文件分布:

checkpoint: 用于保存训练过程中产生的预训练文件

data-end2end-triple: 存放数据集

Annotation: 标注文件xml

images: 图片

triples\_train.tsv: 训练集

triples\_test.tsv: 测试集

tsv文件的格式: img\_id obj\_id sub\_id rel\_id

img\_id为图像的编号

obj\_id为物体1在此图像对应标注文件中的编号/次序

sub\_id为物体2在此图像对应标注文件中的编号/次序

rel\_id为物体1物体2之间关系对应的编好

models：reasoning.py：逻辑规则推理等部分的实现

rel\_models.py：神经网络的实现

train.py：整体训练流程

1.train.py

设置lr, epoch\_num, loss\_fuction, optimizer等

开始每个epoch的训练：

各种评价指标赋初值。

在每个epoch的训练中，将每张图像均训练一遍，从train\_set[]中获取每张图像的训练数据。

* Img为cv2.imread方法读取的图像；
* ann为此图像中所有物体信息的列表，其中每个物体的信息如下{'box': [xmin, ymin, xmax, ymax], 'name': obj\_name}坐标信息和物体名称；
* op\_list是一个根据数据集生成的
* answer为一个向量，其中只有正确的物体对应的编号位置处为1，其余均为0
* type由数据集中的type得到

紧接着就是训练流程，acc和F1等值的计算。

数据集的构建过程*class TripleDataset(Dataset)*的作用：

\_\_init\_\_：根据tsv文件中每行的值形成(img\_id, obj\_id, sub\_id, rel\_id)，并添加到triples\_pos中。例如以判断开关闭合实验为例，根据数据集的xml标注文件，找出每张图像中开关柄(switch\_handle)和开关整体(switch)的编号，并将编号以(img\_id obj\_id sub\_id rel\_id)四元组的形式存储在triples\_up\_train.tsv与triples\_up\_test.tsv文件中，预备生成模型的dataset。随后，将triples\_pos中的每组关系均转换成question。每个question中op\_list列表就是用于模型最后判断时的计算顺序。

\_\_getitem\_\_：通过调用imgtool中load\_img方法解析每个xml标注文件中的物体信息，返回给训练时的数据。

2.reasoning.py

reasoning.py主要分成三个模块

* Predictor集成了谓词与相关神经网络的计算
* Executor实现了每个operator的运算过程
* Reasoning负责整体逻辑规则的顺序计算及逻辑判断

此处以判断开关是否闭合为场景进行描述。

首先定义了concept2id\_name、rel2id、id2rel这三个表。

* concept2id\_name将需要关注的物体(开关整体(switch)'和'开关柄(switch\_handle)')编号设为0、1，其余不需要关注的物体的编号设置为2；
* rel2id将关注的位置进行编号。
* id2rel与rel2id含义相反。

concept2id\_name = {'开关整体(switch)': 0, '开关柄(switch\_handle)': 1, '其他(others)': 2}  
rel2id = {'up': 0, 'down': 1}  
id2rel = {0: 'up', 1: 'down'}

1. **Predictor模块：**

Predictor模块中主要实现数据输入到神经网络中的过程。

* \_\_init\_\_方法：实例化神经网络net\_con = OnClassify\_v1()，OnClassify\_v1()为rel\_models.py中定义的神经网络。IoU指采用bbox相互重叠的办法来判断的策略，此处可忽略。
* on\_classify方法：使用神经网络判断物体A和物体B存在指定关系(连接关系)。首先拼接A、B物体的坐标，坐标从ann中获取。拼接后的特征为[xmin\_a, ymin\_a, xmax\_a - xmin\_a, ymax\_a - ymin\_a, xmin\_b, ymin\_b, xmax\_b - xmin\_b, ymax\_b - ymin\_b]，即为feature: [x1, y1, w1, h1, x2, y2, w2, h2]，此特征维度为[1, 8]。随后，将拼接后的特征送入神经网络，神经网络的返回值y\_pred代表物体A、B间存在连接关系的概率。

1. **Executor模块：**

Executor模块负责具体实现每个operator的运算过程。主要实现了根据物体name和index的过滤filter操作。同时，构建了神经符号学的中间符号表示过程，即构建了concept\_matrix概念矩阵与relate\_matrix关系矩阵这两个矩阵。

* \_make\_concept\_matrix构建概念矩阵concept\_matrix：

概念矩阵的作用是将一个图像中所有物体的物品种类在转换为种类编号后形成矩阵。concept\_matrix的维度为N\*M，M为图像中物体的最大总数，N为物体所属类的总数。其中<i, j>的含义为此图片中第j个物体是第i种物体；concept\_matrix的每行代表每种物体在图像中物体的次序，包含物体的数量与物体的位置编号，如[0, 1, 0, 0, 0, 0,......]表示图像中第2个物体是该种物体。concept\_matrix的每列代表图像中第i个物体对应的种类编号。

构建concept\_matrix时，从ann中依次读取数据，进行构建。

* \_make\_relate\_matrix

关系矩阵的作用是描述了物体之间存在指定关系(开关闭合)的概率。relate\_matrix的维度为N\*N，其中N为图像中物体的最大总数。relate\_matrix中<i, j>处的值代表第i个物体和第j个物体间是开关闭合关系的概率。

在构建relate\_matrix时，当在训练模式下，将A、B两物体的信息传入，调用Predictor模块中的神经网络，将特征输入神经网络，将输出值填写到relate\_matrix的对应位置。

* \_get\_concept\_mask

此处构建了一个mask，目的是为了便于获取concept\_matrix中需要的向量，而减少相应的运算。函数的输入值为concept\_index(物体种类的编号)，返回的是这种物体在该图片中出现的情况，这就是一个mask。

* \_get\_relate\_mask

与\_get\_concept\_mask方法相似，返回的是整个relate\_matrix矩阵。

* filter\_obj\_name

图像中单个物体的位置与排序信息与物体间的位置信息通过神经网络后初步转换为了concept\_matrix和relate\_matrix。但两个矩阵中仅仅存放了物体的所有信息，并且是全部物体信息混在一起存放。因此，需要进行相关操作来筛选出符合要求的物体。filter\_obj\_name方法通过调用\_get\_concept\_mask方法获得相应的向量。

1. **Reasoning模块：**

Reasoning模块根据可微逻辑规则中词语的顺序，依次分析由模型自动生成的自然语言问句的操作词和参数，根据每个操作词的类型调用相关模块进行计算，推理出模型的最终输出，并将推理结果与真实得分进行比较后得到实验操作得分。

针对每个神经谓词与可能含有这种关系的所有物体，引入一个语句来描述存在这个关系的组合。语句由操作词构成，设置5个操作词分别代表物体、关系和存在，语句的的构成形式如下:

*{'op': 'objects', 'param': ''},*

*{'op': 'filter\_name', 'param': '物体A'},*

*{'op': 'relate', 'param': '神经谓词'},*

*{'op': 'filter\_name', 'param': '物体B'},*

*{'op': 'exist', 'param': ''}*

当操作词为object时，构建了一个全为1的行向量，目的是表示此时是对物体进行的操作。

当操作词为filter\_name时，调用关系逻辑执行层中的过滤操作，从概念矩阵中抽取出种类为该物体在图片标注信息中的编号，返回的是一个行向量，对应位置为1即表示为此物体。

当操作词为relate时，调用关系逻辑执行层中的过滤操作，从关系矩阵中获取当前此神经谓词对应关系表。此时返回的是一个矩阵。

在随后的回答与推理中，根据每个子句返回的结果进行运算，将filter\_name操作返回的代表物体编号的结果与relate返回的矩阵相乘，得到的向量即为与该物体存在有语句中relate关系的物体，这就是模型经过逻辑规则推理层推理产生的答案。