**基于深度学习的自动问答**

1. **任务说明**

给定一个问句，在知识库中找出正确答案最相关的一个三元组。打算采用BiLSTM将问句和知识库三元组映射成同一个语义空间中的向量，通过求余弦相似度对三元组进行排序，从而确定正确答案最相关的三元组。

1. **开发环境**

操作系统：Window10、内存64G、CPU E3-1225

开发语言：python3.6

开发环境：PyCharm

Python包：tensorflow1.8.1、numpy1.14.4、nltk3.3、fuzzywuzzy0.16.0、tqdm4.23.4

(ps:下载包时，可以通过以管理员身份运行cmd，然后通过pip工具进行下载，如 pip install tensorflow; pip install numpy; pip install nltk;pip install fuzzywuzzy; pip install tqdm;)

1. **数据准备**

**问题集**：SimpleQuestion（分为训练集、验证集、测试集）

格式如下： 由一个答案三元组以及问题组成，用\t分割

www.freebase.com/m/0jtw9c www.freebase.com/film/writer/film www.freebase.com/m/05szq8z what film is by the writer phil hay?

**知识库**：

Freebase2M：由三元组构成，共包含2M实体，10843106条三元组

www.freebase.com/m/0n1vy1h www.freebase.com/people/person/gender www.freebase.com/m/05zppz

FB5M.name.txt：包含实体对应的name、alias属性，本次只考虑name属性

<fb:m.0f8v12b> <fb:type.object.name> "blue christmas"

**词向量**：glove.6B.100d.txt，包含400000个词、每一个词向量维度为100

glove.6B.50d.txt，包含400000个词、每一个词向量维度为50

1. **数据预处理**

**4.1 未登录词处理说明**

本次使用的词向量文件时glove，其包含400000个词，但在知识库和问题集的处理过程中，会碰到一些glove中没有出现过的词，为了让我们的神经网络模型能够处理这些未知的词条，我们需要维护一个这些词的一致的向量。常用的方式有以下两种：1、将所有这些未登入词统一同一个单一的<UNK>向量代替。2、对每一个未登入词都设置一个单独的向量来代替。为简单起见，本次采用方式1进行处理。

**4.2 知识库预处理**

为了便于模型训练时随机产生反例三元组，对FB2M知识库进行预处理，将其知识库中的实体转换成name，并对三元组进行分词、转换成词ID操作，并将结果使用pickle模块保存成2进制文件，文件见data/freebase/fb2m\_index.pkl。格式示意如下，

三元组分词之后：

('brandon kramer', 'people person gender', 'male')

转换成词ID之后：

([10483, 14252], [70, 900, 5370], [2324])

**4.3训练集预处理**

将SimpleQuestion训练集中的答案三元组和问题进行分词、转换成词ID操作，结果使用pickle模块保存成2进制文件，见data/freebase/traindata.pkl。格式示意如下，

[[103, 15, 1, 540, 1111, 60], [1111], [540, 983, 162, 4777], [60388]]

列表中第一个元素是问题，其后是答案三元组subject、relation、object。

**4.4验证集、测试集预处理**

考虑到时间、内存空间的因素，本次在模型评估时预先对验证集和测试集中的每一个问题基于一定方式产生80个候选三元组，从而减少问题的比较空间，加快运行速度和减少内存消耗。候选三元组生成方式为：通过对问题进行实体识别，初步筛选出一定数量的候选实体，在将候选实体相关的三元组作为候选三元组。

其中实体识别步骤中，通过将问句划分成多个词块(token)，查找包含词块(token)的实体集，通过按词块(token)长度、实体名字和词块相似度对实体集进行排序，从而构成候选实体集。相关代码见entity\_linking.py。

通过该种方式为验证集、测试集获取到80个候选三元组，对验证集和测试集进行分词、转换成词ID操作，结果见/data/test\_sample.pkl、/data/valid\_sample.pkl 。其中验证集包括10807个问题，测试集包括21615个问题。数据格式为，

[ques\_pad, ans\_pad, answer\_index, question\_len, can\_triplet\_len]

ques\_pad 表示问题列表，ans\_pad表示候选三元组列表，answer\_index表示问题正确答案在候选三元组的位置，question\_len表示问题长度，can\_triplet\_len表示候选三元组长度。

1. **模型构建**



模型图如上所示，模型大致分为四层：input层、embedding层、BiLSTM层、Maxpooling层

**5.1 input层**

self.inputQuestions = tf.placeholder(tf.int32, shape=[None, self.num\_step])  
self.inputQuestions\_len = tf.placeholder(tf.int32,shape=[None])  
self.inputTrueAnswers = tf.placeholder(tf.int32, shape=[None, self.num\_step])  
self.inputTrueAnswers\_len = tf.placeholder(tf.int32, shape=[None])  
self.inputFalseAnswers = tf.placeholder(tf.int32, shape=[None, self.num\_step])  
self.inputFalseAnswers\_len = tf.placeholder(tf.int32, shape=[None])

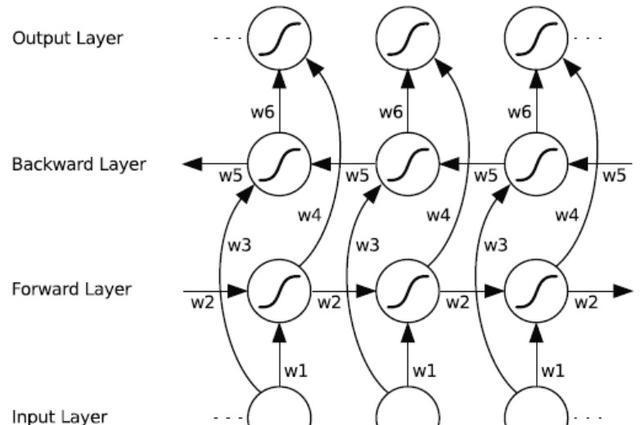
用于问题、正例三元组答案、反例三元组答案输入。

**5.2 embedding 层**

tfEmbedding = tf.Variable(self.embeddings, trainable=True, name="W",dtype=tf.float32)  
questions = tf.nn.embedding\_lookup(tfEmbedding, self.inputQuestions)  
trueAnswers = tf.nn.embedding\_lookup(tfEmbedding, self.inputTrueAnswers)  
falseAnswers = tf.nn.embedding\_lookup(tfEmbedding, self.inputFalseAnswers)

通过tf.nn.embedding\_lookup将词ID转化为词向量，其中tfEmbedding用于存放词向量。

**5.3 BiLSTM层**

****

with tf.variable\_scope("LSTM\_scope", reuse=None):  
 self.question1 = self.biLSTMCell(questions, self.inputQuestions\_len, self.hidden\_neural\_size) # shape = (bitchsize,num\_step,n\_hidden)  
 question2 = tf.nn.tanh(self.max\_pooling(self.question1))  
with tf.variable\_scope("LSTM\_scope", reuse=True):  
 self.trueAnswer1 = self.biLSTMCell(trueAnswers, self.inputTrueAnswers\_len, self.hidden\_neural\_size)  
 trueAnswer2 = tf.nn.tanh(self.max\_pooling(self.trueAnswer1))  
 self.falseAnswer1 = self.biLSTMCell(falseAnswers, self.inputFalseAnswers\_len, self.hidden\_neural\_size)  
 falseAnswer2 = tf.nn.tanh(self.max\_pooling(self.falseAnswer1))

将embeding之后的问句向量和三元组向量传入到BiLSTM中，将BiLSTM的输入通过max\_pooling层、tanh核函数获取最终的向量表示。

**5.4 LOSS函数**

with tf.name\_scope("loss"):  
 losses = tf.maximum(zero, tf.subtract(tfMargin, tf.subtract(trueCosSim, falseCosSim)))

本次训练的loss函数定义如下：

margin 是超参数，是问题和正例三元组之间的余弦值，是问题和反例三元组之间的余弦值。

1. **模型训练及评估**

**6.1超参数定义**

hidden\_neural\_size = 75 # 输入问句的lstm隐藏单元的个数  
embed\_dim = 50 # 预训练的词向量维度  
keep\_prob = 1.0 # dropout rate  
margin = 0.5 # loss 函数中的超参数  
lr = 0.05 # 学习速率  
lrDownRate = 0.5 # 学习速度下降速度  
lrDownCount = 4 # 学习速度下降次数  
batch\_size = 100 # 每批次大小  
num\_step = 36 # 问句对应的lstm 步长，即问句最长长度  
num\_epochs = 60 # 每次学习速度指数下降之前执行的完整epoch次数  
  
numclass = 80 # 正例和反例的数目比，或者候选集和正确答案之间的比列 训练时1，测试时80  
l2\_reg\_lambda = 0  
embeddings\_trainable = True  
max\_grad\_norm = 5 # 设置梯度的最大范数,用于控制梯度膨胀，如果梯度向量的L2模超过max\_grad\_norm，则等比例缩小  
  
word2vec\_file = './data/glove/glove.6B.100d.txt'  
  
trainingFile = "./data/SimpleQuestion/traindata.pkl"  
developFile = "./data/valid\_sample.pkl"  
testingFile = "./data/test\_sample.pkl"  
  
saveFile = "newModel/savedModel"  
trainedModel = "trainedModel/savedModel"

**6.2训练和评估**

for question, trueAnswer, falseAnswer, \_a, \_b, \_c in data\_loader.trainingBatchIter(fb2m\_index, trainset):  
 tqs.append(question), tta.append(trueAnswer), tfa.append(falseAnswer), tqsl.append(\_a), ttal.append(\_b), tfal.append(\_c)  
for question, trueAnswer, falseAnswer, \_a, \_b, \_c, in tqdm(zip(tqs, tta, tfa, tqsl, ttal, tfal),desc="Training epoch "+ str(epoch)+" lr="+str(lr)+" ", total=len(tqs)):  
 startTime = time.time()  
 feed\_dict = {  
 lstm.inputQuestions: question,  
 lstm.inputTrueAnswers: trueAnswer,  
 lstm.inputFalseAnswers: falseAnswer,  
 lstm.inputQuestions\_len: \_a,  
 lstm.inputTrueAnswers\_len: \_b,  
 lstm.inputFalseAnswers\_len: \_c,  
 lstm.keep\_prob: config.keep\_prob  
 }  
 \_, step, trueCosSim, falseCosSim, loss,question1,trueAnswer1, falseAnswer1 = \  
 sess.run([trainOp, globalStep, lstm.trueCosSim, lstm.falseCosSim, lstm.loss ,lstm.question1, lstm.trueAnswer1,lstm.falseAnswer1], feed\_dict)

with open(config.resultFile, 'w') as file:  
 \_correct\_num, \_rr = 0,0  
 for question, answer, lable, question\_len, answer\_len in tqdm(data\_loader.testingBatchIter(test\_sample[0], test\_sample[1], test\_sample[2], test\_sample[3],test\_sample[4]),  
 desc="测试中》》》",total=testsize/config.batch\_size):  
 feed\_dict = {  
 lstm.inputTestQuestions: question,  
 lstm.inputTestQuestions\_len: question\_len,  
 lstm.inputTestAnswers: answer,  
 lstm.inputTestAnswers\_len:answer\_len,  
 lstm.keep\_prob: config.keep\_prob,  
 lstm.input\_y: lable  
 }  
 \_, scores, correct\_num, rr = sess.run([globalStep, lstm.result, lstm.correct\_num, lstm.rr], feed\_dict)  
 \_correct\_num +=correct\_num  
 \_rr += rr  
 print("test acc {:g}, test mrr {:g}".format(\_correct\_num / testsize, \_rr / testsize))

通过遍历训练集80epoch，使用100d词向量的模型在测试集上的正确率达到70.24%，MRR为0.8036；使用50d词向量的模型在测试集上的正确率达到69.12%，MRR为0.7923。

**7、源文件说明**

data/freebase：知识库相关数据

data/glove：预训练的词向量

data/SimpleQuestion：问题集

newModel：新训练好的模型

trainedModel-100d：已训练的模型，使用词向量100d

trainedModel-50d：已训练的模型，使用词向量50d

config.py：配置文件

data\_helper.py：数据预处理相关函数

entity\_linking.py：实体链接

data\_loader.py：模型训练或测试时数据加载相关函数

lstmRNN.py：模型定义

main：程序入口

(ps:程序运行时，记得将当前目录切换到DeepQA目录下，否则可能会报错)

**8、操作步骤**

1、安装python，安装包在软件文件夹中，点击下一步安装即可，记得勾选add path。

2、安装pyhton包时，以管理员身份运行cmd，然后在cmd中依次输入 pip install tensorflow; pip install numpy; pip install nltk;pip install fuzzywuzzy; pip install tqdm;

3、在cmd中，通过cd E:\websoft\DeepQA 进入到源代码目录中，然后

Python ./main.py 执行程序。

**基于模板匹配的自动问答**

**1、任务说明**

实现一个通过模板匹配方式进行自动问答的程序。给定一个问句，通过对问句进行语义解析，将问句和预定义的问句模板进行匹配，从而将问句转化为SPARQL查询语句，然后通过在知识库中查找得出最后的结果。语义解析主要通过分词和正则表达式匹配的方式进行。

**2、开发环境**

操作系统：Window10、内存64G、CPU E3-1225

开发语言：python3.6

开发环境：PyCharm

数据库：Virtuoso7.2.4(Open Source Edition)

Python包：jieba0.39、refo0.13、SPARQLWrapper1.8.2

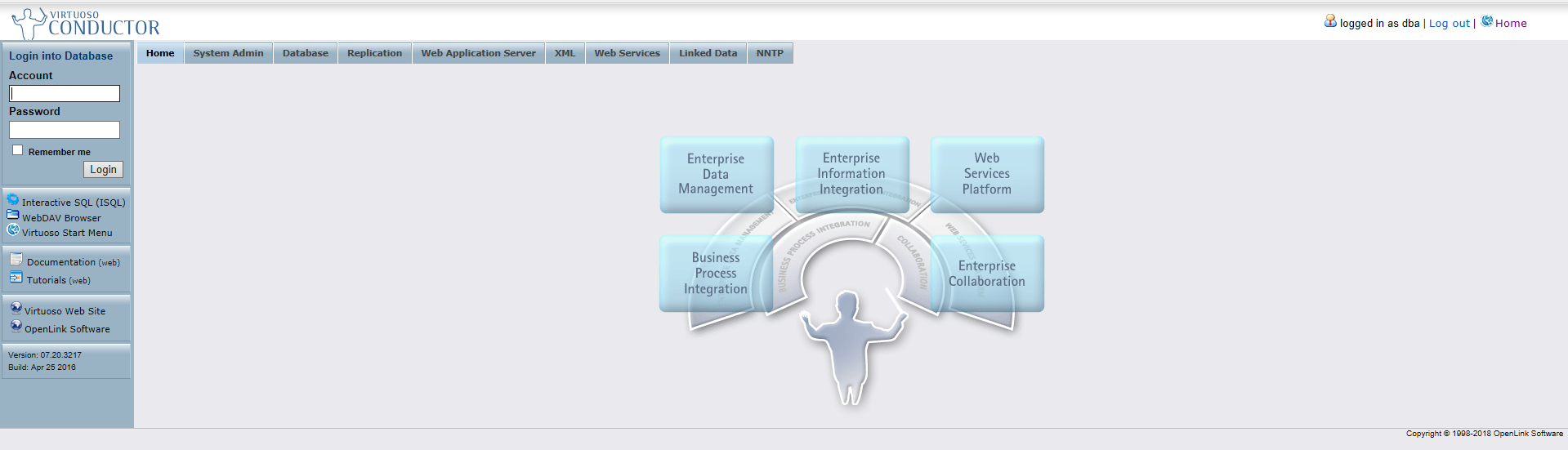
1. **Virtuoso安装和数据导入**

软件下载地址<http://vos.openlinksw.com/owiki/wiki/VOS/VOSDownload>

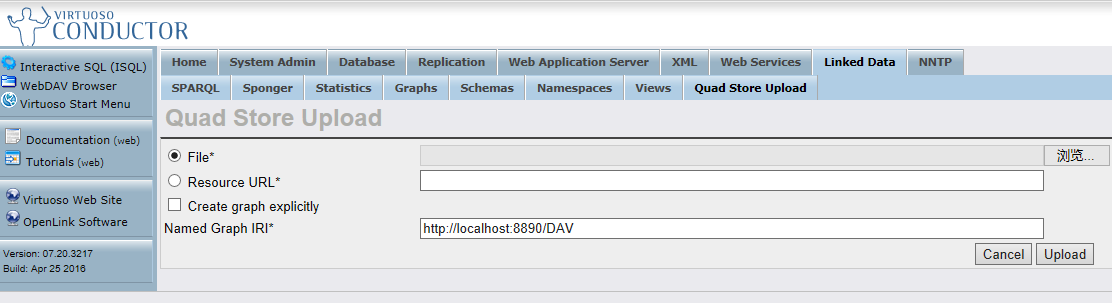
软件安装教程<http://vos.openlinksw.com/owiki/wiki/VOS/VOSUsageWindows#Getting%20Started%20with%20the%20VOS%20Binary%20Distribution%20for%20Windows>

Virtuoso下载安装完成后，以管理员权限开启cmd，在命令行中输入virtuoso-t -? 可以获取virtuoso帮助信息。通过cd %VIRTUOSO\_HOME%/database 进入数据库所在目录，virtuoso-t +service create +instance “New Instance Name” +configfile virtuoso.ini 创建virtuoso服务。通过virtuoso-t +instance “New Instance Name” +service start 启动virtuoso服务。

通过<http://localhost:8890/conductor/>可以登录virtuoso图形管理员界面，默认账号和密码都是dba。



登入成功之后，选择LinkedData菜单—>选择Quad Store Upload菜单，然后可以将本次的知识库上传到virtuoso中，其中在Named Graph IRI中可以对图数据名字进行设置。知识库在./data/kg\_demo\_movie.nt中。



1. **知识库介绍**

本次使用的知识库来源于网上公布的一个中文的关于电影领域的知识库，文件在./data/kg\_demo\_movie.nt中。其基本统计数据如下：

1. 演员数量：505人
2. 电影数量：4518部
3. 电影类型：19类
4. 人物与电影的关系：14451
5. 电影与类型的关系：7898

演员的基本信息包括：姓名、英文名、出生日期、死亡日期、出生地、个人简介。电影的基本信息包括：电影名称、电影简介、电影评分、电影发行日期、电影类型。

1. **问题模板定义**

本次问答中分词和实体识别（人名和电影名）使用jieba完成。通过添加外部字典的方式，来正确识别出问句中出现的人名和电影名。./external\_dict中movie\_title.txt是电影名字典、./person\_name.txt是人名字典。使用REFO来进行模板的匹配，REFO是一个对象级别的正则表达式匹配工具。

在模型中，定义了十种类型的问题，下面将分别解释问题类型以及匹配规则。

**类型1**：某演员演了什么电影?

匹配规则：

Rule(condition\_num=2, condition=person\_entity + Star(Any(), greedy=False) + movie + Star(Any(), greedy=False), action=QuestionSet.has\_movie\_question)

即出现演员实体、电影等movie类关键词。

**类型2**：某电影有哪些演员出演？

匹配规则：

Rule(condition\_num=2, condition=(movie\_entity + Star(Any(), greedy=False) + actor + Star(Any(), greedy=False)) | (actor + Star(Any(), greedy=False) + movie\_entity + Star(Any(), greedy=False)), action=QuestionSet.has\_actor\_question),

即出现电影实体、出演等actor对象关键词

**类型3**：演员A和演员B合作出演了哪些电影?

匹配规则：

Rule(condition\_num=3, condition=person\_entity + Star(Any(), greedy=False) + person\_entity + Star(Any(), greedy=False) + (movie | Star(Any(), greedy=False)), action=QuestionSet.has\_cooperation\_question),

即出现两个演员实体、电影等movie对象关键词。

**类型4**：某演员参演的评分大于X的电影有哪些?

匹配规则：

Rule(condition\_num=4, condition=person\_entity + Star(Any(), greedy=False) + compare + number\_entity + Star(Any(), greedy=False) + movie + Star(Any(), greedy=False), action=QuestionSet.has\_compare\_question),

即出现演员实体、大于等compare对象关键词、数字实体、电影等movie对象关键词。

**类型5：**某演员出演过哪些类型的电影

匹配规则：

Rule(condition\_num=3, condition=person\_entity + Star(Any(), greedy=False) + category + Star(Any(), greedy=False) + movie, action=QuestionSet.has\_movie\_type\_question),

即出现演员实体、类型等category对象关键词、movie对象关键词。

**类型6：**某演员出演的XX类型电影有哪些

匹配规则：

Rule(condition\_num=3, condition=person\_entity + Star(Any(), greedy=False) + genre + Star(Any(), greedy=False) + (movie | Star(Any(), greedy=False)), action=QuestionSet.has\_specific\_type\_movie\_question),

即匹配演员实体、电影类型genre。

**类型7：**某演员出演了多少部电影。

Rule(condition\_num=3, condition=person\_entity + Star(Any(), greedy=False) + several + Star(Any(), greedy=False) + (movie | Star(Any(), greedy=False)), action=QuestionSet.has\_quantity\_question),

即匹配演员实体、多少等several对象关键词、电影等movie关键词。

**类型8：**某演员是喜剧演员吗？

匹配规则：

Rule(condition\_num=3, condition=person\_entity + Star(Any(), greedy=False) + comedy + actor + Star(Any(), greedy=False), action=QuestionSet.is\_comedian\_question),

即匹配演员实体、喜剧等comedy类关键词、演员等actor类关键词。

**类型9**：某演员的生日/出生地/英文名/简介

匹配规则：

Rule(condition\_num=3, condition=(person\_entity + Star(Any(), greedy=False) + (when | where) + person\_basic + Star(Any(), greedy=False)) | (person\_entity + Star(Any(), greedy=False) + person\_basic + Star(Any(), greedy=False)), action=QuestionSet.has\_basic\_person\_info\_question),

即匹配演员实体、when针对生日匹配时间词、where针对出生地匹配哪里等地点询问词、英文名等person\_baisc类关键词。

**类型10**：某电影的简介/上映日期/评分

匹配规则：

Rule(condition\_num=2, condition=movie\_entity + Star(Any(), greedy=False) + movie\_basic + Star(Any(), greedy=False), action=QuestionSet.has\_basic\_movie\_info\_question)

即匹配电影实体、简介/上映日期/评分等movie\_basic类关键词。

1. **源文件说明**

./data/kg\_demo\_movie.nt: 知识库。

./external\_dict：演员名字和电影名字字典。

question.txt：预设的一些问题。

query\_mian.py：程序主入口。

question2sparql.py：用于将自然语言问句转化为对应的SPARQL语句。

question\_temp.py：用于定义问题类型、问题模板、匹配规则。

sparql\_endpoint.py：用于向virtuoso提交SPARQL查询，并处理返回的结果。word\_tagging.py：利用jieba分词，并将问句转化为Word对象。

**7、操作步骤**

1、安装python，安装包在软件文件夹中，点击下一步安装即可，记得勾选add path。

2、安装pyhton包时，以管理员身份运行cmd，然后在cmd中依次输入 pip install jieba、pip install refo、pip install SPARQLWrapper。

1. 3、安装virtuoso，安装包在软件文件夹中，同时安装软件文件夹中的64-bit Microsoft Visual C++ 2010 Redistributable Package ，详情见“3、Virtuoso安装和数据导入”。

3、在cmd中，通过cd E:\websoft\TemplateQA 进入到源代码目录中，然后

Python ./query\_main.py 执行程序。