PPO的Actor损失：

Algorithm 1 PPO Actor Loss Calculation

Input: actor_model, state_vec, reward, action_chosen, old_log_prob, clip_ratio

Output: actor_loss

- 1: advantage \leftarrow ComputeAdvantage(reward)
 - 2: action_probs \leftarrow actor_model(state_vec)
 - 3: m \leftarrow Categorical(action_probs)
 - 4: new_log_prob \leftarrow m.log_prob(action_chosen)
 - 5: ratio \leftarrow exp(new_log_prob - old_log_prob)
 - 6: surr1 \leftarrow ratio \times advantage
 - 7: surr2 \leftarrow clip(ratio, 1 - clip_ratio, 1 + clip_ratio) \times advantage
 - 8: actor_loss \leftarrow - min(surr1, surr2) ▷ Take negative for minimization
 - 9: Log the actor loss for monitoring
 - 10: Update the actor model using gradient descent
-

在这里，actor_loss 代表PPO损失的负值，其目的是通过梯度下降方法最小化策略损失，以优化策略表现。

Actor-Critic 本次复现的Actor-Critic算法中相比原论文的单策略网络模型增加了一个判别模型。这个判别模型的作用是输出状态的价值估计，这与actor模型需要输出的动作概率不同。因此，在保持与actor模型相同的架构的同时，我们在判别模型中去除了最后的Softmax函数。

Listing 2: Critic class

```
1 class Critic(nn.Module):
2     def __init__(self):
3         # ... initialization ...
4         self.linear_relu_stack = nn.Sequential(
5             fc1 ,
6             nn.ReLU() ,
7             fc2 ,
8             nn.ReLU() ,
9             fc3
10            # Softmax is omitted
11        )
```

在本次复现的其余修改中，也尝试去改善状态的表征，在状态输入时对状态进行了正则化处理，确保输入状态呈现均值为0，方差为1的正态分布、以及尝试修改网络结构，增加简单的Layer Normalization、Attention等结构对输入做额外的处理，实践表明，在模型训练次数受限的情况去盲目修改、扩充网络结构并不是一个明智的选择。

4.2 实验环境搭建

1. 创建并激活虚拟环境

python3.7、3.11环境均可，下面仅为其中一个示例。

```
conda activate -n deeppath python==3.7 pytorch==1.10.0
conda activate deeppath
```

2. 安装依赖项

```
pip install setproctitle==1.3.3
pip install numpy==1.21.6
pip install wandb==0.16.1
pip install scikit-learn==1.0.2
```

3. 数据集准备

下载NELL-995数据集³，放置于项目根目录。

4. 文件夹目录

为了组织代码和输出文件，我们在项目的pytorch文件夹下创建以下子文件夹：

³NELL-995链接:https://download.csdn.net/download/HYY_2000/88573274

- `torchmodels` 文件夹，用于存储模型的中间文件。这有助于在实验中保存和加载不同阶段的模型状态。
- `results` 文件夹，用于存储实验结果。这包括链接推理和事实预测的评估指标。
- `log` 文件夹，用于记录实验过程中生成的日志文件。这对于跟踪实验进展和调试潜在问题非常有用。

4.3 运行流程

1. 专家知识监督训练

```
python sl_policy_multi.py
python sl_policy_multi_actor_critic.py
python sl_policy_multi_PPO.py
```

2. 强化学习再训练并测试（生成中间结果文件）

```
python policy_agent_multi.py
python policy_agent_multi_actor_critic.py
python policy_agent_multi_PPO.py
```

3. 链接预测

```
python evaluate_multi.py
python evaluate_multi_ppo.py
python evaluate_multi_AC.py
```

4. 事实预测

```
python fact_prediction_eval_multi.py
python fact_prediction_eval_multi_ppo.py
python fact_prediction_eval_multi_AC.py
```

5 实验结果分析

本次复现先对论文结果进行复现，再在应用PPO agent-only的基础上，测试了算法的关系预测实验。主要展示原RL方法以及PPO算法的主要结果。

FB15K-237					NELL-995				
Tasks	PRA	RL	TransE	TransR	Tasks	PRA	RL	TransE	TransR
teamSports	0.987	0.955	0.896	0.784	athletePlaysForTeam	0.547	0.750	0.627	0.673
birthPlace	0.441	0.531	0.403	0.417	athletePlaysInLeague	0.841	0.960	0.773	0.912
personNationality	0.846	0.823	0.641	0.720	athleteHomeStadium	0.859	0.890	0.718	0.722
filmDirector	0.349	0.441	0.386	0.399	athletePlaysSport	0.474	0.957	0.876	0.963
filmWrittenBy	0.601	0.457	0.563	0.605	teamPlaySports	0.791	0.738	0.761	0.814
filmLanguage	0.663	0.670	0.642	0.641	orgHeadquarterCity	0.811	0.790	0.620	0.657
tvLanguage	0.960	0.969	0.804	0.906	worksFor	0.681	0.711	0.677	0.692
capitalOf	0.829	0.783	0.554	0.493	bornLocation	0.668	0.757	0.712	0.812
organizationFounded	0.281	0.309	0.390	0.339	personLeadsOrg	0.700	0.795	0.751	0.772
musicianOrigin	0.426	0.514	0.361	0.379	orgHiredPerson	0.599	0.742	0.719	0.737
...					...				
Overall	0.541	0.572	0.532	0.540		0.675	0.796	0.737	0.789

Table 2: Link prediction results (MAP) on two datasets.

图 3. 原论文结果

Fact Prediction Results		
Methods	FB15K-237	NELL-995
RL	0.311	0.493
TransE	0.277	0.383
TransH	0.309	0.389
TransR	0.302	0.406
TransD	0.303	0.413

Table 3: Fact prediction results (MAP) on two datasets.

图 4. 原论文事实预测结果

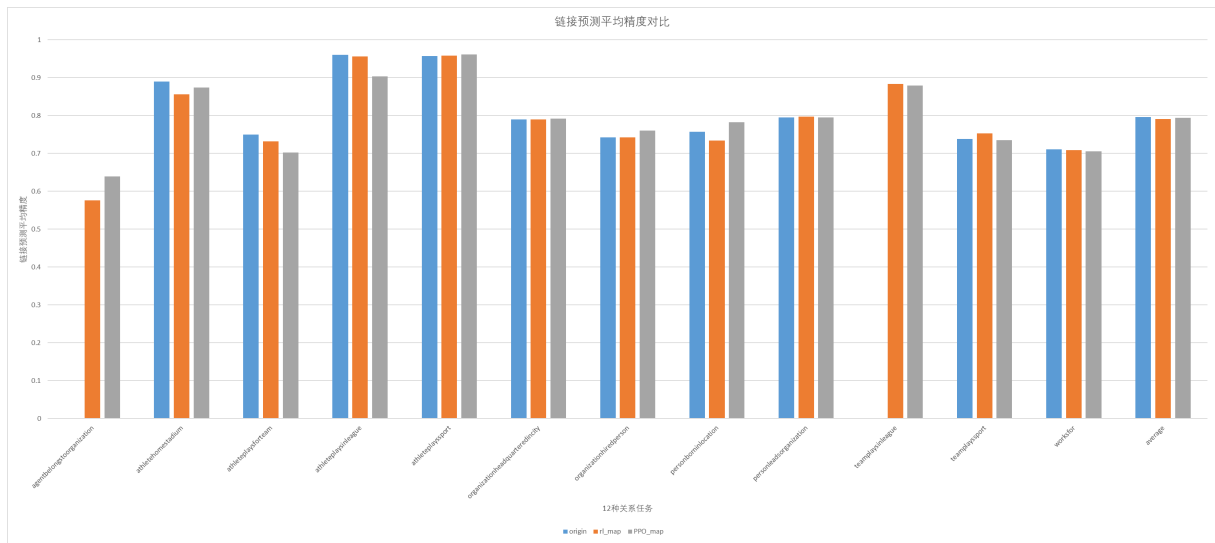


图 5. 链接预测结果

5.1 原文复现结果

我们使用原蒙特卡洛方法实现的关系推理，与原论文的实验数据进行了对比。原论文在链接预测的结果如图3所示。

如图5所示，经过比较，我们可以看到左侧蓝色柱状代表论文原始数据（部分关系结果被隐藏），橙色代表复现的论文结果，灰色代表应用agent-only的PPO算法结果。最右侧的比较显示了平均12个关系推理任务得到的结果（原论文数据为0.796、复现结果为0.790、PPO结果为0.794）。

在原论文的事实预测结果如图4所示，平均精度为0.493。

事实预测的具体数值结果如表1所示。我们注意到在部分关系中AC算法超过了RL和PPO算法的表现，但是存在个别的2个关系（*athleteplaysinleague*、*teamplayssport*）平均精度十分低下。初步估计是由于训练样本对较少，而AC算法对此依赖较大，或AC算法的实现存在遗漏。

原论文的事实预测对比数据如图6所示，左侧蓝色为复现的RL部分，右侧为实现的PPO部分。

表 1. Fact prediction results (MAP) on the dataset.

Relation	RL	PPO	AC
agentbelongstoorganization	0.340	0.338	0.362
athlethomestadium	0.729	0.717	0.717
athleteplaysforteam	0.243	0.319	0.311
athleteplaysinleague	0.527	0.532	0.143
athleteplayssport	0.551	0.495	0.351
organizationheadquarteredincity	0.593	0.642	0.642
organizationhiredperson	0.554	0.451	0.475
personborninlocation	0.289	0.398	0.488
personleadsorganization	0.533	0.496	0.492
teamplaysinleague	0.671	0.702	0.653
teamplayssport	0.399	0.359	0.154
worksfor	0.457	0.453	0.470
average	0.491	0.492	0.438

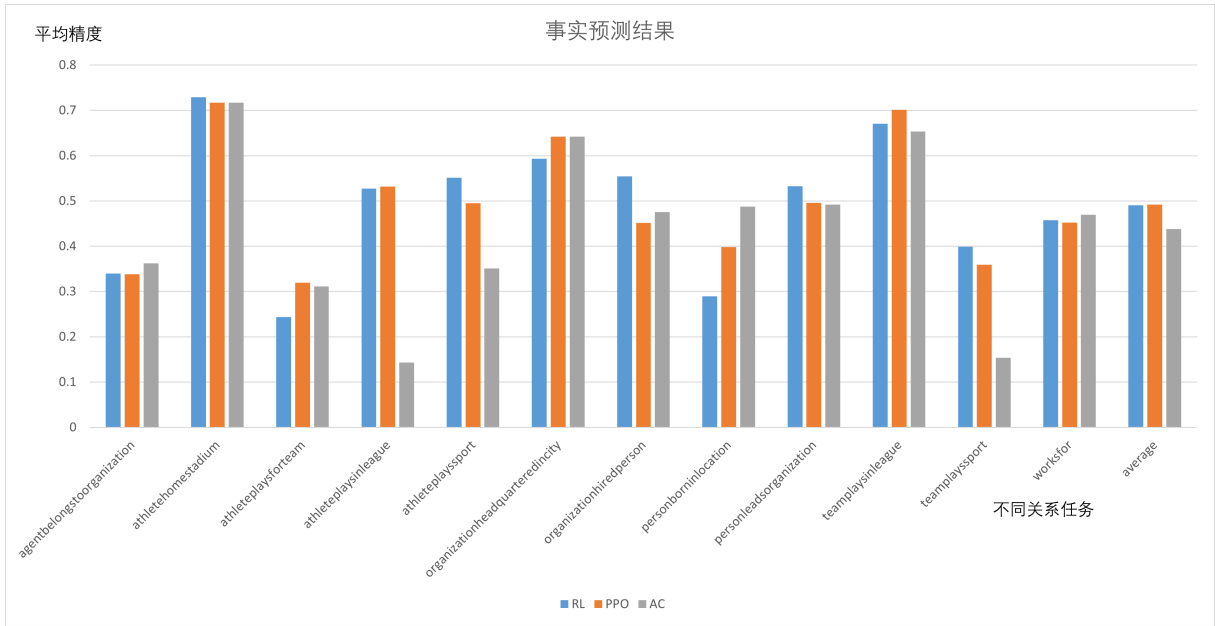


图 6. 事实预测结果

5.2 专家训练结果

为了了解专家训练的预训练效果，我们以单一关系`athleteplaysforteam`为例进行了分析。如图7所示，路径奖励在初期较少，但随着训练的进行，奖励逐渐增多，总体路径长度保持在20以内。

在其他关系任务下，训练的损失函数也逐渐趋近于零。监督学习下的critic和actor损失如图8和图9所示，分别展示了这一趋势。



图 7. 关系athleplaysforteam任务下的专家训练结果

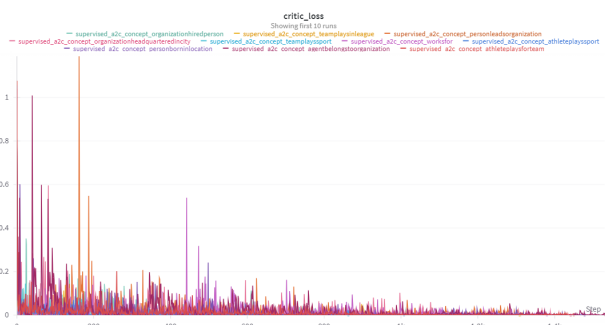


图 8. 监督学习下的critic loss



图 9. 监督学习下的actor loss

6 总结与展望

本研究成功复现并训练了一个在知识图谱中寻找推理路径的强化学习代理。通过这个代理，我们展示了自动化推理路径发现的潜力，并为知识图谱的自动推理和知识发现提供了新的途径。

虽然大语言模型如ChatGPT在处理大规模数据集方面表现出色，但知识图谱在结构化知识表示和可解释性方面仍具有独特优势。未来的工作可以探索将深度学习与知识图谱相结合的方法，以利用两者的互补优势。这种结合不仅有助于提高模型在推理任务上的性能，还可以增强模型的透明度和解释能力，特别是在需要高度可解释性的领域如医疗和法律。

综上所述，本研究验证了知识图谱推理在理论和实践中的应用潜力，为未来自动化知识推理的研究探索了新的方向，期待能激发更多关于智能决策支持系统的深入研究。

参考文献

- [1] Kurt D. Bollacker, Kurt Bollacker, Kurt Bollacker, Colin Evans, Colin Evans, Praveen Paritosh, Praveen Paritosh, Tim Sturge, Tim Sturge, Jamie Taylor, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. *null*, 2008.
- [2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.
- [3] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1306–1313, 2010.
- [4] Andrew Carlson, Andrew Carlson, Justin Betteridge, Justin Betteridge, Bryan Kisiel, Bryan Kisiel, Burr Settles, Burr Settles, Estevam R. Hruschka, Estevam R. Hruschka, Tom M. Mitchell, and Tom M. Mitchell. Toward an architecture for never-ending language learning. *null*, 2010.
- [5] Liang Chen, Chen Liang, Jonathan Berant, Jonathan Berant, Quoc V. Le, Quoc V. Le, Quoc V. Le, Quoc V. Le, Quoc V. Le, Kenneth D. Forbus, Kenneth D. Forbus, Ni Lao, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv: Computation and Language*, 2016.
- [6] Rajarshi Das, Rajarshi Das, Rajarshi Das, Arvind Neelakantan, Arvind Neelakantan, David Belanger, David Belanger, Andrew McCallum, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv: Computation and Language*, 2016.
- [7] Matt Gardner, Partha Talukdar, Bryan Kisiel, and Tom Mitchell. 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*, pages 833–838, 2013.
- [8] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 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)*, pages 397–406, 2014.
- [9] Kelvin Guu, Kelvin Guu, J. J. Miller, John J. Miller, Percy Liang, and Percy Liang. Traversing knowledge graphs in vector space. *arXiv: Computation and Language*, 2015.
- [10] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

- [11] Justin Johnson, Justin Johnson, Bharath Hariharan, Bharath Hariharan, Laurens van der Maaten, Laurens van der Maaten, Judy Hoffman, Judy Hoffman, Feifei Li, Li Fei-Fei, C. Lawrence Zitnick, C. Lawrence Zitnick, Ross Girshick, and Ross Girshick. Inferring and executing programs for visual reasoning. *arXiv: Computer Vision and Pattern Recognition*, 2017.
- [12] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [14] Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539, 2011.
- [15] Ni Lao, Jun Zhu, Liu Liu, Yandong Liu, and William W Cohen. Efficient relational learning with hidden variable detection. *Advances in Neural Information Processing Systems*, 23, 2010.
- [16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [17] Arvind Neelakantan, Arvind Neelakantan, Benjamin Roth, Benjamin Roth, Andrew McCallum, and Andrew McCallum. Compositional vector space models for knowledge base completion. *arXiv: Computation and Language*, 2015.
- [18] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509, 2015.
- [19] William Yang Wang and William Cohen. Joint information extraction and reasoning: A scalable statistical relational learning approach. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 355–364, 2015.
- [20] Zhen Wang, Zhen Wang, Zhen Wang, Jianwen Zhang, Jianwen Zhang, Jianlin Feng, Jianlin Feng, Zheng Chen, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. *null*, 2014.
- [21] Wenhan Xiong, Wenhan Xiong, Thien Hoang, Thien Hoang, William Yang Wang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *Conference on Empirical Methods in Natural Language Processing*, 2017.

- [22] Daojian Zeng, Daojian Zeng, Kang Liu, Kang Liu, Siwei Lai, Siwei Lai, Guangyou Zhou, Guangyou Zhou, Jun Zhao, and Jun Zhao. Relation classification via convolutional deep neural network. *null*, 2014.