# web exp2 report


### PB18111760 王嘉梁

### PB18111782 徐瑞


# 一、实验要求

实现一种利用文本信息的知识图谱的补全方法。需要使用实体和关系描述的文本信息，可以使用tranX等模型进行三元组结构信息的更多利用。

# 二、模型构建

首先使用word2vec对所给实体描述和关系描述进行特征向量抽取，得到了每一个“数字”代表的词的特征向量。（实体描述和关系描述中同一个“数字”代表的实际词汇是一致的）。由于每个实体是由不止一个词汇进行描述的，所以需要对所得词向量进行求和和归一化，才能得到更好描述每个实体特征的类似“doc2vec”的特征向量。

```
def word_2_vec(epoch):
    sentences = word2vec.LineSentence(data_only_path)
    model = word2vec.Word2Vec(
        sentences, hs=1, min_count=1, window=3, vector_size=100, epochs=epoch)
    #model.save(word2vec_model_path)
    model.wv.save_word2vec_format(word2vec_model_path, binary=False)
    # 计算实体描述的embedding
    with open(entity_path, 'r') as f_1,open(entity_full_path,'w') as f_2:
        for line in f_1.readlines():
            vec = 0
            words = line.strip().split('\t')[1].split(" ")
            for word in words:
                vec += model.wv[word]
            vec = vec / np.linalg.norm(vec)
            f_2.write(line.strip().split('\t')[0])
            embedding_matrix[line.strip().split('\t')[0]]=vec
            f_2.write('\t')
            f_2.write(str(vec))
            f_2.write('\n')
```

其次选择**语义匹配模型RESCAL和DistMult**，来构建我们的预测系统

首先介绍RESCAL，此处模型构建参考与Github项目和CSDN，链接放在文末。

### RESCAL

该模型的**得分函数定义为**:

$$Score(h,r,t) = v_h^T M_r v_t$$

其中$v_h \in R^d, v_t \in R^d, M_r \in R^{d \times d}$
$v_h, v_t$都是从实体embedding矩阵（记为$E$）中的**取出**(根据实体id获取)的vector，$E$的形状是(num_entities,d)。
$M_r$是整个关系tensor（三维的）中的根据关系id获取的matrix（二维的）。整个关系tensor记为$R$，形状是(num_relations,d,d)，所以$M_r$的shape是$(d,d)$。

实体描述使用前一部分抽取的实体的特征向量，作为网络层self.E的初始参数，在每一次根据loss进行反向传播时进行修正。

```
        temp_entity = torch.zeros(len(entitys_vector),embedding_dim)
        temp_num = 0
        for x in entitys_vector:
            temp_entity[temp_num] = torch.from_numpy(entitys_vector[x])
            temp_num += 1
        self.E = nn.Embedding(len(entitys_vector), embedding_dim)
        self.E.weight = nn.Parameter(temp_entity)
        self.E.weight.requires_grad = True
```

需要注意的是，RESCAL模型的关系描述是一个len_vec*len_vec的矩阵，不能使用从文本抽取的特征向量，只能进行随机生成。

```
        # self.R = nn.Embedding(len(rela_vector), embedding_dim*embedding_dim)
        # self.R.weight.requires_grad = True
        # self.embedding_dim=embedding_dim
        # self.entitys_vector=entitys_vector
        # self.rela_vector=rela_vector
```

使用train.txt作为训练数据，将三元组分批次输入网络层中，计算最终结果与实际期望的误差，反向传播。几轮训练结束后将需要预测的head-relation-?作为输入，与网络层参数相乘的到预测值，写入文件中。

```
        # RESCAL
        # head_embed=head_emb.view(-1,1,100)
        # rel_embed=rel_emb.view(-1,100,100)
        #
score=torch.matmul((torch.squeeze(torch.matmul(head_embed,rel_embed),dim=1)),sel
f.E.weight.transpose(1,0))
        # #print(score.shape)
```

DisMult模型是RESCAL的简化版，但对于我们来说，效果可能更好。因为DisMult模型的关系表示采用的是向量，所以我们可以直接使用word2vec抽取出的特性向量作为初始参数，这样可以更快收敛、更好拟合。

### DistMult

DistMult是RESCAL的简化。具体来说就是RESCAL中每一个head和tail实体之间的关系r是用一个matrix表示。而DistMult中则用一个vector表示两个实体间的关系。
所以得分函数是三个vector之间的内积：

$$< v_h, v_r, v_t >$$

$v_h, v_r, v_t \in R^d$

```
        temp_rela = torch.zeros(len(rela_vector),embedding_dim)
        temp_num = 0
        for x in rela_vector:
            temp_rela[temp_num] = torch.from_numpy(rela_vector[x])
            temp_num += 1
        self.R = nn.Embedding(len(rela_vector), embedding_dim)
        self.R.weight = nn.Parameter(temp_rela)
        self.R.weight.requires_grad = True
```

失败尝试：看见有群友说结构信息比文本信息多，试图使用TransE等使用结构信息的模型尝试，可惜效果恰似random(TAT)

2021-12-09 13:28:02　124_1639027682_result.txt　0.000049　0.000244

不知道哪里出了问题。。。

# 三、实验结果

最终效果最好的是参数设置为vector_length=100,epochs=6,model=Dismult，Hit@1=0.171895，Hit@5=0.309782

您已经提交 **10** 次，剩余提交次数为 **0** 次。

| 提交时间 | 文件名称 | Hit@1 | Hit@5 |
| --- | --- | --- | --- |
| 2021-12-12 21:04:38 | 124_1639314278_result.txt | 0.171895 | 0.309782 |
| 2021-12-12 16:04:27 | 124_1639296267_result_1.txt | 0.161927 | 0.257549 |
| 2021-12-11 11:50:48 | 124_1639194648_result.txt | 0.164810 | 0.272989 |
| 2021-12-10 19:59:10 | 124_1639137550_result.txt | 0.163100 | 0.267712 |
| 2021-12-09 18:28:18 | 124_1639045698_result.txt | 0.171602 | 0.307486 |
| 2021-12-09 18:21:40 | 124_1639045300_result.txt | 0.155477 | 0.279146 |
| 2021-12-09 18:11:06 | 124_1639044666_result.txt | 0.166764 | 0.299179 |
| 2021-12-09 17:31:37 | 124_1639042297_result.txt | 0.142969 | 0.242549 |
| 2021-12-09 15:18:20 | 124_1639034300_result.txt | 0.167693 | 0.304700 |
| 2021-12-09 13:28:02 | 124_1639027682_result.txt | 0.000049 | 0.000244 |

RESCAL对于关系的描述比DisMult复杂，但只能随机生成，所以需要多轮训练。但是因为针对实体我们已经有提前提取的特征向量，所以如果训练轮数过多，可能会出现过拟合的现象。故而可能导致最后效果不如更简单的模型DisMult.

# 四、实验总结

通过本次实验，我们学习了知识图谱有关知识，将word2vec进行了更为实际的应用。同时培养了一定的搭建简单网络的能力，能够根据需要调整已有模型的初始化参数，做出一定的改进。但是因为个人能力，未能很好的使用tranX系列模型，故而预测准确度不高。

参考链接：

https://blog.csdn.net/m0_45478865/article/details/121195480?spm=1001.2014.3001.5501

https://github.com/malllabiisc/EmbedKGQA