信息与知识获取作业三

唐文柏

2021211235

# 作业要求

作业3要求：   
  
基本要求：自己动手设计实现一个信息检索系统，中、英文皆可，数据源可以自选，数据通过开源的网络爬虫获取，规模不低于100篇文档，进行本地存储。中文可以分词（可用开源代码），也可以不分词，直接使用字作为基本单元。英文可以直接通过空格分隔。构建基本的倒排索引文件。实现基本的向量空间检索模型的匹配算法。用户查询输入可以是自然语言字串，查询结果输出按相关度从大到小排序，列出相关度、题目、主要匹配内容、URL、日期等信息。最好能对检索结果的准确率进行人工评价。界面不做强制要求，可以是命令行，也可以是可操作的界面。提交作业报告和源代码。  
  
扩展要求：鼓励有兴趣和有能力的同学积极尝试多媒体信息检索以及优化各模块算法，也可关注各类相关竞赛。自主开展相关文献调研与分析，完成算法评估、优化、论证创新点的过程。  
  
  
评分标准如下（按照100分计算）：  
  
1、  完成基本的信息检索功能且有对环境和社会可持续发展影响的考虑，系统能够正常运行，并提交源代码和实验报告：60分；  
  
2、  完成要求的信息检索功能且有对环境和社会可持续发展影响的考虑，系统能够正常运行，并按时提交源代码和实验报告：61~70分；  
  
3、  在2的基础上，且实验报告撰写认真、思路清晰、表达准确：71~80分;  
  
4、  在3的基础上，支持检索结果准确率人工评价：81~90分；  
  
5、  在4的基础上，融入了自己的创新性思考、优化算法或对多媒体信息检索进行了尝试：91-100分。  
  
  
提交时间：2024年6月14日之前  
  
提交文件命名要求：作业2-组长姓名-成员1姓名-成员2姓名

# 系统实现

## 数据集

为了方便实现，我们选择了现有的数据集，而不是自己手动从网络中爬取数据作为被检索的文档。经过多方比较，我们选择了使用比较广泛的 BBC 数据集。其包含了 2004 年 BBC 发表的一些关于运动的新闻。

### Dataset: BBCSport

All rights, including copyright, in the content of the original articles are owned by the BBC.

* Consists of 737 documents from the [BBC Sport](http://news.bbc.co.uk/sport1/hi/default.stm) website corresponding to sports news articles in five topical areas from 2004-2005.
* Class Labels: 5 (athletics, cricket, football, rugby, tennis)

- D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006.

我选择了400个样本放在data目录下

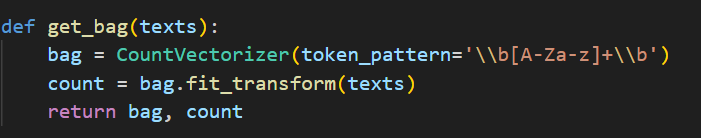
## 构建空间向量

我们使用 sklearn 来构建文档的空间向量。sklearn使用词袋（Bag of Words）和TF-IDF模型来表示文本数据，这两个模型都是One-Hot表示的应用，其中，词袋模型对应的就是文档向量。当前实现为了方便，我们使用词袋模型来构建空间向量，也就是说，我们只关注文档中是否出现给定的单词和单词出现频率，而舍弃文本的结构、单词出现的顺序和位置。

一般来说，对于一个文本语料库，构建词袋模型有四个步骤：

1. 文本分词：把每个文档中的文本进行分词
2. 构建词汇表：把文本分词得到的单词构建为一个词汇表，包含文本语料库中的所有单词，并对单词进行编号，假设词汇表有n个单词，单词编号从0开始，到n-1结束，可以把单词编号看作是单词的索引，通过单词编号可以唯一定位到该单词。
3. 词向量表示：每个单词都表示为一个n列的向量，在单词编号（词汇索引）位置上的列值为1，其他列的值为0
4. 统计频次：统计每个文档中每个单词出现的频次。

使用 sklearn 即可快速构建模型，其核心代码如下：



CountVectorizer 即为 sklearn 提供的构建词向量模型的类别，使用它可以很方便的构建出我们的模型。

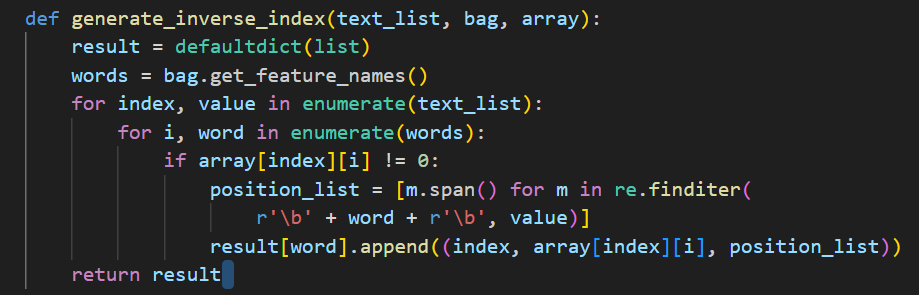
## 构建倒排索引

所谓倒排索引，就是按照索引去反向查找文件的过程。其思路为从单词角度看文档，标识每个单词分别在那些文档中出现(文档ID)，以及在各自的文档中每个单词分别出现了多少次（词频）及其出现位置（相对于该文档首部的偏移量）。

创建倒排索引，分为以下几步：

* 创建文档列表
* 创建倒排索引列表：对文档中数据进行分词，得到词条。对词条进行编号，以词条创建索引。然后记录下包含该词条的所有文档编号（及其它信息）。

核心代码如下：



## 检索

构建了逆向索引 inverse\_index，它将单词映射到文档中的位置。对用户查询进行处理，将查询词与逆向索引中的文档进行匹配。根据匹配到的文档，计算了每个文档的相关性和相似度，然后进行排名。



1. \*\*查询处理\*\*： - 将查询字符串分词并生成词列表`temp`和频率列表`freq`。 - 从`inverse\_index`中查找每个查询词的文档索引及其频率，并统计每个词的出现频率。

2. \*\*文档评分\*\*： - 创建一个结果字典`result\_dict`，用来存储匹配的文档及其相关信息（如频率、排名等）。 - 遍历`temp`中的文档索引，更新每个文档的频率、排名和出现位置。

3. \*\*相似度计算\*\*： - 使用`CountVectorizer`将查询转换为向量`search\_vec`。 - 计算查询向量和每个文档向量的余弦相似度，并更新文档的相似度值。

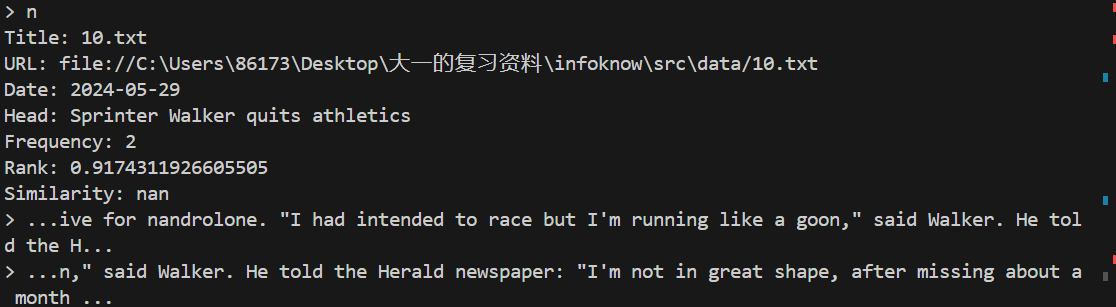
4. \*\*排序和返回结果\*\*： - 根据文档的排名和计数对结果列表`result\_list`进行排序。 - 返回排序后的结果列表。

查询结果输出按相关度从大到小排序，列出相关度、题目、主要匹配内容、URL、日期等信息。

示例：

**Title**: 文件名，对应于文档的标题或名称。

* **URL**: 文件的URL地址，如果文件位于本地，则是文件的本地路径。
* **Date**: 文件的日期，通常是文件的创建日期或修改日期。
* **Head**: 文件的开头部分，通常是文件的第一行或者前几个字。
* **Frequency**: 查询词在文件中出现的频率。
* **Rank**: 文件在搜索结果中的排名，根据查询词的频率和匹配程度计算得出。
* **Similarity**: 文件与查询字符串的相似度，根据向量空间模型计算得出。
* **Occurrence**: 查询词在文件中的出现位置，以文本片段的形式展示，前后各50个字符。



D.系统的命令解析：

运行环境python scikit-learn

运行main.py

**- n（next）**：继续显示下一个搜索结果。

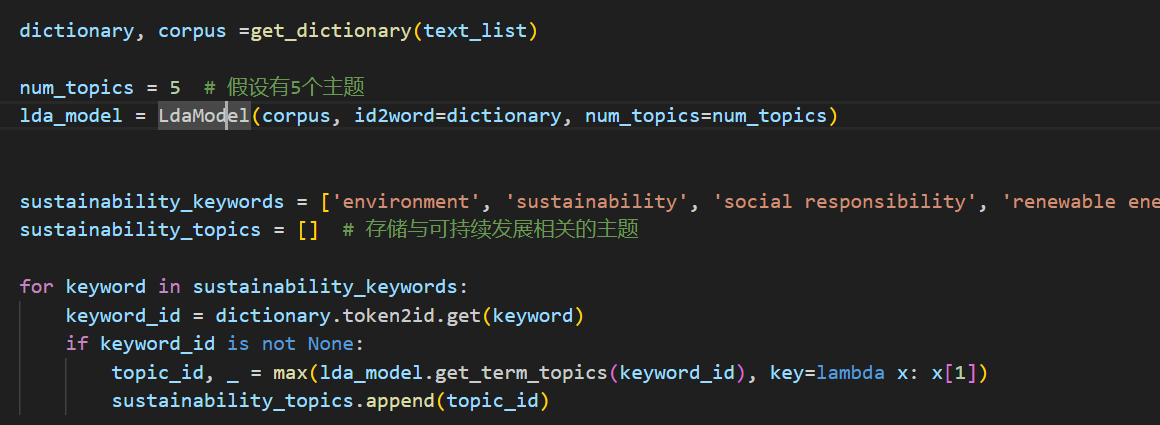
* **q**：quit
* r(rate): 评价这个推荐结果（1-5）评分最后会输出准确率和召回率

##### E.评分机制

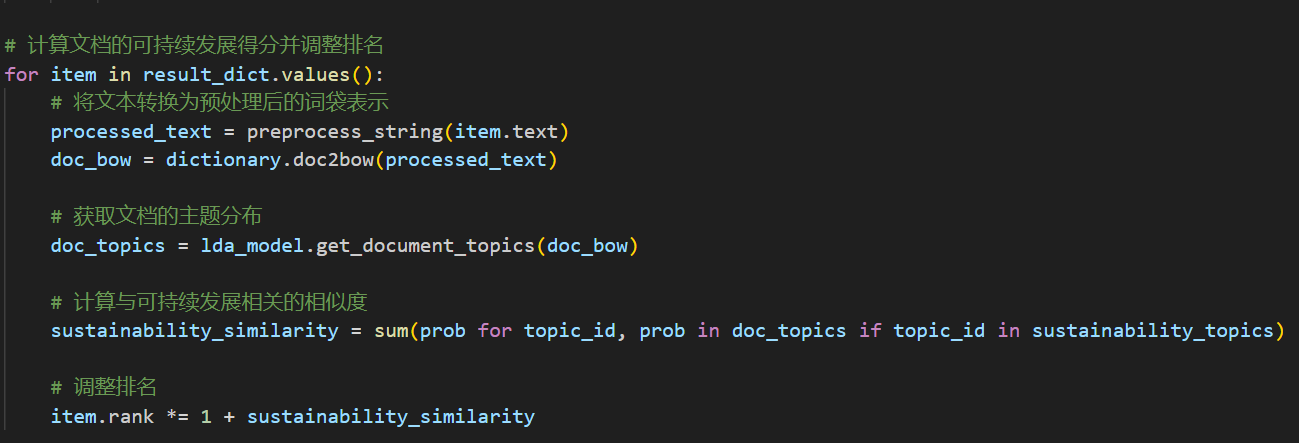
# 体现环境和社会可持续发展影响的考虑

由于这是个信息检索系统

根据文档的环境和社会可持续发展相关性，调整搜索结果的排名。可以为符合可持续发展原则的文档分配更高的权重，以便它们在搜索结果中更加突出。 使用这个函数来计算权重



同时我还会使用lda来抽取主题建模使用Python中的gensim库实现LDA主题建模，并识别与可持续发展相关的主题



# 界面演示

## 搜索出多个项目，通过按键进入下一个

可以进行评分得出召回率

# 

# 可行的优化与展望

实际上，我们当前的实现是非常简单，甚至可以说得上事简陋的。这主要是因为这学期的作业量实在太多，没法将时间花在打磨实现上。下面我将简要列出可能的优化方式：

2. 允许动态更新数据集，正常的搜索引擎实际上是可以加入新的文档的，我们当前的每次启动时生成索引的方式显然不合理，应该加入动态更新的机制，例如 Lucene 的实现效果。