

## Text Processing

Notebook: 我的第一个笔记本

Created: 1/27/2020 6:09 AM

Updated: 1/28/2020 9:12 AM

Author: Alvin

---

## 第二部分：IR

# 1.Information Retrieval 的定义

**The task of finding terms that describe documents well**

两种 Document indexing方法：

- **1.manual approach indexing:**

indexing by humans 人工索引

labour and training intensive需要大量训练资源

优点：1.高准确率，

2. 对封闭的数据很友好（比如图书馆里的书）

问题：1.研究员需要懂这些词汇来达到高准确率。

2.标签需要被训练来达到连续性。

3.数据集会经常发生变化

- **2.automatic approach :**

Term manipulation(特定词汇算作同一词)

Term Weighting (部分词汇比其他词汇权重更高)

Index terms must only derive from text（只对文中出现的词汇进行索引）

### **automatic indexing的特点：**

**1.No predefined set of index terms, 使用Natural Language来作为索引。**

**2.文本里的词汇给文本内容提供信息。**

### 3.使用Inverted files 来进行indexing

#### **Inverted Files**的具体方法:

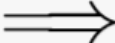
(略过简单版本, 直接看2个复杂的版本)

第一个版本:

1.记录每个词汇出现的DOC的ID, 和每个出现ID的出现次数

2.能够帮助发现文件与query的**相关性 Relevance**

<i>Doc</i>	<i>Text</i>
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old



<i>Num</i>	<i>Token</i>	<i>Docs</i>
1	cold	1:1, 4:1
2	days	3:1, 6:1
3	hot	1:1, 4:1
4	in	2:1, 5:1
5	it	4:2, 5:1
6	like	4:2, 5:1
7	nine	3:1, 6:1
8	old	3:1, 6:1
9	pease	1:2, 2:1
10	porridge	1:2, 2:1
11	pot	2:1, 5:1
12	some	4:2, 5:1
13	the	2:1, 5:1

第二个版本:

1.记录每个出现的**DOC**的**ID**, 和每个词汇在**DOC**里面出现的位置(从1开始, 标点不算)

2.用于寻找文件里的段落。

<i>Doc</i>	<i>Text</i>		<i>Num</i>	<i>Token</i>	<i>Docs</i>
1	Pease porridge hot, pease porridge cold		1	cold	1:(6), 4:(8)
2	Pease porridge in the pot		2	days	3:(2), 6:(2)
3	Nine days old		3	hot	1:(3), 4:(4)
4	Some like it hot, some like it cold		4	in	2:(3), 5:(4)
5	Some like it in the pot		5	it	4:(3, 7), 5:(3)
6	Nine days old		6	like	4:(2, 6), 5:(2)
			7	nine	3:(1), 6:(1)
			8	old	3:(3), 6:(3)
			9	pease	1:(1, 4), 2:(1)
			10	porridge	1:(2, 5), 2:(2)
			11	pot	2:(5), 5:(6)
			12	some	4:(1, 5), 5:(1)
			13	the	2:(4), 5:(5)

## 2.Retrieval Models:

### 1.Bag-of-Words Approach

- 记录哪些词汇出现过
- 记录词汇在每个doc里的出现次数
- 忽略词汇间的关联性

## 2.Retrieval Models:

### 1.Boolean Search:

- 文件是否相关
- 词汇的出现和数量足够用来**match**
- Boolean operators are set operations (AND, OR)  
使用布尔操作符

Boolean query 提供了一个逻辑计算结果，来决定是否返回document，基于以下几点：

- basic terms of query 是否出现在文本中
- 逻辑运算符的意义

Boolean总结：

- 文件是否匹配：

1.取决于研究人员的需要建立高精度的query，对于研究人员很友好，常用于bibliographic search engines.（通俗讲就是图书馆查书）

- 对于大部分用户不好：

- 1.大部分用户不懂怎么写boolean queries
- 2.效率不高，用户不会愿意去看大量的没排序的列表。比如web search里的数据太多。

## **2.Ranked Algorithm:**

- 关注文本中词汇的出现频率
- 不需要所有的搜索词汇都出现在文本中

The vector space model 属于 Ranked Algorithm

### **The vector space model特点：**

- Documents 是在high-dimensional高维度的数据点。
- 每一个索引里的词汇是一个维度dimension -> 稀疏向量
- 值value是文本里词汇的频率

### **具体方法：**

- 选择document-query similarity最高的文本
- document-query similarity 是用于计算相关性 relevance(ranking)的模型
- 有了ranking，返回多少文本不重要->用户只要从top rank往下看，直到匹配。

如何计算向量之间的相似度 **similarity**:

方法一: 使用欧式距离 **euclidean distance**.

- Each document and the query are represented as a vector of  $n$  values:

$$\vec{d}^i = (d_1^i, d_2^i, \dots, d_n^i), \quad \vec{q} = (q_1, q_2, \dots, q_n)$$

- Many metrics of similarity between 2 vectors, e.g.: **Euclidean**

$$\sqrt{\sum_{k=1}^n (q_k - d_k)^2}$$

- E.g.: Distance between:

$$Doc_1 \text{ and } Q = \sqrt{(9-0)^2 + (0-1)^2 + (1-0)^2 + (0-1)^2} = \sqrt{84} = 9.15$$

$$Doc_2 \text{ and } Q = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (10-1)^2} = \sqrt{81} = 9$$

$$Doc_3 \text{ and } Q = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (2-1)^2} = \sqrt{1} = 1$$

**Doc 3 is the closest (shortest distance)**

这个并不是一个好方法, 哪怕只有一个词汇, 不同长度的向量的距离太大, 而且**词汇频率freq of terms的影响太大**

方法二: 使用**cosine**来测量向量X和Y的角度:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

可以被理解为归一化关联因子 **Normalise correlation coefficient**

归一化:

The vector  $\vec{x}$  is normalised by dividing its components by its length:

$$|\vec{x}| = \sqrt{\sum_{i=1}^n x_i^2}$$

$$\text{sim}(\vec{q}, \vec{d}) = \cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$$

本质上是计算term i 的出现次数在query和document中的关联程度，

### 3.Term manipulation

**Pre-process** 预处理的三种方法：

**Tokenisation**: 去掉标点符号

**Capitalisation**: 把所有的词汇都变成小写，lower-case

**Lemmatisation**: 把单词转换成基本形式，比如had变成have，cats变成cat

**Stemming**: 去掉单词的尾缀affix，比如connecting变成connect

**Normalisation**: 取消掉一些断字，拼写，空格等等，比如U.S.A和USA都为USA

**Stop Words**:

使用stoplist来消除一些非内容类的词

a	always	both
about	am	being
above	among	co
across	amongst	could

这样做的好处是减小了inverted index的size

**Single vs. Multi-word Terms**:

为了能够识别句段，要允许搜索词组，使用**multi-word indexing**，同时也需要**Positional Indexes**（参见inverted files）

## Term Weighting:

binary weights - 0/1: 只判断词汇是否出现在文本中，但是query关键词出现更多的文本也许关联性更强

Frequency of term in document: 参见上面Ranked Algorithm.

Frequency in document vs in collection: 如果词汇在相关文本中出现频率高，但是在整个数据集中频率不高，weight terms highly，加大词汇的权值。

## Key concepts:

document collection	$D$	collection (set) of documents
size of collection	$ D $	total number of documents in collection
term freq	$tf_{w,d}$	number of times $w$ occurs in document $d$
collection freq	$cf_w$	number of times $w$ occurs in collection
document freq	$df_w$	number of documents containing $w$

## The informativeness of terms:

- 更少出现的词汇对于找到关联文本更有用，也就是说这些词汇更有意义，informative. 反之亦然

Word	$cf_w$	$df_w$
insurance	10440	3997
try	10422	8760

◇ term *insurance* semantically focussed, term *try* very general

在上面这个例子中，document frequency反应出了区别，但是collection frequency区别不大，无法区分。

## Document Frequency计算:

- Compute metric such as:  $\frac{|D|}{df_w}$ 
  - ◇ Value reduces as  $df_w$  gets larger, tending to 1 as  $df_w$  approaches  $|D|$ 
    - e.g.  $\frac{10000}{3997} = 2.5$  (insurance)       $\frac{10000}{8760} = 1.14$  (try)
  - ◇ Value very large for small  $df_w$  — **over**-weights such cases
    - e.g.  $\frac{10000}{350} = 28.6$  (mischief)

这个公式会让 $df_w$ 值小的词汇的结果非常大，所以引入了一个概念叫做Inverse document Frequency

**Inverse document frequency**计算:

$$idf_{w,D} = \log \frac{|D|}{df_w}$$

$\log \frac{10000}{3997} = 0.398$  (insurance)     $\log \frac{10000}{8760} = 0.057$  (try)     $\log \frac{10000}{350} = 1.456$  (mischief)

本质就是在Document Frequency前面加了一个log，这样起到了归一化的作用，让数值看起来没那么夸张。

**TF.IDF**计算:

Term	<i>tf</i>	<i>df</i>	$ D $	<i>idf</i>	<i>tf.idf</i>
the	312	28,799	30,000	0.018	5.54
in	179	26,452	30,000	0.055	9.78
general	136	179	30,000	2.224	302.50
fact	131	231	30,000	2.114	276.87
explosives	63	98	30,000	2.486	156.61
nations	45	142	30,000	2.325	104.62
haven	37	227	30,000	2.121	78.48

For term **the**:

$$idf(the) = \log_{10}\left(\frac{30,000}{28,799}\right) = 0.018$$

$$tf.idf(the) = 312 \cdot 0.018 = 5.54$$

接下来计算**cosine similarity** 分数:



Example: Vector Space Model, tf.idf term weighting, cosine similarity

- tf.idf values for words in two documents  $D_1$  and  $D_2$ , and in a query  $Q$  “hunter gatherer Scandinavia”:

	Q	$D_1$	$D_2$	
hunter	19.2	56.4	112.2	
gatherer	34.5	122.4	0	
Scandinavia	13.9	0	30.9	
30,000	0	457.2	0	
years	0	12.4	0	
BC	0	200.2	0	
prehistoric	0	45.3	0	
deer	0	0	23.6	
rifle	0	0	452.2	
Mesolithic	0	344.2	0	
$\sqrt{\sum_{i=1}^n x_i^2}$	41.9	622.9	467.5	(i.e. length of vector)

- $$\text{sim}(\vec{q}, \vec{d}) = \cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$$

比较两个**DOC**对于**Q**的相关性:

- $$\text{sim}(\vec{q}, \vec{d}) = \cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$$

$$\cos(Q, D_1) = \frac{(19.2 * 56.4) + (34.5 * 122.4) + \dots + (0 * 0) + (0 * 344.2)}{41.9 * 622.9}$$

$$= \frac{5305.68}{26071.72}$$

$$= 0.20$$

$$\cos(Q, D_2) = \frac{(19.2 * 112.2) + (34.5 * 0) + \dots + (0.0 * 452.2) + (0.0 * 0.0)}{41.9 * 467.5}$$

$$= \frac{2583.8}{19570.0}$$

$$= 0.13$$

这里的结论是D1与Q的关联性更强（相对于D2与Q）。

## 4.Web Search Ranking:

特点:

- Web docs contain info beyond their mere “textual content”
- HTML contains clues that some terms are more important
- Link text — commonly provide description of target doc
- Link structure of web more broadly

这一部分个人觉得主要看PageRank这个算法，16-17考卷上有考题。

在PageRank中，每一个网页都有一个PageRank分数，可以代表这个网页的authority 或者 quality

PageRank的主要特点解释：

- 如果A网页有link可以跳转到B网页，则B confers authority
- 究竟有多少authority可以被conferred取决于：

1. A网页的 authority (PageRank 分数)，和跳转到其他网页链接的数量

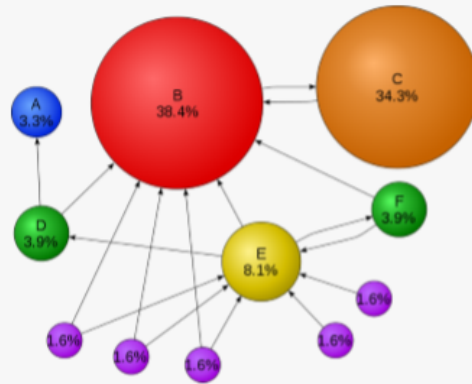
那就是说，A的authority被跳转其他网页的链接分享（下面图更好说明）。

2.这个计算分数是会递归计算的，意思就是一个网页的分数取决于其他网页的分数。

PageRank有另外一个解释：

随机游走打开那个网页的概率（有一个叫随机游走模型的东西）。这个概括来说就是，一个人从一个随机的网页开始，随机打开下一个链接，以此反复。

- Graphical intuition:



- During retrieval, rank score of doc  $d$  is a *weighted combination* of:
  - ◇ its PageRank score: a measure of its authority
  - ◇ its IR-Score: how well  $d$  matches the query  $q$ , based on
    - Vector Space model, TF.IDF, *up-weighting* of important terms, etc

这个图就是说明B是一个PageRank score非常高的网站，因为它和C互相链接（双向），所以C分享了B的PageRank score。

但是这里注意，除了C以外的网站，其他网站都是单向跳转到B，所以B的分数才会这么高，但是这些单向跳转的网页本身不会得到很高的PageRank。

## 5.Evaluate:

给模型评分这个直接看IR4的PDF吧，三言两语说不清，个人觉得不是重点。这个PDF整篇都是在讲如何evaluate。F分数，precision，recall这些必会，没啥好说的。