基于Doc2Vec Model的文档相似度算法

1.Introduction

原理：

本方法参考了Quoc Le and Tomas Mikolov: “Distributed Representations of Sentences and Documents”.

实现

本方法调用了gensim库中的**doc2vec**接口

运行方法

> 1.安装python的gensim，docx库
>
> 2.修改文件路径fn1 = r'D:\GitHubFile\HomeWork\DOC\Walden.docx'为文件所在的路径
>
> 3.运行learnner.py

如何学习表征，为什么这样学习?

Bag-of-words模型（将词转化为缺乏语义的词向量）

本方法将文档转为向量的过程基于：词袋模型 bag-of-words model ，本方法将每个文档转为一个定长的向量，比如：

- `John likes to watch movies. Mary likes movies too.`
- `John also likes to watch football games. Mary hates football.`

那么模型的转化向量为：

- `[1, 2, 1, 1, 2, 1, 1, 0, 0, 0, 0]`
- `[1, 1, 1, 1, 0, 1, 0, 1, 2, 1, 1]`

每个向量都有十个元素，每一个元素代表该文档某个特定词出现的次数，元素之间的顺序是随机的，对于上边的例子，元素的顺序对应着：

`["John", "likes", "to", "watch", "movies", "Mary", "too", "also", "football", "games", "hates"]`.

词袋模型确实很好，但是存在几个缺点：

- 首先，转码的过程丢失了语序信息，“John likes Mary” 和 “Mary likes John” 有完全不同的意思，但是转码向量却是相同的

  解决办法：使用bag of n-grams models，这个模型在表征文档到定长向量的过程中，考虑了词序n，可以捕捉本地词序，但是会受数据稀疏性和过高纬度的影响

- 另外，词袋模型并不会试着去学习潜在词的意思，这代表着，向量之间的距离并不会映射着语义之间的不同

  解决办法：Word2Vec可以解决

`Word2Vec` 模型（将词转化为可用的词向量）

word2Vec模型利用浅层神经网络，把词嵌入到低维向量之中

word2Vec的**输出**为一系列的词向量，而在向量空间中更接近的词向量就是语义更相近的词，词向量之间的距离也更有意义，strong和powerful的词向量会更接近，而strong和paris的词向量则会远一些

Paragraph Vector（将段落转化为向量）

word2Vec产生的词向量无法直接用来衡量两个文档是否相似，如果计算某文档所有向量的平均值，将其与另一向量的平均值作比较，必然会丢失很多信息，产生过大误差

解决办法：使用Le and Mikolov在2014年提出的 Doc2Vec algorithm，带来的的性能和直接平均计算比较提升很大

基本思路：

> The basic idea is: act as if a document has another floating word-like vector, which contributes to all training predictions, and is updated like other word-vectors, but we will call it a doc-vector.

对文本进行表征学习，本文是通过分布式内存模型(Paragraph Vector - Distributed Memory)和分布式词袋模型(Paragraph Vector - Distributed Bag of Words)学习段落和文档嵌入表征，这类型的算法要么使用分级softmax，要么使用负采样：

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean: “Efficient Estimation of Word Representations in Vector Space, in Proceedings of Workshop at ICLR, 2013”
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean: “Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013”.

> PV-DM is analogous to Word2Vec CBOW. The doc-vectors are obtained by training a neural network on the synthetic task of predicting a center word based an average of both context word-vectors and the full document’s doc-vector.
>
> PV-DBOW is analogous to Word2Vec SG. The doc-vectors are obtained by training a neural network on the synthetic task of predicting a target word just from the full document’s doc-vector. (It is also common to combine this with skip-gram testing, using both the doc-vector and nearby word-vectors to predict a single target word, but only one at a time.)

2.what's new?

- 将需要对比相似性的docx文档转化为document文件格式，对于其中的每个段落或每个文档赋予索引，使用Doc2Vec Model训练对应的语料库模型；
- 利用训练好的语料库模型，进行表征学习，学习目标文档对应的高维向量表征；
- 再通过余弦相似性计算其文档之间的相似性，即可找出其他文档和本文的相似度
- 在多文档对比的情况下，可得到相似性排名，方便对文档相似性进行排序。
- 由于使用了gensim库，在百万级别文档的对比下，仍然可以保持准确度和时间效率，比利用传统numpy库实现的doc2vec速度快70倍

Experiment Details

Doc2Vec是一个把每个文档表征为向量的模型，本实验分为：

- 加入日志操作
- 加载数据，预处理语料库

- 使用语料库训练模型
- 从训练好的模型中得到向量
- 评估模型

1.import logging:

```
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)
```

2.Processing data

原始数据包含5个.docx文档，分别节选自以下文本：

- War and Peace By Leo Tolstoy
- Anna Karenina By Leo Tolstoy
- Crime and Punishment By Fyodor Dostoevsky
- Walden By Henry David Thoreau
- A Tale of Two Cities By Charles Dickens

定义一个函数来读取和预处理数据

- 打开文档
- 按行读取文件
- 预处理每一行

```
#input : file dir which you want to compare
#output : a List which contains all words in this file
def wholedoccreater(fn):
    doc = docx.Document(fn)
    wordList = []
    for paragraph in doc.paragraphs:
        tempList = []
        tempList = paragraph.text.split(" ")
        if len(tempList) > 1:
            for i in tempList:
                wordList.append(i)
    return wordList

#input : file dirs which you want to compare
#output : a document you can feed in Doc2Vec
def documentreader(fn1, fn2, fn3, fn4, fn5):
    wholeDoc1 = wholedoccreater(fn1)
    wholeDoc2 = wholedoccreater(fn2)
    wholeDoc3 = wholedoccreater(fn3)
    wholeDoc4 = wholedoccreater(fn4)
    wholeDoc5 = wholedoccreater(fn5)
    wordList = [wholeDoc1, wholeDoc2, wholeDoc3, wholeDoc4, wholeDoc5]

    documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(wordList)]
    return documents
```

数据描述

title	War and Peace	Anna Karenina	Crime and Punishment	Walden	A Tale of Two Cities
word numbers	9434	940	3396	6105	4757
word numbers after preprocess	9261	936	3169	6100	4752

分配索引规则

读取的文件是一个语料库，所以每一行都是一个文档，为了训练数据，每一行都需要一个索引，可以给每一行分配一个索引即可

```
import smart_open

def read_corpus(fname, tokens_only=False):
    with smart_open.open(fname, encoding="iso-8859-1") as f:
        for i, line in enumerate(f):
            tokens = gensim.utils.simple_preprocess(line)
            if tokens_only:
                yield tokens
            else:
                # For training data, add tags
                yield gensim.models.doc2vec.TaggedDocument(tokens, [i])

train_corpus = list(read_corpus(lee_train_file))
test_corpus = list(read_corpus(lee_test_file, tokens_only=True))
```

举例:语料库长什么样子?

```
print(train_corpus[:2])
```

```
[TaggedDocument(words=['hundreds', 'of', 'people', 'have', 'been', 'forced',
'to', 'vacate', 'their', 'homes', 'in', 'the', 'southern', 'highlands', 'of',
'new', 'rain', 'is', 'concerned', 'in', 'fact', 'they', 've', 'probably',
'hampered', 'the', 'efforts', 'of', 'the', 'firefighters', 'more', 'because',
'of', 'the', 'wind', 'gusts', 'that', 'are', 'associated', 'with', 'those',
'thunderstorms'], tags=[0]), TaggedDocument(words=['indian', 'security',
'forces', 'have', 'shot', 'dead', 'eight', 'suspected', 'militants', 'in',
'night', 'long', 'encounter', 'in', 'southern', 'kashmir', 'the', 'shootout',
'took', 'place', 'at', 'dora',  'police', 'in', 'karachi', 'say', 'it', 'is',
'likely', 'more', 'raids', 'will', 'be', 'launched', 'against', 'the', 'two',
'groups', 'as', 'well', 'as', 'other', 'militant', 'organisations', 'accused',
'of', 'targetting', 'india', 'military', 'tensions', 'between', 'india', 'and',
'pakistan', 'have', 'escalated', 'to', 'level', 'not', 'seen', 'since', 'their',
'war'], tags=[1])]
```

3.Training the model

初始化模型为一个64维，并迭代训练40次，抛弃出现频率小于3的词

```
model = gensim.models.doc2vec.Doc2Vec(vector_size=64, min_count=3, epochs=40)
```

建立词汇表

```
model.build_vocab(train_corpus)
```

最终，词汇表是一个包含所有独特词的列表（使用model.wv.index_to_key），每个词的额外的信息包含在model.wv.get_vecattr()

比如要看penalty出现频率

```
print(f"Word 'penalty' appeared {model.wv.get_vecattr('penalty', 'count')} times
in the training corpus.")
```

输出

```
Word 'penalty' appeared 4 times in the training corpus.
```

下一步，在语料库中训练模型，使用BLAS库会更快

```
model.train(train_corpus, total_examples=model.corpus_count,
epochs=model.epochs)
```

现在就可以用训练好的模型得到一个单词列表的向量表示，之后就可以使用这个向量表示计算相似性

```
vector = model.infer_vector(['only', 'you', 'can', 'prevent', 'forest',
'fires'])
print(vector)
```

输出

```
[-0.08478509  0.05011684  0.0675064  -0.19926868 -0.1235586   0.01768214
 -0.12645927  0.01062329  0.06113973  0.35424358  0.01320948  0.07561274
 -0.01645093  0.0692549   0.08346193 -0.01599065  0.08287009 -0.0139379
 -0.17772709 -0.26271465  0.0442089  -0.04659882 -0.12873884  0.28799203
 -0.13040264  0.12478471 -0.14091878 -0.09698066 -0.07903259 -0.10124907
 -0.28239366  0.13270256  0.04445919 -0.24210942 -0.1907376  -0.07264525
 -0.14167067 -0.22816683 -0.00663796  0.23165748 -0.10436232 -0.01028251
 -0.04064698  0.08813146  0.01072008 -0.149789    0.05923386  0.16301566
  0.05815683  0.1258063 ]
```

计算相似性:

```
def documenttrainer():
    train_corpus = list(documentreader(fn1, fn2, fn3, fn4, fn5))
    model = gensim.models.doc2vec.Doc2Vec(vector_size=64, min_count=3,
epochs=40)
    model.build_vocab(train_corpus)
    model.train(train_corpus, total_examples=model.corpus_count,
epochs=model.epochs)
    fname = get_tmpfile("my_doc2vec_model")
    model.save(fname)
    ranks = []
    second_ranks = []

    for doc_id in range(len(train_corpus)):
        inferred_vector = model.infer_vector(train_corpus[doc_id].words)
        sims = model.docvecs.most_similar([inferred_vector],
topn=len(model.docvecs))
        rank = [docid for docid, sim in sims].index(doc_id)
        ranks.append(rank)
        second_ranks.append(sims[1])
        print('Document ({}): «{}»\n'.format(doc_id, '
'.join(train_corpus[doc_id].words)))
        print(u'SIMILAR/DISSIMILAR DOCS PER MODEL %s:\n' % model)
        for label, index in [('MOST', 0), ('SECOND-MOST', 1), ('MEDIAN',
len(sims) // 2), ('LEAST', len(sims) - 1)]:
            print(u'%s %s: «%s»\n' % (label, sims[index], '
'.join(train_corpus[sims[index][0]].words)))
```

文档相似度计算结果

其中, 计算了**最相似的文档(MOST)**, 次相似的文档(SECOND-MOST),**最不相似的文档(LEAST)**,**空值**为真实情况下次相似的文档.

值得注意的是，文档自身和自身的相似性总是很高，接近1，对于第二名的相似性分数就会低了很多

title	War and Peace	Anna Karenina	Crime and Punishment	Walden	A Tale of Two Cities
War and Peace	0.9884666204452515	0.4948118329048157	0.3094167113304138	0, 0.1593853235244751	
Anna Karenina		0.909547746181488	0.6572633981704712	0.023658521473407745	0.7205044031143188
Crime and Punishment		0.7337720990180969	0.959261953830719	-0.135994553565979	0.5950512886047363
Walden		0.09636926651000977	-0.16651076078414917	0.9970352649688721	0.505054235458374
A Tale of Two Cities	0.07437780499458313	0.7354865074157715	0.5285435914993286		0.9432753324508667

4.Result analysis

其中, 与瓦尔登湖最相似的双城记的相似度不高(0.5950512886047363),而其余俄国作品和瓦尔登湖的相似度接近0

title	最相似的文档	次相似的文档	最不相似的文档
War and Peace	Anna Karenina	Crime and Punishment	Walden
Anna Karenina	A Tale of Two Cities	Crime and Punishment	Walden
Crime and Punishment	Anna Karenina	A Tale of Two Cities	Walden
Walden	A Tale of Two Cities	无	Crime and Punishment Anna Karenina War and Peace
A Tale of Two Cities	Anna Karenina	Crime and Punishment	War and Peace

5.conclosion:

本实验达成了以下目标

- 将需要对比相似性的docx文档转化为document文件格式，对于其中的每个段落或每个文档赋予索引，使用Doc2Vec Model训练对应的语料库模型；
- 利用训练好的语料库模型，进行表征学习，学习目标文档对应的高维向量表征；
- 再通过余弦相似性计算其文档之间的相似性，即可找出其他文档和本文的相似度
- 在多文档对比的情况下，可得到相似性排名，方便对文档相似性进行排序。

另外, 通过结果发现 , 由于梭罗本人的超验主义写作手法较为独特,通过余弦相似度计算,他与同时代的其余作家的作品的相似度都很低, 而其余作品之间的相似度都很高,间接也证明了算法的 可靠性.

6.附录

原始输出:

```
2021-06-23 14:25:56,075 : INFO : worker thread finished; awaiting finish of 2
more threads
2021-06-23 14:25:56,079 : INFO : worker thread finished; awaiting finish of 1
more threads
2021-06-23 14:25:56,079 : INFO : worker thread finished; awaiting finish of 0
more threads
2021-06-23 14:25:56,079 : INFO : EPOCH - 1 : training on 13472 raw words (5596
effective words) took 0.0s, 1027128 effective words/s
2021-06-23 14:25:56,082 : INFO : worker thread finished; awaiting finish of 2
more threads
2021-06-23 14:25:56,085 : INFO : worker thread finished; awaiting finish of 1
more threads
2021-06-23 14:25:56,086 : INFO : worker thread finished; awaiting finish of 0
more threads
2021-06-23 14:25:56,086 : INFO : EPOCH - 2 : training on 13472 raw words (5577
effective words) took 0.0s, 985806 effective words/s
...
...

2021-06-23 14:25:56,342 : INFO : worker thread finished; awaiting finish of 2
more threads
2021-06-23 14:25:56,345 : INFO : worker thread finished; awaiting finish of 1
more threads
2021-06-23 14:25:56,345 : INFO : worker thread finished; awaiting finish of 0
more threads
2021-06-23 14:25:56,345 : INFO : EPOCH - 40 : training on 13472 raw words (5676
effective words) took 0.0s, 1074288 effective words/s

2021-06-23 14:25:56,345 : INFO : training on a 538880 raw words (225740 effective
words) took 0.3s, 827253 effective words/s
2021-06-23 14:25:56,346 : INFO : saving Doc2Vec object under
C:\Users\lenovo\AppData\Local\Temp\my_doc2vec_model, separately None
2021-06-23 14:25:56,353 : INFO : saved
C:\Users\lenovo\AppData\Local\Temp\my_doc2vec_model
2021-06-23 14:25:56,456 : INFO : precomputing L2-norms of doc weight vectors
Document (0): «At a certain season of our life we are accustomed to consider
every spot as the possible site of a house.
...
...
```

```
so by the divining-rod and thin rising vapors I judge; and here I will begin to
mine.»

SIMILAR/DISSIMILAR DOCS PER MODEL Doc2Vec(dm/m,d64,n5,w5,mc3,s0.001,t3):

MOST (0, 0.9970352649688721): «At a certain season of our life we are accustomed
to consider every spot as the possible site of a house.
...
...
so by the divining-rod and thin rising vapors I judge; and here I will begin to
mine.»

SECOND-MOST (1, 0.505054235458374): «It was the best of times, it was the worst
of times, it was the age of wisdom, it was the age of foolishness
...
...
Thus did the year one thousand seven hundred and seventy-five conduct there
Greatnesses, and myriads of small creatures—the creatures of this chronicle among
the rest—along the roads that lay before them.»

MEDIAN (2, 0.09636926651000977): «Happy families are all alike; every unhappy
family is unhappy in its own way.
...
...
"It's that idiotic smile that's to blame for it all," thought Stepan
Arkadyevitch. "But what's to be done? What's to be done?" he said to himself in
despair, and found no answer.»

LEAST (3, -0.16651076078414917): «On an exceptionally hot evening early in July a
young man came out of the garret in which he lodged in S.
...
...
He was sitting apart, now and then sipping from his pot and looking round at the
company. He, too, appeared to be in some agitation.»

Document (1): «It was the best of times, it was the worst of times, it was the
age of wisdom, it was the age of foolishness
...
...
Thus did the year one thousand seven hundred and seventy-five conduct there
Greatnesses, and myriads of small creatures—the creatures of this chronicle among
the rest—along the roads that lay before them.»

SIMILAR/DISSIMILAR DOCS PER MODEL Doc2Vec(dm/m,d64,n5,w5,mc3,s0.001,t3):

MOST (1, 0.9432753324508667): «It was the best of times, it was the worst of
times, it was the age of wisdom, it was the age of foolishness
...
...
Thus did the year one thousand seven hundred and seventy-five conduct there
Greatnesses, and myriads of small creatures—the creatures of this chronicle among
the rest—along the roads that lay before them.»

SECOND-MOST (2, 0.7354865074157715): «Happy families are all alike; every unhappy
family is unhappy in its own way.
...
...
```

```
"It's that idiotic smile that's to blame for it all," thought Stepan
Arkadyevitch. "But what's to be done? What's to be done?" he said to himself in
despair, and found no answer.»

MEDIAN (3, 0.5285435914993286): «On an exceptionally hot evening early in July a
young man came out of the garret in which he lodged in S.
...
...
He was sitting apart, now and then sipping from his pot and looking round at the
company. He, too, appeared to be in some agitation.»

LEAST (4, 0.07437780499458313): «"Well, Prince, so Genoa and Lucca are now just
family estates of the Buonapartes.
...
...
"I'll speak to Lise, young Bolkonski's wife, this very evening, and perhaps the
thing can be arranged. It shall be on your family's behalf that I'll start my
apprenticeship as old maid."»

Document (2): «Happy families are all alike; every unhappy family is unhappy in
its own way.
...
...
"It's that idiotic smile that's to blame for it all," thought Stepan
Arkadyevitch. "But what's to be done? What's to be done?" he said to himself in
despair, and found no answer.»

SIMILAR/DISSIMILAR DOCS PER MODEL Doc2Vec(dm/m,d64,n5,w5,mc3,s0.001,t3):

MOST (2, 0.909547746181488): «Happy families are all alike; every unhappy family
is unhappy in its own way.
...
...
"It's that idiotic smile that's to blame for it all," thought Stepan
Arkadyevitch. "But what's to be done? What's to be done?" he said to himself in
despair, and found no answer.»

SECOND-MOST (1, 0.7205044031143188): «It was the best of times, it was the worst
of times, it was the age of wisdom, it was the age of foolishness
...
...
Thus did the year one thousand seven hundred and seventy-five conduct there
Greatnesses, and myriads of small creatures—the creatures of this chronicle among
the rest—along the roads that lay before them.»

MEDIAN (3, 0.6572633981704712): «On an exceptionally hot evening early in July a
young man came out of the garret in which he lodged in S.
...
...
There was another man in the room who looked somewhat like a retired government
clerk. He was sitting apart, now and then sipping from his pot and looking round
at the company. He, too, appeared to be in some agitation.»

LEAST (0, 0.023658521473407745): «At a certain season of our life we are
accustomed to consider every spot as the possible site of a house.
...
...
```

```
so by the divining-rod and thin rising vapors I judge; and here I will begin to
mine.»

Document (3): «On an exceptionally hot evening early in July a young man came out
of the garret in which he lodged in S.
...
...
He was sitting apart, now and then sipping from his pot and looking round at the
company. He, too, appeared to be in some agitation.»

SIMILAR/DISSIMILAR DOCS PER MODEL Doc2Vec(dm/m,d64,n5,w5,mc3,s0.001,t3):

MOST (3, 0.959261953830719): «On an exceptionally hot evening early in July a
young man came out of the garret in which he lodged in S.
...
...
He was sitting apart, now and then sipping from his pot and looking round at the
company. He, too, appeared to be in some agitation.»

SECOND-MOST (2, 0.7337720990180969): «Happy families are all alike; every unhappy
family is unhappy in its own way.
...
...
"It's that idiotic smile that's to blame for it all," thought Stepan
Arkadyevitch. "But what's to be done? What's to be done?" he said to himself in
despair, and found no answer.»

MEDIAN (1, 0.5950512886047363): «It was the best of times, it was the worst of
times, it was the age of wisdom, it was the age of foolishness
...
...
Thus did the year one thousand seven hundred and seventy-five conduct there
Greatnesses, and myriads of small creatures—the creatures of this chronicle among
the rest—along the roads that lay before them.»

LEAST (0, -0.135994553565979): «At a certain season of our life we are accustomed
to consider every spot as the possible site of a house.
...
...
so by the divining-rod and thin rising vapors I judge; and here I will begin to
mine.»

Document (4): «"Well, Prince, so Genoa and Lucca are now just family estates of
the Buonapartes.
...
...
It shall be on your family's behalf that I'll start my apprenticeship as old
maid."»

SIMILAR/DISSIMILAR DOCS PER MODEL Doc2Vec(dm/m,d64,n5,w5,mc3,s0.001,t3):

MOST (4, 0.9884666204452515): «"Well, Prince, so Genoa and Lucca are now just
family estates of the Buonapartes.
...
...
It shall be on your family's behalf that I'll start my apprenticeship as old
maid."»
```

```
SECOND-MOST (2, 0.4948118329048157): «Happy families are all alike; every unhappy
family is unhappy in its own way.
...
...
"But what's to be done? What's to be done?" he said to himself in despair, and
found no answer.»

MEDIAN (3, 0.3094167113304138): «On an exceptionally hot evening early in July a
young man came out of the garret in which he lodged in S.
...
...
He was sitting apart, now and then sipping from his pot and looking round at the
company. He, too, appeared to be in some agitation.»

LEAST (0, 0.1593853235244751): «At a certain season of our life we are accustomed
to consider every spot as the possible site of a house.
...
...
I think that the richest vein is somewhere hereabouts; so by the divining-rod and
thin rising vapors I judge; and here I will begin to mine.»
```