

## TF-IDF与余弦相似性的应用（三）：自动摘要

作者： 阮一峰

日期: 2013年3月26日

有时候，很简单的数学方法，就可以完成很复杂的任务。

这个系列的前两部分就是很好的例子。仅仅依靠统计词频，就能找出关键词和相似文章。虽然它们算不上效果最好的方法，但肯定是最简便易行的方法。

今天，依然继续这个主题。讨论如何通过词频，对文章进行[自动摘要](#)（Automatic summarization）。

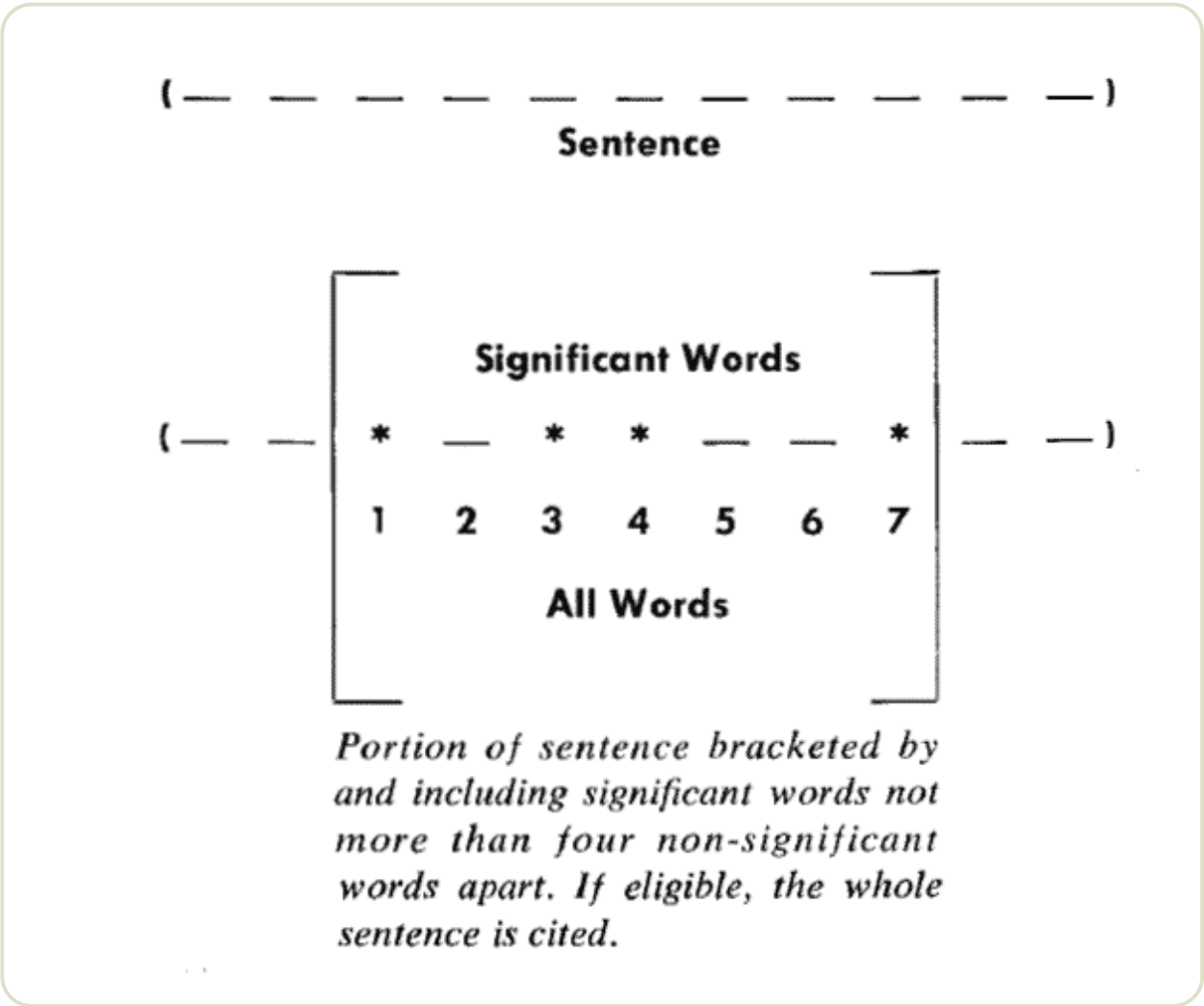


如果能从3000字的文章，提炼出150字的摘要，就可以为读者节省大量阅读时间。由人完成的摘要叫"人工摘要"，由机器完成的就叫"自动摘要"。许多网站都需要它，比如论文网站、新闻网站、搜索引擎等等。2007年，美国学者的论文[《A Survey on Automatic Text Summarization》](#)（Dipanjan Das, Andre F.T. Martins, 2007）总结了目前的自动摘要算法。其中，很重要的一种就是词频统计。

这种方法最早出自1958年的IBM公司科学家H.P. Luhn的论文《[The Automatic Creation of Literature Abstracts](#)》。

Luhn博士认为，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。Luhn提出用"簇"（cluster）表示关键词的聚集。所谓"簇"就是包含多个关键词的句子片段。



上图就是Luhn原始论文的插图，被框起来的部分就是一个"簇"。只要关键词之间的距离小于"阈值"，它们就被认为处于同一个簇之中。Luhn建议的阈值是4或5。也就是说，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。

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

以前图为例，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  $(4 \times 4) / 7 = 2.3$ 。

然后，找出包含分值最高的簇的句子（比如5句），把它们合在一起，就构成了这篇文章的自动摘要。具体实现可以参见 [《Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites》](#)（O'Reilly, 2011）一书的第8章，python代码见[github](#)。

Luhn的这种算法后来被简化，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。

```
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')...]
    wordFrequencies = getWordCounts(originalText)

    // 过滤掉停用词，数组变成[(3, 'language'), (8, 'code')...]
    contentWordFrequencies = filterStopWords(wordFrequencies)

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

    // 将文章分成句子
    sentences = getSentence(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
```

类似的算法已经被写成了工具，比如基于Java的[Classifier4J](#)库的[SimpleSummariser](#)模块、基于C语言的[OTS](#)库、以及基于[classifier4J](#)的[C#实现](#)和[python实现](#)。

（完）

## 文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2013年3月26日
- 更多内容： [档案](#) » [算法与数学](#)
- 购买文集：  《如何变得有思想》
- 社交媒体：  twitter,  weibo
- Feed订阅： 



## 相关文章

---

- **2016.07.22:** [如何识别图像边缘？](#)

图像识别（image recognition）是现在的热门技术。

- **2015.09.01:** [理解矩阵乘法](#)

大多数人在高中，或者大学低年级，都上过一门课《线性代数》。这门课其实是教矩阵。

- **2015.07.27:** [蒙特卡罗方法入门](#)

本文通过五个例子，介绍蒙特卡罗方法（Monte Carlo Method）。

- **2015.06.10:** [泊松分布和指数分布：10分钟教程](#)

大学时，我一直觉得统计学很难，还差点挂科。

---

联系方式 | [ruanyifeng.com](http://ruanyifeng.com) 2003 - 2017