

13 | 基于关键词的召回：如何使用关键词吸引用户？

2023-05-15 黄鸿波 来自北京

《手把手带你搭建推荐系统》



你好，我是黄鸿波。

在讲解了基于时间的召回和基于热度的召回后，今天我们进入到基于规则召回的最后一种——基于关键词的召回，我将本节课分为了下面两个大的部分。

1. 基于关键词的召回。
2. 提取关键词的几种获取方式。

那么话不多说，我们直接开始本节课的内容。

基于关键词的召回

对于任何一篇文章、段落、标题等，只要是以文字形式展现的内容，我们都能够通过一系列的词语组合起来代表一篇文章的主旨，而这一系列的词语就是文章的**关键词**。

文章的标题是一篇文章最精辟的概括，摘要是一篇文章的主旨，正文是文章的详细信息。对于一篇文章来说，我们最先看到的是关键词，其次是摘要，然后才是文章正文的本身。因此，在做关键词提取和文章的特征时，我们一般需要分开来做关键词的提取和保存。

基于关键词的召回往往与用户画像或搜索关键词息息相关。在 [🔗 用户画像](#) 中实际上也会有用户常看的内容关键词的集合，甚至还可以对这些关键词的出现频率进行排序，从而来确定这些关键词的重要性。而在基于关键词召回的算法中，我们可以将文章的关键词和用户画像中用户标签的关键词进行结合，从而进行基于关键词的召回。

除了用户画像外，用户的搜索内容往往是基于关键词召回的关键所在。一般来讲，一旦用户搜索了一个关键词，说明用户对这个关键词表示的内容是非常感兴趣的。因此，我们在做推荐的时候就需要尽可能把关键词相关的内容给展示出来，从而增加用户的黏性和点击率。

在与用户搜索内容结合的时候，一般来讲，我们会采取多种不同的方式来结合。最简单直接的方式是**关键词与热度结合**。

当用户在进行关键词搜索时，我们在后台就可以知道用户目前搜索的关键词是什么，然后将关键词放到内容画像中进行搜索，这个时候我们可能会找到一个、多个或者根本没有相关关键词的内容。当有多个和该关键词相关的内容之后，我们就可以将这些内容按照热度降序进行排列，然后展现给用户。

当与之匹配的关键词内容只有一个或者没有时，我们就得用另外一种形式对内容进行补充：**结合协同过滤或基于关键词相似度的 embedding 召回**。

这两块的召回部分会在后面的课程中详细讲解，在这里我先来说一下大致的原理。我们在操作的时候如果发现召回的内容非常少，这个时候有两种策略：**不给用户推荐**和**给用户推荐相似的内容**。

如果我们要推荐相似内容，我们实际上最简单的方法就是去找关键词的相似关键词，或者直接拿这个关键词与我们关键词库的每一个词做距离计算，找到距离最相机的几个关键词，然后再将这些关键词作为目标关键词进行关键词召回，这样就能够召回更多的内容进行推荐。

你能看到，关键词召回需要联合其他形式才能起到最好的效果。

提取关键词的几种常见算法

说完关键词召回怎么做之后，我们进入到提取关键词的算法部分。

TF-IDF 算法

shikey.com转载分享

要了解 TF-IDF 算法，需要先澄清三个概念：词频、去停用词和逆文档频率。以文档 D 为例，用 ω 来表示一个关键词，可用以下公式统计出 ω 在文档 D 中出现的频率，即“词频 (Term Frequency, TF)”。

$$TF_{\omega, D_i} = \frac{\text{count}(\omega)}{D_i}$$

下面这个公式就是上面公式的文字化表示。

$$\text{词频(TF)} = \frac{\text{某个词在文章中出现的次数}}{\text{文章中词的总数}}$$

我们经常会发现，用词频算法提取的词往往是“的”“是”“在”这类词，这类词即为“停用词”。在进行关键词提取的任务中，去停用词是一个非常必要的前提条件，只有做完去停用词之后，统计出来的结果才是相对准确的。

假定文档题目是《中国人均收入调查》，那么提取的关键词很有可能是“中国”“收入”

“GDP”等词，而这类词虽然是这篇文章中的关键词，但并不能代表整篇文章的特性，因为这类词在其他的文章中也是高频词。那么怎么才能找到一个词，它既是关键词又能代表这篇文章的特性呢？这个时候我们需要一个新的名词，即“逆文档频率” (Inverse Document Frequency, IDF)，文档总数 n 与词 ω 所出现文件数 $\text{doc}(\omega, D)$ 比值的对数，可以表示为下面这个公式。

$$IDF_{\omega} = \log_2 \left(\frac{n}{\text{docs}(\omega, D)} \right)$$

shikey.com 转载分享

下面这个公式就是上面公式的文字化表示。

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

从上面公式可以看出，如果一个词越常见（如“中国”“收入”“GDP”），那么其逆文档频率就会越小。也就是说，逆文档频率越大，关键词在其他的文档中出现概率越小，意味着越是这篇文章的关键词。为了避免逆文档频率大到所有的文档都不包含该关键词，上述公式的分母加上 1。

那么，一篇文档的 TF-IDF 值就与一个词在文档中出现的次数成正比，与该词在其他文档中出现的次数成反比。

$$TF - IDF = \text{词频}(TF) \times \text{逆文档频率}(IDF)$$

因此，TF-IDF 算法即对关键词的 TF-IDF 值做降序排序，最后取前 N 个关键词作为最终的结果。

TextRank 算法

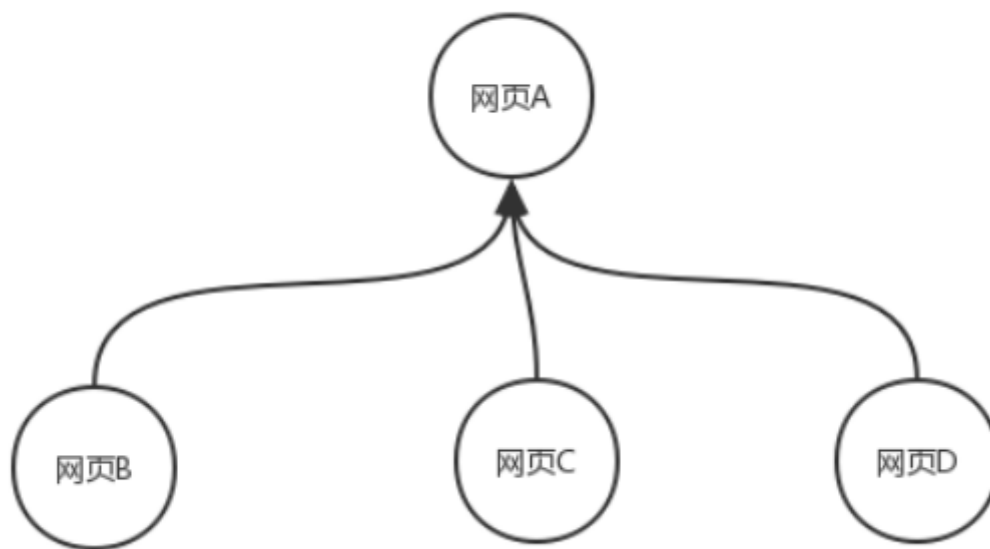
与 TF-IDF 不同，TextRank 在进行关键词提取时能够考虑到相邻词语之间的语义关系，而 TF-IDF 只是针对每个词的词频进行统计，并没有充分考虑词之间的语义信息。这就衍生出 TextRank 的另外一个优势：**脱离语料库**。

脱离语料库是指 TextRank 并不需要像 TF-IDF 一样使用语料库进行训练，而是直接针对单篇文章就可以进行关键词的提取，因此 TextRank 算法对于小批量数据来说是一个非常理想的选择。

从原理上来讲，TextRank 算法是基于 PageRank 算法衍生出来的。其核心思想是通过词之间的相邻关系构建网络，然后用 PageRank 迭代计算每个节点的 rank 值，排序 rank 值即可得到关键词。

TextRank 算法是一个无向图的算法。所谓的无向是指两个词之间的边是无向的，这也是 TextRank 算法与 PageRank 算法最大的区别。

我们先简要地介绍一下 PageRank 算法。PageRank 算法是 Google 用来计算网页权重值的一种算法，在 PageRank 算法中，如果一个链接的权重值越大，说明这个链接就越重要。假设现在有四个网页分别为网页 A、网页 B、网页 C 和网页 D，在这四个网页中，所有除了网页 A 的网页都只链接到了网页 A，那么 A 的 PR (PageRank) 值就是网页 B、网页 C 和网页 D 的 PageRank 值的总和，如图所示。

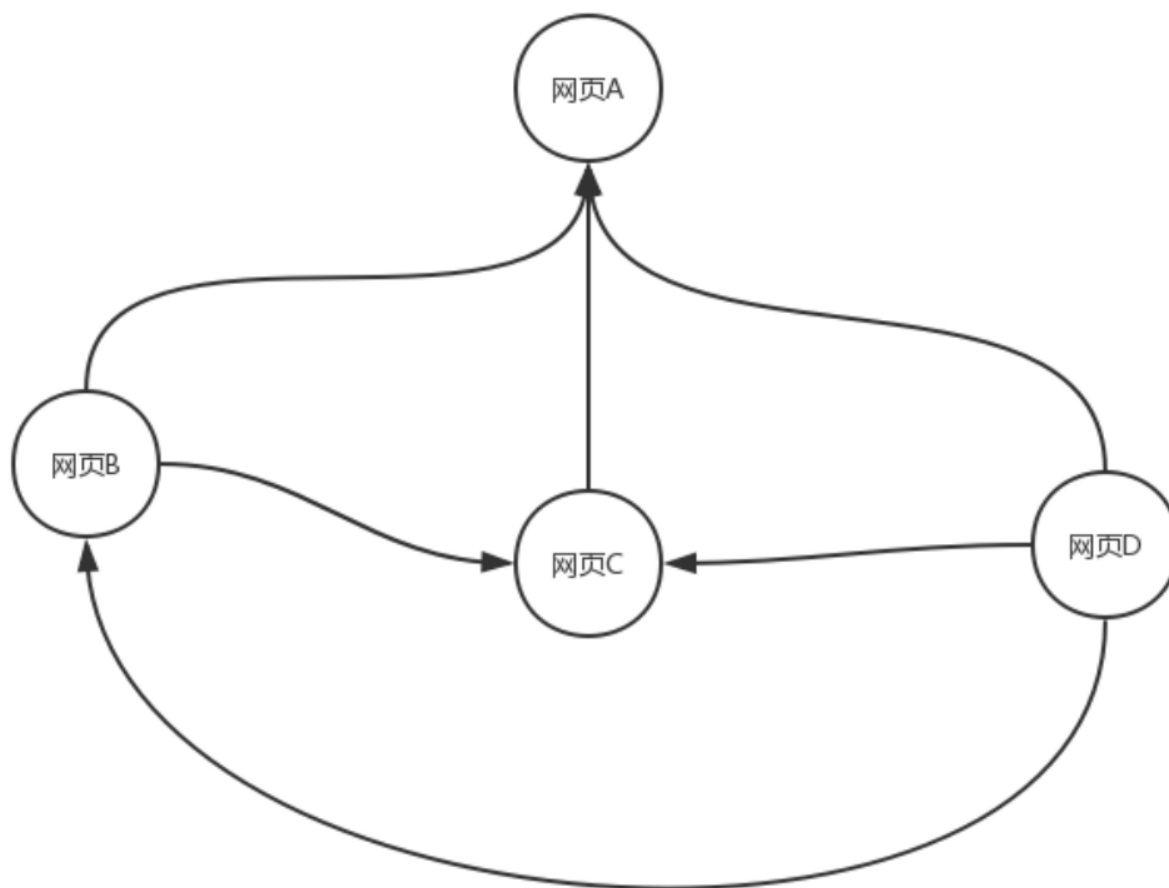


可以用下面这个公式来表示。

$$PR(A) = PR(B) + PR(C) + PR(D)$$

shikey.com转载分享

如果我们重新定义一下这四个网页的指向关系，网页 B 指向网页 A 和网页 C、网页 C 指向网页 A，网页 D 指向其他的所有网页，那么就有下图。



网页 B 有两个向外的链接，其中一个链接到了网页 A，因此，网页 A 对于网页 B 的 PR 值为 $\frac{PR(B)}{2}$ ；网页 C 只有 1 个向外的链接（即网页 A），因此，网页 C 对于网页 A 的 PR 值为 $\frac{PR(C)}{1}$ ；网页 D 有 3 个向外的链接，其中一个链接到了网页 A，因此，网页 D 对于网页 A 的 PR 值为 $\frac{PR(D)}{3}$ 。由此得到网页 A 的总的 PR 值为下面这个公式。

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

shikey.com转载分享

在实际 PR 值的计算中我们还要考虑到，当用户到达某个页面之后，是否还会继续通过这个页面向前。因此我们需要在 PR 值的计算中增加一项阻尼系数 d ，因此我们可以得到下面这个 PR 算法的推广公式。

$$PR(A) = \frac{(1-d)}{N} + d(PR(T_1)/C(T_1) + \dots + PR(T_n)/C(T_n))$$

其中 d 为阻尼系数，表示在任意时刻，用户到达某页面后并继续向后浏览的概率； $PR(T_i)$ 代表页面 T_i 的 PR 值； $C(T_i)$ 代表页面 T_i 指向其他页面的边的个数。

当然，这只是 PR 值计算的最基本的形式，在实际 Google 的 PR 算法中还有很多其他的变形以及其他系数，并且每隔一段时间这个系数和公式还会有微调。然后 Google 利用 PR 算法，当用户在 Google 上进行搜索时，就会计算与搜索关键词相关联的网页并计算出这些网页的 PR 值，然后将计算出来的 PR 值倒序排序，根据 PR 值从大到小的结果展示给用户。

TextRank 算法实际上是 PageRank 算法的一个变体，其中最大的变化在于 PageRank 算法是基于网页，而 TextRank 算法是基于词；另外 PageRank 算法中网页之间是一个有向关系，也就是说网页之间是有很明确的方向性，而 TextRank 算法是一个无向图，也就是说我们在表示两个词之间的权重时，对于两个词来讲是相同的。

因此，我们可以类比 PageRank 算法来说明一下 TextRank 提取关键词的具体思路。

1. 对整个文本进行句子切分。

2. 将待提取关键词的文本语句进行数据清洗，包括去停用词、无用词和标点符号、然后对其进行分词处理，形成候选关键词。
3. 构建候选关键词的图，这个图可以表示为 $G(V, E)$ ，其中 V 是由第 2 步生成的结果集，采用共现结果的方式； E 是由两个节点之间的边组成，可以把 E 看出 $V \times V$ 的集合；图中任意两个点之间的权重 V_i 和 V_j 之间边的权重可以表示为 W_{ij} 。对于给定的点 V_i 来说， $In(V_i)$ 为指向该点的点集合， $Out(V_i)$ 为点 V_i 指向的点集合。由此，我们可以通过下面这个公式算出点 V_i 的权重值。

$$WS(V_i) = (1 - d) + d * \sum_{v_j \in In(V_i)} \frac{\omega_{ji}}{\sum_{v_k \in In(V_j)} \omega_{jk}} WS(V_j)$$

4. 根据第三步中给出的公式，初始化各个节点的权重，然后迭代计算各个节点的权重，直至收敛。在这里的收敛是指经过 N 轮迭代，使得得到一个相对稳定的误差，而这个误差小于某个规定的阈值之内（一般我们将这个阈值设置为 0.001）。
5. 对上述得到的各个节点的权重进行倒序排序，从而得到这个文本中最重要的 N 个词，作为候选关键词。
6. 将第五步得到的候选关键词在原始的文本中进行标记，如果形成了相邻的词组，则组成一个多词关键词。

到这里，使用 TextRank 算法得到关键词的整套流程也就结束了。

两种算法联合提取关键词

前面的实现全部基于一种算法，但是在真正的企业工程化中，我们往往使用多种算法进行联合关键词提取。

在进行关键词提取之前，首先要对自己准备提取关键词的文本有比较全面的了解。虽然每个算法都能进行关键词提取，但是针对不同的文本内容及文本量来说，每种关键词提取算法的表现是不同的。

比如，当文本量大到可组成一个数据集时，我们可以考虑使用 TF-IDF 算法作为关键词提取的主要算法；但是如果文本量相对比较少（或者针对某一篇文章来进行关键词提取），那么常见做法就是使用 TextRank 算法。

单独使用一种算法的话，关键词提取的准确率会大打折扣，那么最好的做法就是联合使用几种算法，常见的做法是 TF-IDF+TextRank。


不同算法关注的重点不同，就会导致结果差异比较大。例如针对下面同样一段文本，我们使用 TF-IDF 和 TextRank 来做一个对比。

北京时间 5 月 28 日消息，据《北京青年报》官微透露，北京中赫国安归化球员李可将入选新一期国家队的大名单。而他将成为国足历史首位归化国脚，目前相关手续应该已经得到落实，意味着李可具备代表中国队参赛的资格。北京时间 5 月 28 日消息，据《北京青年报》官微透露，北京中赫国安归化球员李可将入选新一期国家队的大名单。而他将成为国足历史首位归化国脚，目前相关手续应该已经得到落实，意味着李可具备代表中国队参赛的资格。

对这段文本进行数据清洗后，得到如下文本。

北京时间 5 月 28 日消息，据《北京青年报》官微透露，北京中赫国安归化球员李可将入选新一期国家队的大名单。而他将成为国足历史首位归化国脚，目前相关手续应该已经得到落实，意味着李可具备代表中国队参赛的资格。北京时间 5 月 28 日消息，据《北京青年报》官微透露，北京中赫国安归化球员李可将入选新一期国家队的大名单。而他将成为国足历史首位归化国脚，目前相关手续应该已经得到落实，意味着李可具备代表中国队参赛的资格。

然后使用开源 jieba 库编写的整体算法进行处理，代码如下。

 复制代码

```
1 import jieba
2 import jieba.analyse
3 from segment import Segment
4
5
6 class Model(object):
7     def __init__(self):
8         self.seg = Segment("../common/data/stopword.txt", "../common/data/user_di
```

```

9
10 def process_text(self, text):
11     words_list = self.seg.cut(text)
12     words_list = ' '.join(words_list)
13     return words_list
14
15 def get_keyword(self, words_list, param, use_pos=True):
16     if use_pos:
17         # 选定部分词性
18         allow_pos = ('n', 'nr', 'nr1', 'nr2', 'ns', 'nsf', 'nt', 'nz', 'nl',
19     else:
20         allow_pos = ()
21     if param == 'tfidf':
22         tfidf_keywords = jieba.analyse.extract_tags(words_list, topK=10, with
23     return tfidf_keywords
24     elif param == 'textrank':
25         textrank_keywords = jieba.analyse.textrank(words_list, topK=10, withw
26     return textrank_keywords
27
28 def keyword_interact(self, tfidf_keyword, textrank_keyword):
29     return list(set(tfidf_keyword).intersection(set(textrank_keyword)))
30
31 def keyword_topk(self, tfidf_keyword, textrank_keyword, k):
32     combine = list(tfidf_keyword)
33     for word in textrank_keyword:
34         combine.append(word)
35     return list(set(combine))
36
37
38 if __name__ == "__main__":
39     model = Model()
40     text = '北京时间5月28日消息，据《北京青年报》官微透露，北京中赫国安归化球员李可将入选新一期
41     words_list = model.process_text(text)
42     tfidf_keyword = model.get_keyword(words_list, param='tfidf')
43     print('tfidf_keyword', tfidf_keyword)
44     textrank_keyword = model.get_keyword(words_list, param='textrank')
45     print("textrank_keyword", textrank_keyword)
46     keyword_interact = model.keyword_interact(tfidf_keyword, textrank_keyword)
47     print('keyword_interact', keyword_interact)
48     keyword_topk = model.keyword_topk(tfidf_keyword, textrank_keyword, 3)
49     print('keyword_topk', keyword_topk)
50

```

使用 TFIDF 算法提取的关键词如下。

```
1 tfidf_keyword ['归化', '李可', '官微', '中赫', '国脚', '北京青年报', '中国队', '北京']
```

复制代码

```
1 textrank_keyword ['归化', '北京', '国安', '消息', '历史', '参赛', '时间', '李可', '代表']
```

在企业中有两种方法联合算法，一种是取两种算法的交集（即这两种算法都认为是关键词的部分），那么我们认为这个关键词一定就能代表文章；另外一种是用两种算法的交集与每种算法的 Top3 关键词求并集，这样做默认了每种算法的前三个关键词都是真正的关键词。

方法一：两种算法的交集，我们可以得出的结果关键词集合如下。

复制代码

```
1 keyword_interact ['李可', '北京', '参赛', '归化']
```

方法二：两种算法的交集与每种算法的 Top3 关键词求并集，我们可以得出的结果关键词集合如下。

复制代码

```
1 keyword_topk ['归化', '李可', '国安', '北京', '官微']
```

从结果可以看出，方法一（取算法交集）的效果会更好一些。但是实际上在企业中，效果的好坏要根据实际的文本类型进行实验才能得出，而不是通过其他人的实验结果简单粗暴地做出结论。

总结

到这里，这节课就已经接近尾声了，我们来对今天的课程做一个简单的总结。学完今天的课程，你应该知道下面这三个要点。

1. 什么是基于关键词的召回算法。
2. 基于关键词的召回算法一般是如何实现的。
3. 提取关键词中两种常见的算法：TF-IDF 和 TextRank 算法，以及这两种算法的实现原理。

课后题

shikey.com转载分享

最后你布置两个小作业。

1. 使用关键词提取技术提取我们爬取的文章内容。
2. 想一想，我们还可以用关键词召回算法配合哪些算法进行使用。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (3)



peter

2023-05-16 来自北京

关键词的获取，是否有现成的可用的工具？就是说拿来就能用、基本不用开发。比如我要搭建一个推荐系统，也用到了关键词获取，导入一个库然后调用其API就可以直接获取；或者运行一个工具软件，可以直接获取；或者某个平台提供该服务。等等。

作者回复：可以的，其实有很多基于深度学习和机器学习的关键词提取的库，你可以去百度下这些库的用法，jieba就是其中一个。



Geek_ccc0fd

2023-05-16 来自广东

我装的jieba==0.42.1可以直接对句子提取关键词，分词的部分已经封装在jieba代码里面了，修改了一下关键词提取代码：

```
import jieba
from jieba.analyse import extract_tags
from jieba.analyse import textrank
```

```

class KeywordModel(object):
    def __init__(self):
        jieba.load_userdict('../data/user_dict.csv')
        jieba.analyse.set_stop_words('../data/stopWord.txt')

    def get_keywords(self, sentence, type, topK=10, pos=('ns', 'n', 'vn', 'v')):
        """
            shikey.com转载分享
            获取关键词
            :param sentence: 文本
            :param type: 使用哪种关键词算法, 可选: tfidf, textrank
            :param topK: 获取topK关键词
            :param pos: 分词保留的词性类型, eg: ('ns', 'n', 'vn', 'v')
            :return:
            """
        if type == 'tfidf':
            tfidf_keywords = extract_tags(sentence, topK=topK, allowPOS=pos)
            return tfidf_keywords
        elif type == 'textrank':
            textrank_keywords = textrank(sentence, topK=topK, allowPOS=pos)
            return textrank_keywords

    def keyword_interact(self, tfidf_keyword, textrank_keyword):
        """
            关键词交集
            :param tfidf_keyword:
            :param textrank_keyword:
            :return:
            """
        return list(set(tfidf_keyword).intersection(set(textrank_keyword)))

    def keyword_combine(self, tfidf_keyword, textrank_keyword):
        """
            关键词并集
            :param tfidf_keyword:
            :param textrank_keyword:
            :param k:
            :return:
            """

```



```
combine = list(tfidf_keyword)
for word in textrank_keyword:
    combine.append(word)
return list(set(combine))
```

```
def keyword_combine_topk(self, tfidf_keyword, textrank_keyword, k):
```

```
    """
```

```
    关键词topk并集
```

```
    :param tfidf_keyword:
```

```
    :param textrank_keyword:
```

```
    :param k:
```

```
    :return:
```

```
    """
```

```
    combine = list(tfidf_keyword[:k])
```

```
    for word in textrank_keyword[:k]:
```

```
        combine.append(word)
```

```
    return list(set(combine))
```

shickey.com转载分享

作者回复: 不错, 可以推广给同学们



翡翠虎

2023-05-15 来自广西

有常用的停用词表吗

作者回复: 一般常用的停用词表可以在网上找到, 或者找敏感词表, 有些github上面也会公布, 可以搜关键词stopwords。

共 2 条评论 >

