2.文本预处理

CONTENTS

一、英文文本预处理

01

英文文本预处理步骤

英文文本预处理步骤

文本数据清理
(Clearning)

去掉常用
或罕见的词

检查拼写
(Spelling)

词干化
及词形还原

N元模型
(N-gram)

高频词提取
及词云图绘制

数据清洗

- 原始的文本数据中往往包含很多**标点符号**和**数字**，尤其当文本数据来源于网络时，数据中通常会混杂很多的**网页标记符号**，因此需要事先对这些内容进行清洗，**只保留其中的汉字和字母（有时也包括数字）等具有实际意义的内容**。

去掉常用或罕见的词

中国人民大学
RENMIN UNIVERSITY OF CHINA

- 英文文本中的停用词**主要包括“the”、“a”、“is”**等。除了停用词外，有时也认为一些出现频率很低的罕见词语对分析文本没有较大帮助，因此可以去掉在整个文本数据集中**出现频率小于某个特定阈值的词**。

检查拼写

中国人民大学
RENMIN UNIVERSITY OF CHINA

- 英文文本中可能含有拼写错误的词，如不将这些词语去除或修正，可能会影响后续分析结果，因此要在预处理时**对文本进行拼写检查**。

词干化及词形还原

- **词干化（Stemming）**是指提取单词的词根，例如将“movie”提取词干为“movi”。
- **词形还原（Lemmatization）**则是将单词的复杂形态转变成最基础的形态。比如，将英文名词的复数形式还原为单数形式，将英文形容词的比较级还原为原型等。

N元模型（N-gram）

- 虽然英文中各个单次自然的被空格分开，但是有时我们也希望在英文文本中**以“词组”为单位进行分析**。
- 例如，将英文文本中，任意**前后相连的两个词作为一个词组**，这就是二元语法分词（bigram），或者将任意**前后相连的三个词作为一个词组**，这就是三元语法分词（trigram）。
- 二元模型和三元模型都属于N元模型（还原），**常用来评估一个句子是否合理以及评估两个字符串之间的差异程度。**

高频词提取以及词云图绘制

中国人民大学
RENMIN UNIVERSITY OF CHINA

经过上述处理后，可以计算每个英文单词的词频，然后从中找到高频词。最后通过**绘制词云图展示英文文本的主要内容。**

02

英文文本预处理示例

英文文本示例：R

下面以英文电影评论数据为例，介绍如何在R中进行英文文本预处理

第一步 安装R包

- **quanteda包**

在R中对英文文本进行预处理主要使用quanteda包进行。quanteda包是专门用于对文本数据进行定量分析的R包。比起直接使用R自带的字符串函数处理文本，quanteda包中的函数具有高效的优点，更适于处理大量的文本数据。

- **textstem包**

使用textstem包对文本进行词形还原。

- **qdap包**

使用qdap包进行拼写检查。

- **jiebaR包**

使用jiebaR包提取词频。

- **wordcloud2包**

使用wordcloud2包来绘制美观的词云图。

- **RColorBrewer包**

使用RColorBrewer包中的调色盘美化词云图。

```
library("quanteda")
library("textstem")
library("qdap")
library("jiebaR")
library("wordcloud2")
library("RColorBrewer")
```

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第二步 除去HTML标签

由于该数据是从网页上爬取获得，因此常含有**HTML标签**，需要除去。HTML是超文本标记语言，其形式通常是由尖括号包围的关键词，比如“<html>”。利用**正则表达式以及R语言自带的字符串函数gsub()函数即可将其除去。**

gsub()函数第一个参数表示**要识别替换的内容**，第二个参数是**替换后的内容**，第三个参数是**作用的对象**。所使用的正则表达式中，“.” 匹配除换行符 \n 之外的任何单字符；“*”匹配前面的子表达式零次或多次；“？”在此处是表示非贪婪匹配，即尽可能少的匹配。“.*?”表示匹配任意数量的重复，但是在能使整个匹配成功的前提下使用最少的重复。

```
#对每条评论文本进行匹配，除去HTML标签
for (i in 1:length(review)){
  #“<.*?>”能匹配最短的以“<”开始，以“>”结束的字符串，
  #所以下一行可以除去<>包围的关键词
  review[i] <- gsub("<.*?>","",review[i])
  review[i] <- gsub("Â.*","",review[i]) #除去特殊符号Â
  review[i] <- gsub("Ã.*","",review[i]) #除去特殊符号Ã
}
```

英文文本示例：R

中国人民大學
RENMIN UNIVERSITY OF CHINA

第三步 词形还原

R中用于进行词形还原的函数为**textstem包中的lemmatize_strings()**函数，得到的结果是字符型变量，会对后续处理造成影响，因此在进行分词前先进行词形还原。

```
review[1] #查看词形还原前的第一条评论
review <- lemmatize_strings(review) #实现词形还原
review[1] #查看词形还原后的第一条评论
```

以第一条评论的第一句话为例，词形还原前后的结果对比如下（[1]为词形还原前，[2]为词形还原后）

*[1] "Ant-Man **is** great Heist film **disguised** as a superhero film. With great **performances** all around, I'm **surprised** it doesn't get the attention it **deserves**. Definitely a must-see-movie."*

*[2] "Ant - Man **be** great Heist film **disguise** as a superhero film. With great **performance** all around, I'm **surprise** it doesn't get the attention it **deserve**. Definitely a must - see - movie."*

可以看到“is”被还原成了“be”，过去时“disguised”被还原成了一般现在时“disguise”，复数“performances”被还原成了原型“performance”。

英文文本示例：R

第四步 存储成tokens

使用**quanteda包中的tokens()函数**将英文文档中的每个单词提取出来，tokens()会产生一个以**字符向量形式存在的单词表**，表中的每个元素都对应于一个输入文档，此表的数据类型为“tokens”，在tokens内部，各个单词会作为单独的字符串进行存储。值得一提的是，tokens()具有保守性，它不会从文本中删除任何东西，除非有指令。

```
review <-  tokens(review) #实现分词
review[1] #查看分词后的第一条评论
class(review) #查看分词后的变量类型
```

第一条电影评论分词后的结果如下，可以看到成功实现了句子的分词：

"Ant" "-" "Man" "be" "great" "Heist" "film" "disguise" "as" "a"
"superhero" "film"
[... and 25 more]

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第五步 文本数据清洗

使用quanteda包中的tokens()进行清洗

实现词形还原和分词后的文本中还含有数字、标点、符号、网址、分隔符等非文本内容。通过设置tokens()函数中的参数，可以实现将文本中的数字（numbers）、标点（punctuations）、符号（symbols）、网址（URLs）以及分隔符（separators）除去。

其中，“remove_punct”表示**除去标点**；“remove_symbols”表示**除去符号**，“remove_numbers”表示**除去数字**，“remove_url”表示**除去网址**；“remove_separators”表示**除去分隔符**。

```
#去除标点、符号、数字、url网址和分隔符
review <- tokens(review,remove_punct = TRUE,
                        remove_symbols = TRUE,
                        remove_numbers = TRUE,
                        remove_url = TRUE,
                        remove_separators = TRUE)
```

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第六步 统一小写

为了使文本信息更为整齐、统计词频时结果更可靠，**可使用quanteda包中的tokens_tolower()函数将文本统一转换为小写。**

```
review <- tokens_tolower(review) #统一小写
review[1] #查看统一小写后的第一条评论
```

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第七步 去掉停用词、专有名词、低频词

■ 去掉停用词

使用quanteda包中的tokens_remove函数，并将第二个参数设置为**quanteda包里的stopwords**即可去掉停用词。还可以查看quanteda包里的英文停用词具体有哪些。

```
review1 <- tokens_remove(review,stopwords()) #去掉停用词
stopwords(language = "en") #查看英文的停用词表
```

i	me	my	myself	we	our	ours	ourselves	you	your
yours	yourself	yourselves	he	him	his	himself	she	her	hers
herself	it	its	itself	they	them	their	theirs	themselves	what
which	who	whom	this	that	these	those	am	is	are
was	were	be	been	being	have	has	had	having	do
does	did	doing	would	should	could	ought	i'm	you're	he's
she's	it's	we're	they're	i've	you've	we've	they've	i'd	you'd
he'd	she'd	we'd	they'd	i'll	you'll	he'll	she'll	we'll	they'll
isn't	aren't	wasn't	weren't	hasn't	haven't	hadn't	doesn't	don't	didn't
won't	wouldn't	shan't	shouldn't	can't	cannot	couldn't	mustn't	let's	that's

who's	what's	here's	there's	when's	where's	why's	how's	a	an
the	and	but	if	or	because	as	until	while	of
at	by	for	with	about	against	between	into	through	during
before	after	above	below	to	from	up	down	in	out
on	off	over	under	again	further	then	once	here	there
when	where	why	how	all	any	both	each	few	more
most	other	some	such	no	nor	not	only	own	same
so	than	too	very	will					

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第七步 去掉停用词、专有名词、低频词

■ 去掉出现频率小于k的词

在R中要除去频率较小的词需要两步：第一步是**计算词频并将词频较高的词记录到一个变量中**；第二步是**通过quanteda包中的tokens_keep()**函数来保留词频较高的词，去除词频较低的词。计算词频可以通过jiebaR包里的freq()函数实现。

需要注意的是，由于freq()函数需要传入的参数是词向量，对tokens类型不适用，因此**要先使用unlist()函数将评论数据转换成字符（即“character”）类型**。**tokens_keep()**函数可以在第一个参数中保留第二个参数中的内容并除去其它内容。

```
#以去除频率小于10的词为例
freq1 <-  freq(unlist(review))
#提取词频；由于review为tokens类型
#所以使用unlist()函数将其转化为字符类型
frwords <- freq1[which(freq1$freq>=10),]$char
#使用which提取出频率超过或等于10的词语和相应的词频
#提取其中的char变量即为频率超过或等于10的词语
review2 <- tokens_keep(review,frwords)
#保留频率高于等于10的词，去掉频率小于10的词
```

英文文本示例：R

中国人民大学 RENMIN UNIVERSITY OF CHINA

第八步 检查拼写

文本数据中可能存在拼写错误，可以使用**qdap**包里的**check_spelling**函数提取出拼写错误的单词并将其除去。

```
review <- review1 #已经去掉了停用词
chreview <- unlist(review) #将review转为字符类型
ck <- check_spelling(chreview) #检查拼写
review <- tokens_remove(review,ck$not.found) #去掉拼写错误的词
```

```
ck$not.found #查看可能拼写错误的词语，这里仅展示部分输出
```

```
 [1] "dra"           "pygmy"          "mc"
 [4] "mc"            "characters"     "mc"
 [7] "casette"       "humorously"     "superintelligences"
[10] "vilipending"   "d"              "reachable"
[13] "mc"            "pygmy"          "pygmy"
[16] "pyemias"       "multiservice"   "c"
[19] "mutineers"     "ultra"          "seamless"
[22] "langue"        "langue"         "billionaires"
[25] "kind"          "stroll"         "yeomanries"
[28] "strolls"       "favorite"       "mc"
[31] "penal"         "pygmy"          "sora"
[34] "pygmy"         "r"              "neon"
[37] "apologies"     "pygmy"          "atman"
[40] "concussed"     "clunk"          "mica"
[43] "wouldn't"      "usques"         "allot"
[46] "undersell"     "allot"          "haven't"
[49] "penal"         "scion"          "fib"
[52] "pygmy"         "penal"          "mc"
[55] "c"             "favorite"       "peas"
[58] "heartbeat"     "c"              "quippu"
[61] "etch"          "penal"          "pygmy"
```

```
ck$suggestion #查看可能拼写错误的词语的更正建议，这里仅展示部分输出
```

```
 [1] "dr"            "pym"            "mcu"
 [4] "mcu"           "charcters"      "mcu"
 [7] "castthe"       "humormusical"   "scoreeditingnegatives"
[10] "villainpacing" "dc"             "rewatchable"
[13] "mcu"           "pym"            "pym"
[16] "pym's"         "multiverse"     "cgi"
[19] "mutiverse"     "ultron"         "seamlessly"
[22] "lange"         "lange"          "billionaire"
[25] "kinda"         "stoll"          "yellowjacket"
[28] "stoll's"       "favourite"      "mcu"
[31] "pena"          "pym"            "sorta"
[34] "pym"           "rd"             "non"
[37] "apologise"     "pym"            "antman"
[40] "consused"      "clunky"         "mkcay"
[43] "wouldnt"       "usues"          "alot"
[46] "underwhelm"    "alot"           "havent"
[49] "pena"          "sci"            "fi"
[52] "pym"           "pena"           "mcu"
[55] "cgi"           "favourite"      "pena's"
[58] "heartfelt"     "cgi"            "quippy"
[61] "etc"           "pena"           "pym"
```

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第八步 检查拼写

例如，第五个拼写错误的词是“charcters”，改正的建议是“characters”。

对于拼写错误的词语，我们可以结合实际情况，选择用更正建议进行替换或直接去除拼写错误的词语。这里选择去掉拼写错误的词语。

```
review <- tokens_remove(review,ck$not.found) #去掉拼写错误的词
```

英文文本示例：R

第九步 词干化

在R中可以直接使用**quanteda**包中的**tokens_wordstem()**函数进行词干化。

```
reviewst <- tokens_wordstem(review)  #词干化
```

查看第一条评论词干化前后的结果对比如下（上方为词干化前，下方为词干化后）：
词干化前：text1 :
*[1] "ant" "man" "great" "heist" "film" "**disguise**"*
*[7] "superhero" "film" "great" "**performance**" "around" "**surprise**"*
[... and 7 more]

词干化后：text1 :
*[1] "ant" "man" "great" "heist" "film" "**disguis**" "superhero"*
*[8] "film" "great" "**perform**" "around" "**surpris**"*
[... and 7 more]

英文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第十步 N元模型

在R中，使用**tokens_ngrams()**函数并**设置第二个参数**可实现二元、三元语法分词。

```
bigram <- tokens_ngrams(review,n=2) #实现二元语法分词，n表示2元
bidfm <- dfm(bigram) #转换为dfm类型
topfeatures(bidfm,5) #展示词频最高的5个词
```

两元语法分词后词频最高的五个词。

```
          ant_man         paul_rudd  michael_douglas
               69                36               15
evangeline_lilly       scott_lang
                9                 9
```

二、中文文本预处理

01

中文文本预处理步骤

中文文本预处理步骤

中国人民大學
RENMIN UNIVERSITY OF CHINA

数据清洗　文本分词　去停用词　统计词频　画词云图　词性分析

数据清洗

- 原始的文本数据中往往包含很多标点符号和数字，尤其当文本数据来源于网络时，数据中通常会混杂很多的网页标记符号，因此需要事先对这些内容进行清洗，**只保留其中的汉字和字母（有时也包括数字）等具有实际意义的内容**。

文本分词概述

中国人民大學
RENMIN UNIVERSITY OF CHINA

- 中文分词是中文文本处理的一个基础步骤，是将连续的字序列按照一定的规范重新组合成词序列的过程。
- 不同于英文分词的是，**中文句子中没有明显的分隔符，而英文的单词与单词之间有自然的空格符来分隔**。因此在进行中文自然语言处理时，通常需要先进行分词。

常见的分词算法：概述

中国人民大學
RENMIN UNIVERSITY OF CHINA

- 常见的分词算法一般都是使用机器学习算法和词典算法相结合。这样一方面能够提高分词准确率，另一方面能够改善领域适应性。
- 常见的中文分词工具有：**jieba、SnowNLP（MIT）、pynlpir（大数据搜索挖掘实验室）、thulac（清华大学自然语言处理与社会人文计算实验室）等**。其中最常用的是jieba分词工具。

去停用词

中国人民大学
RENMIN UNIVERSITY OF CHINA

- 停用词是分词过程中**不具有分析意义且经常出现的介词连词**等，比如：我、你、的、是等。这些词在统计词频的时候意义不大，且会增加噪音，因此需要在分析前先将这些词进行过滤，只保留具有实际分析意义的词。
- 停用词大致可以分为两类：
  - ✓ 语言中使用的功能词（语气助词、副词、介词、连接词），这些功能词通常极其普遍，并没有什么实际含义，比如：“你、我、他、了、的”
  - ✓ 有特定含义，但是应用十分广泛的词，比如“想、做、来、去”等

去停用词

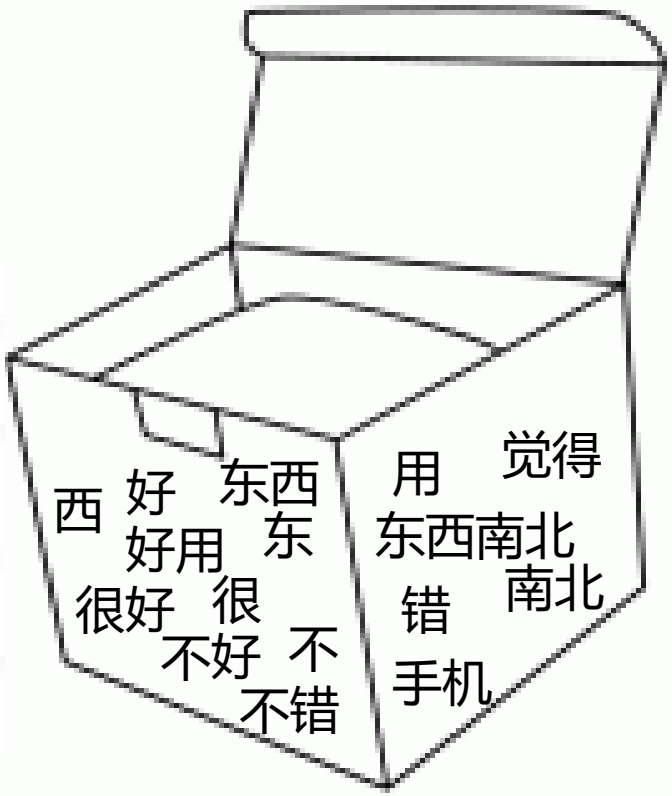
中国人民大学
RENMIN UNIVERSITY OF CHINA

- 对停用词的总结
  - ✓ 很多机构都给出了自己的停用词表，如“百度”，“哈工大”，“四川大学机器自然实验室”等
  - ✓ 汇总结果： stopwords.dat
- 值得注意的是，停用词的定义并没有统一确定的标准。在实际进行数据分析时，研究者可以根据处理文本的实际情况定义自己的停用词词典。例如，对手机评论数据进行分析，“手机”这个词会频繁出现，但在该分析场景中意义不大，因此也可以考虑将“手机”作为一个停用词去掉。

自定义字典

- 绝大多数分词方法都是基于“字典”

东西|不错|很|好|用



- 通过添加用户自定义字典，丰富原始字典

自定义字典

- 从原始分词结果中总结
- 有自己领域的行业词汇
- 搜狗细胞词库

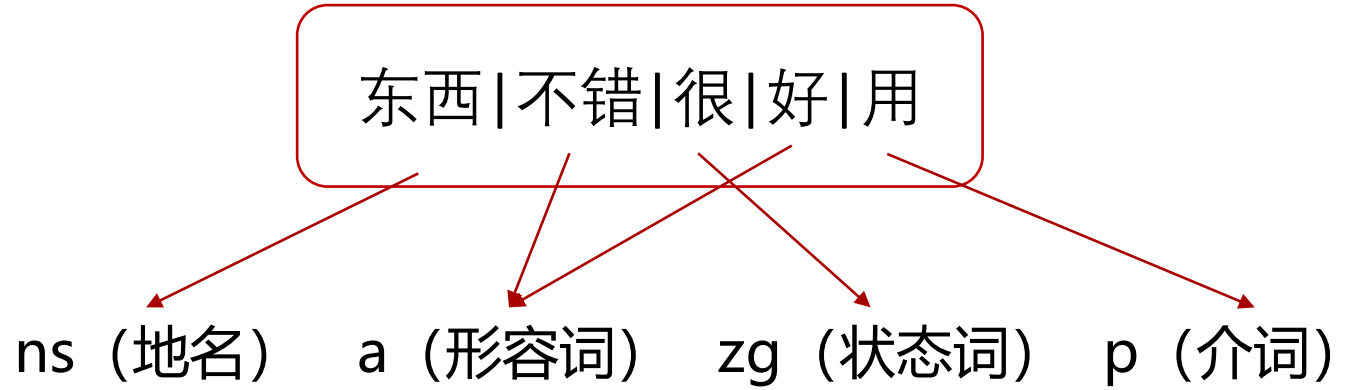
词性分析

中国人民大学
RENMIN UNIVERSITY OF CHINA

- 在文本分析时有时会利用到词语的词性。词语的词性包括：名词、动词、形容词、介词、连词、数词等等。在某些研究问题中，可能会只关注某些特定词性的词语。例如，需要了解最近热门的行业，由于行业通常都是名词，因此可以只筛选名词进行分析。
- 常用的分词算法在对文本进行分词时往往会给出词语的词性。**例如，jieba分词方法对词性进行了非常详细的分类，其主要参照中科院ICTCLAS 汉语词性标注方法。在使用jieba分词后就可以输出各个中文词语的词性。**

词性分析

- Jieba分词可以给出每个分词结果的词性
  - Jieba词性标注表.txt



- 对分词结果按词性进行筛选
  - 假设我们只关注“名词、动词、形容词、副词”

统计词频

- 文本分词之后，可以计算各个词语在整个文本数据集中出现的频数。
- 统计所有词语的频数，进行降序排列，从中**选出频数最高的词语**，这些词语就代表了整个文本数据集中被提及最多的词语，因此可以在一定程度上代表该文本数据集的主要含义，透过这些高频词可以对文本数据集进行**宏观的大致了解。**

TF-IDF（词频-逆向文件频率）

中国人民大学
RENMIN UNIVERSITY OF CHINA

- TF(Term Frequency)指的是某一个给定的词语在该文件中出现的次数
- IDF (inverse document frequency) 是一个词语普遍重要性的度量。某一特定词语的IDF，可以由总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到。
- TF-IDF = TF * IDF

TF-IDF（词频-逆向文件频率）

[1] 还|不错|吧
[2] 东西|不错|很|好|用
[3] 用|着|还|不错
[4] 物流|很|给力
[5] 质量|不错
[6] 东西|很|好|售后|也|非常|给力|哦

- 6个文件中出现的总词数：3+5+5+3+2+8=26
- “不错”出现的总次数=4
- 词频TF=4/26=0.154
- “不错”出现的文件个数=4
- 逆文件频率 IDF=log(6/4)=0.405
- TF-IDF=TF*IDF=0.06237

画词云图

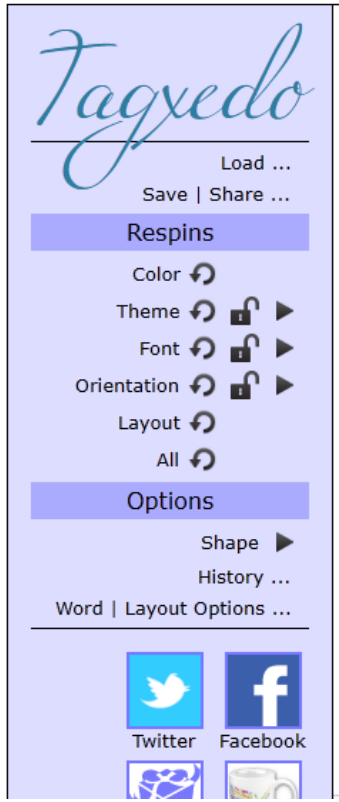
- **词云图是一种常用的展示文本主要信息的可视化方法。**通常可以将文本中出现频率较高的词绘制成词云图，每个词的大小与其出现频率正相关，从而使读者快速领略文本的主旨。

画词云图

- 绘制词云图有许多工具。在R和Python中都有相应的软件包。此外，也有很多词云的在线编辑和展示工具，例如Wordle(wordle.net/)、Tagxedo(tagxedo.com/)、ToCloud(tocloud.com/)、图悦(picdata.cn/)、BDP个人版(me.bdp.cn/home.html)等。在这些工具中，使用者可以通过设置各种要求绘制出需要的词云图样式，从而使词云图更加美观。

画词云图

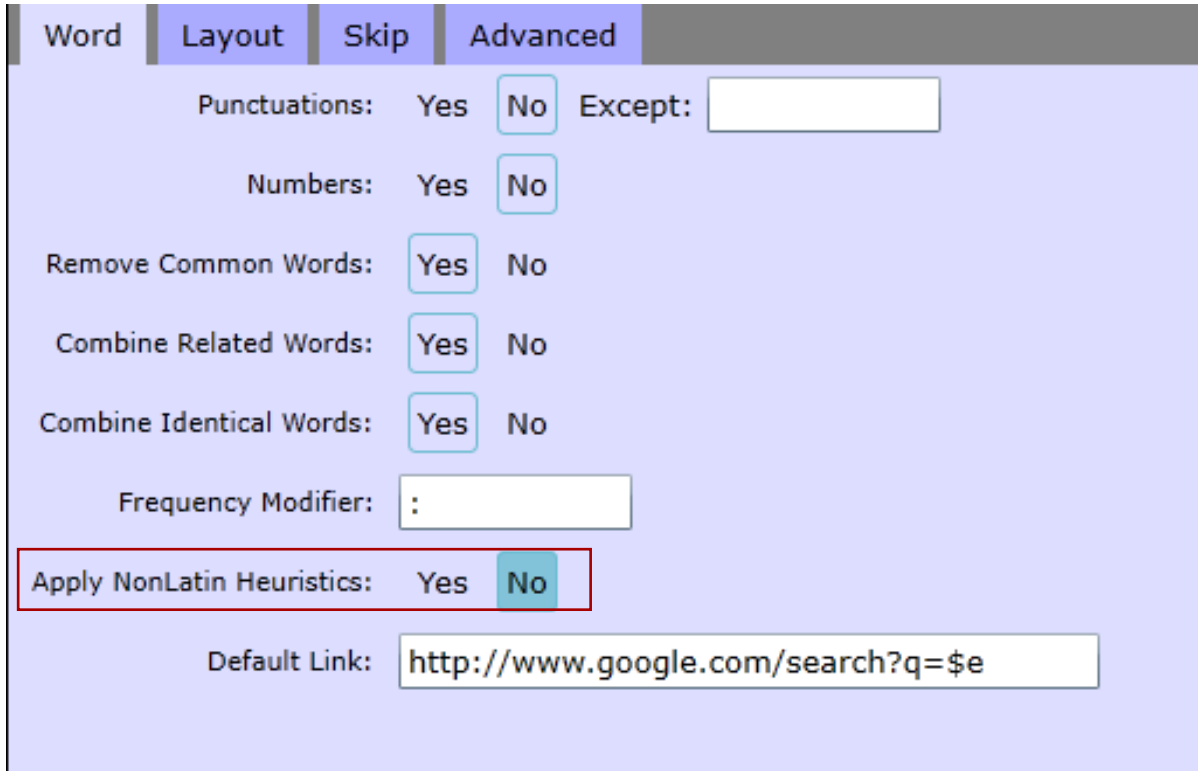
- Tagxedo：一个专门绘制词云图的网站
  - http://www.tagxedo.com/app.html
- 常用选项
  - Load:输入要画词云的数据
  - Save|share：保存词云图
  - Color: 随机变化颜色
  - Theme：变化主题色系
  - Font：变化字体
  - Shape：选择词云形状



画词云图

中國人民大學
RENMIN UNIVERSITY OF CHINA

- 中文使用时的特殊处理
  - Word | Layout Options – Word- Apply NoLatin Heuristics - No



02

中文文本预处理示例

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第一步 前期准备——安装R包

在R中进行中文分词主要使用**jiebaR**包进行。此外，还需要使用**stringr**包进行正则表达式操作（字符匹配、字符替换等）；使用**wordcloud2**包进行词云图的绘制。

```
# 加载需要的R包
library("jiebaR")
library("stringr")
library("wordcloud2")
```

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第一步 前期准备——处理文本中的数字

在中文分词前，我们需要对文本数据进行预处理，用**stringr包中的str_replace_all( )**函数将文本中所有【数字】匹配出来并全部替换为“”(空值)

```
contents <- str_replace_all(contents,"[0-9]","")  #将所有数字替换为空字符串
```

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第一步 前期准备——自定义词典准备

绝大多数分词方法都是基于“词典”，R中有着自己的默认词典，可以满足绝大多数的分词需求，但对于一些专业领域的特有词汇无法正确切分。例如，本文中所使用到新闻文本中出现“汤臣倍健”一词，会被分词为“汤臣”、“倍”、“健”三个词，这显然不是我们想要得到的，因此需要添加自定义词典以满足我们的分词需求。添加自定义词库有三种方法：

（1）使用new_user_word函数

使用new_user_word( )函数将需要保留的词语直接添加至词库中。这种方法在处理短文本是是比较方便的，但是一旦遇到长文本，需要添加至词库的词多了，就不适用了。

```
work_new_word<-worker()   #初始化分词器
user_word(work_new_word, c("汤臣倍健"))   #将“汤臣倍健”一词添加至词库
```

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第一步 前期准备——自定义词典准备

（2）从原始分词结果中总结并自己新建词库

自定义一个新词库，然后从词库里面直接读取新词进行分词。利用Notepad++编写一个名为ciku.txt的文档。这里需要注意，如果你的词库是用记事本写的话，因为编码有时不是UTF-8，使用时会出现各种错误，甚至软件奔溃。因此，建议使用notepad++编辑，将编码设置为UTF-8，再另存为txt文件。然后在ciku.txt文档中逐行添加词语，从而形成新词库。这种方法的好处是可以精确分词，但缺点也很明显，就是在处理长文本时费时费力，不太适用。

```
dictpath="ciku.txt" #指明词库的路径
work_user<-worker(user="ciku.txt")  #调用新的词库
```

（3）借用搜狗细胞词库

将搜狗细胞词库中下载的词典放到R的分词库里实现调用。示例代码如下：

```
dictpath="D:/沪深A股上市公司名单.txt"  #从搜狗细胞词库中下载得到沪深A股上市公司名单
cutter = worker(bylines = TRUE,user = dictpath)  #放到R的分词库中调用
```

中文文本示例：R

中国人民大學
RENMIN UNIVERSITY OF CHINA

第一步 前期准备——停用词准备

在jiebaR中，过滤停用词有2种方法，一种是通过配置stop_word文件，另一种是使用filter_segment()函数。

（1）配置stopwords文件

编写一个名为stopwords.txt的文档，里面包含“的”、“和”等不需要作为结果输出的词。利用worker()函数调用。

```
stoppath="D:/stopwords.txt"  #包含停用词的文档
cutter = worker(bylines = TRUE,stop_word=stoppath)  #调用分词器进行逐行分词
```

（2）filter_segment()函数

在上面的过滤基础上，利用filter_segment()继续过滤掉“年”、“月”、“日”等一些在文本中出现频率很高，但对文本分析意义不大的词。

```
filter<-c("年","月","日")  #定义需要过滤掉的词
res_stop<-filter_segment(res_stop,filter)  #去除“年”、“月”、“日”等词
```

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第二步 jieba中文分词——初始化分词器

在导入自定义词典和停用词词典后，用到worker( )函数初始化分词引擎并结合上述步骤进行分词。

```
cutter = worker(bylines = TRUE,stop_word=stoppath,user = dictpath)#初始化分词器
res_stop = cutter[contents]#分词
filter<-c("年","月","日")#过滤年、月、日等在文本中出现频率很高，但对文本分析意义不大的词。
res_stop<-filter_segment(res_stop,filter)
head(res_stop,1)#展示第一篇文章分词结果
```

第一篇文章的原文（部分）如下：

3月5日,汤臣倍健(300146,股吧)发布了2020年度业绩报告,公司实现营业收入60.95亿元,较去年同期增长15.83%,实现归母净利润15.24亿元,同比增长528.29%。上市10 年以来,其销售收入和归母净利润则分别增长了超过16倍和15倍。

去掉数字和标点符号后，相应的分词结果如下：

"月" "日" "汤臣倍健" "股" "发布" "年度" "业绩" "报告" "公司" "营业" "收入" "亿元" "去年同期" "增长" "归母" "净利润" "亿元" "同比" "增长" "上市" "年" "销售收入" "归母" "净利润" "增长" "超过" "倍" "倍"

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第二步 jieba中文分词——词频统计

先用unlist( )函数将list结构数据变成非list结构数据，再用table()函数统计各个词语出现的次数，并用as.data.frame()将统计结果转化为数据框结构；用order()函数使词频按照降序排列，并展示前50个高频词。

```
freq = as.data.frame(table(unlist(res_stop)),
                     stringsAsFactors = F)#进行词频统计，并以转为数据框
freq = freq[order(freq[,2],decreasing = TRUE),]#按词频个数降序排列
top50 = freq[1:50,] #挑选前50个高频词
colnames(top50)=c('text','Freq')
top50#展示前50个高频词
```

中文文本示例：R

第二步 jieba中文分词——词频统计

其中，前20个高频词输出结果如下图所示：

	text	Freq
86006	年	48460
46971	公司	45317
139324	月	36528
103786	市场	27640
39553	发展	24420
90933	企业	22676
147262	中国	21024
96355	日	20861
133646	亿元	19423
125172	新	18302
47964	股	17289
141165	增长	15857
113552	投资	15309
23289	产品	15261
65677	经济	14543
126380	行业	14219
42762	服务	13437
105931	数据	12893
58358	基金	12835
69569	科技	12261

中文文本示例：R

词云图绘制主要是基于R中的**wordcloud**包或**wordcloud2**包。最简单的是通过worldcloud2()函数，以上面统计出来的词频top50作为参数绘制词云图

词云图中，每个词语的大小与其词频成正比。例如“公司”一词的面积较大，说明“公司”的词频较高，此外还有“市场”“中国”等词。

```
wordcloud2(top50)   #用频率最高的前50个词绘制词云图
```

中文文本示例：R

第二步 jieba中文分词——绘制词云图

另外，还可以通过设置其他参数改变词云图的形状字体颜色等。常用参数包括：

- **size**：字体大小；
- **fontFamily**：字体；
- **color**：字体颜色色系；
- **shape**：词云形状选择，默认是‘circle’，即圆形。还可以选择‘cardioid’（苹果形或心形），‘star’（星形），‘diamond’（钻石），‘triangle-forward’（三角形），‘triangle’（三角形），‘pentagon’（五边形）等等；

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第二步 jieba中文分词——词性提取

要提取词性，需要构筑词性标注环境。与之前的分词原理类似，在词性提取时也要用到初始化分词器worker()函数，构筑词性标注环境。同时在这里使用到了dplyr包和tibble包中的enframe()函数辅助处理数据。

```
# 加载包
library(dplyr)
library(tibble)
seg <- worker('tag',stop_word=stoppath,user=dictpath)#构筑词性标注环境
seg_contents = segment(contents,seg) #对所有的新闻词性标注
seg_table <- enframe(seg_contents)#将数据存储为tibble数据框,转换为具有名称和值的数据框
head(seg_table,15)#显示前15个词的词性
```

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第二步 jieba中文分词——词性提取

前15个词的词性输出结果如下图所示：第一列为词性，第二列为具体的词，其中：m表示数词，x表示非语素词，q表示量词，v表示动词，n表示名词。

```
# A tibble: 15 x 2
   name  value
   <chr> <chr>
 1 m     月
 2 m     日
 3 x     汤臣倍健
 4 q     股
 5 v     发布
 6 n     年度
 7 n     业绩
 8 n     报告
 9 n     公司
10 n     营业
11 v     收入
12 m     亿元
13 n     去年同期
14 v     增长
15 x     归母
```

中文文本示例：R

中国人民大学
RENMIN UNIVERSITY OF CHINA

第二步 jieba中文分词——词性提取

为进一步分析文本数据中的内容，还可以用filter函数筛选不同词性的词语进行进一步分析。此处以名词n为例筛选，得到前15个名词的输出结果。也可以筛选其他词性的词语。

```
n_seg <- seg_table %>%
  select(name,value)%>%  #保留词性和词语
  filter(name == "n")  #筛选其中的名词，n表示名词
head(n_seg,15)#展示前十五个名词
```

```
# A tibble: 15 x 2
   name  value
   <chr> <chr>
 1 n     年度
 2 n     业绩
 3 n     报告
 4 n     公司
 5 n     营业
 6 n     去年同期
 7 n     净利润
 8 n     销售收入
 9 n     净利润
10 n     数据
11 n     膳食
12 n     补充剂
13 n     行业
14 n     市场份额
15 n     疫情
```

END