用 Doc2Vec 得到文档 / 段落 / 句子的向量表达

原创 Alice 机器学习X计划 2017-07-14

The Ocean

Mike Perry;Shy Martin - The Ocean (Radio Edit)

本文结构：

- Doc2Vec 有什么用
- 两种实现方法
- 用 Gensim 实现 Doc2Vec 训练

Doc2Vec 或者叫做 paragraph2vec, sentence embeddings，是一种非监督式算法，可以获得 sentences/paragraphs/documents 的向量表达，是 word2vec 的拓展。

学出来的向量可以通过计算距离来找 sentences/paragraphs/documents 之间的相似性，
或者进一步可以给文档打标签。

例如首先是找到一个向量可以代表文档的意思，
然后可以将向量投入到监督式机器学习算法中得到文档的标签，
例如在**情感分析** sentiment analysis 任务中，标签可以是“negative”, “neutral”,” positive”。

2013 年 Mikolov 提出了 word2vec 来学习单词的向量表示，
主要有两种方法，**cbow (continuous bag of words) 和 skip-gram** ，
一个是用语境来预测目标单词，另一个是用中心单词来预测语境。

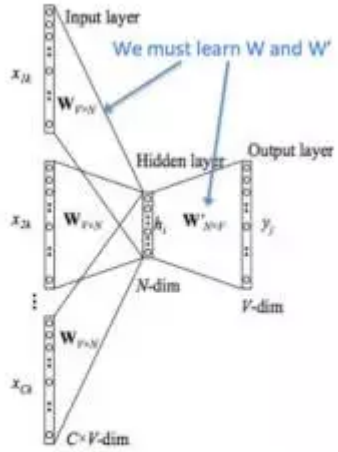

Figure 1: This image demonstrates how CBOW works and how we must learn the transfer matrices

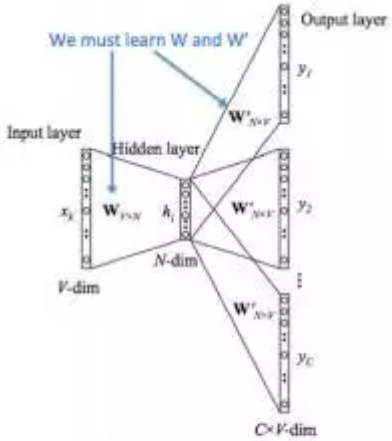

Figure 2: This image demonstrates how Skip-Gram works and how we must learn the transfer matrices

---

既然可以将 word 表示成向量形式，那么句子 / 段落 / 文档是否也可以只用一个向量表示?

一种方式是可以先得到 word 的向量表示，然后用一个简单的平均来代表文档。
另外就是 Mikolov 在 2014 提出的 Doc2Vec。

Doc2Vec 也有两种方法来实现。

dbow (distributed bag of words)

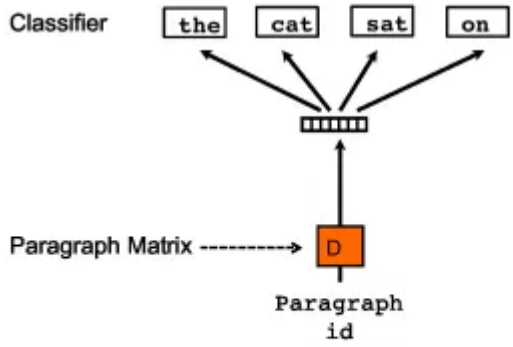

Figure 3. Distributed Bag of Words version of paragraph vectors. In this version, the paragraph vector is trained to predict the words in a small window.

gensim 实现:

```
model = gensim.models.Doc2Vec(documents,dm = 0, alpha=0.1, size= 20, min_alpha
```

dm (distributed memory)

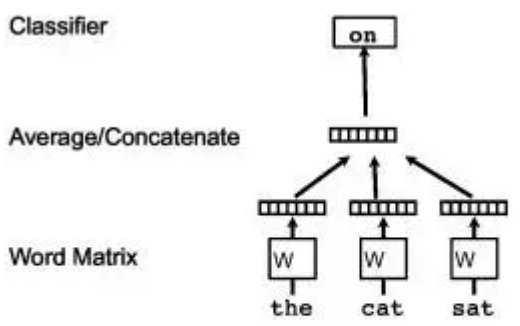

Figure 1. A framework for learning word vectors. Context of three words ("the," "cat," and "sat") is used to predict the fourth word ("on"). The input words are mapped to columns of the matrix W to predict the output word.

gensim 实现:

```
model = gensim.models.Doc2Vec(documents,dm = 1, alpha=0.1, size= 20, min_alpha
```

二者在 gensim 实现时的区别是 dm = 0 还是 1.

Doc2Vec 的目的是获得文档的一个固定长度的向量表达。

数据：多个文档，以及它们的标签，可以用标题作为标签。
影响模型准确率的因素：语料的大小，文档的数量，越多越高；文档的相似性，越相似越好。

这里要用到 Gensim 的 Doc2Vec：

```
import gensim
LabeledSentence = gensim.models.doc2vec.LabeledSentence
```

- 先把所有文档的路径存进一个 array 中，docLabels：

```
from os import listdir
from os.path import isfile, join
docLabels = []
docLabels = [f for f in listdir("myDirPath") if f.endswith('.txt')]
```

- 把所有文档的内容存入到 data 中：

```
data = []
for doc in docLabels:
    data.append(open(“myDirPath/” + doc, ‘r’)
```

- 接下来准备数据，
  如果是用句子集合来训练模型，则可以用：

```
class LabeledLineSentence(object):
    def __init__(self, filename):
        self.filename = filename
    def __iter__(self):
        for uid, line in enumerate(open(filename)):
            yield LabeledSentence(words=line.split(), labels=[‘SENT_%s’ % uid]
```

如果是用文档集合来训练模型，则用：

```
class LabeledLineSentence(object):
    def __init__(self, doc_list, labels_list):
       self.labels_list = labels_list
       self.doc_list = doc_list
    def __iter__(self):
        for idx, doc in enumerate(self.doc_list):
            yield LabeledSentence(words=doc.split(),labels=[self.labels_list[i
```

在 gensim 中模型是以单词为单位训练的，所以不管是句子还是文档都分解成单词。

- 训练模型：

将 data, docLabels 传入到 LabeledLineSentence 中，
训练 Doc2Vec，并保存模型：

```
it = LabeledLineSentence(data, docLabels)

model = gensim.models.Doc2Vec(size=300, window=10, min_count=5, workers=11,alp

model.build_vocab(it)

for epoch in range(10):
    model.train(it)
    model.alpha -= 0.002             # decrease the learning rate
    model.min_alpha = model.alpha         # fix the learning rate, no deca
    model.train(it)

model.save(“doc2vec.model”)
```

- 测试模型：

Gensim 中有内置的 most_similar：

```
print model.most_similar(“documentFileNameInYourDataFolder”)
```

- 输出向量：

```
model[“documentFileNameInYourDataFolder”]
```

- 得到向量后，可以计算相似性，输入给机器学习算法做情感分类等任务了。

资料：
https://arxiv.org/abs/1405.4053
https://rare-technologies.com/doc2vec-tutorial/
https://medium.com/@klintcho/doc2vec-tutorial-using-gensim-ab3ac03d3a1

相关文章：

CS224d - Day 3: word2vec 模型思想和代码实现

怎样做情感分析

推荐 阅读原文
也许可以找到你想要的：