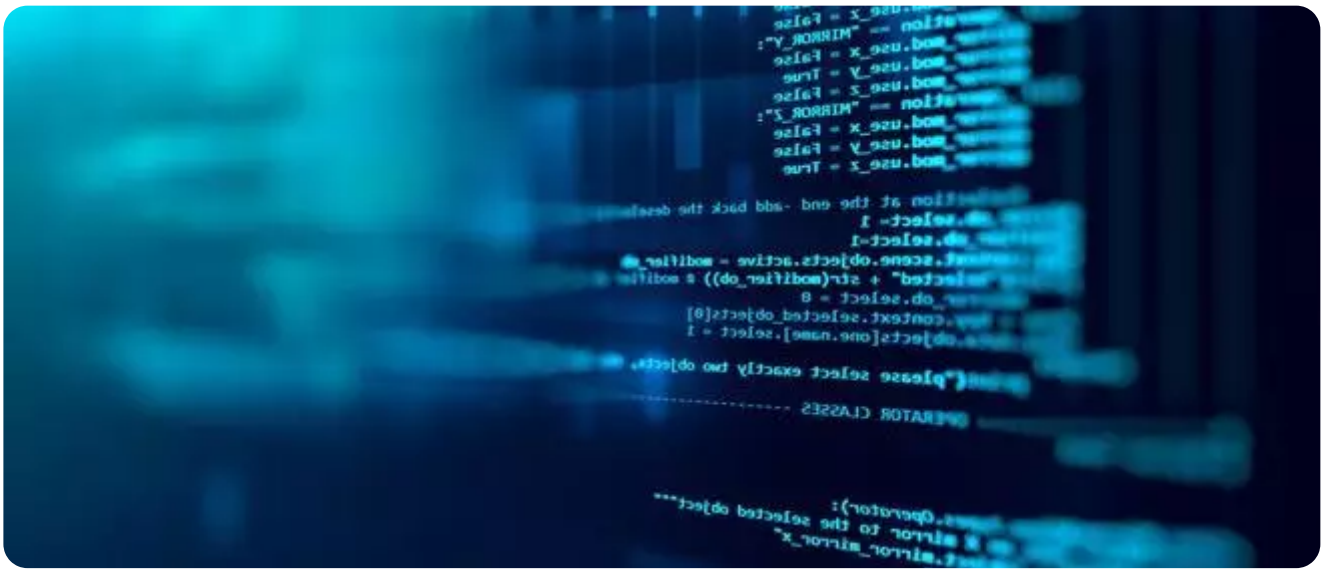
NLP（TF-IDF）---关键词提取算法实现

壹沓科技 壹沓科技 2019-03-08

壹沓第95篇，8， Mar



随着现代社会信息化进程的加快，如今的企业机构通常会有意识地储存一些数据，这些数据数量庞大而又种类不一，可能来自客户、邮件，也可能来自媒体和其他APP平台。为了从这些数据中分类筛选出有效内容，企业往往需要投入大量的时间和精力，而为了能够快速便捷地进行内容筛选，他们通常会采用一项从文本数据中提取信息的核心技术——自然语言处理（NLP或TF-IDF）。

一般情况下，这项技术往往会出现在下面这种应用场景中：

Q:现在我有一篇文章，需要从中提取它的关键词（Automatic Keyphrase extraction），如果完全不用人工介入，那么我应该怎么做？

A:这个问题将会涉及到数据挖掘、文本处理、信息检索等多个计算机前沿领域，但通过NLP（TF-IDF）算法我们可以快速实现并达到非常满意的结果。

下面我们一起来具体了解下这项核心技术：

一、 TF_IDF实现的原理

简单来说，TF-IDF是一种统计方法，它通常用以评估，在一个文件集或一个语料库中，其中某一份文件的某一字词的重要程度。

它由两部分组成，TF和IDF。TF（Term Frequency）词频，是指一个词在文章中出现的频率，假如一篇文章中的某个词出现了很多次，那么说明这个词可能是一个比较重要的词，但是，如果把“的、得、你、我”这一类常用词也全算上，那我们的计算就没有价值了。所以计算机还需要排除这些词。

IDF（inverse document frequency）逆文档频率，指的是是“权重”的度量，在词频的基础上，如果一个词在多篇文档中出现的频率较低，也就表示这是一个比较少见的词，但在某一篇文章中却出现了很多次，则代表这个词IDF值越大，在这篇文章中的“权重”也越大。所以当一个词越常见，它的IDF值反而越低。

概括来讲， IDF反应了一个词在所有文本中出现的频率，如果一个词在很多的文本中出现，那么它的IDF值应该低，比如上文中的“to”。而反过来说，如果一个词在比较少的文本中出现的次数多，那么它的IDF值应该高。比如一些专业的名词如“Machine Learning”等等。这样的词IDF值应该高。因此，还有一个极端的情况，那就是如果一个词在所有的文本中都出现，那么它的IDF值应该为0。

当计算出TF和IDF的值后，两个一乘就得到TF-IDF，这个词的TF-IDF指数越高就表示，就表示这个词在这篇文章中的重要性越大，越有可能就是文章的关键词。

二、 TF-IDF的实际应用

让我们用一个实际的例子来讲吧。比如《上海金融学习班》这样一篇文章，我们准备用计算机提取它的关键词。

首先，一个比较容易能想到的方法，就是找出文章中出现次数最多的词。如果某个词很重要，它应该在这篇文章中多次出现。于是，我们进行"词频"（Term Frequency，缩写为TF）统计。

结果你肯定猜到了，出现次数最多的词是----"的"、"是"、"在"----这一类最常用的词。它们被称为"停用词"（stop words），在计算机中表示对找到结果毫无帮助、必须过滤掉的词。

那么假设我们把“停用词”都过滤掉，只考虑剩下的有实际意义的词。这样的话又会遇到了另一个问题，我们可能会发现"上海"、"金融"、"学习班"这三个词出现的次数一样多。那么，这是不是就意味着，作为关键词，它们的重要性是一样的呢？

显然不是这样。因为"上海"是很常见的词，相对而言，"金融"和"学习班"不那么常见。如果这三个词在一篇文章的出现次数一样多，我们有理由认为，"学习班"和"金融"的重要程度要大于"上海"，也就是说，在关键词排序上面，"学习班"和"金融"应该排在"上海"的前面。

所以，我们需要用一个重要性调整系数来衡量一个词是不是常见词。如果某个词比较少见，但是它在这篇文章中多次出现，那么它很可能就反映了这篇文章的特性，也正是我们所需要的关键词。

用统计学语言表达，就是在词频的基础上，要对每个词分配一个"重要性"权重。最常见的词（"的"、"是"、"在"）给予最小的权重，较常见的词（"上海"）给予较小的权重，较少见的词（"金融"、"学习班"）给予较大的权重。这个权重就叫做"逆文档频率"（Inverse Document Frequency，缩写为IDF），它的权重大小与一个词的常见程度成反比。

在了解了"词频"（TF）和"逆文档频率"（IDF）以后，我们将这两个值相乘，就得到了一个词的TF-IDF值。某个词对文章的重要性越高，它的TF-IDF值就越大。所以，排在最前面的几个词，就是这篇文章的关键词。

第一步，计算词频

词频（TF）=某个词在文章中出现的次数

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

以上式子中 ，分子是指这个词在文章中出现的次数，而分母是指文章中所有字数之和。最后得出词频（TF）

第二步，计算逆文档频率（IDF）

$$\mathrm{idf_i} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

在这个地方需要一个语料库，才可以完成IDF的计算，而在这个式子中，分子是指语料库中文件总数，而分母是指包含该词的文档数。

第三步，计算TF-IDF。

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。

还是以《上海金融学习班》为例，假定该文长度为800个词，"上海"、"金融"、"学习班"各出现10次，则这三个词的"词频"（TF）都为0.0125。然后，搜索Google发现，包含"的"字的网页共有250亿张。假定这就是中文网页总数，其中包含"上海"的网页共有62.3亿张，包含"金融"的网页为0.484亿张，包含"学习班"的网页为0.973亿张。则它们的逆文档频率（IDF）和TF-IDF如下：

	包含该词的文档数（亿）	IDF	TF-IDF
上海	62.3	0.603	0.0075
金融	0.484	2.713	0.0033
学习班	0.973	2.410	0.0030

从上表可见，"学习班"的TF-IDF值最高，"金融"其次，"上海"最低。（如果还计算"的"字的TF-IDF值，那将是一个极其接近0的值。）所以，如果只选择一个词，"学习班"就是这篇文章的关键词。

三、 实际代码演示效果

实际代码截图：

```
var nodejieba = require("nodejieba");
var fs = require('fs');
var topN = 100;
var result;
var data = fs.readFileSync('t.txt', 'utf8');
console.log(data);
result = nodejieba.extract(data, topN);
console.log("11==>",result);
```

需要分析的文章

```
据中国之声《新闻纵横》报道，在刚刚过去的中秋之夜，一颗“火流星”滑亮了云南省迪庆州的夜空。根据相关天文机构公布的信息，陨石坠落的

事发一周之后，昨天（11日）下午，记者专访了巴拉格宗景区相关人员。对方称，目前还是没有确定陨石坠落的具体位置。最近，有很多人员都

巴拉格宗景区的工作人员洛桑培楚说，事发当时，景区的多位工作人员都目睹了那颗“火流星”，“因为我们酒店的位置，刚好是在一个U字型的峡
```

通过NLP（TF-IDF）算法最终实现自动提取关键词

```
liuyugang:NodeJieBa apple$ node nodenlp.js
....
11==> [ { word: '陨石', weight: 45.6077707943 },
  { word: '格宗', weight: 35.21761292125063 },
  { word: '景区', weight: 32.27518069876 },
  { word: '巴拉', weight: 29.735080816230003 },
  { word: '火流星', weight: 24.582479479 },
  { word: '坠落', weight: 18.22637181838 },
  { word: '事发', weight: 16.80701885336 },
  { word: '工作人员', weight: 13.28734988976 },
  { word: '震感', weight: 12.5143832909 },
  { word: '迪庆', weight: 11.9547675029 },
  { word: '11', weight: 11.739204307083542 },
  { word: '培楚', weight: 11.739204307083542 },
  { word: '有个', weight: 11.739204307083542 },
  { word: '人员', weight: 11.18200151198 },
  { word: '新闻纵横', weight: 11.0103058941 },
  { word: '具体位置', weight: 10.8096351986 },
  { word: '飞过来', weight: 10.765183436 },
  { word: '香格里拉', weight: 10.642581114 },
  { word: '洛桑', weight: 10.2630914922 },
  { word: '字型', weight: 10.0088573539 },
  { word: '相关', weight: 9.67141986604 },
  { word: '崖壁', weight: 9.65218240993 },
  { word: '没有', weight: 9.338470695449999 },
  { word: '目睹', weight: 8.79473217808 },
  { word: '之后', weight: 8.7536825453 },
  { word: '夜空', weight: 8.75318317516 },
  { word: '之夜', weight: 8.65893063692 },
  { word: '中秋', weight: 8.55357012126 },
```

（以上内容整理by壹沓-产品部 帆哥）

END

阅读原文