数据表示技术

朱小飞

如何对文档进行相似性计算?

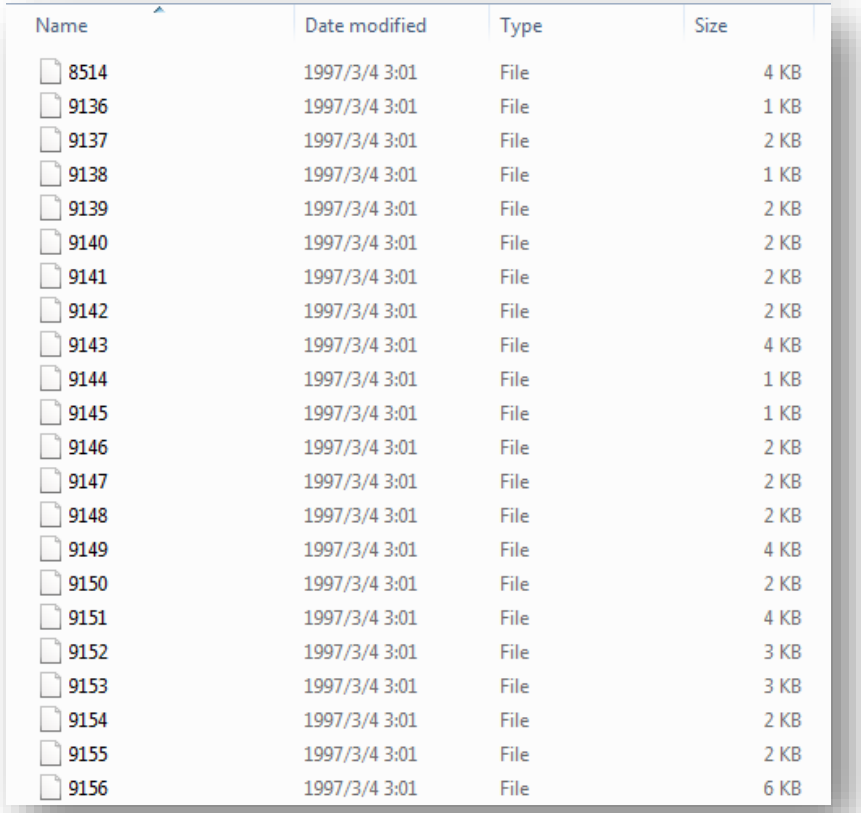
- 文档表示

Name	Date modified	Type
alt.atheism	2018/1/20 10:29	File folder
comp.graphics	2018/1/20 10:29	File folder
comp.os.ms-windows.misc	2018/1/20 10:29	File folder
comp.sys.ibm.pc.hardware	2018/1/20 10:29	File folder
comp.sys.mac.hardware	2018/1/20 10:29	File folder
comp.windows.x	2018/1/20 10:29	File folder
misc.forsale	2018/1/20 10:29	File folder
rec.autos	2018/1/20 10:29	File folder
rec.motorcycles	2018/1/20 10:29	File folder
rec.sport.baseball	2018/1/20 10:29	File folder
rec.sport.hockey	2018/1/20 10:29	File folder
sci.crypt	2018/1/20 10:29	File folder
sci.electronics	2018/1/20 10:29	File folder
sci.med	2018/1/20 10:29	File folder
sci.space	2018/1/20 10:29	File folder
soc.religion.christian	2018/1/20 10:29	File folder
talk.politics.guns	2018/1/20 10:29	File folder
talk.politics.mideast	2018/1/20 10:29	File folder
talk.politics.misc	2018/1/20 10:29	File folder
talk.religion.misc	2018/1/20 10:29	File folder

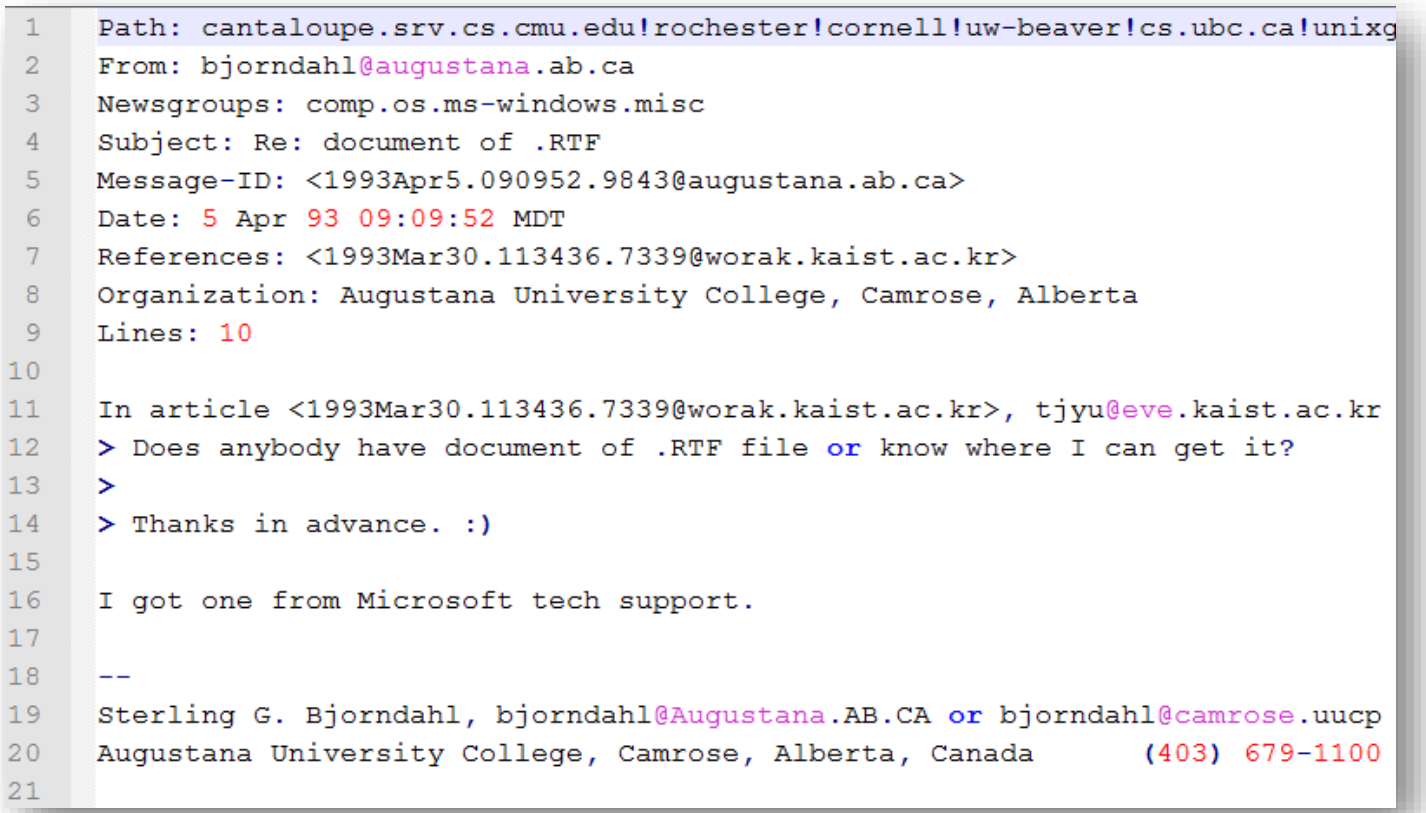
49960	51060	51119
51121	51122	51123
51125	51126	51127
51129	51130	51131
51133	51134	51135
51137	51138	51139
51141	51142	51143
51145	51146	51147
51149	51150	51151
51153	51154	51155
51157	51158	51159
51161	51162	51163
51165	51166	51167
51169	51170	51171
51173	51174	51175
51177	51178	51179
51181	51182	51183
51185	51186	51187
51189	51190	51191
51193	51194	51195
51197	51198	51199
51201	51202	51203

数据表示

- 如何对文档进行相似性计算
  - 文档表示



Name	Date modified	Type	Size
8514	1997/3/4 3:01	File	4 KB
9136	1997/3/4 3:01	File	1 KB
9137	1997/3/4 3:01	File	2 KB
9138	1997/3/4 3:01	File	1 KB
9139	1997/3/4 3:01	File	2 KB
9140	1997/3/4 3:01	File	2 KB
9141	1997/3/4 3:01	File	2 KB
9142	1997/3/4 3:01	File	2 KB
9143	1997/3/4 3:01	File	4 KB
9144	1997/3/4 3:01	File	1 KB
9145	1997/3/4 3:01	File	1 KB
9146	1997/3/4 3:01	File	2 KB
9147	1997/3/4 3:01	File	2 KB
9148	1997/3/4 3:01	File	2 KB
9149	1997/3/4 3:01	File	4 KB
9150	1997/3/4 3:01	File	2 KB
9151	1997/3/4 3:01	File	4 KB
9152	1997/3/4 3:01	File	3 KB
9153	1997/3/4 3:01	File	3 KB
9154	1997/3/4 3:01	File	2 KB
9155	1997/3/4 3:01	File	2 KB
9156	1997/3/4 3:01	File	6 KB



```
1  Path: cantaloupe.srv.cs.cmu.edu!rochester!cornell!uw-beaver!cs.ubc.ca!unixg
2  From: bjorndahl@augustana.ab.ca
3  Newsgroups: comp.os.ms-windows.misc
4  Subject: Re: document of .RTF
5  Message-ID: <1993Apr5.090952.9843@augustana.ab.ca>
6  Date: 5 Apr 93 09:09:52 MDT
7  References: <1993Mar30.113436.7339@worak.kaist.ac.kr>
8  Organization: Augustana University College, Camrose, Alberta
9  Lines: 10
10
11 In article <1993Mar30.113436.7339@worak.kaist.ac.kr>, tjyu@eve.kaist.ac.kr
12 > Does anybody have document of .RTF file or know where I can get it?
13 >
14 > Thanks in advance. :)
15
16 I got one from Microsoft tech support.
17
18 --
19 Sterling G. Bjorndahl, bjorndahl@Augustana.AB.CA or bjorndahl@camrose.uucp
20 Augustana University College, Camrose, Alberta, Canada      (403) 679-1100
21
```

文本处理

文本处理

- 词法分析：识别文档中的词
  - 概念词条（Tokens）/词项（Terms）
  - 英文/中文
- 停用词消除(stop words)
  - 查表法
  - 基于文档频率
- 词项归一化
- 词干还原(stemming)：去除单词两端词缀
  - Porter算法：规则
  - 提高召回率，但是会降低准确率
- 词形归并(Lemmatization)： am, is, are → be

词法分析(Lexical Analysis)

- 文本可以表示为
  - 一个字符串
  - 词的集合
  - 语言单元 (例如：名词、短语)
- 简单的表示 (如：单个词项) 效果好
  - 以往的一些研究显示：基于短语的索引不如基于词的索引
  - 短语可能太特殊了

词法分析(Lexical Analysis)

- 将文档的字符串序列变成词序列
  - 英文词法分析：书写时英文词之间通常通过空格或者标点进行区分，因此从英文字符串变成英文词是相对比较容易的。
  - 中文词法分析：书写时通常没有空格，需要分词。

停用词

- **文本的特点：**有些词在文本中出现的频率非常高，而且对文本所携带的信息基本不产生影响
  - 例如英文中的“a，the，of”，中文的“的，了，着”，字符串“http”、“.com”以及各种标点符号等，这样的词称为停用词(stop words)。

Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

停用词

- 消除停用词问题和方法
  - 优点：停用词消除可以减少term的个数
  - 缺点：有时消除的停用词对检索是有意义的。
    - “的士”、“to be or not to be”

- 消除方法：
  - 查表法：停用词表
  - 基于文档频率
  - 将词项按照文档集频率（collection frequency），从高到底排列

词项归一化

- 我们需要将文档和查询中的词条“归一化”成一致的形式
  - 希望USA和U.S.A.之间也能形成匹配

- 归一化的结果：
  - 在IR系统的词项词典中，形成多个近似词项的一个等价类

- 隐式的建立等价类
  - 例如将USA和U.S.A.映射为USA

- 其他
  - 文中日期的表示7月30日vs. 英文中7/30

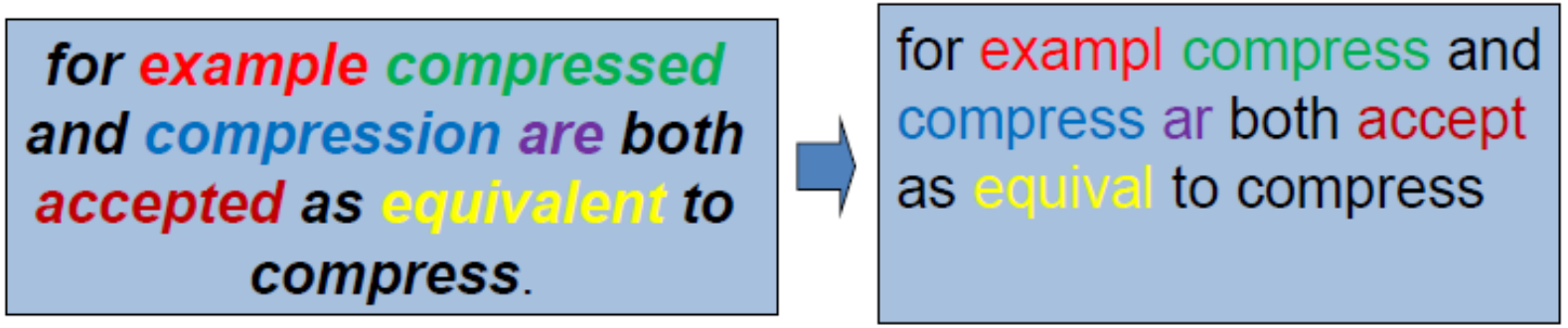
词项归一化

- 大小写转换
  - 将所有字母转换为小写

- 词项归一化的策略：建立同义词扩展表
  - e.g.：手工建立同义词词表
    - car = automobile
    - color = colour

词干还原（Stemming）

- 通常指的就粗略的去除单词两端词缀的启发式过程。
  - e.g., automate(s), automatic, automation, automat.



词干还原（Stemming）

- 中文重叠词还原——可视为“词干还原”

形容词(AB)	ABAB 式	AABB 式	A 里 AB 式
高兴	高兴高兴	高高兴兴	
明白	明白明白	明明白白	
热闹	热闹热闹	热热闹闹	
潇洒	潇洒潇洒	潇潇洒洒	
糊涂		糊糊涂涂	糊里糊涂
流气			流里流气
粘乎	粘乎粘乎	粘粘乎乎	
凉快	凉快凉快	凉凉快快	

形容词（A）	ABB 式	ABCD 式
黑	黑压压	黑不溜秋
白	白花花	白不吡咧
红	红彤彤	
亮	亮晶晶	
恶	恶狠狠	
香	香喷喷	
滑	滑溜溜	

词干还原工具：Porter算法1980

- 英文处理中最常用的词干还原算法。
  - 经过实践证明是高效性的算法。

  - 算法包括5个按照顺序执行的词项约简步骤：
  - 每个步骤都是按照一定顺序执行的
  - 每个步骤中包含了选择不同的规则的约定
  - 比如，从规则组中选择作用时词缀最长的那条规则

词干还原工具：Porter算法1980

- Porter算法中的典型规则
  - sses → ss　　caresses → caress
  - ies → i　　ponies → poini
  - ational → ate　　national → nate

- 词干还原能够提高召回率，但是会降低准确率
  - e.g.:
    - operative ⇒oper
  - 词干还原对于芬兰语，西班牙语，德语，法语都有明显的作用，其中对芬兰语的提高达到30%（以MAP平均准确率来计算）

词形归并（Lemmatization）

- 减少词变化的形式，将其转变为基本形式。
- 例如
  - am, are,is → be
  - car, cars, car's, cars' → car
  - the boy's cars are different colors → the boy car be different color

- 词形归并可以减少词项词典中的词项数量

索引

引言

- 索引：
  - 是一种数据结构，它在关键词与包含关键词的文档之间建立了一种映射关系，从而加快检索的速度。
- 建立索引的目的：
  - 加快检索速度
- 常用的索引技术
  - 倒排文档（或倒排索引）是一种最常用的索引机制
  - 后缀数组
  - 签名文件

倒排索引倒排索引

- 什么是倒排索引
- 倒排索引的建立
- 基于倒排索引的检索
- 倒排索引的维护

实例

文档编号	题目	关键词
1	…	知识管理，管理信息系统，企业信息化
2	…	知识管理，知识链，学习型组织，知识创新
3	…	知识管理，知识创新，知识共享
4	…	知识管理，学习型组织
5	…	技术创新，知识空间，知识创新
6	…	企业信息化，信息结构，竞争情报
7	…	知识管理，知识创新，竞争情报
8	…	管理信息系统，企业文化
9	…	管理信息系统，竞争情报
10	…	知识管理，企业文化，管理创新

建立索引

	关键词	目长	文档集合
1	管理创新	1	10
2	管理信息系统	3	1;8;9
3	技术创新	1	5
4	竞争情报	3	6;7;9
5	企业文化	2	8;10
6	企业信息化	2	1;6
7	信息结构	1	6
8	学习型组织	2	2;4
9	知识创新	4	2;3;5;7
10	知识管理	6	1;2;3;4;7;10
11	知识共享	1	3
12	知识空间	1	5
13	知识链	1	2

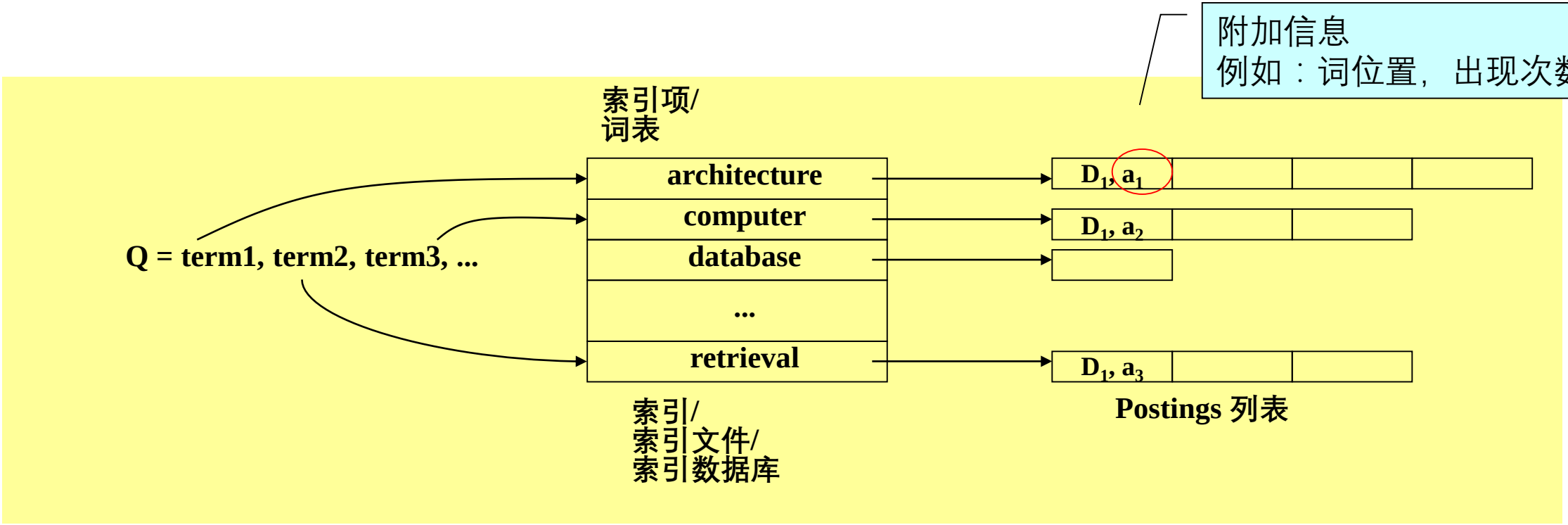
倒排索引的定义

- 倒排索引（或倒排文档）是一种最常用的索引机制
  - 也称倒排文档，是从关键词快速查询到文档的索引结构。文档正常表示为关键词的集合，建立倒排索引是把每个关键词表示为其出现的文档的集合，这个过程称为inversion，即倒排。倒排索引是一种最常用的索引机制

- 倒排索引一般由两部分组成：
  - 词汇表（vocabulary）：是文本或文本集合中所包含的所有不同单词的集合。
  - 记录表（posting list）：对于词汇表中的每一个单词，其在文本中出现的位置或者其出现的文本编号构成一个列表。

一般的倒排索引

- 倒排文档组成
  - 词汇表（vocabulary）
  - 记录表（posting list）



- 索引文件可以用任何文件结构来实现
- 索引文件中的词项是文档集合中的词表

距离约束：需要位置信息为记录表

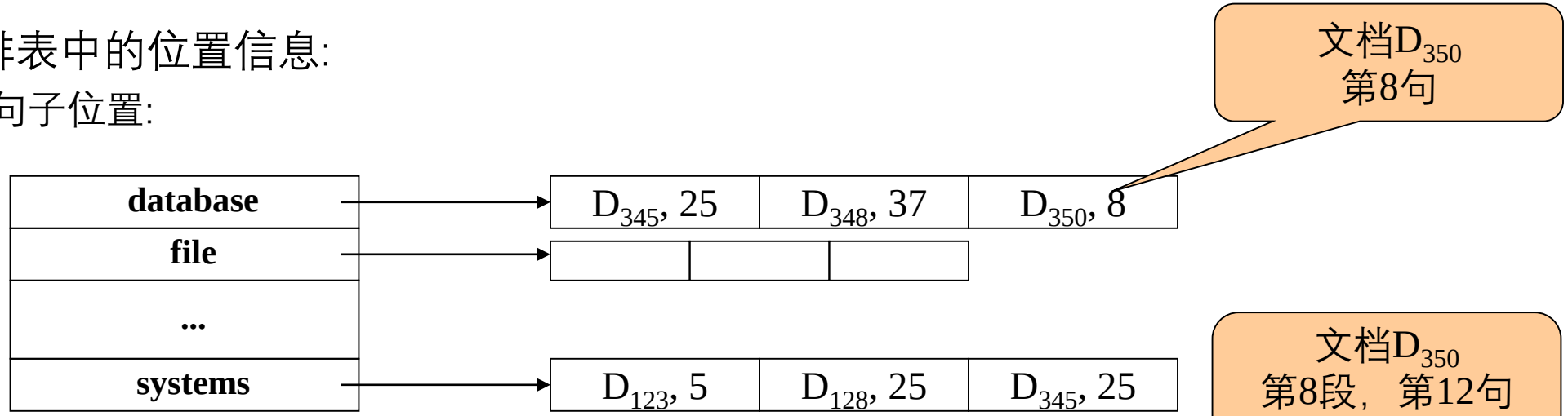
- 常常需要知道邻接条件，例如：
  - “database” 后面紧跟着“systems”
    - 例如：短语搜索 “database systems”
  - “database”和 “systems”之间不能间隔超过3个词
  - “database”和“architecture”在同一个句子里

- 需求扩展：
  - 倒排索引中保存着关键词在**文档中的位置**，文档的**组成单元**(标题, 小标题, 句子分割标记等)
  - 检索算法和位置信息相关联，并需检查文档的组成单元

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
这	是	一本	关于	信息	检索	的	教材	。	介绍	了	检索	的	基本	技术	。	…

以位置信息为记录表

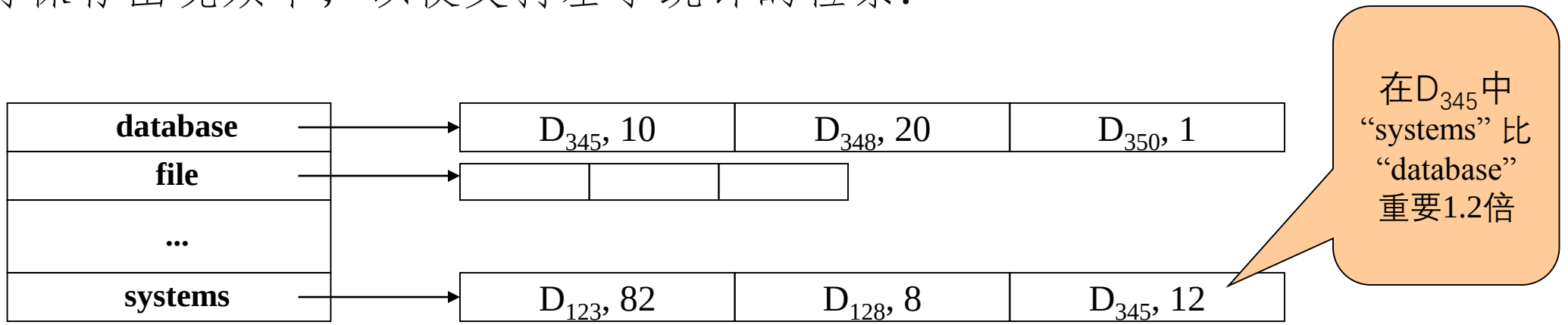
- 保存倒排表中的位置信息:
  - 保存句子位置:



  - 保存段落、句子和词的位置:

以权重信息为记录表

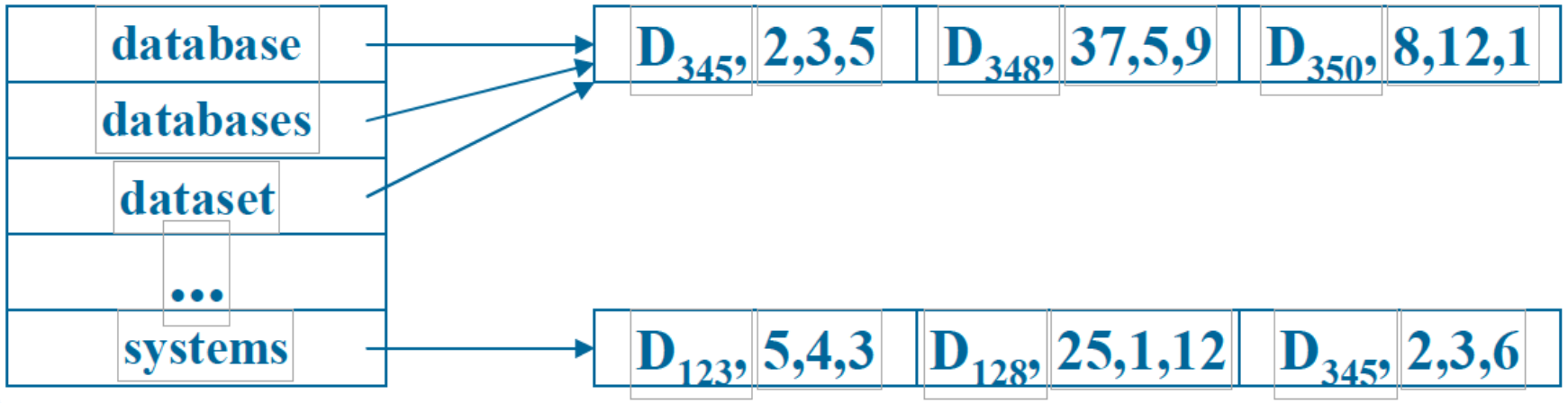
- 可保存出现频率，以便支持基于统计的检索:



- Postings中的第二个单元可以是该term的权重 (例如, 可以被归一化在0和1之间)，或者是该term的出现频率

同义词扩展词汇表

- 同义词对于提高检索的召回率很有意义
- 同义词可以通过指针指向同一个记录表



倒排索引的建立

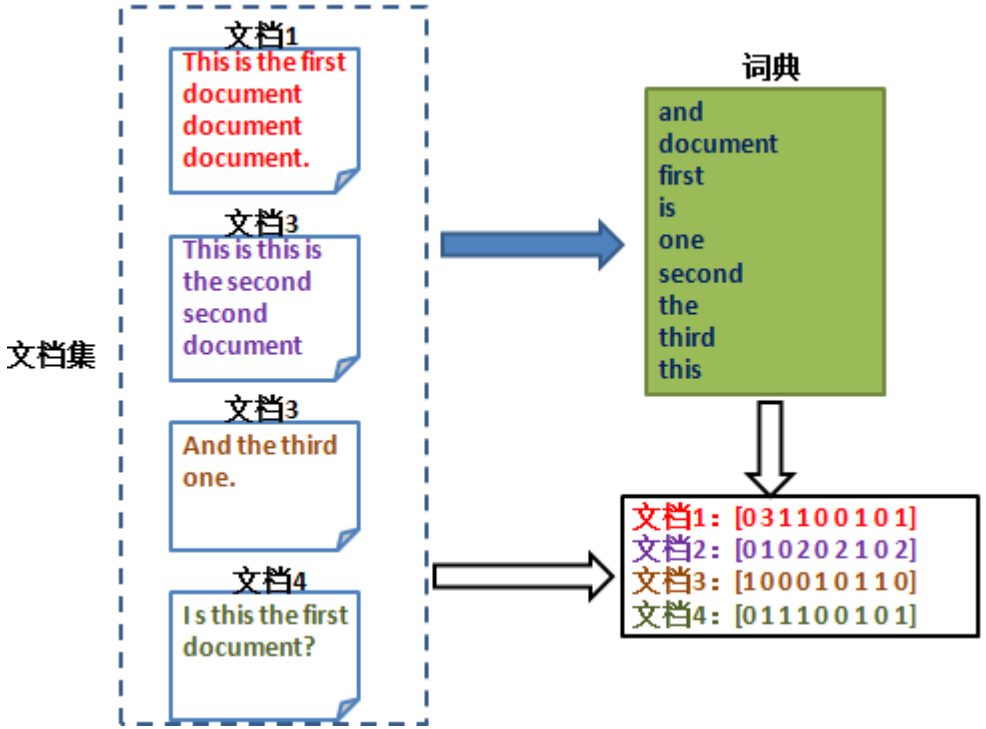
- 步骤
  - 在文档中抽取关键词，并在其后附上其文档编号
  - 对抽出的关键词进行排序，使之便于归并相同关键词
  - 对相同关键词进行归并，把合并后的关键词放入倒排文档的词汇表。统计每一关键词的文档频率，把每一关键词后的记录号顺序放在记录表中

数据表示

一、文档向量空间表示

- 向量空间中的N个文档可以用一个矩阵表示
- 矩阵中的一个元素对应于文档中一个词项的权重。“0”意味着该词项在文档中没有意义，或该词项不在文档中出现。

	T_1	T_2	....	T_t
D_1	d_{11}	d_{12}	...	d_{1t}
D_2	d_{21}	d_{22}	...	d_{2t}
:	:	:		:
:	:	:		:
D_n	d_{n1}	d_{n2}	...	d_{nt}



数据表示

二、文档TFIDF表示

- 根据词项在文档(*tf*)和文档集(*idf*)中的频率(frequency)计算词项的权重

$$\mathrm{tfidf}_{i,k} = \mathrm{tf}_{i,k} * \log\left(\frac{N}{\mathrm{df}_i}\right)$$

词项的重要性

词项的区分能力

tf_{iK}= 词项i在文档k中的频率

df_i = 词项i的文档频率，即包含词项i的文档数量

N: 文档集中文档总数

数据表示

- 三、文档概率模型表示BM25
  - 查询词i 在文档d 中的权重为：

$$W_{id}^{bm25} = \log\left[\frac{N}{df_i}\right] * \frac{tf_{id}}{tf_{id}+k_1((1-b)+ b\cdot\frac{dl}{avgdl})}$$

IDF

TF

数据表示

- 三、文档概率模型表示BM25
  - 查询词i 在文档d 中的权重为：

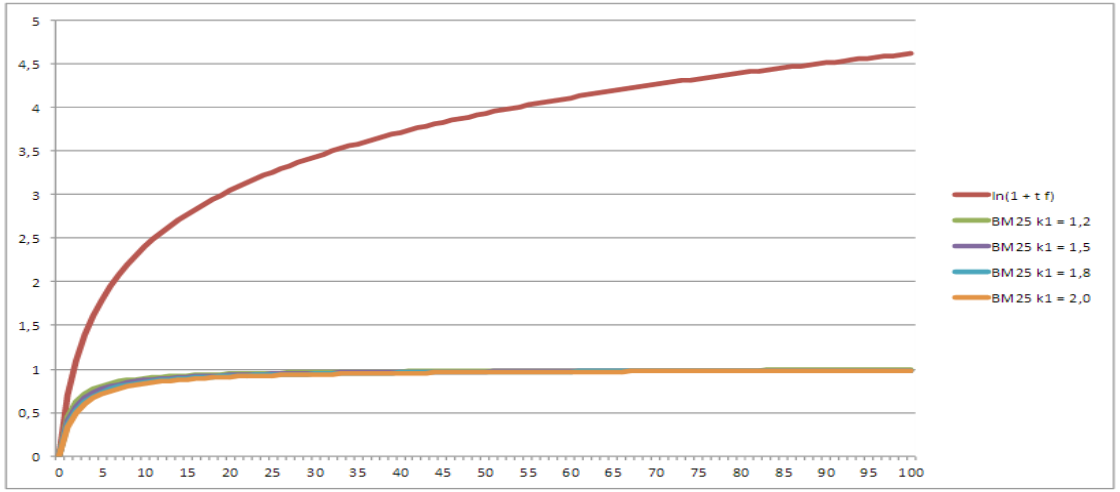
$$W_{id}^{bm25} = \log\left[\frac{N}{df_i}\right] * \frac{tf_{id}}{tf_{id}+k_1((1-b)+ b\cdot\frac{dl}{avgdl})}$$

IDF

TF

➢**高频词**：对于高频词而言，其词频的增加对相关性的影响不大:

$$Saturation_{BM25}(tf_{id}) = \frac{tf_{id}}{tf_{id}+k_1}$$



数据表示

- 三、文档概率模型表示BM25
  - 查询词i 在文档d 中的权重为：

$$W_{id}^{bm25} = \log\left[\frac{N}{df_i}\right] * \frac{tf_{id}}{tf_{id}+k_1((1-b)+ b\cdot\frac{dl}{avgdl})}$$

IDF

TF

➢**高频词**：对于高频词而言，其词频的增加对相关性的影响不大:

$$Saturation_{BM25}(tf_{id}) = \frac{tf_{id}}{tf_{id}+k_1}$$

➢**长文档**：对于长文档而言，其词频通常要高于短文档中的词频，其与相关性无关。

$$B = (1-b) + b\cdot\frac{dl}{avgdl}$$

default value: b = 1.5, k=2.0

数据表示

● 四、语言模型表示Language Model

● 给定词序列 $(w_1, \cdots, w_m)$ 言模型描述该词序列的概率分布：

$$P(w_1, \cdots, w_m) = P(w_1)P(w_2|w_1)\cdots P(w_m|w_{m-1}\cdots w_1)$$

独立性假设

一元语言模型

$$P(w_1, \cdots, w_m) = P(w_1)P(w_2)\cdots P(w_m)$$

文档生成查询的概率

查询的概率

$$P(\text{query}|\text{doc}) = \prod_{\text{term in query}} P(\text{term}|\text{doc})$$

数据表示

● 四、语言模型表示Language Model

● 文档模型：词w 在文档d中出现的概率。

文档d


Deep Learning Tutorial, Release 0.1

5.4.1 Nonlinearity

Two of the most common ones are the *sigmoid* and the *tanh* function. For reasons explained in Section 4.8, nonlinearities that are symmetric around the origin are preferred because they tend to produce zero-mean inputs to the next layer (which is a desirable property). Empirically, we have observed that the *tanh* has better convergence properties.

5.4.2 Weight initialization

At initialization we want the weights to be small enough around the origin so that the activation function operates in its linear regime, where gradients are the largest. Other desirable properties, especially for deep networks, are to conserve variance of the activation as well as variance of back-propagated gradients from layer to layer. This allows information to flow well upward and downward in the network and reduces discrepancies between layers. Under some assumptions, a compromise between these two constraints leads to the following initialization: $uniform[-\sqrt{\frac{6}{fan_{in}+fan_{out}}}, \sqrt{\frac{6}{fan_{in}+fan_{out}}}]$ for tanh and $uniform[-4*\sqrt{\frac{6}{fan_{in}+fan_{out}}}, 4*\sqrt{\frac{6}{fan_{in}+fan_{out}}}]$ for sigmoid. Where fan_{in} is the number of inputs and fan_{out} the number of hidden units. For mathematical considerations please refer to [Xavier10].

5.4.3 Learning rate

There is a great deal of literature on choosing a good learning rate. The simplest solution is to simply have a constant rate. Rule of thumb: try several log-spaced values ($10^{-1}, 10^{-2}, \ldots$) and narrow the (logarithmic) grid search to the region where you obtain the lowest validation error.

Decreasing the learning rate over time is sometimes a good idea. One simple rule for doing that is $\frac{\epsilon_0}{1+dt}$ where ϵ_0 is the initial rate (chosen, perhaps, using the grid search technique explained above), d is a so called "decrease constant" which controls the rate at which the learning rate decreases (typically, a smaller positive number, 10^{-3} and smaller) and t is the epoch/stage.

Section 4.7 details procedures for choosing a learning rate for each parameter (weight) in our network and for choosing them adaptively based on the error of the classifier.

5.4.4 Number of hidden units

This hyper-parameter is very much dataset-dependent. Vaguely speaking, the more complicated the input distribution is, the more capacity the network will require to model it, and so the larger the number of hidden units that will be needed (note that the number of weights in a layer, perhaps a more direct measure of capacity, is $D \times D_h$ (recall: D is the number of inputs and D_h is the number of hidden units).

Unless we employ some regularization scheme (early stopping or L1/L2 penalties), a typical number of hidden units vs. generalization performance graph will be U-shaped.

5.4.5 Regularization parameter

Typical values to try for the L1/L2 regularization parameter λ are $10^{-4}, 10^{-3}, \ldots$. In the framework that we described so far, optimizing this parameter will not lead to significantly better solutions, but is worth

5.4. Tips and Tricks for training MLPs 49

词	文档d词概率分布
a	0.09
world	0.15
likes	0.03
we	0.03
share	0.2
...	...

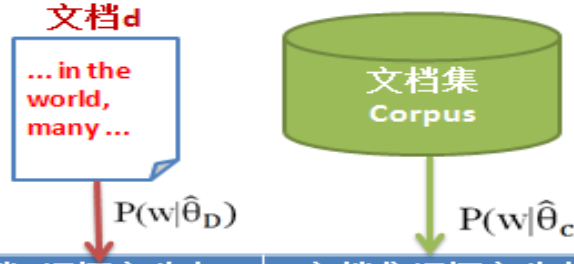
根据最大使然估计，我们有：

$$P(w \mid d) = \frac{count(w,d)}{|d|}$$

其中 count(w,d) 为词w在文档d中出现的次数， |d| 为文档d的长度

平滑Smoothing

- 四、语言模型表示Language Model
  - 对文档中未出现的词赋以一定的概率(零概率问题)



词	文档d词概率分布	文档集词概率分布
a	0.1	0.001
world	0.2	0.003
likes	0.05	0.002
we	0.05	0.02
share	0.3	0.05
...	...	...

Jelinek-Mercer	$p(w\|\hat{\theta}_D)=\lambda p_{ML}(w\|\hat{\theta}_D)+(1-\lambda)p(w\|\hat{\theta}_C)$
Dirichlet	$p(w\|\hat{\theta}_D)=\frac{c(w,D)+\mu p(w\|\hat{\theta}_C)}{\|D\|+\mu}$
Absolute Discounting	$p(w\|\hat{\theta}_D)=\frac{\max(c(w,D)-\delta,0)}{\|D\|}+\frac{\delta\|D\|_u}{\|D\|}p(w\|\hat{\theta}_C)$

文档TFIDF表示

文档TFIDF表示

- **1. 数据清洗**

  1) 大小写转换； 2) 标点符号替换为空格； 3) 词干还原； 4) 分词

```
import nltk, string
import pickle
#nltk.download('punkt')

remove_punctuation_map = dict((ord(char), ' ') for char in string.punctuation) #标点符号
stemmer = nltk.stem.porter.PorterStemmer() #Porter词干还原
def stem_tokens(tokens):
    return [stemmer.stem(item) for item in tokens]
def normalize(text):
    words = stem_tokens(nltk.word_tokenize(text.lower().translate(remove_punctuation_map)))
    return [word for word in words if len(word)>1]
```

文档TFIDF表示

- **1. 数据清洗**

1) 大小写转换； 2) 标点符号替换为空格； 3) 词干还原； 4) 分词

```
25 texts = ["I'd like an apple, do you like?",
26          "An apple a day keeps the doctor away",
27          "Never compare an apple to an orange",
28          "I prefer scikit-learn, to Orange"]
29
30 texts, labels = pickle.load(open('20newsgroups.pkl', 'rb'))
31 """
32 2. 清洗20newsgroups数据
33 """
34 def proceed(texts):
35     docs = []
36     for i,text in enumerate(texts):
37         if i%100==0:
38             print(i)
39         doc = normalize(text)
40         docs.append(doc)
41     return docs
42 docs = proceed(texts)
```

文档TFIDF表示

- 2. 特征提取

使用sklearn模块提供的TFIDF 特征提取工具

```
47 from sklearn.feature_extraction.text import TfidfVectorizer
48
49 vectorizer = TfidfVectorizer(tokenizer=normalize, stop_words='english')
50
51 #vectorizer = TfidfVectorizer(tokenizer=normalize, stop_words='english',
52 #                             max_df=0.9, min_df=3, max_features=5000)
53
54 dvecs = vectorizer.fit_transform(texts)
55 print(dvecs.toarray())
```

1. 大小写转换
2. 标点符号替换为空格
3. 词干还原

1. 去停用词

TFIDF Representation

```
[[ 0.30403549  0.          0.          0.          0.          0.
   0.9526607   0.          0.          0.        ]
 [ 0.34578314  0.5417361   0.          0.5417361   0.5417361   0.          0.
   0.          0.          0.        ]
 [ 0.44809973  0.          0.70203482  0.          0.          0.          0.
   0.55349232  0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.52547275
   0.          0.41428875  0.52547275  0.52547275]]
```

文档相似度

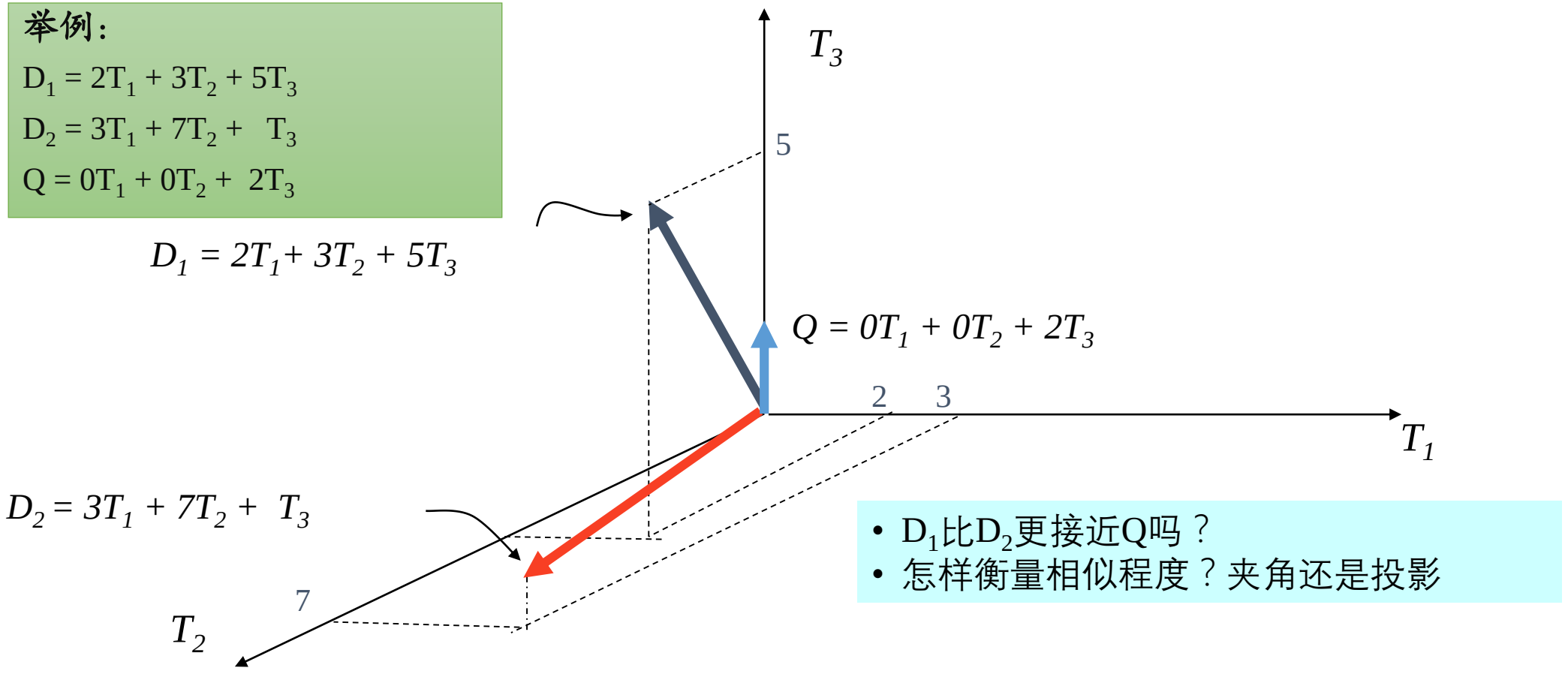
三、文档相似度度量

举例：

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



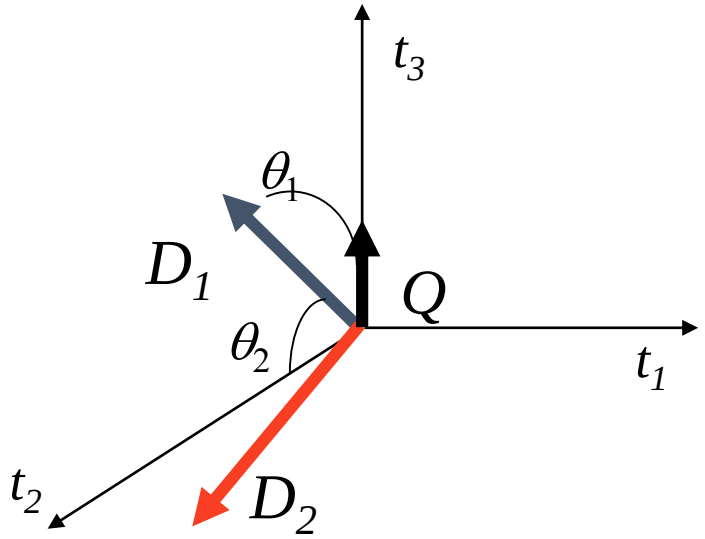
- D_1比D_2更接近Q吗？
- 怎样衡量相似程度？夹角还是投影

文档相似度

三、文档相似度度量

- 余弦（Cosine）相似度计算两个向量的夹角

$$\text{CosSim}(D_i, Q) = \frac{\sum_{k=1}^{t}(d_{ik} \cdot q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^2 \cdot \sum_{k=1}^{t} q_k^2}}$$



$D_1 = 2T_1 + 3T_2 + 5T_3$ $\quad \text{CosSim}(D_1, Q) = 5 / \text{sqrt}(38) = 0.81$

$D_2 = 3T_1 + 7T_2 + T_3$ $\quad \text{CosSim}(D_2, Q) = 1 / \text{sqrt}(59) = 0.13$

$Q = 0T_1 + 0T_2 + 2T_3$