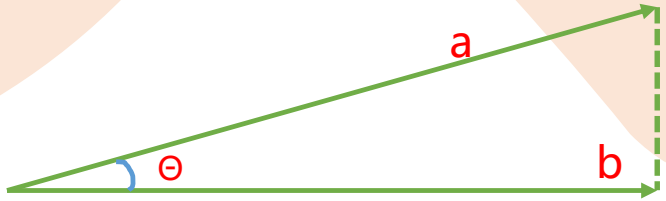
NLP文本相似度

OutLine

- 余弦相似度、向量空间模型
- TFIDF
- 【实践】TFIDF
- LCS
- 【实践】LCS

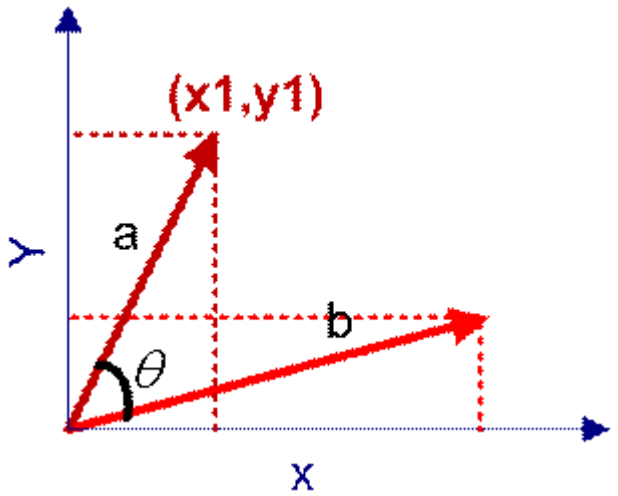
相 似 度

- 相似度度量：计算个体间相似程度
- 相似度值越小，距离越大，相似度值越大，距离越小
- 最常用——**余弦相似度**
  - 一个向量空间中两个向量夹角的余弦值作为衡量两个个体之间差异的大小
  - 余弦值接近1，夹角趋于0，表明两个向量越相似



$$\cos(\theta) = \frac{b}{a}$$

相 似 度



如果向量a和b不是二维而是n维

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

$$= \frac{a \bullet b}{||a|| \times ||b||}$$

一个例子

- 句子1：这只皮靴号码大了，那只号码合适
- 句子2：这只皮靴号码不小，那只更合适

分词

- 句子1：这只/皮靴/号码/大了，那只/号码/合适。
- 句子2：这只/皮靴/号码/不/小，那只/更/合适。

列出所有词

这只，皮靴，号码，大了，那只，合适，不，小，更

计算词频

- 句子1：这只1，皮靴1，号码2，大了1，那只1，合适1，不0，小0，更0
- 句子2：这只1，皮靴1，号码1，大了0，那只1，合适1，不1，小1，更1

词频向量化

- 句子A：(1，1，2，1，1，1，0，0，0)
- 句子B：(1，1，1，0，1，1，1，1，1)

套公式计算

$$\cos(\theta) = \frac{1\times1+1\times1+2\times1+1\times0+1\times1+1\times1+0\times1+0\times1+0\times1}{\sqrt{1^2+1^2+2^2+1^2+1^2+1^2+0^2+0^2+0^2}\times\sqrt{1^2+1^2+1^2+0^2+1^2+1^2+1^2+1^2+1^2}}$$

$$= \frac{6}{\sqrt{7}\times\sqrt{8}}$$

$$= 0.81$$

处 理 流 程

- 得到了文本相似度计算的处理流程是:
  - 找出两篇文章的关键词;
  - 每篇文章各取出若干个关键词，合并成一个集合，计算每篇文章对于这个集合中的词的词频
  - 生成两篇文章各自的词频向量;
  - 计算两个向量的余弦相似度，值越大就表示越相似。

OutLine

- 余弦相似度、向量空间模型
- TFIDF
- 【实践】TFIDF
- LCS
- 【实践】LCS

词 频 — — T F

- 假设：如果一个词很重要，应该会在文章中多次出现
- 词频——TF（Term Frequency）：一个词在文章中出现的次数
- 也不是绝对的！出现次数最多的是“的” “是” “在”，这类最常用的词，叫做停用词（stop words）
- 停用词对结果毫无帮助，必须过滤掉的词
- 过滤掉停用词后就一定能接近问题么？
- 进一步调整假设：如果某个词比较少见，但是它在这篇文章中多次出现，那么它很可能反映了这篇文章的特性，正是我们所需要的关键词

反文档频率——IDF

- 在词频的基础上，赋予每一个词的权重，进一步体现该词的重要性，
- 最常见的词（“的”、“是”、“在”）给予最小的权重
- 较常见的词（“国内”、“中国”、“报道”）给予较小的权重
- 较少见的词（“养殖”、“维基”、“涨停”）

- 将TF和IDF进行相乘，就得到了一个词的TF-IDF值，某个词对文章重要性越高，该值越大，于是排在前面的几个词，就是这篇文章的关键词。

计 算 步 骤

词频（TF）= 某个词在文章中出现的次数

词频标准化 → 词频（TF）= $\frac{\text{某个词在文章中出现的次数}}{\text{文章的总词数}}$

词频标准化 → 词频（TF）= $\frac{\text{某个词在文章中出现的次数}}{\text{该文出现次数最多的词的出现次数}}$

反文档频率（IDF）= log（$\frac{\text{语料库的文档总数}}{\text{包含该词的文档数+1}}$）

TF-IDF = 词频（TF）*反文档频率（IDF）

TF-IDF与一个词在文档中的出现次数成正比，与包含该词的文档数成反比。

相似文章

- 使用TF-IDF算法，找出两篇文章的关键词；
- 每篇文章各取出若干个关键词（比如20个），合并成一个集合，计算每篇文章对于这个集合中的词的词频（为了避免文章长度的差异，可以使用相对词频）；
- 生成两篇文章各自的词频向量；
- 计算两个向量的余弦相似度，值越大就表示越相似。

自 动 摘 要

- 文章的信息都包含在句子中，有些句子包含的信息多，有些句子包含的信息少。"自动摘要"就是要找出那些包含信息最多的句子。
- 句子的信息量用"关键词"来衡量。如果包含的关键词越多，就说明这个句子越重要。
- 只要关键词之间的距离小于“门槛值”，它们就被认为处于同一个簇之中，如果两个关键词之间有5个以上的其他词，就可以把这两个关键词分在两个簇。
- 下一步，对于每个簇，都计算它的重要性分值。

$$\text{簇的重要性} = \frac{(\text{包含的关键词数量})^2}{\text{簇的长度}}$$

- **例如：**其中的簇一共有7个词，其中4个是关键词。因此，它的重要性分值等于（4 x 4）/ 7 = 2.3
- 简化：不再区分"簇"，只考虑句子包含的关键词。下面就是一个例子（采用伪码表示），只考虑关键词首先出现的句子。

自动摘要

```
Summarizer(originalText, maxSummarySize):

    // 计算原始文本的词频，生成一个数组，比如[(10,'the'), (3,'language'), (8,'code')...]
    wordFrequences = getWordCounts(originalText)

    // 过滤掉停用词，数组变成[(3, 'language'), (8, 'code')...]
    contentWordFrequences = filtStopWords(wordFrequences)

    // 按照词频进行排序，数组变成['code', 'language'...]
    contentWordsSortbyFreq = sortByFreqThenDropFreq(contentWordFrequences)

    // 将文章分成句子
    sentences = getSentences(originalText)

    // 选择关键词首先出现的句子
    setSummarySentences = {}
    foreach word in contentWordsSortbyFreq:
        firstMatchingSentence = search(sentences, word)
        setSummarySentences.add(firstMatchingSentence)
        if setSummarySentences.size() = maxSummarySize:
            break

    // 将选中的句子按照出现顺序，组成摘要
    summary = ""
    foreach sentence in sentences:
        if sentence in setSummarySentences:
            summary = summary + " " + sentence

    return summary
```

总 结

- 优点：简单快速，结果比较符合实际情况
- 缺点：单纯以“词频”做衡量标准，不够全面，有时重要的词可能出现的次数并不多
  - 这种算法无法体现词的位置信息，出现位置靠前的词与出现位置靠后的词，都被视为重要性相同，这是不正确的。（一种解决方法是，对全文的第一段和每一段的第一句话，给予较大的权重。）

OutLine

- 余弦相似度、向量空间模型
- TFIDF
- 【实践】TFIDF
- LCS
- 【实践】LCS

OutLine

- 余弦相似度、向量空间模型
- TFIDF
- 【实践】TFIDF
- LCS
- 【实践】LCS

LCS定义

- 最长公共子序列（Longest Common Subsequence）
- 一个序列S任意删除若干个字符得到的新序列T，则T叫做S的子序列
- 两个序列X和Y的公共子序列中，长度最长的那个，定义为X和Y的最长公共子序列
  - 字符串12455与245576的最长公共子序列为2455
  - 字符串acdfg与adfc的最长公共子序列为adf
- 注意区别最长公共子串（Longest Common Substring）
  - 最长公共子串要求连接

LCS作用

- 求两个序列中最长的公共子序列算法
  - 生物学家常利用该算法进行基因序列比对，以推测序列的结构、功能和演化过程。
- 描述两段文字之间的“相似度”
  - 辨别抄袭，对一段文字进行修改之后，计算改动前后文字的最长公共子序列，将除此子序列外的部分提取出来，该方法判断修改的部分

求解——暴力穷举法

- 假定字符串X，Y的长度分别为m，n；
- X的一个子序列即下标序列{1,2，……，m}严格递增子序列，因此，X共有2^m个不同子序列；同理，Y有2^n个不同子序列;
- 穷举搜索法时间复杂度O($2^m * 2^n$);
- 对X的每一个子序列，检查它是否也是Y的子序列，从而确定它是否为X和Y的公共子序列，并且在检查过程中选出最长的公共子序列;
- 复杂度高，不可用!

求解——动态规划法

- 字符串X，长度为m，从1开始数；
- 字符串Y，长度为n，从1开始数；
- $X_i=<x_1, \ldots\ldots, x_i>$即X序列的前i个字符（1<=i<=m）（$X_i$计作“字符串X的i前缀”）
- $Y_i=<y_1, \ldots\ldots, y_i>$即Y序列的前j个字符（1<=j<=n）（$Y_j$计作“字符串Y的j前缀”）
- LCS(X,Y)为字符串X和Y的最长公共子序列，即为$Z=<z_1, \ldots\ldots, z_k>$

- 如果$x_m = y_n$（最后一个字符相同），则：X_m与Y_n的最长公共子序列Z_k的最后一个字符必定为$x_m(=y_n)$
- $z_k = x_m = y_n$
- $LCS(X_m, Y_n)=LCS(X_{m-1}, Y_{n-1})+x_m$

求解——动态规划法

如果$x_m = y_m$

	1	2	3	4	5	6	7
X	B	D	**C**	A	**B**	A	
Y	A	B	**C**	**B**	D	A	B

- 对于上面的字符串X和Y：
- $x_3 = y_3$= ‘C’ 则有：LCS(BD**C**, AB**C**)=LCS(BD, AB)+ ‘**C**’
- $x_5 = y_4$= ‘B’ 则有：LCS(BDCA**B**, ABC**B**)=LCS(BDCA, ABC)+ ‘**B**’

求解——动态规划法

- 如果$x_m \neq y_n$，则LCS(X_m, Y_n)=LCS(X_{m-1}, Y_n)，或者LCS(X_m, Y_n)=LCS(X_m, Y_{n-1})
- 即LCS(X_m, Y_n)=max{LCS(X_{m-1}, Y_n), LCS(X_m, Y_{n-1})}

	1	2	3	4	5	6	7
X	B	D	C	A	B	A	
Y	A	B	C	B	D	A	B

- 对于上面的字符串X和Y：
- $x_2 \neq y_2$则有：LCS(B**D**, A**B**)=max{LCS(B**D**, A), LCS(B, A**B**)}
- $x_4 \neq y_5$则有： LCS(BDC**A**, ABCB**D**)=max{LCS(BDC**A**, ABCB), LCS(BDC, ABCB**D**)}

LCS总结分析

$$LCS(X_m, Y_n) = \begin{cases} LCS(X_{m-1}, Y_{n-1}) + x_m & 当 x_m = y_n \\ \max\{LCS(X_{m-1}, Y_n), LCS(X_m, Y_{n-1})\} & 当 x_m \neq y_n \end{cases}$$

- 属于**动态规划**问题!

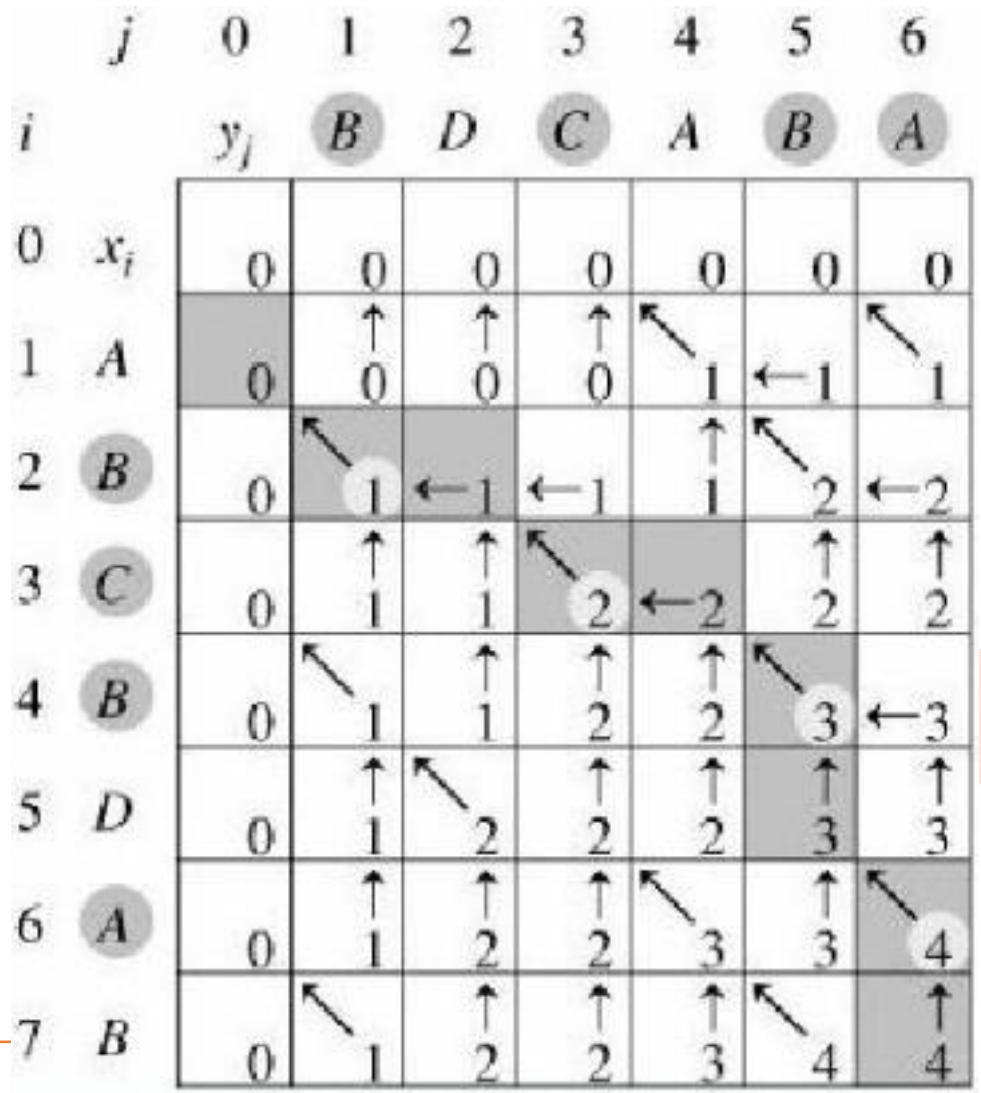
数据结构——二维数组

- 使用二维数组C[m,n]
- C[i,j]记录序列X_i和Y_j的最长公共子序列的**长度**
  - 当i=0或j=0时，空序列是X_i和Y_j的最长公共子序列，故C[i,j]=0

$$c(i,j)=\begin{cases}0 & \text{当}i=0\text{或者}j=0\\ c(i-1,j-1)+1 & \text{当}i>0,j>0,\text{且}x_i=y_j\\ \max\{c(i-1,j),c(i,j-1)\} & \text{当}i>0,j>0,\text{且}x_i\neq y_j\end{cases}$$

例 子

- X=<A, B, C, B, D, A, B>
- Y=<B, D, C, A, B, A>



OutLine

- 余弦相似度、向量空间模型
- TFIDF
- 【实践】TFIDF
- LCS
- 【实践】LCS

Q & A

@八斗学院