Lab2 说明

深度学习模型

关于**word2vec**怎么用

1. 用`entity_with_text`做语料库, 提取描述

```
from gensim.models import word2vec
with open("entity_with_text.txt", 'r') as fr, open("entity_co.txt", 'w') as
fw:
    for line in fr.readlines():
        fw.write(line.strip().split('\t')[1])
        fw.write("\n")
```

2. 训练

```
sentences = word2vec.LineSentence('entity_co.txt')
model = word2vec.Word2Vec(sentences, hs=1,min_count=1,window=3,size=100)
```

1) sentences: 我们要分析的语料，可以是一个列表，或者从文件中遍历读出。后面我们会有从文件读出的例子。

2) size: 词向量的维度，默认值是100。这个维度的取值一般与我们的语料的大小相关，如果是不大的语料，比如小于100M的文本语料，则使用默认值一般就可以了。如果是超大的语料，建议增大维度。

3) window：即词向量上下文最大距离，这个参数在我们的算法原理篇中标记为cc，window越大，则和某一词较远的词也会产生上下文关系。默认值为5。在实际使用中，可以根据实际的需求来动态调整这个window的大小。如果是小语料则这个值可以设的更小。对于一般的语料这个值推荐在[5,10]之间。

4) sg: 即我们的word2vec两个模型的选择了。如果是0， 则是CBOW模型，是1则是Skip-Gram模型，默认是0即CBOW模型。

5) hs: 即我们的word2vec两个解法的选择了，如果是0， 则是Negative Sampling，是1的话并且负采样个数negative大于0， 则是Hierarchical Softmax。默认是0即Negative Sampling。

6) negative:即使用Negative Sampling时负采样的个数，默认是5。推荐在[3,10]之间。这个参数在我们的算法原理篇中标记为neg。

7) cbow_mean: 仅用于CBOW在做投影的时候，为0，则算法中的xwxw为上下文的词向量之和，为1则为上下文的词向量的平均值。在我们的原理篇中，是按照词向量的平均值来描述的。个人比较喜欢用平均值来表示xwxw,默认值也是1,不推荐修改默认值。

8) min_count:需要计算词向量的最小词频。这个值可以去掉一些很生僻的低频词，默认是5。如果是小语料，可以调低这个值。

9) iter: 随机梯度下降法中迭代的最大次数，默认是5。对于大语料，可以增大这个值。

10) alpha: 在随机梯度下降法中迭代的初始步长。算法原理篇中标记为ηη，默认是0.025。

11) min_alpha: 由于算法支持在迭代的过程中逐渐减小步长，min_alpha给出了最小的迭代步长值。随机梯度下降中每轮的迭代步长可以由iter，alpha， min_alpha一起得出。这部分由于不是word2vec算法的核心内容，因此在原理篇我们没有提到。对于大语料，需要对alpha, min_alpha,iter一起调参，来选择合适的三个值。

3. 得到词向量

```
model.wv['6581'] # 6581 即为描述中某个token
```

4. 对实体的描述，使用(sum token)得到向量， 这里的归一化是可选的

```
import numpy as np
with open("entity_with_text.txt", 'r') as fr:
    for line in fr.readlines():
        vec = 0
        words = line.strip().split('\t')[1].split(" ")
        for word in words:
            vec += model.wv[word]
        vec = vec / np.linalg.norm(vec)
```

Embedding + CNN

https://github.com/moguizhizi/tail_entity_predict_CaRe

几何模型

知识图谱嵌入的Translate模型汇总（TransE，TransH，TransR，TransD） - 知乎 (zhihu.com)

翻译模型（一）（TransE、TransH、TransR） - 胡萝不青菜 - 博客园 (cnblogs.com)

本处仅叙述TransE

参考transE知识图谱补全，FB15K-237数据集(python实现) - cpaulyz - 博客园 (cnblogs.com)

缺点， 并没有用到实体、关系的描述，而我们的实验给了关系、实体的描述，使用TransE相当于仅使用数据中的(head, relation, tail) 三元组

流程

1 加载数据集

考虑FB15K-237数据集, 与本次实验数据集的对应请同学们自己思考:)

分为以下四个文件

- **entity2id.txt**

  实体和id对

  ```
  /m/06rf7    0
  /m/0c94fn   1
  /m/016ywr   2
  /m/01yjl    3
  /m/02hrh1q  4
  ```

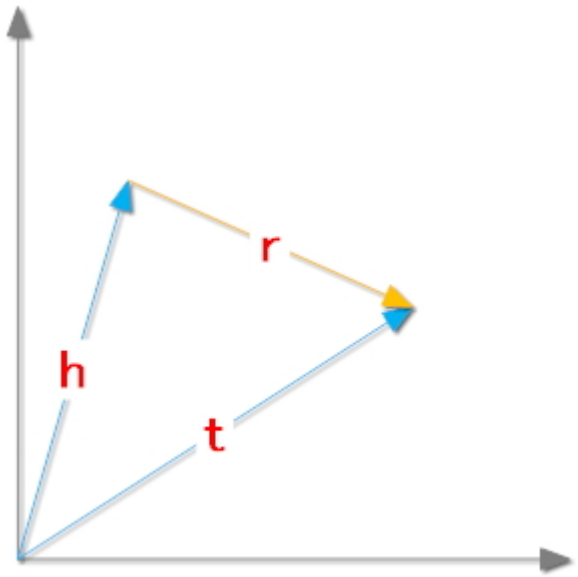
- **relation2id.txt**

  关系和id对

  ```
  /people/appointed_role/appointment./people/appointment/appointed_by 0
  /location/statistical_region/rent50_2./measurement_unit/dated_money_value/currency  1
  /tv/tv_series_episode/guest_stars./tv/tv_guest_role/actor   2
  /music/performance_role/track_performances./music/track_contribution/contributor    3
  ```

- **train.txt**

  训练集三元组（实体，实体，关系）



- **test.txt**

  测试集三元组（实体，实体，关系）

2 TransE

2.1 原理

TransE将起始实体，关系，指向实体映射成同一空间的向量，如果（head,relation,tail）存在，那么h+r≈t

```
/m/027rn    /m/06cx9    /location/country/form_of_government
/m/017dcd   /m/06v8s0   /tv/tv_program/regular_cast./tv/regular_tv_appearance/actor
/m/07s9rl0  /m/0170z3   /media_common/netflix_genre/titles
/m/01sl1q   /m/044mz_   /award/award_winner/awards_won./award/award_honor/award_winner
/m/0cnk2q   /m/02nzb8   /soccer/football_team/current_roster./sports/sports_team_roster/position
```

目标函数为：

$$f_r(h,t) = \|h + r - t\|_2^2$$

2.2 算法

Algorithm 1 Learning TransE

input Training set $S = \{(h,\ell,t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$

  $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$

  $\mathbf{e} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

loop

  $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$

  $S_{batch} \leftarrow$sample(S, b) // sample a minibatch of size b

  $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets

  for $(h,\ell,t) \in S_{batch}$ **do**

    $(h',\ell,t') \leftarrow$sample$(S'_{(h,\ell,t)})$ // sample a corrupted triplet

    $T_{batch} \leftarrow T_{batch} \cup \{((h,\ell,t),(h',\ell,t'))\}$

  end for

  Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})]_+$

end loop

3 训练

3.1初始化

根据维度，为每个实体和关系初始化向量，并归一化

```
    def emb_initialize(self):
        relation_dict = {}
        entity_dict = {}

        for relation in self.relation:
            r_emb_temp = np.random.uniform(-6 / math.sqrt(self.embedding_dim),
                                           6 / math.sqrt(self.embedding_dim),
                                           self.embedding_dim)
            relation_dict[relation] = r_emb_temp / np.linalg.norm(r_emb_temp,
ord=2)

        for entity in self.entity:
            e_emb_temp = np.random.uniform(-6 / math.sqrt(self.embedding_dim),
                                           6 / math.sqrt(self.embedding_dim),
                                           self.embedding_dim)
            entity_dict[entity] = e_emb_temp / np.linalg.norm(e_emb_temp, ord=2)
```

3.2 选取batch

设置`nbatches`为batch数目，`batch_size = len(self.triple_list) // nbatches`

从训练集中随机选择`batch_size`个三元组，并随机构成一个错误的三元组S'，进行更新

```
    def train(self, epochs):
        nbatches = 400
        batch_size = len(self.triple_list) // nbatches
        print("batch size: ", batch_size)
        for epoch in range(epochs):
            start = time.time()
            self.loss = 0

            # Sbatch:list
            Sbatch = random.sample(self.triple_list, batch_size)
            Tbatch = []

            for triple in Sbatch:
                corrupted_triple = self.Corrupt(triple)
                if (triple, corrupted_triple) not in Tbatch:
                    Tbatch.append((triple, corrupted_triple))
            self.update_embeddings(Tbatch)
```

3.3 梯度下降

定义距离*d(x,y)*来表示两个向量之间的距离，一般情况下，我们会取L1,或者L2 normal。

在这里，我们需要定义一个距离，对于正确的三元组*(h,r,t)*,距离*d(h+r,t)*越小越好；对于错误的三元组*(h',r,t')*,距离*d(h'+r,t')*越小越好。

```
14
mean rank: 353.06935721419984
hit@3: 0.12181950534103028
hit@10: 0.2754989758087725
```

之后，使用梯度下降进行更新

4 预测

通过transE建模后，我们得到了每个**实体**和**关系**的嵌入向量，利用嵌入向量，我们可以进行知识图谱的链接预测

将三元组(head,relation,tail)记为(h,r,t)

链接预测分为三类

1. 头实体预测：**(?,r,t)**
2. 关系预测：**(h,?,t)**
3. 尾实体预测：**(h,r,?)**

但原理很简单，利用向量的可加性即可实现。以**(h,r,?)**的预测为例：

假设***t'=h+r***，则在所有的实体中选择与***t'***距离最近的向量，即为***t***的的预测值

计算距离

```
def distance(h, r, t):
    h = np.array(h)
    r = np.array(r)
    t = np.array(t)
    s = h + r - t
    return np.linalg.norm(s)
```

排序

```
def mean_rank(entity_set, triple_list):
    triple_batch = random.sample(triple_list, 100)
    mean = 0
    hit10 = 0
    hit3 = 0
    for triple in triple_batch:
        dlist = []
        h = triple[0]
        t = triple[1]
        r = triple[2]
        dlist.append((t, distance(entityId2vec[h], relationId2vec[r],
entityId2vec[t])))
        for t_ in entity_set:
            if t_ != t:
                dlist.append((t_, distance(entityId2vec[h], relationId2vec[r],
entityId2vec[t_])))
        dlist = sorted(dlist, key=lambda val: val[1])
        for index in range(len(dlist)):
            if dlist[index][0] == t:
                mean += index + 1
```

```
                if index < 3:
                    hit3 += 1
                if index <10:
                    hit10 += 1
                print(index)
                break
    print("mean rank:", mean / len(triple_batch))
    print("hit@3:", hit3 / len(triple_batch))
    print("hit@10:", hit10 / len(triple_batch))
```

综合模型

TransE+Word2Vec

[word2vec + transE 知识表示模型 - bbking - 博客园 (cnblogs.com)]

[GitHub - chenbjin/RepresentationLearning: 知识表示相关学习算法]