使用TF-IDF算法给新闻自动生成关键字

实验目标

使用中文分词和TF-IDF算法，给几篇新闻自动地生成关键字。

介绍TF-IDF

TF-IDF *(term frequency-inverse document frequency)* 算法是一种统计方法，常用于资讯检索领域。基本原理是“单词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率反比下降”，由此得出的提取关键单词的算法。

关键概念

词频(term frequency, TF) ：某一个给定的单词在该文件中出现的次数。

$$TF(w,d) = \frac{count(w,d)}{size(d)}$$

其中$count(w,d)$是指单词w在文档d中出现的次数，$size(d)$是文档d中出现的总的单词个数

逆向文件频率(inverse document frequency, IDF) ： 用来衡量一个单词的普遍重要性。IDF是一个全局因子，其考虑的不是文档本身的特征，而是特征单词之间的相对重要性。特征单词出现的文档数目越多，IDF值越低，这个词区分不同文档的能力就越差。举个例子，“的”、”了“这样的词在很多文章中的词频都特别高，所以它们的IDF值就较低，不列入关键字的计算。

$$IDF(w) = log(\frac{n}{docs(w,D)})$$

n是文档总数，$docs(w,D)$是词w出现过的文件数。

介绍分词

分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。

中文和英文的不同之处在于，英文单词之间有明确的空格作为分界，而中文只有在句、段间才有分界符来分界。而且一句中文可能有多个切分成词的方案，若是单纯使用词典匹配可能会引起歧义。

例如：“南京市长江大桥”这一句，就有：南京市/长江/大桥，和： 南京/市长/江大桥 两种划分方案。

我们先不讨论分词算法的具体实现，在这里我们使用第三方分词工具来实现分词效果

PKUSeg 分词工具介绍

pkuseg 是由北京大学语言计算与机器学习研究组研制推出的一套全新的中文分词工具包。它简单易用，支持多领域分词，在不同领域的数据上都大幅提高了分词的准确率。目前仅支持python3

安装使用

我们在系统命令行使用pip来安装pkuseg:

```
$ pip install pkuseg
```

在python3文件里导入pkuseg包使用

```
import pkuseg
seg = pkuseg.pkuseg()

if __name__ == "__main__":
    print(seg.cut("北京大学语言计算与机器学习研究组研制推出了一套全新的中文分词工具包。它简单易用，支持多领域分词，在不同领域的数据上都大幅提高了分词的准确率。"))
##['北京', '大学', '语言', '计算', '与', '机器', '学习', '研究组', '研制', '推出', '了', '一', '套', '全新', '的', '中文', '分词', '工具包', '。', '它', '简单', '易用', '，', '支持', '多', '领域', '分词', '，', '在', '不同', '领域', '的', '数据', '上都', '大幅', '提高', '了', '分词', '的', '准确率', '。']
```

输出结果应该和注释中的一样。

在这里我们发现，分词工具将标点符号也算进了词组中，所以我们需要提前手动将标点符号从内容中删除：

```
def is_chinese(uchar):
    if uchar >= u'\u4e00' and uchar <= u'\u9fa5':
        return True
    else:
        return False

def format_str(content):
    content_str = ''
    for i in content:
        if is_chinese(i):
            content_str = content_str + i
    return content_str
```

使用`format_str()`函数就可以将字符串中不属于中文的字符全部剔除。

我们在当前目录下建立三个文件`a.txt`,`b.txt`和`c.txt`，每个文件中都有一篇新闻，我们将对这三篇新闻生成关键字。

```
filenames = ["a.txt","b.txt","c.txt"]

if __name__ == "__main__":
    corpus = []
    for name in filenames:
        with open(name,'r') as f:
            str = f.read()
            str = format_str(str)
            str = seg.cut(str)
            corpus.append(" ".join(str))
```

这样corpus中就有了三篇已经被分过词的文章，我们将使用sklearn自带的tfidf工具计算这三篇文章中各自的tfidf权重。

计算TF-IDF

首先在命令行目录下安装sklearn和numpy：

```
$ pip install scikit-learn
$ pip install numpy
```

在python文件中引用：

```
from sklearn import feature_extraction
from sklearn.feature_extraction.text import
TfidfTransformer
from sklearn.feature_extraction.text import
CountVectorizer
import numpy as np
```

在给corpus加入内容之后使用`CountVectorizer()`建立词频矩阵，`TfidfTransformer()`统计每个词语的tf-idf权值。

```
    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j词在i类文本下的词频
    transformer=TfidfTransformer()  #该类会统计每个词语的tf-idf权值

 tfidf=transformer.fit_transform(vectorizer.fit_transform(corpus))
    #第一个fit_transform是计算tf-idf，第二个fit_transform是将文本转为词频矩阵
    word=vectorizer.get_feature_names()

    #获取词袋模型中的所有词语
    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-idf权重
```

最后我们对每个文章，按权重排序，输出排名前五的词：

```
for (name, w) in zip(filenames,weight):
        print(name,": ")
        loc = np.argsort(-w)
        for i in range(5):
            print(i+1,word[loc[i]], w[loc[i]])
```

输出结果：

```
a.txt :
1 书店 0.799779102995797
2 实体 0.311778294388192
3 扶持 0.19655588124472975
4 北京市 0.1897780922362908
5 特色 0.155889147194096
b.txt :
1 生猪 0.45271828638081124
2 猪肉 0.31342035210979236
3 生产 0.20894690140652825
4 广西 0.17412241783877355
5 保障 0.13929793427101883
c.txt :
1 韩国 0.5025237118258845
2 美国 0.3517665982781192
3 防卫费 0.25126185591294226
4 报道 0.25126185591294226
5 朝鲜 0.20100948473035382
```

使用的三篇新闻报导：

http://www.xinhuanet.com/2019-11/21/c_1125257388.htm

https://k.sina.com.cn/article_5993531560_1653e08a802000nkhm.html

http://www.xinhuanet.com/2019-11/22/c_1125260006.htm