doc2vec计算文本相似度--python实现

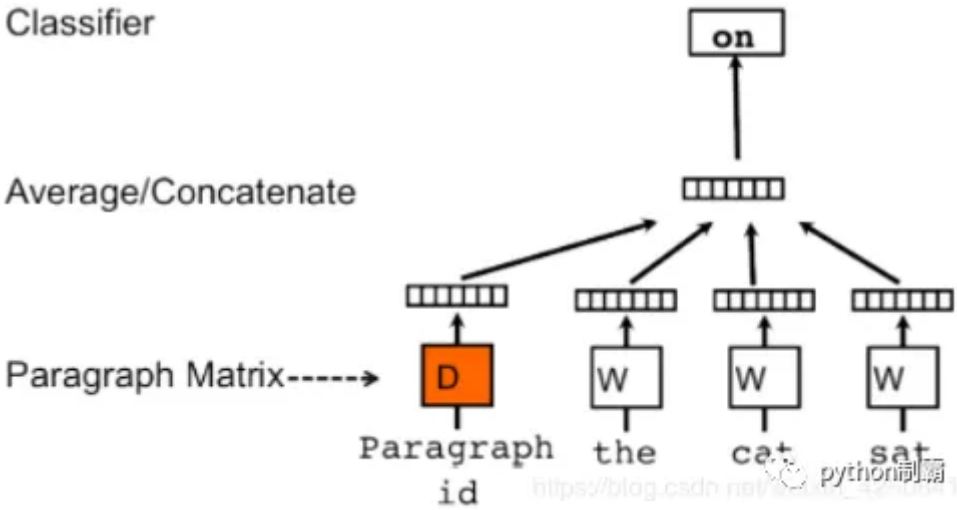
boblee python制霸 2020-05-06

Boblee人工智能硕士毕业，擅长及爱好python，基于python研究人工智能、群体智能、区块链等技术，并使用python开发前后端、爬虫等。

1.背景

doc2vec的目标是创建文档的向量化表示，而不管其长度如何。但与单词不同的是，文档并没有单词之间的逻辑结构，因此必须找到另一种方法。

Mikilov和Le使用的概念很简单但很聪明：他们使用了word2vec模型，并添加了另一个向量（下面的段落ID），如下所示：

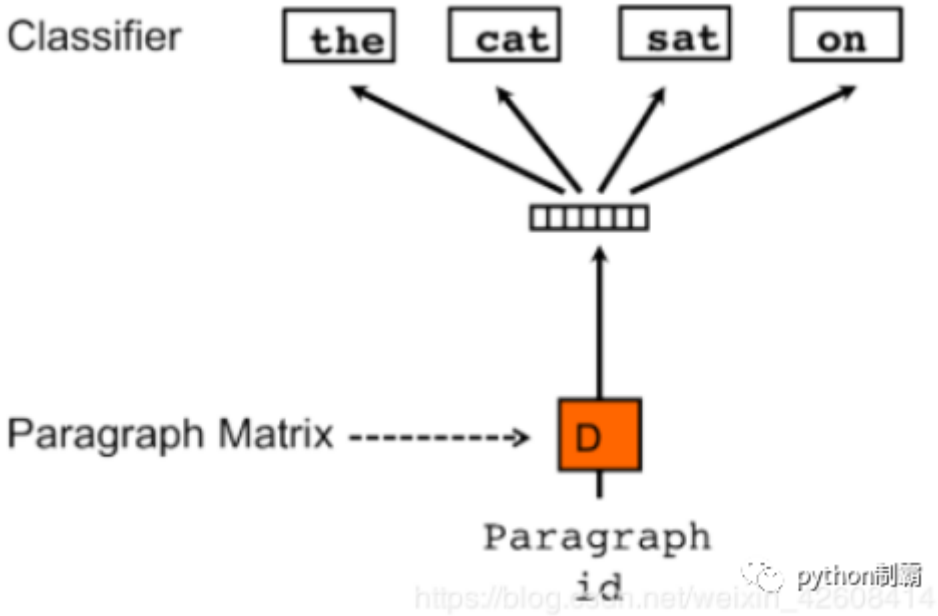


上图是word2vec中CBOW模型的一个小扩展。它不是仅是使用一些单词来预测下一个单词,我们还添加了另一个特征向量，即文档Id。

因此，当训练单词向量W时，也训练文档向量D，并且在训练结束时，它包含了文档的向量化表示。

上面的模型称为段落向量的分布式记忆的版本（PV-DM）。它充当记忆器，它能记住当前上下文中缺少的内容 - 或者段落的主题。虽然单词向量表示单词的概念，但文档向量旨在表示文档的概念。

如在doc2vec中，另一种类似于skip-gram的算法,段落向量的分布式词袋版本（PV-DBOW）。



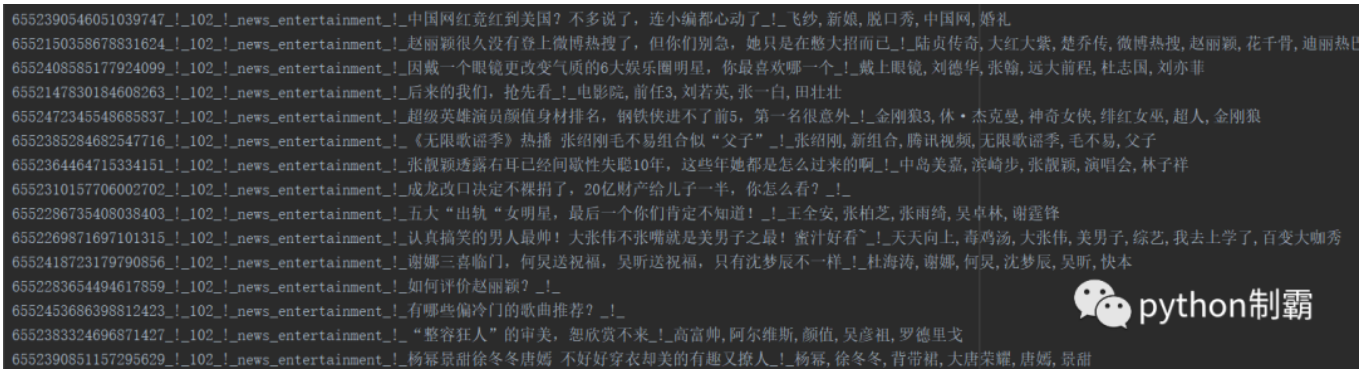
该算法实际上更快（与word2vec相反）并且消耗更少的内存，因为不需要保存词向量。

在论文中，作者建议使用两种算法的组合，尽管PV-DM模型是优越的，并且通常会自己达到最优的结果。

doc2vec模型的使用方式：对于训练，它需要一组文档。为每个单词生成词向量W，并为每个文档生成文档向量D. 该模型还训练softmax隐藏层的权重。在推理阶段，可以呈现新文档，并且固定所有权重以计算文档向量。

2.python实现

本文使用今日头条提供的文本分类数据集进行实验，https://github.com/skdjfla/toutiao-text-classfication-dataset。



python中提供了doc2vec、word2vec封装好的库sklearn。sklearn使用doc2vec请见https://radimrehurek.com/gensim/models/doc2vec.html。

```
pip install sklearn
```

1.句子分词

```
import gensim
import numpy as np
import jieba
```

```
from gensim.models.doc2vec import Doc2Vec

def jieba_tokenize(text):
    """
    文本分词
    :param text: 文本
    :return: 分词list
    """
    return jieba.lcut(text)
```

2.获取训练集

```
def get_datasest():
    """
    获取doc2vec文本训练数据集
    :return: 文本分词list，及id
    """
    TaggededDocument = gensim.models.doc2vec.TaggedDocument
    x_train = []
    for file in open('toutiao_cat_data.txt', encoding='utf8'):
        file = file.split('_!_')
        if len(file) > 3:
            document = TaggededDocument(file[3], tags=[int(file[1])])
            x_train.append(document)
    return x_train
```

3.训练

```
def train(x_train, size=2000, epoch_num=10):
    model_dm = Doc2Vec(x_train,min_count=1, window = 3, size = size, sample=1e
    model_dm.train(x_train, total_examples=model_dm.corpus_count, epochs=epoch
    model_dm.save('model')
    return model_dm
```

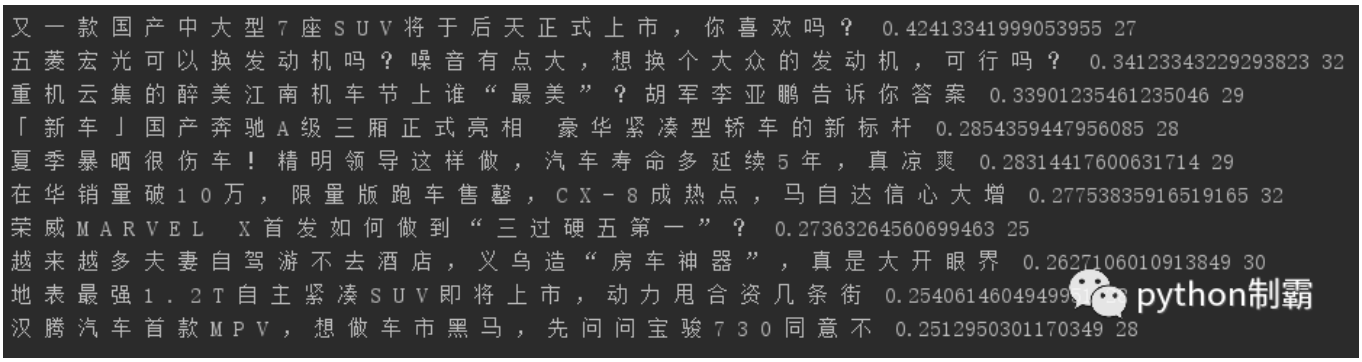
4.测试

```
def getVecs(model, corpus, size):
    vecs = [np.array(model.docvecs[z.tags[0]].reshape(1, size)) for z in corp
```

```
    return np.concatenate(vecs)
def test():
    model_dm = Doc2Vec.load("model")
    test_text = ['想换个', '30', '万左右', '的', '车', ',', '现在', '开科鲁兹',
    inferred_vector_dm = model_dm.infer_vector(test_text)
    sims = model_dm.docvecs.most_similar([inferred_vector_dm], topn=10)
    return sims

if __name__ == '__main__':
    x_train = get_datasest()
    model_dm = train(x_train)

    sims = test()
    for count, sim in sims:
        sentence = x_train[count]
        words = ''
        for word in sentence[0]:
            words = words + word + ' '
        print (words, sim, len(sentence[0]))
```



感觉效果还是不错。对应车的句子能够提取出来。