法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：大数据分析挖掘
  - 新浪微博：ChinaHadoop

分布式爬虫

大纲

- 文本分词
- 主题爬取
- 网页排重

文本分词

网页排重

重复网页类型

互联网存在大量的内容转载，也存在不同域名指向同一网站的情况，因此抓取回来的网页会有大量的重复。除了全文重复之外，还存在主要内容重复等情况，关于重复，我们定义以下4种类型：

- 如果2篇文章内容和格式上毫无差别，则这种重复叫做“完全重复页面”
- 如果2篇文章内容相同，但是格式不同，则叫做“内容重复页面”
- 如果2篇文章有部分重要的内容相同，并且格式相同，则称为“布局重复页面”
- 如果2篇文章有部分重要的内容相同，但是格式不同，则称为“部分重复页面”

查重的目的

- 就能节省一部分存储空间，提高检索的质量
- 对以往搜集信息的分析，预先发现重复网页，在今后的网页搜集过程中就可以避开这些网页
- 对重复次数多的网页，给予更高的 page rank
- 提高容错能力。比如一个网页的图片不存在了，我们可以到它类似的网页上尝试提取图片

查重的目的

- 就能节省一部分存储空间，提高检索的质量
- 对以往搜集信息的分析，预先发现重复网页，在今后的网页搜集过程中就可以避开这些网页
- 对重复次数多的网页，给予更高的 page rank
- 提高容错能力。比如一个网页的图片不存在了，我们可以到它类似的网页上尝试提取图片

查重算法 – 文本相似度

将一篇文章分词后，词汇加权行成一个向量，称之为这篇文章的特征向量，如果两篇文章的相似度很高，那么他们在高维空间里的夹角应该就比较小

计算向量的夹角：

$$v1 \times v2 = |v1||v2|cos\theta$$

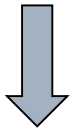
查重算法 – 文本相似度

从附加赛第三轮以3-0力擒素可泰，到小组赛首轮以1-0绝杀首尔FC，再到此役5-1大比分战胜西悉尼流浪者，上海上港在新赛季的亚冠赛场上一举拿到三连胜，为球队在赛季初始阶段开了个好头。此役战胜西悉尼流浪者后，上港不但继续保持着主场不败的金身，也成为在亚冠改制后，继泰达、国安和恒大之后第三支前2轮全胜的球队。

从附加赛以3-0力擒素可泰，到小组赛以1-0绝杀首尔FC，再到此役5-1大比分战胜西悉尼流浪者，上海上港在新赛季的亚冠赛场上拿到三连胜，为球队在赛季开了个好头。战胜西悉尼流浪者后，上港不但继续保持着主场不败的金身，也成为在亚冠改制后，继泰达、国安和恒大之后第三支前2轮全胜的球队。

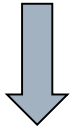
文本相似度计算

附加赛、素可泰、小组赛、首轮、绝杀、首尔FC、此役、大比分、战胜、西悉尼流浪者、上海上港、赛季、亚冠、赛场、、连胜、球队、赛季、好头、战胜、西悉尼流浪者、上港、主场、不败、金身、亚冠、改制、泰达、国安、恒大、全胜、球队



提取特征向量

[30234　23425　234235　234232　...]

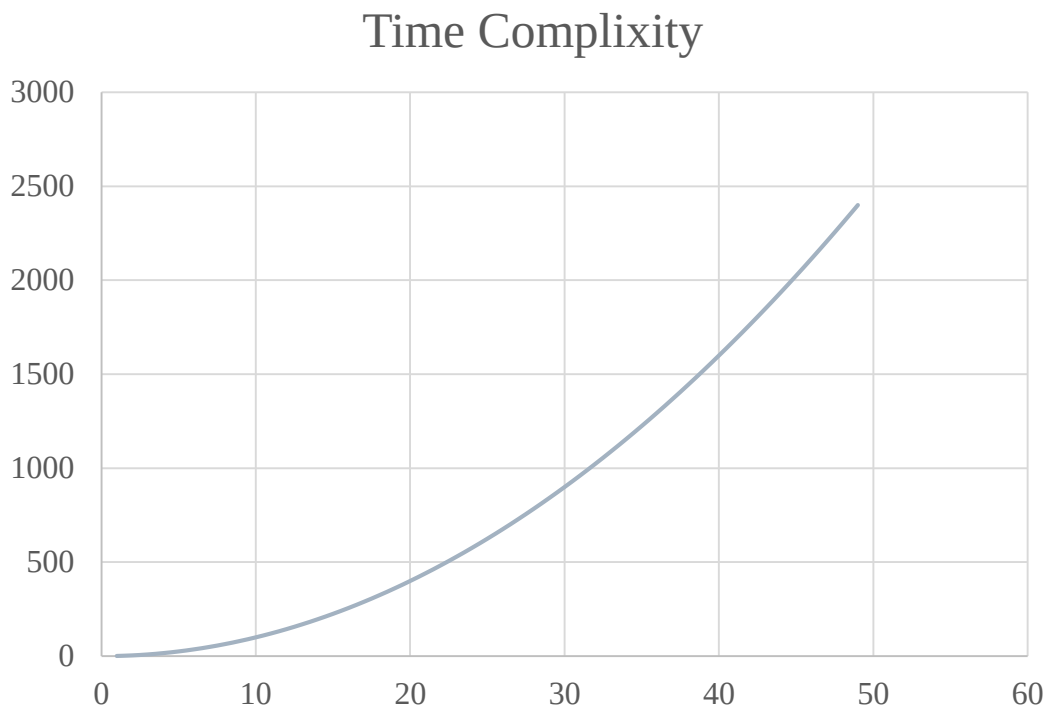


计算夹角

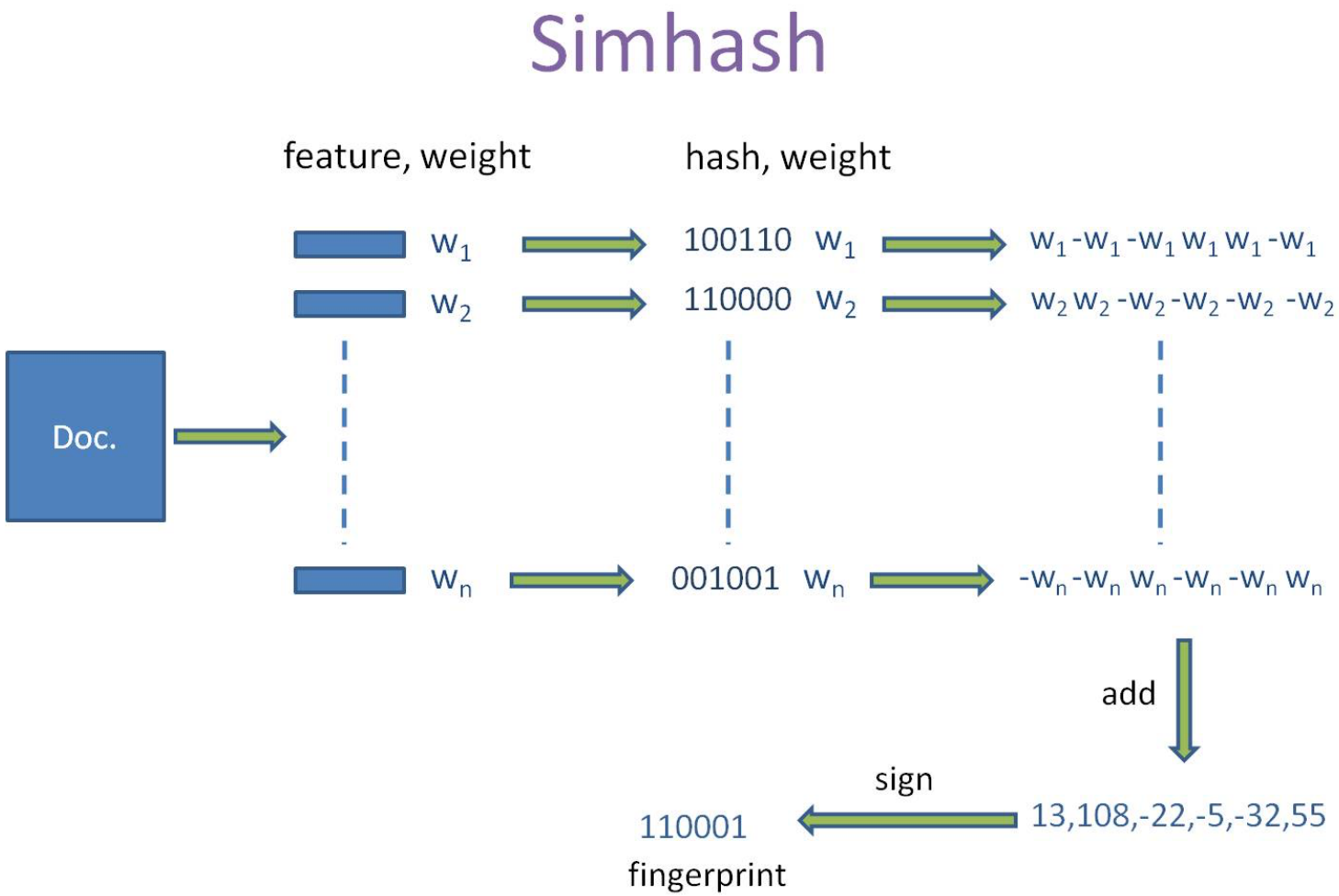
$$\theta = arcos(\frac{v1 \times v2}{|v1||v2|})$$

文本相似度计算

时间复杂度太高，n^2



查重算法 - SimHash



SimHash – I 分词

分词，把需要判断文本分词形成这个文章的特征单词。最后形成去掉噪音词的单词序列并为每个词加上权重，我们假设权重分为5个级别（1~5）

附加赛	3
小组赛	4
首轮	1
绝杀	3
首尔FC	4
西悉尼流浪者	4
上海上港	5
赛季	5
亚冠	5

SimHash – II Hash

通过hash算法把每个词变成 Hash 值，比如“亚冠”通过 Hash 算法计算为 111000，“上海上港”通过 Hash 算法计算为 101011。这样我们的字符串就变成了一串串数字，要把文章变为数字计算才能提高相似度计算性能，现在是降维过程进行时

附加赛	3	101001
小组赛	4	101110
首轮	1	110001
绝杀	3	101000
上海上港	5	101011
赛季	5	101100
亚冠	5	111000

SimHash – III 加权

Hash 生成结果，需要按照单词的权重形成加权数字串，bit 为 1，权重值乘以1，bit 位为0，权重值乘以-1，比如“亚冠”的 Hash 值为 111000，通过加权计算为5 5 5 -5 -5 -5

附加赛	3	101001	3 -3 3 -3 -3 3
小组赛	4	101110	4 -4 4 4 4 -4
首轮	1	110001	.
绝杀	3	101000	.
上海上港	5	101011	.
赛季	5	101100	
亚冠	5	111000	5 5 5 -5 -5 -5

SimHash – VI 合并

把上面各个单词算出来的序列值累加，变成只有一个序列串，也就是每一个数字位相加

附加赛	3	101001	3 -3 3 -3 -3 3
小组赛	4	101110	4 -4 4 4 4 -4
首轮	1	110001	1 1 -1 -1 -1 1
绝杀	3	101000	3 -3 3 -3 -3 -3
上海上港	5	101011	5 -5 5 -5 5 5
赛季	5	101100	5 -5 5 5 -5 -5
亚冠	5	111000	5 5 5 -5 -5 -5
			26 -24 24 -8 -8 -8

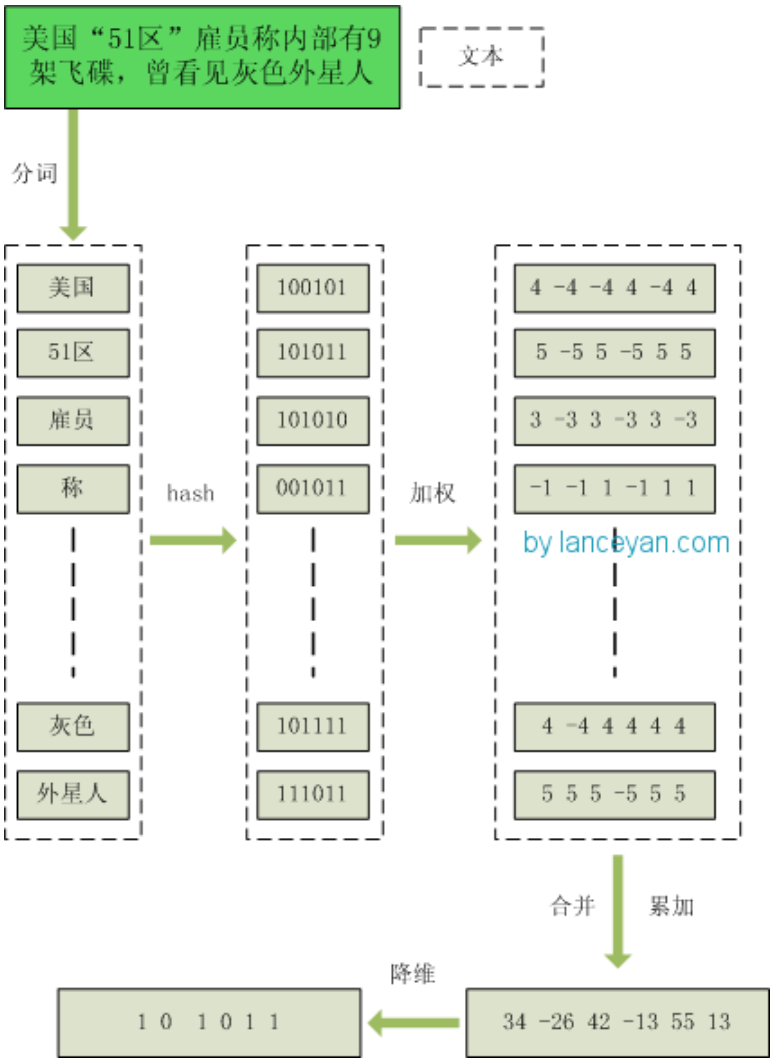
SimHash – V 降维

将结果转换为 0 1 这样的序列，规则是如果一个位上的和（例如第一位的和是26）> 0，则该位设置为1；否则为 0

26 -24 24 -8 -8 -8

1 0 1 0 0 0

SimHash – 图解



SimHash – 意义

将加权值合并的时候进行了第一步的降维，在最后一步根据正负换算为1和0（不考虑数值本身的大小，只考虑正负）进行了第二次降维，因此最终的Hash 值对字符串的改变变得不是很敏感。

传统 Hash 算法，任意字符的变化，都会导致整个 Hash 结果的剧烈变化

上港不但继续保持着主场不败的金身，也成为在亚冠改制后
上港不但继续保持主场不败的金身，成为在亚冠改制后

SimHash
1101100100010011101000101010110001010101101
1101000100000011100001010111100010101101101

md5:
68254e797b2bdc7e022b0004649a8c96
a500532daefa7bdbc50159fce4687208

SimHash – 海明距离

海明距离：两个二进制串中不同位的数量

可以通过异或，然后求bit为1的个数，即 A xor B 后二进制中1的个数

Python Library: SimHash

__init__: 构造函数，文本

build_by_features(): 输入一个 dictionary 或者 包含tuple的数组，以 key – weight 的方式输入，例如 {'Apache':20, 'hadoop':25} 或 [('Apache', 20), ('hadoop', 25)]

distance(another)：计算与另一个Simhash 对象的海明距离

SimHash – 海明距离

```
pip install simhash

from simhash import Simhash

str0 = 'The Apache Hadoop software library is a framework that allows for
the distributed processing large data'
str1 = 'The Apache Hadoop software library is a framework that allows for
the distributed processing big data'

# 构造 SimHash 对象
sh0 = Simhash(str0)
sh1 = Simhash(str1)

# 构造特征值，关键字加权
features = [('Apache', 10),('Hadoop', 15),('framework', 3), ('distributed',
10), ('data', 6)]
# 不加权计算
sh0.distance(sh1)
# 加权计算海明距离
sh0.build_by_features(features)
sh1.build_by_features(features)
sh0.distance(sh1)
```

处理数百万的网页？

假设我们需要计算海明距离在3以内的网页，可以将64位 Hash 所有变化在3以内组合列出来，一共是

$$C_{64}^{3} = \frac{64 \times 63 \times 62}{3 \times 2}$$
$$= 41664$$

当海明距离增加，所需要的空间也指数级增加，思考找一种更加高效的方法？

处理数百万的网页？

问题：一个80亿的64-bit指纹组成的集合Q，对于一个给定64-bit的指纹F，如何在毫秒级找到Q中和 f 至多只有 $k(k=3)$位 差别的指纹？

如果将 f 的 64位 Hash 所有变化在3以内组合列出来，一共是

$$C_{64}^{3}+C_{64}^{2}+C_{64}^{1}+C_{64}^{0}=43745$$

也就是说，所有与文档F，指纹距离在3以内的文档一定存在于这43745个指纹组成的列表里

两种简单粗暴的办法

- online，实时把 F 的 43745种距离为3个指纹结果计算出来，然后依次查找，因此需要计算出43745个结果并进行43745次查找
- offline，离线把 F 的 43745个距离为3个指纹结果缓存起来。意味着，如果一万个网页，每个网页需要存储 43745个SIMHASH拷贝，总量为 437,450,000 个 SIMHASH值，这样可以在$O(1)$的时间查找出所有相似网页

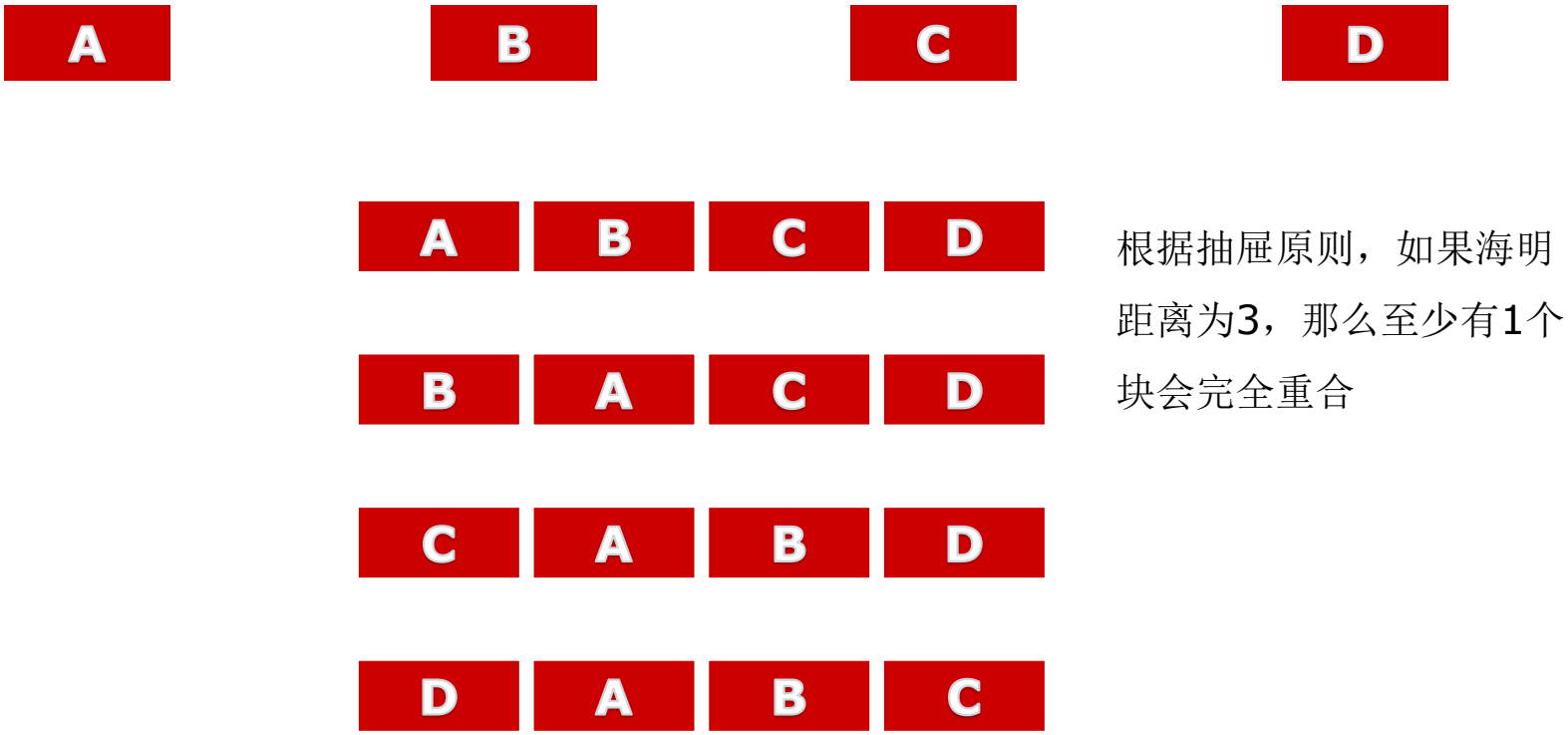
优化查找过程

把上面的粗暴方法结合一下，不要实时计算所有的 SimHash 可能性并逐一比较，提前进行一些离线计算，但同时也全部都算出来，以至于存储43745倍个拷贝，我们做一个折中

假设我们计算的是海明距离为3以内的网页，那么根据抽屉原则，如果把 SimHash 的64 位分为4块，ABCD，如果2个海明距离为3以内的网页，它们至少有一个块是完全一样的

分为4等份

1100101100100010　1100100000000010　1100111100000010　0000101100101100



根据抽屉原则，如果海明距离为3，那么至少有1个块会完全重合

算法复杂度

空间会增加到 4 倍，搜索时间，由于排除了16个bit位，因此总的可能性降低从 2^{64} 降低到了4×2^{48}，即$4 \times \frac{2^{48}}{2^{64}} = \frac{4}{2^{16}}$，假设网页总数为$1B$，现在需要比较的次数为 61000 次

与之前的粗暴运算的方法比，这种方法不但增加了空间，比较的次数还增加了。注意到，排除16个bit位后，需要比较的次数是指数方式在减少，从总共10亿次比较减少到了6万次，我们是不是可以想办法继续增加排除的bit位？

疑问

- 问题答疑： http://www.xxwenda.com/
  - 可邀请老师或者其他人回答问题

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop