使用三种方法计算TFIDF向量

原创 班长 NLP学习教室 2018-10-14

一. 摘要

这篇文章主要介绍了计算TF-IDF的不同方法实现，主要有三种方法：

> - 用gensim库来计算tfidf值
> - 用sklearn库来计算tfidf值
> - 用python手动实现tfidf的计算

关于TFIDF的算法原理我就不过多介绍了，看这篇博客即可——TF-IDF原理（**http://www.ruanyifeng.com/blog/2013/03/tf-idf.html**）。阮一峰大佬写的，浅显易懂，看了这么多篇就这篇最好懂。

二. 正文

1.使用gensim提取文本的tfidf特征

首先来看我们的语料库

```
corpus = [
    'this is the first document',
    'this is the second second document',
    'and the third one',
    'is this the first document'
]
```

接下来看我们的处理过程
1)把语料库做一个分词的处理

```
[输入]:
word_list = []
for i in range(len(corpus)):
    word_list.append(corpus[i].split(' '))
print(word_list)

[输出]:
```

```
[['this', 'is', 'the', 'first', 'document'],
 ['this', 'is', 'the', 'second', 'second', 'document'],
 ['and', 'the', 'third', 'one'],
 ['is', 'this', 'the', 'first', 'document']]
```

2) 得到每个词的id值及词频

```
[输入]:
from gensim import corpora
# 赋给语料库中每个词(不重复的词)一个整数id
dictionary = corpora.Dictionary(word_list)
new_corpus = [dictionary.doc2bow(text) for text in word_list]
print(new_corpus)

# 元组中第一个元素是词语在词典中对应的id，第二个元素是词语在文档中出现的次数
[输出]:
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)],
 [(0, 1), (2, 1), (3, 1), (4, 1), (5, 2)],
 [(3, 1), (6, 1), (7, 1), (8, 1)],
 [(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)]]

 [输入]:
 # 通过下面的方法可以看到语料库中每个词对应的id
 print(dictionary.token2id)
 [输出]:
 {'document': 0, 'first': 1, 'is': 2, 'the': 3, 'this': 4, 'second': 5, 'and':
 'one': 7,   'third': 8}
```

3)训练gensim模型并且保存它以便后面的使用

```
[输入]:
# 训练模型并保存
from gensim import models
tfidf = models.TfidfModel(new_corpus)
tfidf.save("my_model.tfidf")

# 载入模型
tfidf = models.TfidfModel.load("my_model.tfidf")

# 使用这个训练好的模型得到单词的tfidf值
tfidf_vec = []
for i in range(len(corpus)):
    string = corpus[i]
    string_bow = dictionary.doc2bow(string.lower().split())
    string_tfidf = tfidf[string_bow]
    tfidf_vec.append(string_tfidf)
print(tfidf_vec)

[输出]:
```

```
[[(0, 0.33699829595119235),
  (1, 0.8119707171924228),
  (2, 0.33699829595119235),
  (4, 0.33699829595119235)],
 [(0, 0.10212329019650272),
  (2, 0.10212329019650272),
  (4, 0.10212329019650272),
  (5, 0.9842319344536239)],
 [(6, 0.5773502691896258), (7, 0.5773502691896258), (8, 0.5773502691896258)],
 [(0, 0.33699829595119235),
  (1, 0.8119707171924228),
  (2, 0.33699829595119235),
  (4, 0.33699829595119235)]]
```

通过上面的计算我们发现这向量的维数和我们语料单词的个数不一致呀，我们要得到的是每个词的tfidf值，为了一探究竟我们再做个小测试

4) 小测试现出gensim计算的原形

```
[输入]:
# 我们随便拿几个单词来测试
string = 'the i first second name'
string_bow = dictionary.doc2bow(string.lower().split())
string_tfidf = tfidf[string_bow]
print(string_tfidf)

[输出]:
[(1, 0.4472135954999579), (5, 0.8944271909999159)]
```

结论

> - gensim训练出来的tf-idf值左边是词的id，右边是词的tfidf值
> - gensim有自动去除停用词的功能，比如the
> - gensim会自动去除单个字母，比如i
> - gensim会去除没有被训练到的词，比如name
> - 所以通过gensim并不能计算每个单词的tfidf值

2.使用sklearn提取文本tfidf特征

我们的语料库不变，还是上面那个

```
corpus = [
    'this is the first document',
    'this is the second second document',
    'and the third one',
    'is this the first document'
]
```

然后来看我们的处理过程

```
[输入]:
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vec = TfidfVectorizer()
tfidf_matrix = tfidf_vec.fit_transform(corpus)

# 得到语料库所有不重复的词
print(tfidf_vec.get_feature_names())

# 得到每个单词对应的id值
print(tfidf_vec.vocabulary_)

# 得到每个句子所对应的向量
# 向量里数字的顺序是按照词语的id顺序来的
print(tfidf_matrix.toarray())

[输出]:
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']

{'this': 8, 'is': 3, 'the': 6, 'first': 2, 'document': 1, 'second': 5, 'and':

[[0.         0.43877674 0.54197657 0.43877674 0.         0.
  0.35872874 0.         0.43877674]
 [0.         0.27230147 0.         0.27230147 0.         0.85322574
  0.22262429 0.         0.27230147]
 [0.55280532 0.         0.         0.         0.55280532 0.
  0.28847675 0.55280532 0.        ]
 [0.         0.43877674 0.54197657 0.43877674 0.         0.
  0.35872874 0.         0.43877674]]
```

3.python提取文本的tfidf特征

我们的语料库依旧不变

```
corpus = [
    'this is the first document',
    'this is the second second document',
    'and the third one',
    'is this the first document'
]
```

1) 对语料进行分词

```
[输入]:
word_list = []
for i in range(len(corpus)):
    word_list.append(corpus[i].split(' '))
print(word_list)

[输出]:
[['this', 'is', 'the', 'first', 'document'],
 ['this', 'is', 'the', 'second', 'second', 'document'],
 ['and', 'the', 'third', 'one'],
 ['is', 'this', 'the', 'first', 'document']]
```

2) 统计词频

```
[输入]:
countlist = []
for i in range(len(word_list)):
    count = Counter(word_list[i])
    countlist.append(count)
countlist

[输出]:
[Counter({'document': 1, 'first': 1, 'is': 1, 'the': 1, 'this': 1}),
 Counter({'document': 1, 'is': 1, 'second': 2, 'the': 1, 'this': 1}),
 Counter({'and': 1, 'one': 1, 'the': 1, 'third': 1}),
 Counter({'document': 1, 'first': 1, 'is': 1, 'the': 1, 'this': 1})]
```

3) 定义计算tfidf公式的函数

```
# word可以通过count得到，count可以通过countlist得到

# count[word]可以得到每个单词的词频， sum(count.values())得到整个句子的单词总数
def tf(word, count):
    return count[word] / sum(count.values())

# 统计的是含有该单词的句子数
def n_containing(word, count_list):
    return sum(1 for count in count_list if word in count)

# len(count_list)是指句子的总数，n_containing(word, count_list)是指含有该单词的句子
def idf(word, count_list):
    return math.log(len(count_list) / (1 + n_containing(word, count_list)))

# 将tf和idf相乘
def tfidf(word, count, count_list):
    return tf(word, count) * idf(word, count_list)
```

4) 计算每个单词的tfidf值

```
[输入]:
import math
for i, count in enumerate(countlist):
    print("Top words in document {}".format(i + 1))
    scores = {word: tfidf(word, count, countlist) for word in count}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    for word, score in sorted_words[:]:
        print("\tWord: {}, TF-IDF: {}".format(word, round(score, 5)))

[输出]:
Top words in document 1
    Word: first, TF-IDF: 0.05754
    Word: this, TF-IDF: 0.0
    Word: is, TF-IDF: 0.0
    Word: document, TF-IDF: 0.0
    Word: the, TF-IDF: -0.04463
Top words in document 2
    Word: second, TF-IDF: 0.23105
    Word: this, TF-IDF: 0.0
    Word: is, TF-IDF: 0.0
    Word: document, TF-IDF: 0.0
    Word: the, TF-IDF: -0.03719
Top words in document 3
    Word: and, TF-IDF: 0.17329
    Word: third, TF-IDF: 0.17329
    Word: one, TF-IDF: 0.17329
    Word: the, TF-IDF: -0.05579
Top words in document 4
    Word: first, TF-IDF: 0.05754
    Word: is, TF-IDF: 0.0
    Word: this, TF-IDF: 0.0
    Word: document, TF-IDF: 0.0
    Word: the, TF-IDF: -0.04463
```