NARS第一层试写体会

徐升

2017-08-08

以下内容纯属个人观点，所有内容都欢迎质疑和讨论

1:自我介绍

- 我的专业/技术背景：
  - 心理学
  - **自学**编程(python c# matlab javascript)
  - 有一定的深度学习的知识和经验

2:人工智能里的深度学习VSNARS

- 在比较了深度学习和NARS后，我觉得NARS是更好的人工智能方案。换句话说人类的智能更有可能是类似深度学习，还是类似NARS的？
  - NARS和深度学习为代表的机器学习方案都是对人类智能的模拟，深度学习模拟的是行为，NARS模拟的是思维规则[这个说法也许并不准确，因为并没有定论说人类就是按照三段论的方式思考]
  - NARS是简洁优美的[奥卡姆剃刀]，讽刺的是，第一次听到这个名词是从机器学习的专著里。而机器学习这个方向现在的发展是一定程度上背离了这一原则。

3.我为什么对NARS感兴趣

- NARS本身具有重大的价值：
  - 大家发挥想象……
- NARS还具有重要的理论研究价值：
  - 可以帮助我们更好地理解、发现人类智能。
  - 心理学曾经、也一直在对动物（老鼠，鸽子，大猩猩）的研究中获益,那么以NARS为对象研究人类的思维，情感甚至意识也是非常有意义的。

4. NARS：从逻辑到控制

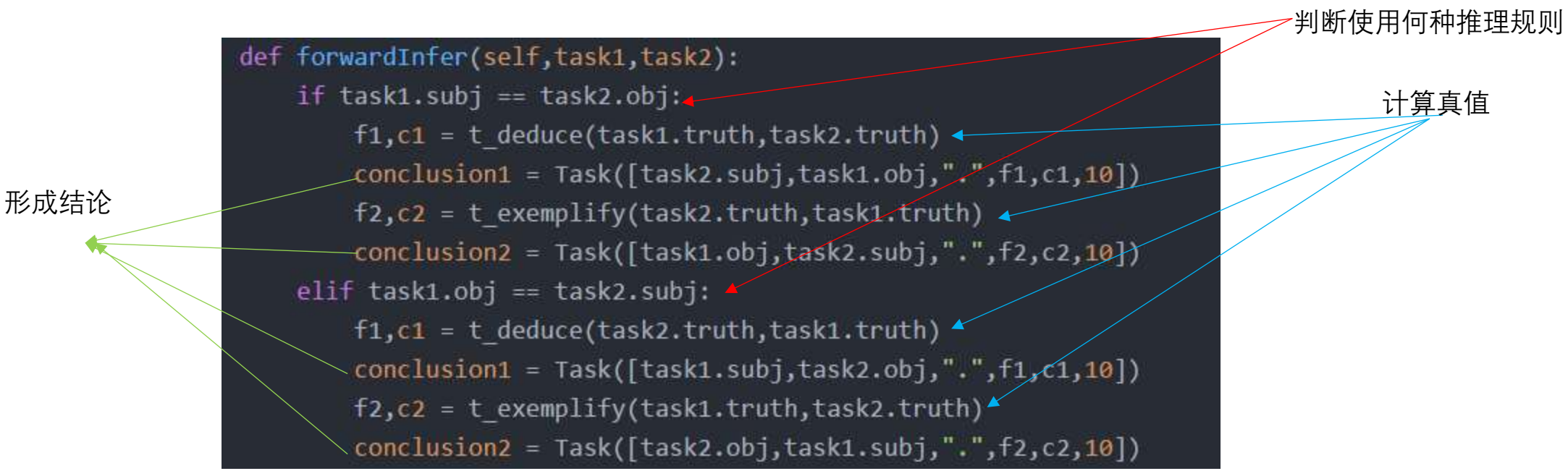
- 在最开始接触NARS的时候，我关注更多地是NARS的逻辑部分
  - 相对来说，逻辑部分更加具体，因为有推理规则，有真值计算公式等
  - 但值得一提的是，我们现在看到的NARS的逻辑部分可能只是冰山露出水面的一角
- 然后，控制部分成了不可回避的一部分
  - 第一层的推理规则十分简单，但掌握这些推理规则并不能使这样一个智能系统运转
  - 这也是我试写第一层的主要动机

5.试写NARS的必要性与可行性

- 取决于目的：
  - 想要达到什么效果，以及现有的nars能不能满足[重写完整的，独立的NARS非常非常困难]
  - 而我只是想检验一下自己的NARS的理解。

6.试写NARS第一层：逻辑部分

- 这一部分基本就是把nars第一层的推理规则实现，包括真值计算。
- 比如当输入a->b , b->c时，能够得到相应的两个结论和其真值。
- 每一对陈述都可以产生两个结论。



6.试写NARS第一层：逻辑部分

- 这一部分基本就是把nars第一层的推理规则实现，包括真值计算。
- 比如当输入a->b , b->c？时，能够得到相应的两个结论和其真值。
- 每一对陈述都可以产生两个结论。

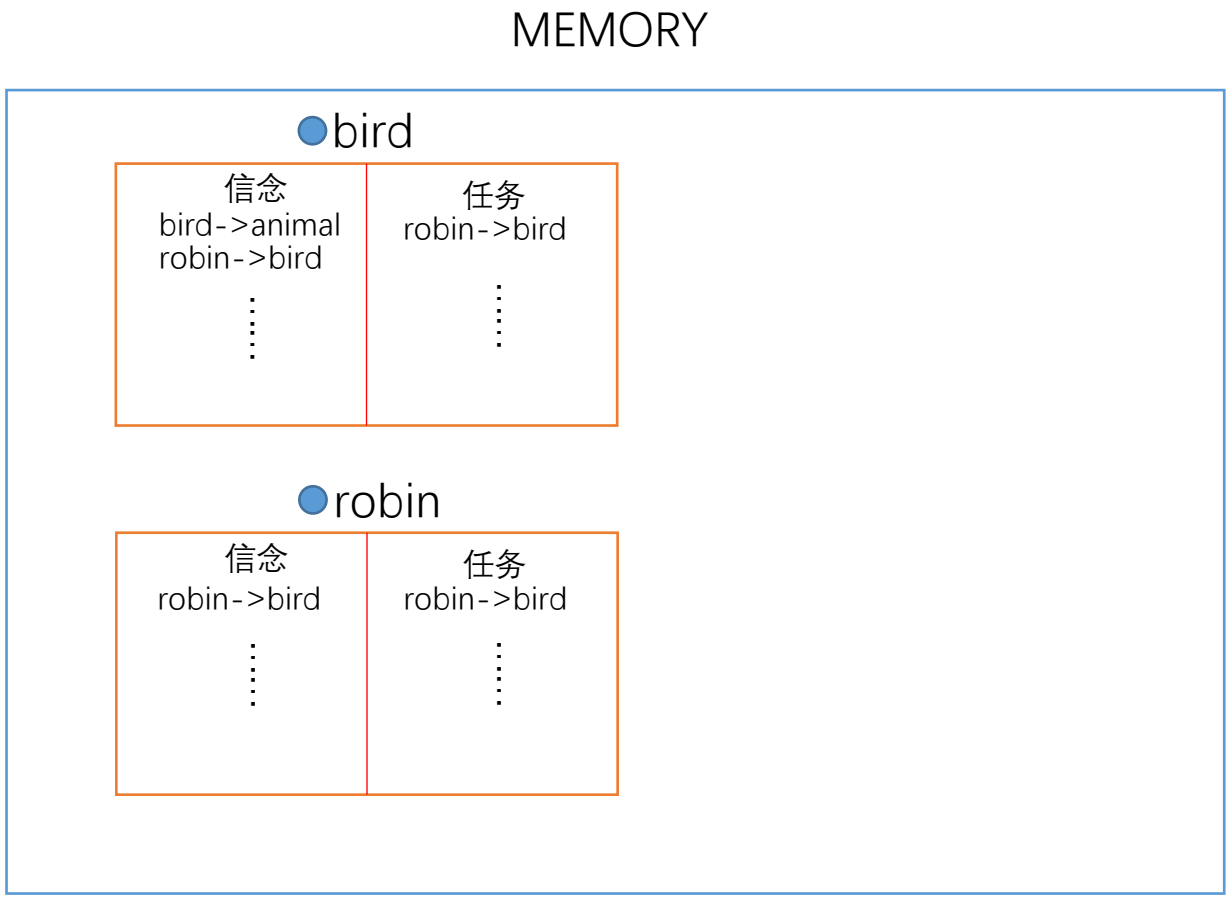
```
def backwardInfer(self,belf,task):
    t1 = set([belf.subj,belf.obj])
    t2 = set([task.subj,task.obj])
    pkey = t1.intersection(t2)
    u1 = list(t1.difference(pkey))[0]
    u2 = list(t2.difference(pkey))[0]
    derivedQuestion1 = Task([u1,u2,"?",1.0,0.9,10])
    derivedQuestion2 = Task([u2,u1,"?",1.0,0.9,10])
    derivedQuestion1.parent = task
    derivedQuestion2.parent = task
    self.taskin(derivedQuestion1)
    self.taskin(derivedQuestion2)
```

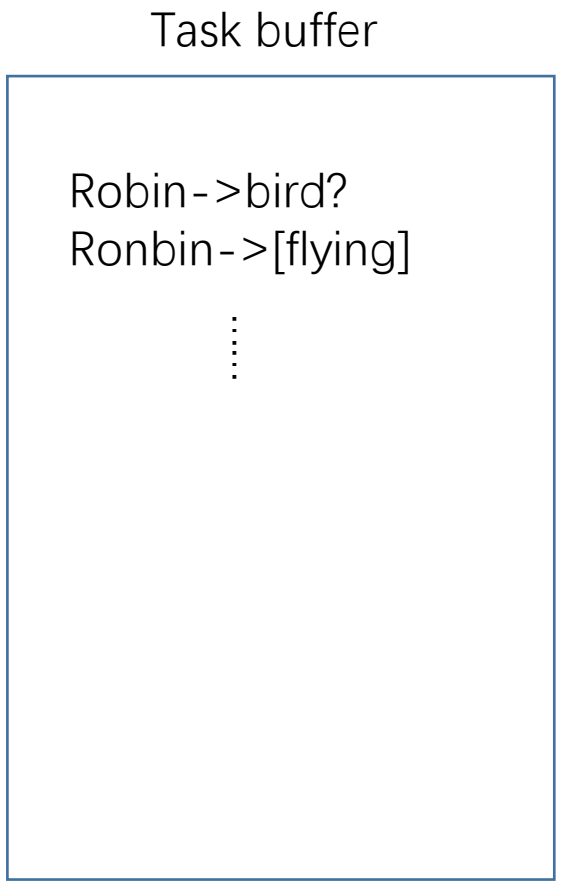
7.试写NARS第一层：控制部分

- 1.控制部分的基本结构
  - 一个通用的数据结构是这样一张有容量上限的键值对表，键是概念、陈述或者问题；值是其优先度。这张表的主要功能是：根据优先度的分布取出一个键，并在达到容量上限的时候删除一个键
  - 记忆[memory]
    - 两级表。第一级存储的是概念，第二级是这个概念所包含的所有陈述
  - 任务缓冲器[task buffer]
    - 一级表，存储了当前的任务，比如用户输入，或者系统产生的。

7.试写NARS第一层：控制部分

- 1.控制部分的基本结构





7.试写NARS第一层：控制部分

As a reasoning system, the running process of NARS consists of an unlimited number of inference steps, each consists of the following procedure:

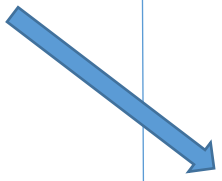
(1) get a concept from the memory,
(2) get a task from the concept,
(3) get a belief from the concept,
(4) derive new tasks from the selected task and belief,
(5) put the involved items back into the corresponding bags,
(6) put the new tasks into the corresponding bags.

7.试写NARS第一层：控制部分

- 2.系统工作流程
  - 1.无论是系统推理产生[正\反向推理]或者用户输入首先进入task buffer
  - 2.从task buffer中取出一个任务
  - 3.取出任务的预处理
    - 如果是陈述
      - 变成系统的一个信念[加入memory]
      - 如果有相同的陈述存在，revise it
      - 回答一个直接的问题[回答的是tf和memory中的问题吗？]
    - 如果是问题
      - 如果可以，直接回答

7.试写NARS第一层：控制部分

- 2.系统工作流程
  - 1.无论是系统推理产生[正\反向推理]或者用户输入首先进入task buffer
  - 2.从task buffer中取出一个任务
  - 3.取出任务的预处理
  - 4.取出的任务的继续处理 robin->bird
    - 如果是陈述，进行一步正向推理
    - 如果是问题，进行一步反向推理



1.根据优先度选择概念 robin或者bird。[bird]

2.在memory里找到bird，在bird下根据优先度选择一个信念

3.

7.试写NARS第一层：控制部分

- 2.系统工作流程
  - 1.无论是系统推理产生[正\反向推理]或者用户输入首先进入task buffer
  - 2.从task buffer中取出一个任务
  - 3.取出任务的预处理
  - 4.取出的任务的继续处理
  - 5.将生成的新任务或者信念加入到task buffer中

7.试写NARS第一层：控制部分

- 2.系统工作流程
  - 1.无论是系统推理产生[正\反向推理]或者用户输入首先进入task buffer
  - 2.从task buffer中取出一个任务
  - 3.取出任务的预处理
  - 4.取出的任务的继续处理
  - 5.将生成的新任务或者信念加入到task buffer中
  - **期间，所有使用过的信念和任务的优先度都会做相应的调整
    - 随时间衰减
    - 当前情境中涉及的项目[概念，信念]优先度更高
    - 当前推理步的结果将作为反馈调整优先度

7.试写NARS第一层：控制部分

- 2.系统工作流程
  - 1.无论是系统推理产生[正\反向推理]或者用户输入首先进入task buffer
  - 2.从task buffer中取出一个任务
  - 3.取出任务的预处理
  - 4.取出的任务的继续处理
  - 5.将生成的新任务或者信念加入到task buffer中
  - **期间，所有使用过的信念和任务的优先度都会做相应的调整
  - **在做推导时，我们应该记录一定长度其“父母”信念，这样才能避免
    - 循环推理；revise or choice

8:总结

- 前面的内容应该足够可以大体描绘出NARS第一层的样子。
- 但是实际上在真正动手的时候还是会遇到很多问题。
- 所以现在回过头来看，我想能更好地回答这样一个问题：
  - 自己写nars的必要性？
  - 我觉得值得尝试[写第一层，或者前面几层]

谢谢
欢迎指正