



Eastmount的专栏

无知 · 乐观 · 谦逊 · 低调 · 生活

目录视图

摘要视图

RSS 订阅

个人资料



Eastmount



访问：1479290次

积分：16777

等级：BLOG>7

排名：第577名

原创：252篇 转载：10篇

译文：0篇 评论：1244条

文章搜索

Q

个人博客

作者：杨秀璋
学历：本科-北京理工大学
 硕士-北京理工大学
现任教于贵财财经大学信息学院
<http://www.eastmountyxz.com>

简介：自幼受贵州大山的熏陶，养成了诚实质朴的性格。经过寒窗苦读，考入BIT，为完成自己的教师梦，放弃IT、航天等工作，成为贵财一名大学教师，并想把自己所学所感真心传授给自己的学生，帮助更多陌生人。

贵州纵美路迢迢，
为负劳心此一遭。
收得破书三四本，
也堪将去教尔曹。

娜美人生，醉美生活。
他和她经历风雨，慢慢变老。

博客专栏

- 数据库实战开发设计与优化
文章：13篇
阅读：63364
- HTML网站前端设计
文章：12篇
阅读：48178

评论送书 | 7月书讯：众多畅销书升级！ 征文 | 你会为 AI 转型么？ 赠书 | AI专场（AI圣经！《深度学习》中文版）

[python] 使用scikit-learn工具计算文本TF-IDF值

标签：TF-IDF python 权重计算 Scikit-learn 源码分析

2016-08-08 16:46

8930人阅读

评论(3)

收藏

举报

分类：

知识图谱（14）

机器学习（20）

版权声明：本文为博主原创文章，转载请注明CSDN博客源地址！共同学习，一起进步~

目录(?)

[+]

在文本聚类、文本分类或者比较两个文档相似程度过程中，可能会涉及到TF-IDF值的计算。这里主要讲述基于Python的机器学习模块和开源工具：scikit-learn。

希望文章对你有所帮助，相关文章如下：

[\[python爬虫\] Selenium获取百度百科旅游景点的InfoBox消息盒](#)

[Python简单实现基于VSM的余弦相似度计算](#)

[基于VSM的命名实体识别、歧义消解和指代消解](#)

[\[python\] 使用Jieba工具中文分词及文本聚类概念](#)

目录：

一.Scikit-learn概念

- 1.概念知识
- 2.安装软件

二.TF-IDF基础知识

- 1.TF-IDF
- 2.举例介绍

三.TF-IDF调用两个方法

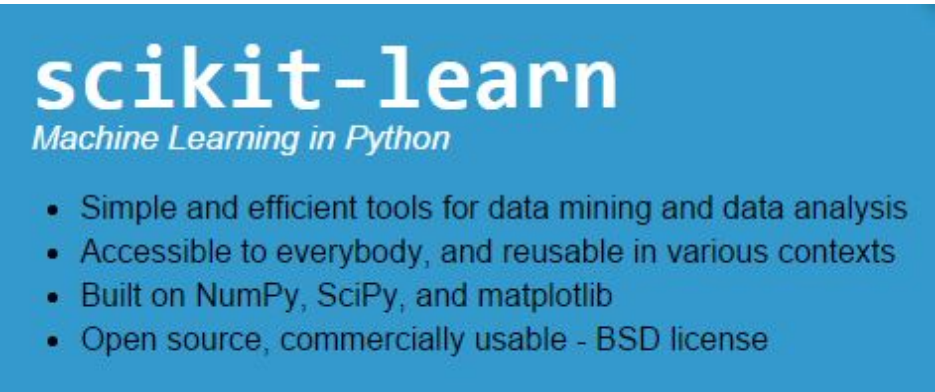
- 1.CountVectorizer
- 2.TfidfTransformer
- 3.别人示例

一. Scikit-learn概念

1.概念知识

官方网址：<http://scikit-learn.org/stable/>

Scikit-learn是一个用于数据挖掘和数据分析的简单且有效的工具，它是基于Python的机器学习模块，基于BSD开源许可证。



Scikit-learn的基本功能主要被分为六个部分：分类(Classification)、回归



知识图谱、web数据挖掘及NLP
文章：36篇
阅读：272589



Python爬虫之Selenium+PhantomJS
文章：29篇
阅读：252451



MFC应用及图像处理
文章：18篇
阅读：133245



PHP基础知识及网站开发
文章：12篇
阅读：59736



Android实例开发与学习
文章：19篇
阅读：140845



Python学习系列
文章：16篇
阅读：74779



C# 系统应用知识
文章：20篇
阅读：109915



C# 网络编程知识
文章：13篇
阅读：81570

- 文章分类
- 汇编知识 (4)

MFC基础知识 (11)

C#基础知识 (5)

设计模式 (4)

C#网络编程 (13)

商务智能 (1)

C#数据库知识 (4)

C#窗体操作 (19)

PHP基础知识 (11)

C#文件操作 (2)

算法知识 (3)

MFC图像知识 (14)

数字图像处理 (7)

C#系统应用 (20)

Error报告 (4)

C/C++基础知识 (6)

编程生活 (8)

Python基础知识 (19)

数据挖掘 (15)

Android (21)

编程杂谈 (8)

软件项目管理 (1)

机器学习 (21)

百度地图开发 (5)

学习排序 (3)

自然语言处理 (4)

读书笔记 (5)

Java网站开发 (9)

知识图谱 (15)

(Regression)、聚类(Clustering)、数据降维(Dimensionality reduction)、模型选择(Model selection)、数据预处理(Preprocessing)。

Scikit-Learn中的机器学习模型非常丰富，包括SVM，决策树，GBDT，KNN等等，可以根据问题的类型选择合适的模型，具体可以参考官网文档，推荐大家从官网中下载资源、模块、文档进行学习。

<div>Classification</div> <div>Identifying to which category an object belongs to.</div> <div>Applications: Spam detection, Image recognition.</div> <div>Algorithms: SVM, nearest neighbors, random forest, ...</div> <div>Examples</div>	<div>Regression</div> <div>Predicting a continuous-valued attribute associated with an object.</div> <div>Applications: Drug response, Stock prices.</div> <div>Algorithms: SVR, ridge regression, Lasso, ...</div> <div>Examples</div>	<div>Clustering</div> <div>Automatic grouping of similar objects into sets.</div> <div>Applications: Customer segmentation, Grouping experiment outcomes</div> <div>Algorithms: k-Means, spectral clustering, mean-shift, ...</div> <div>Examples</div>
<div>Dimensionality reduction</div> <div>Reducing the number of random variables to consider.</div> <div>Applications: Visualization, Increased efficiency</div> <div>Algorithms: PCA, feature selection, non-negative matrix factorization.</div> <div>Examples</div>	<div>Model selection</div> <div>Comparing, validating and choosing parameters and models.</div> <div>Goal: Improved accuracy via parameter tuning</div> <div>Modules: grid search, cross validation, metrics.</div> <div>Examples</div>	<div>Preprocessing</div> <div>Feature extraction and norm</div> <div>Application: Transforming i</div> <div>text for use with machine lea</div> <div>Modules: preprocessing, fei</div>

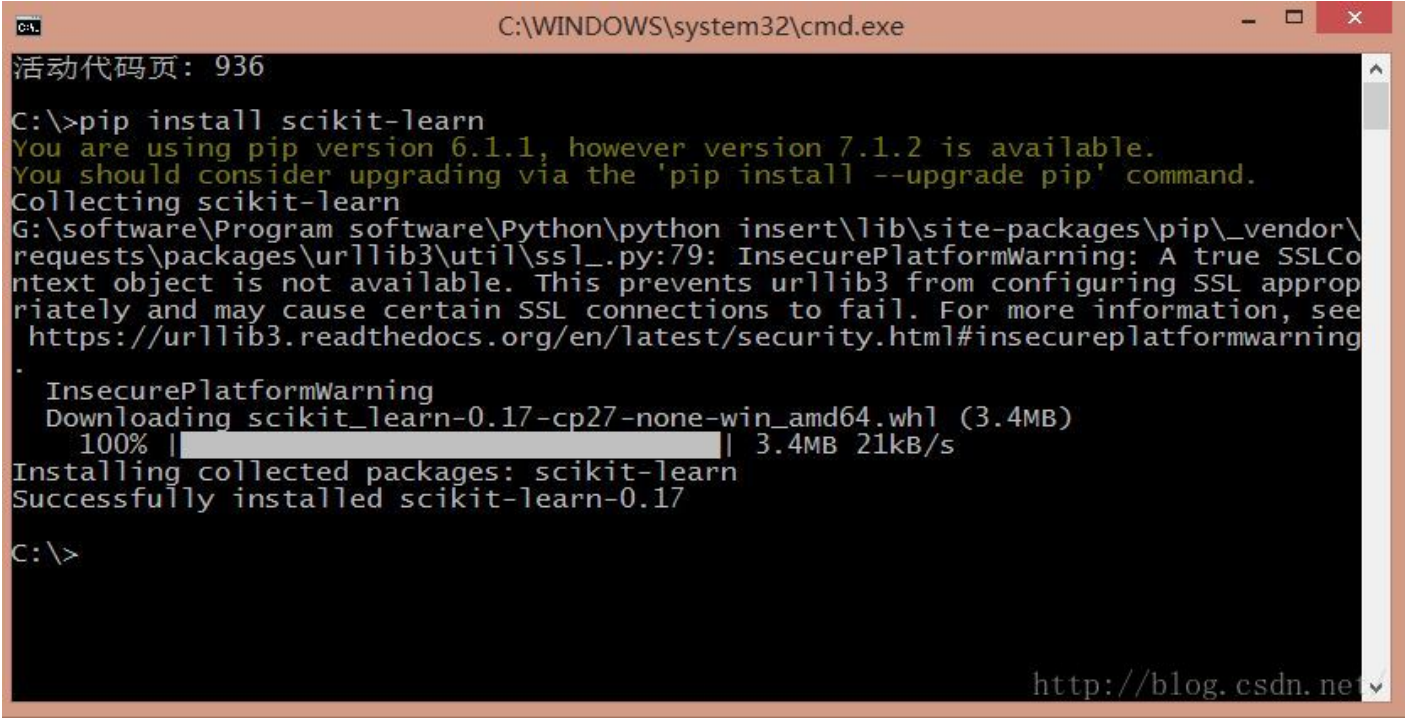
Scikit-Learn的安装需要numpy、scipy、matplotlib等模块，windows用户可以到：<http://www.lfd.uci.edu/~gohlke/pythonlibs>直接下载编译好的安装包以及源码，可以到这个网站下载：http://sourceforge.jp/projects/sfnet_scikit-learn/。

参考文章：[开源机器学习工具scikit-learn入门 - 轩辕森](#)

2.安装软件

Python 2.0我推荐使用"pip install scikit-learn"或"easy_install scikit-learn"全自动安装，再通过"from sklearn import feature_extraction"导入。

安装时如果出现错误"unknown encoding: cp65001"，输入"chcp 936"将编码方式由utf-8变为简体中文gbk。



二. TF-IDF基础知识

参考官方文档：
gensim中TF-IDF：<http://radimrehurek.com/gensim/models/tfidfmodel.html>

1.TF-IDF

TF-IDF (Term Frequency-InversDocument Frequency) 是一种常用于信息处理和数据挖掘的加权技术。该技术采用一种统计方法，根据字词的在文本中出现的次数和在整个语料中出现的文档频率来计算一个字词在整个语料中的重要程度。它的优点是能过滤掉一些常见的却无关紧要本的词语，同时保留影响整个文本的重要字词。计算方法如下面公式所示。

面试工作 (6)

Python爬虫 (29)

LeetCode (15)

数据库 (13)

HTML网页知识 (9)

Python数据挖掘课程 (18)

Putty学习 (1)

个人网站搭建 (4)

黑科技 (4)

Echarts可视化 (2)

Office (1)

Python网站开发 (0)

渗透&攻防 (3)

文章存档

2017年07月 (3)

2017年06月 (3)

2017年05月 (2)

2017年04月 (4)

2017年03月 (11)

展开

阅读排行

word2vec词向量训练及中 (29783)

[python] 基于k-means和I (23373)

Android百度地图之位置 (20157)

[C/C++基础知识] main函 (19865)

[python] 常用正则表达式 (18589)

[python] 使用Jieba工具中 (18177)

C# 系统应用之TreeView: (17192)

Java+MyEclipse+Tomcat (17141)

中文知识图谱研讨会的学 (17092)

Echarts字体和线条颜色 (16664)

评论排行

2016年总结：教师路的牙 (89)

2016年总结：教师路的牙 (74)

[python爬虫] Selenium爬 (73)

再见北理工：忆北京研究 (68)

回忆自己的大学四年得与 (42)

[Python爬虫] Selenium (31)

[python] LDA处理文档主 (31)

【学习排序】 Learning to (30)

[python] 基于k-means和I (29)

【数字图像处理】七.MFI (23)

推荐文章

* CSDN日报20170721——《为什么我们创业失败了和选择创业公司的思考》

* 深入剖析基于并发AQS的重入锁(ReentrantLock)及其Condition实现原理

* Android版本的"Wannacry"文件加密病毒样本分析(附带锁机)

* 工作与生活真的可以平衡吗？

$$tfidf_{i,j} = tf_{i,j} \times idf_j$$

其中，式中 $tfidf_{i,j}$ 表示词频 $tf_{i,j}$ 和倒文本词频 idf_j 的乘积。TF-IDF值越大表示该特征词对这个文本的重要性越大。

TF（Term Frequency）表示某个关键词在整篇文章中出现的频率。

IDF（InversDocument Frequency）表示计算倒文本频率。文本频率是指某个关键词在整个语料所有文章中出现的次数。倒文档频率又称为逆文档频率，它是文档频率的倒数，主要用于降低所有文档中一些常见但对文档影响不大的词语的作用。

下面公式是TF词频的计算公式。

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

其中， $n_{i,j}$ 为特征词 t_i 在文本 d_j 中出现的次数，是文本 d_j 中所有特征词的个数结果即为某个特征词的词频。

下面公式是IDF的计算公式。

$$idf_{i,j} = \log \frac{|D|}{1 + |D_{t_i}|}$$

其中， $|D|$ 表示语料中文本的总数，表示文本中包含特征词 t_i 的数量。为防止该词语在语料库中不存在，即分母为0，则使用1作为分母。

2.示例

示例参考仿造阮一峰大神的例子进行简单讲解，推荐大家去阅读：[TF-IDF与余弦相似性的应用（一）：自动提取关键词](#)

下面通过一个示例进行讲解TF-IDF权重计算的方法。

假设现在有一篇文章《贵州的大数据分析》，这篇文章包含了10000个词组，其中“贵州”、“大数据”、“分析”各出现100次，“的”出现500次（假设没有去除停用词），则通过前面TF词频计算公式，可以计算得到三个单词的词频，即：

词频 = $\frac{\text{某个词组在文章中出现的次数}}{\text{该文章的总词组数}}$

TF(贵州)=100/10000=0.01

TF(大数据)=100/10000=0.01

TF(分析)=100/10000=0.01

T(的)=500/10000=0.05

现在预料库中共存在1000篇文章，其中包含“贵州”的共99篇，包含“大数据”的共19篇，包含“分析”的共“59”篇，包含“的”共“899”篇。则它们的IDF计算如下：

逆文档频率 = $\log(\frac{\text{语料库中文档总数}}{\text{包含该词组的文章个数}+1})$

IDF(贵州)=log(1000/100)=1.000

IDF(大数据)=log(1000/20)=1.700

IDF(分析)=log(1000/60)=1.221

IDF(的)=log(1000/900)=0.046

By: Eastmount CSDN

由IDF可以发现，当某个词在语料库中各个文档出现的次数越多，它的IDF值越低，当它在所有文档中都出现时，其IDF计算结果为0，而通常这些出现次数非常多的词或字为“的”、“我”、“吗”等，它对文章的权重计算起不到一定的作用。

* 《Real-Time Rendering 3rd》提炼总结——高级着色：BRDF及相关技术

* 《三体》读后思考-泰勒展开/维度打击/黑暗森林

牛人博客

【算法知识】v_july_v
【C# .NET】杨友山 孟子E章
【游戏开发】拳四郎 lufy
 秦元培 毛星云
【Android】罗升阳 欧阳鹏
 郭霖 CodingSnail
【正能量&导师】陆其明 刘未鹏
 沈逸 贺利坚 金旭亮
【图形&视频处理】小魏 雷霄骅
【各种知识】21aspnet 传智播客
 实在科技 Rachel-Zhang
【MFC】friendan
【数据挖掘】cowboy_wz
【Linux】倦飞L
【.NET MF&嵌入式】叶帆
【架构&模式】DarrenF 丁国华
【操作系统】garryxin
【IOS开发】Colin
【CSDN扫地僧】zzwu

最新评论

[Python爬虫] Selenium爬取新浪rookie95: 博主，运行登录后进入的不是微博界面而是新浪新闻界面，也找不到mobile元素了。不知该如何修改？

【数字图像处理】一.MFC详解Mce_周: @Eastmount:博主？我这有个问题，我现在是用你的打开、显示、保存、另存为图片的所有函数，但是...

[python爬虫] BeautifulSoup和SeEastmount: @a393134314:学弟好，在论坛BIT的好像见过几个，但也不多，哈哈

【数字图像处理】三.MFC实现图Eastmount: @Mce_19:这部分我还没试过，保存到时再Save函数写就好的，但是画图在保存，应该是需要鼠标操作...

网站开发之HTML基础表格TableEastmount: @github_36326955:你好，这部分网站开发是我的弱项，可能可以试试采用计数器的方式，相当...

【数字图像处理】一.MFC详解图Eastmount: @Mce_19:你好，这个还不太是，属于底层代码 相当于是实现GDI类似功能，后面的文章是基于这个底...

再见北理工：忆北京研究生的编Eastmount: @MrMoGu:哈哈 哈，学弟或学妹，你好，认识你很开心，这是当时自己的一些感慨，希望你也做些自己喜欢...

[python爬虫] BeautifulSoup和SeLeonhard_: 居然是一个学长，支持一下

【数字图像处理】三.MFC实现图Mce_周: 楼主，如果在图片上面画上线然后在保存图片如何处理？请指教。

再见北理工：忆北京研究生的编MrMoGu: 看的眼角有一丝热泪划过。。。同院校友

同时计算TF-IDF值如下：



通过TF-IDF计算，“大数据”在某篇文章中出现频率很高，这就能反应这篇文章的主题就是关于“大数据”方向的。如果只选择一个词，“大数据”就是这篇文章的关键词。所以，可以通过TF-IDF方法统计文章的关键词。同时，如果同时计算“贵州”、“大数据”、“分析”的TF-IDF，将这些词的TF-IDF相加，可以得到整篇文档的值，用于信息检索。

TF-IDF算法的优点是简单快速，结果比较符合实际情况。缺点是单纯以该词的重要性，不够全面，有时重要的词可能出现次数并不多。而且，这种算法没有考虑词的位置信息。

三. TF-IDF计算

Scikit-Learn中TF-IDF权重计算方法主要用到两个类：CountVectorizer和TfidfTransformer。

1.CountVectorizer

CountVectorizer类会将文本中的词语转换为词频矩阵，例如矩阵中包含一个元素a[i][j]，它表示j词在i类文本下的词频。它通过fit_transform函数计算各个词语出现的次数，通过get_feature_names()可获取词袋中所有文本的关键字，通过toarray()可看到词频矩阵的结果。

代码如下：

```
[python]
01. # coding:utf-8
02. from sklearn.feature_extraction.text import CountVectorizer
03.
04. #语料
05. corpus = [
06.     'This is the first document.',
07.     'This is the second second document.',
08.     'And the third one.',
09.     'Is this the first document?',
10. ]
11. #将文本中的词语转换为词频矩阵
12. vectorizer = CountVectorizer()
13. #计算个词语出现的次数
14. X = vectorizer.fit_transform(corpus)
15. #获取词袋中所有文本关键词
16. word = vectorizer.get_feature_names()
17. print word
18. #查看词频结果
19. print X.toarray()
```

输出如下所示：

```
[python]
01. >>>
02. [u'and', u'document', u'first', u'is', u'one', u'second', u'the', u'third', u'this']
03. [[0 1 1 1 0 0 1 0 1]
04.  [0 1 0 1 0 2 1 0 1]
05.  [1 0 0 0 1 0 1 1 0]
06.  [0 1 1 1 0 0 1 0 1]]
07. >>>
```

从结果中可以看到，总共包括9个特征词，即：

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

同时在输出每个句子中包含特征词的个数。例如，第一句“This is the first

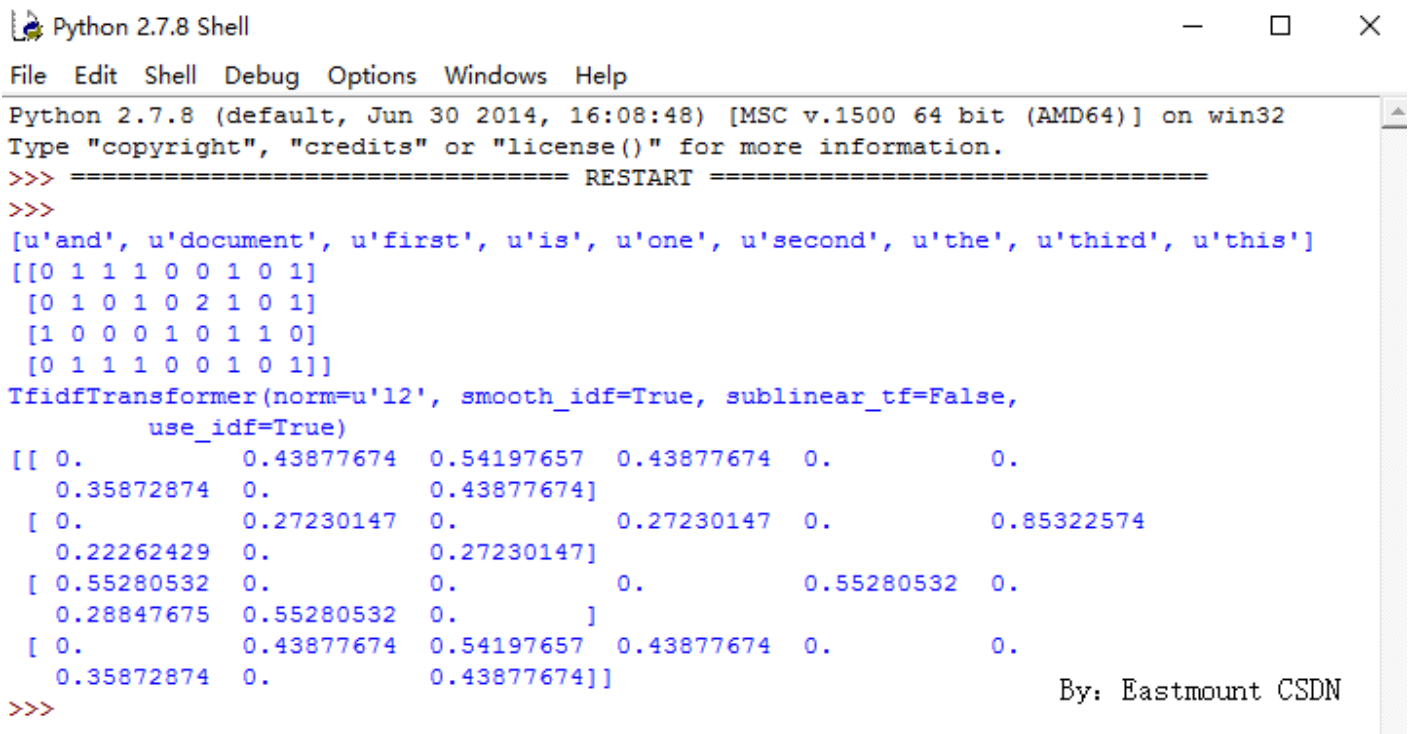
document.”，它对应的词频为[0, 1, 1, 1, 0, 0, 1, 0, 1]，假设初始序号从1开始计数，则该词频表示存在第2个位置的单词“document”共1次、第3个位置的单词“first”共1次、第4个位置的单词“is”共1次、第9个位置的单词“this”共1词。所以，每个句子都会得到一个词频向量。

2.TfidfTransformer

TfidfTransformer用于统计vectorizer中每个词语的TF-IDF值。具体用法如下：

```
[python]
01. # coding:utf-8
02. from sklearn.feature_extraction.text import CountVectorizer
03.
04. #语料
05. corpus = [
06.     'This is the first document.',
07.     'This is the second second document.',
08.     'And the third one.',
09.     'Is this the first document?',
10. ]
11. #将文本中的词语转换为词频矩阵
12. vectorizer = CountVectorizer()
13. #计算个词语出现的次数
14. X = vectorizer.fit_transform(corpus)
15. #获取词袋中所有文本关键词
16. word = vectorizer.get_feature_names()
17. print word
18. #查看词频结果
19. print X.toarray()
20.
21. from sklearn.feature_extraction.text import TfidfTransformer
22.
23. #类调用
24. transformer = TfidfTransformer()
25. print transformer
26. #将词频矩阵X统计成TF-IDF值
27. tfidf = transformer.fit_transform(X)
28. #查看数据结构 tfidf[i][j]表示i类文本中的tf-idf权重
29. print tfidf.toarray()
```

输出结果入下所示：



3.别人示例

如果需要同时进行词频统计并计算TF-IDF值，则使用核心代码：

```
vectorizer=CountVectorizer()
transformer=TfidfTransformer()
tfidf=transformer.fit_transform(vectorizer.fit_transform(corpus))
```

下面给出一个liuxuejiang158大神的例子，供大家学习，推荐大家阅读原文：

[python scikit-learn计算tf-idf词语权重 - liuxuejiang](#)

```
[python]
01. # coding:utf-8
02. __author__ = "liuxuejiang"
03. import jieba
04. import jieba.posseg as pseg
05. import os
06. import sys
07. from sklearn import feature_extraction
08. from sklearn.feature_extraction.text import TfidfTransformer
09. from sklearn.feature_extraction.text import CountVectorizer
10.
11. if __name__ == "__main__":
12.     corpus=["我 来到 北京 清华大学",#第一类文本切词后的结果，词之间以空格隔开
13.            "他 来到 了 网易 杭研 大厦",#第二类文本的切词结果
14.            "小明 硕士 毕业 与 中国 科学院",#第三类文本的切词结果
15.            "我 爱 北京 天安门"]#第四类文本的切词结果
16.     vectorizer=CountVectorizer()#该类会将文本中的词语转换为词频矩阵，矩阵元素a[i][j]表示i词在i类
文本下的词频
17.     transformer=TfidfTransformer()#该类会统计每个词语的tf-idf权值
18.     tfidf=transformer.fit_transform(vectorizer.fit_transform(corpus))#第一个f
是计算tf-idf，第二个fit_transform是将文本转为词频矩阵
19.     word=vectorizer.get_feature_names()#获取词袋模型中的所有词语
20.     weight=tfidf.toarray()#将tf-idf矩阵抽取出来，元素a[i][j]表示j词在i类文本中的tf
21.     for i in range(len(weight)):#打印每类文本的tf-idf词语权重，第一个for遍历所有文本，第二个for便
利某一类文本下的词语权重
22.         print u"-----这里输出第",i,u"类文本的词语tf-idf权重-----"
23.         for j in range(len(word)):
24.             print word[j],weight[i][j]
```

输出如下所示：

```
[python]
01. -----这里输出第 0 类文本的词语tf-idf权重----- #该类对应的原文本是："我来到北京清华大
学"
02. 中国 0.0
03. 北京 0.52640543361
04. 大厦 0.0
05. 天安门 0.0
06. 小明 0.0
07. 来到 0.52640543361
08. 杭研 0.0
09. 毕业 0.0
10. 清华大学 0.66767854461
11. 硕士 0.0
12. 科学院 0.0
13. 网易 0.0
14. -----这里输出第 1 类文本的词语tf-idf权重----- #该类对应的原文本是："他来到了网易杭研
大厦"
15. 中国 0.0
16. 北京 0.0
17. 大厦 0.525472749264
18. 天安门 0.0
19. 小明 0.0
20. 来到 0.414288751166
21. 杭研 0.525472749264
22. 毕业 0.0
23. 清华大学 0.0
24. 硕士 0.0
25. 科学院 0.0
26. 网易 0.525472749264
27. -----这里输出第 2 类文本的词语tf-idf权重----- #该类对应的原文本是："小明硕士毕业于中
国科学院"
28. 中国 0.4472135955
29. 北京 0.0
30. 大厦 0.0
31. 天安门 0.0
32. 小明 0.4472135955
33. 来到 0.0
34. 杭研 0.0
35. 毕业 0.4472135955
36. 清华大学 0.0
37. 硕士 0.4472135955
38. 科学院 0.4472135955
39. 网易 0.0
40. -----这里输出第 3 类文本的词语tf-idf权重----- #该类对应的原文本是："我爱北京天安
门"
41. 中国 0.0
42. 北京 0.61913029649
```



```
43. 大厦 0.0
44. 天安门 0.78528827571
45. 小明 0.0
46. 来到 0.0
47. 杭研 0.0
48. 毕业 0.0
49. 清华大学 0.0
50. 硕士 0.0
51. 科学院 0.0
52. 网易 0.0
```

推荐几篇机器学习和NLP领域的大神博客：

[应用scikit-learn做文本分类 - Rachel-Zhang](#)

[python scikit-learn计算tf-idf词语权重 - liuxuejiang](#)

[用Python开始机器学习（5：文本特征抽取与向量化）（强推） - lsldd](#)

[再谈word2vec - Felven \(强推\)](#)

[利用word2vec对关键词进行聚类 - Felven \(强推\)](#)

[Python 对文档内容TFIDF处理](#)

[Python TF-IDF计算100份文档关键词权重 - chenbjin](#)

最后希望文章对你有所帮助，如果文章中存在不足或错误的地方，还请海涵~还是那句话，挺享受现在的老师生活，不论科研、项目，还是教学，很充实，加油！

但行好事，莫问前程。

待随满天李桃，再追学友趣事。

(By:Eastmount 2016-08-08 下午5点 <http://blog.csdn.net/eastmount/>)

顶

6

踩

0

上一篇

《统计自然语言处理》读书笔记 一.基础知识及概念介绍

下一篇

[python] 专题九.Mysql数据库编程基础知识

相关文章推荐

- | | |
|---|--|
| <ul style="list-style-type: none">python scikit-learn计算tf-idf词语权重 | <ul style="list-style-type: none">sklearn之sklearn.feature_extraction.text.CountVe... |
| <ul style="list-style-type: none">短文本分析----基于python的TF-IDF特征词标签自... | <ul style="list-style-type: none">python 使用sklearn计算TF-IDF权重 |
| <ul style="list-style-type: none">使用sklearn进行文本TF-IDF处理 | <ul style="list-style-type: none">Sklearn TFIDF中文计算问题以及解决方法 |
| <ul style="list-style-type: none">sklearn 计算tf-idf | <ul style="list-style-type: none">三十三、利用scikit-learn计算tf-idf做文本词频分析 |
| <ul style="list-style-type: none">Matplotlib Tutorial(译) | <ul style="list-style-type: none">利用sklearn计算词频 |

猜你在找

- | | |
|---|---|
| 【直播】机器学习&深度学习系统实战（唐宇迪） | 【直播】Kaggle 神器：XGBoost 从基础到实战（冒教授） |
| 【直播回放】深度学习基础与TensorFlow实践（王琛） | 【直播】计算机视觉原理及实战（屈教授） |
| 【直播】机器学习之凸优化（马博士） | 【直播】机器学习之矩阵（黄博士） |
| 【直播】机器学习之概率与统计推断（冒教授） | 【直播】机器学习之数学基础 |

查看评论

【直播】TensorFlow实战进阶（智亮）

【直播】深度学习30天系统实训（唐宇迪）

2楼 [HEZCHAO](#) 2017-05-04 22:58发表



您改，感谢您可以分享这么有趣的知识。我的专业是人文地理学，目前也在研究TF-IDF算法，并将其应用到POI赋权。关于上文，我有一个疑惑，比如《3别人示例》中，在算词组的IDF时，北京的TF-IDF为 0.52640543361，可是我根据算法的定义怎么也算不出这个值。您可以详细说说吗？谢谢。

Re: [Eastmount](#) 2017-05-06 20:18发表



回复HEZCHAO：你好！具体的值，我没有仔细研究过，我是直接根据sklearn包进行计算的。人文地理学也搞Python研究了吗？不过确实用来解决人文类问题很好的，这篇文章一个例子可能对你有帮助。
<http://blog.csdn.net/eastmount/article/details/49898133>

1楼 [Danielntz](#) 2017-04-24 19:57发表




赞

发表评论

用户名：[haijunz](#)

评论内容：



提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

