文本分词（HMM）、词云

覃雄派

文本分词（HMM）、词云

提纲

- 问题背景、分词意义
- 分词的困难
- 分词方法
  - 规则分词
  - 统计分词
- HMM模型与维特比算法
  - 维特比算法效率分析
  - 在分词中采用HMM模型
- 分词的实践

文本分词（HMM）、词云

- 文本分词：中文文本特点
- 英文（以及一些国家/地区语言文字）词与词之间有**空格（分隔符）**，分词处理相对容易
  - 例如：This is a book.
- 中文基于单字，中文书面表达方式以汉字作为最小单位的
  - 字与字之间、词与词之间**紧密连接**，词与词之间没有显性的界限标志
  - 词是**最小并且能独立活动**的语言成分，文章以词为基本单位来形成有意义的篇章
  - 添加合适的显性的词语边界标志使得所形成的词串反映句子的本意
  - 所以分词是汉语文本分析处理中首先要解决的问题

文本分词（HMM）、词云

- 文本分析任务之词法分析
  - 词法分析是将构成句子的**字符序列**转换为**词的序列**，并对每个词加上语法或语义标记
    - **分词**：对句子进行分词，完成该功能的软件称为分词器(Tokenizer)
    - **词性标注**： Part-of-Speech Tagger，（简称POS Tagger）分析某种语言的文本，然后针对每个词(Word或者Token)赋予POS标记，比如名词(Noun)、动词(Verb)、形容词(Adjective)等
    - **命名实体识别**
    - **词义消歧**

文本分词（HMM）、词云

- 分词的意义
- 正确的机器自动分词是正确的中文信息处理的基础
  - 文本检索
    - 和服 | 务 | 于三日后裁制完毕，并呈送将军府中。
    - 王府饭店的设施 | 和 | 服务 | 是一流的。

    如果不分词或者“和服务”分词有误，都会导致荒谬的检索结果

  - 文语转换
    - 他们是来 | 查 | 金泰 | 撞人那件事的。(“查”读音为cha）
    - 行侠仗义的 | 查金泰 | 远近闻名。（“查”读音为zha，姓氏)

文本分词（HMM）、词云

文本分词（HMM）、词云

- 中文分词面临的主要难题
  - 如何面向大规模开放应用是汉语分词研究亟待解决的挑战
    - 如何识别未登录词
    - 如何利用语言学知识
    - 词语边界歧义处理
    - 实时性应用中的效率问题

南京市　长江大桥

还好我一把　把把(四声)　把住了

我也想过　过儿　过过　的生活

校长说衣服上除了校徽　别别（四声）　别的

文本分词（HMM）、词云

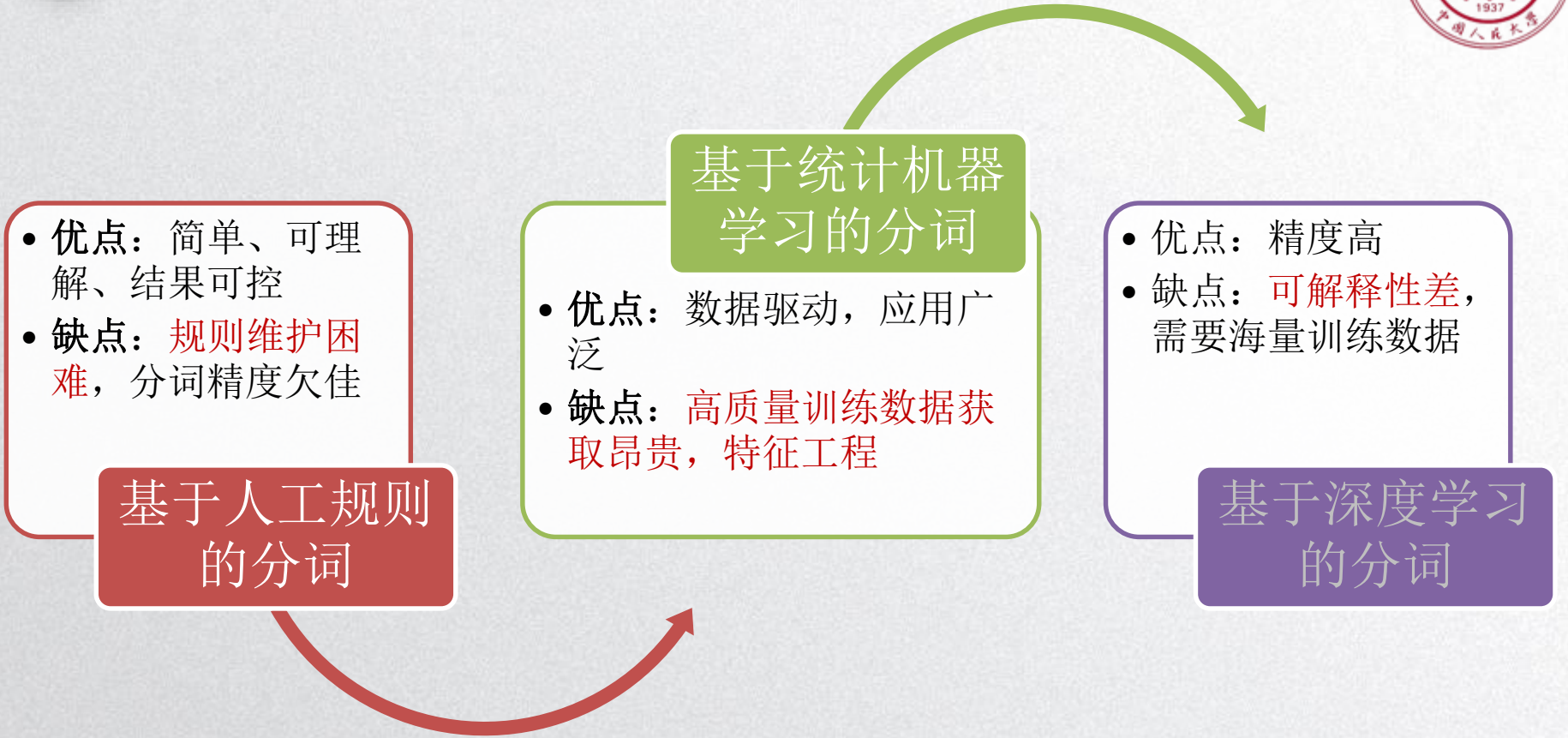
- 未登录词
  - 虽然一般的词典都能覆盖大多数的词语，但有相当一部分的词语不可能穷尽地收入系统词典中，这些词语称为未登录词
  - 分类：
    - 专有名词：中文人名、地名、机构名称、外国译名、时间词
    - 重叠词：“高高兴兴”、“研究研究”
    - 派生词：“一次性用品”
    - 与领域相关的术语：“互联网”

文本分词（HMM）、词云

- 分词歧义
  - 交集型切分歧义
    - 汉字串AJB被称作交集型切分歧义，如果满足**AJ、JB同时为词**(A、J、B分别为汉字串)。此时汉字串J被称作交集串。
      - [例] “结合成分子” 结合、合成
        - » 结合 | 成分|子 |
        - » 结合|成|分子|
        - » 结 | 合成 |分子|
      - [例] “美国会通过对台售武法案” 美国、国会
      - [例] “乒乓球拍卖完了” 球拍、拍卖
  - 组合型切分歧义
    - 汉字串AB被称作组合型切分歧义，如果满足条件：A、B、**AB同时为词**
      - [例]组合型切分歧义： “起身”
      - 他站 | 起 | 身 | 来。
      - 他明天 | 起身 | 去北京

文本分词（HMM）、词云

分词方法


基于人工规则的分词
• 优点：简单、可理解、结果可控
• 缺点：规则维护困难，分词精度欠佳
基于统计机器学习的分词
• 优点：数据驱动，应用广泛
• 缺点：高质量训练数据获取昂贵，特征工程
基于深度学习的分词
• 优点：精度高
• 缺点：可解释性差，需要海量训练数据

文本分词（HMM）、词云

- 分词方法：正向最大匹配分词(Forward Maximum Matching method, FMM)
- 基本思想：
  - 1.设自动分词词典中**最长词条所含汉字个数为I**；
  - 2.取被处理句子当前字符串序数中的I个字作为匹配字段，查找分词词典
    - 若词典中有这样的一个I字词，则匹配成功，匹配字段作为一个词被切分出来，转6；
  - 3.如果词典中找不到这样的一个I字词，则匹配失败；
  - 4.匹配字段去掉最后一个汉字，I--；
  - 5.重复2-4，直至切分成功为止；
  - 6.I重新赋初值，转2，直到切分出所有词为止

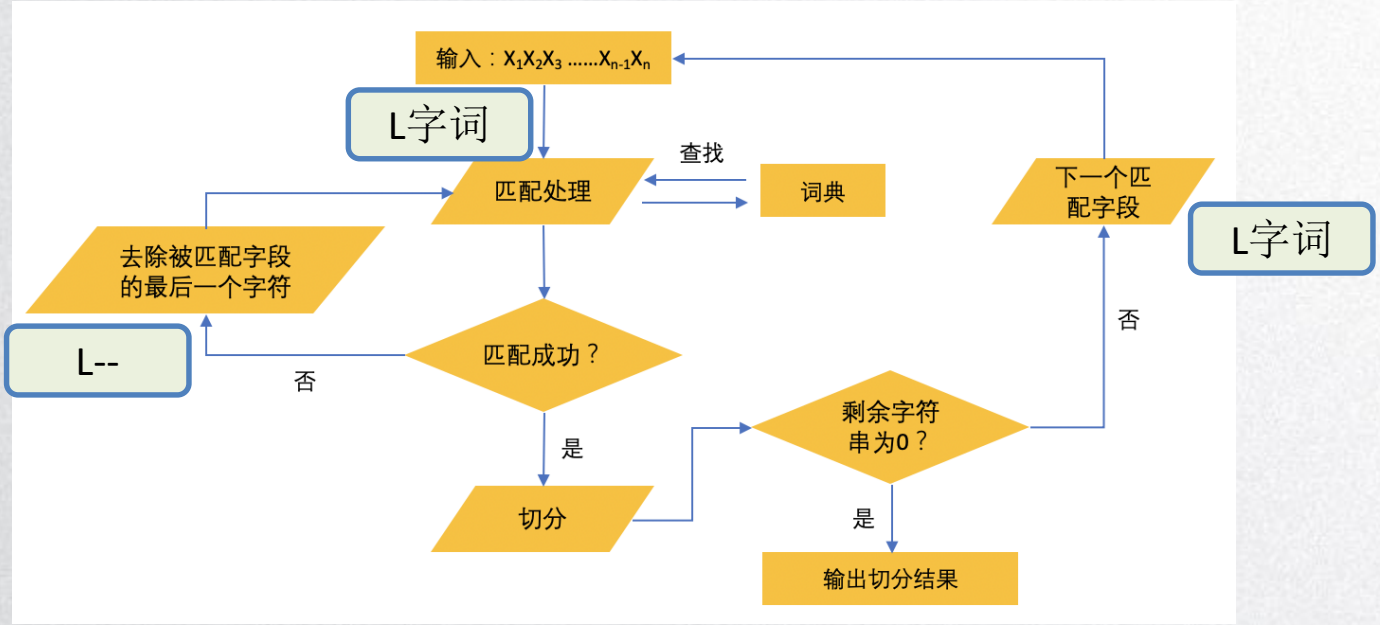
例句：南京市长江大桥

词典：{南京，市，市长，长江，大桥，江，长江大桥}

A. 南京/市/长江/大桥
B. 南京/市长/江/大桥
C. 南京/市/长江大桥

文本分词（HMM）、词云

- 分词方法：正向最大匹配分词(Forward Maximum Matching method, FMM)



文本分词（HMM）、词云

- 分词方法：正向最大匹配存在的问题
  - **维护词典困难**
    - 新词层出不穷，人工维护费时费力
    - 不能保证词典能很好地覆盖到所有可能出现的词
    - 信息爆炸的时代每天新词出现的速度，使得人工维护词典更加困难
  - **执行效率底下**
    - 为了能找到一个合适的窗口，会循环往复的进行下去直到找到一个合适的匹配
    - 在**词典非常大、初始窗口也大**的情况下，匹配词段寻找的时间和循环次数会相应增加
  - **歧义问题**
    - 假设最长词长度为5的词典，词典中有“南京市长”和“长江大桥”两个词语
    - “南京市长江大桥”通过上述正向最大匹配算法进行切分
      - 首先通过对前五个字符进行匹配，发现没有符合合适的，那么此时就会去掉最后一个汉字，变成前面四个汉字进行匹配，发现匹配到了“南京市长”
      - 用剩下的“江大桥”继续匹配，可能得到的结果是“江”和“大桥”这两个词语
      - 分词结果：南京市长 / 江 / 大桥

文本分词（HMM）、词云

- 分词方法：基于统计机器学习的方法
  - 机器学习：研究一类算法，使之
    - 在某些任务上(task)
    - 通过已有的观测经验(数据)(experience)
    - 提升算法效果(performance)
  - 两个过程
    - 离线训练：基于标注数据，发现规则（确定模型参数）
    - 在线预测：基于已发现的规则，对新数据进行预测（如：标注）
  - 人工规则 → 从标注数据中自动发现规则

文本分词（HMM）、词云

- 分词方法：基于统计机器学习的分词流程

市场/中/国有/企业/才能/发展

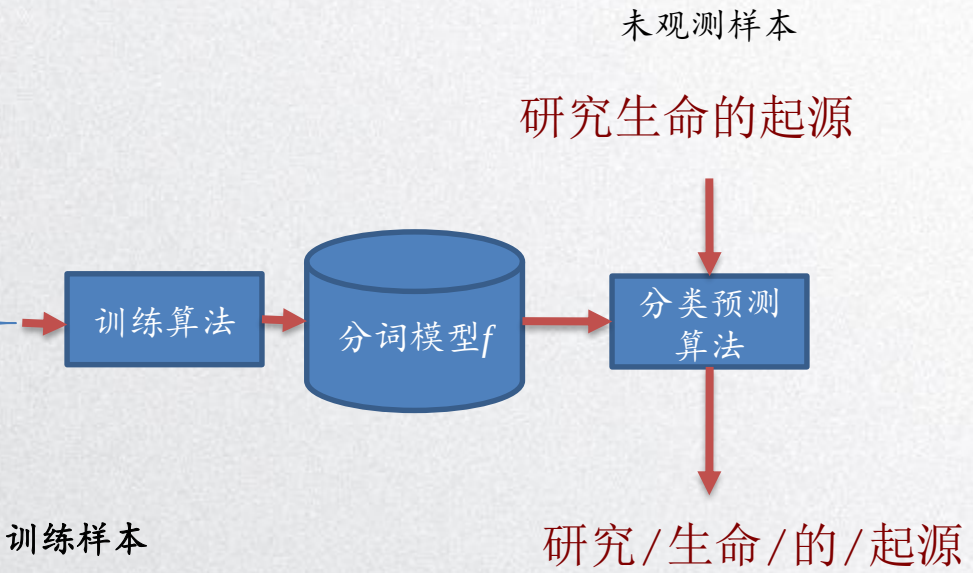
吃/两/顿/饭

跳/新疆/舞

……

时间/就/是/生命/

失败/是/成功/之/母



文本分词（HMM）、词云

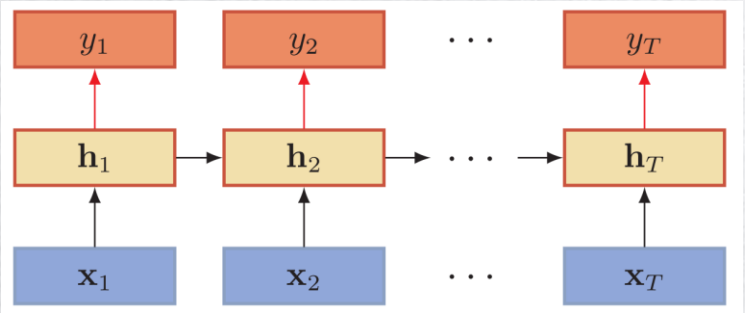
- 分词方法：问题建模→序列标注
- 为每一个字打上一个标签{B, I, E, S}
  - B：这个字是某个词的开头
  - I：这个字是某个词的中间部分
  - E：这个字是某个词的结尾
  - S：这个字单独成词

输入X：市 场 中 国 有 企 业 才 能 发 展
标签Y：B E S B E B E B E B E
结果 ：市场/中/国有/企业/才能/发展

输入X：南 京 市 长 江 大 桥
标签Y： B I E B E B E
结果 ：南京市/长江/大桥

文本分词（HMM）、词云

- 深度学习方法逐渐替代传统方法
  - 2014年以前
    - 特征工程问题
    - 用分布式语义表示来代替传统的离散特征
  - 2015年
    - 长距离依赖问题
    - 循环神经网络
    - 循环神经网络与条件随机场的结合，e.g., BiLSTM + CRF



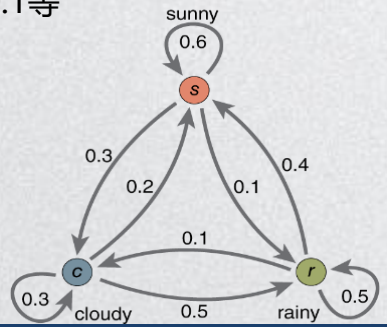
文本分词（HMM）、词云

- 问题依旧
  - 未登录词问题
  - 语言学知识的融入
  - 语义理解问题
    - 涉及语义理解的歧义情况，仍然无法解决
  - 分词标准问题
    - 标准差异
    - 粒度差异
  - 实时性应用中的效率问题
  - 评价问题

文本分词（HMM）、词云

文本分词（HMM）、词云

- HMM模型与维特比算法
  - 马尔可夫模型，是通过马尔可夫链进行建模的一种状态空间模型
    - 马尔可夫链服从马尔可夫性质，也就是没有长期记忆性
    - 换句话说，某一个时刻的状态，受到而且只受到前一时刻状态的影响，不受更往前的时刻的状态的影响
  - 图中给出了一个简单的天气模型
    - 在这个模型中，存在三种状态，包括Sunny、Rainy和Cloudy等
    - 图上还给出了各个状态之间的转移概率
      - 比如当前状态为Sunny，那么下一个状态为Sunny的概率为0.6，为Cloudy的概率为0.3，为Rainy的概率为0.1等



文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型
  - 我们通过一个实例来介绍隐马尔可夫模型
    - 1.背景情况
    - 从前有个村子，村民的身体状况有两种可能：健康或者发烧
      - 假设这个村子里的人没有其他检测设备比如温度计等，村民判断身体状况的唯一办法，就是到小诊所里询问那里的一位大夫，大夫的名字叫做月儿
    - 月儿通过望闻问切诊断病情，村民只需回答正常、头晕或者感觉冷就可以了
      - 有一位村民去诊所询问身体状况，第一天他说感觉正常，第二天他说感觉冷，第三天他说感觉头晕。现在的问题是，月儿如何根据这位村民的描述推断这三天中他的身体状况呢?

文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 2.已知条件
    - 现在月儿已经知道的情况如下：
      - 隐含的身体状态集合={健康，发烧}
      - 可以观察的感觉集合={正常，冷，头晕}
    - 月儿预判的村民的身体状况的概率分布={健康：0.6，发烧：0.4}。
      - 月儿还掌握了村民身体健康状态的转移概率={健康→健康：0.7，健康→发烧：0.3，发烧→健康：0.4，发烧→发烧：0.6}

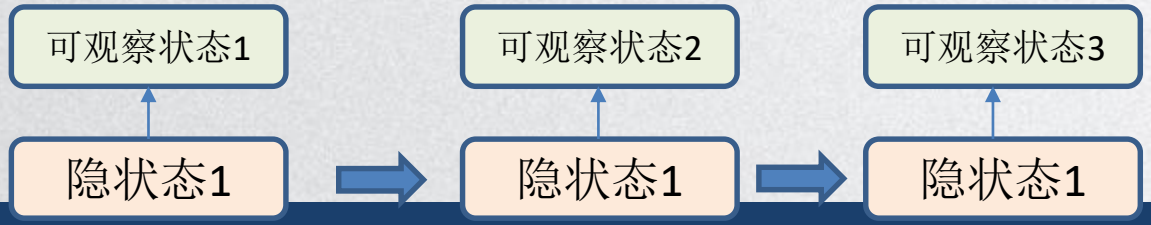
先验分布

昨天 \ 今天	健康	发烧
健康	0.7	0.3
发烧	0.4	0.6

文本分词（HMM）、词云

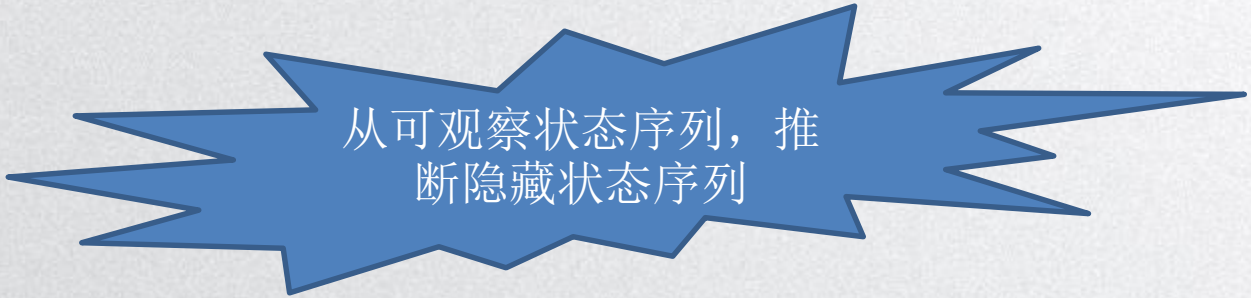
- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 2.已知条件
    - 此外，月儿认为
    - 在相应的健康状况下，村民的**感觉**的概率分布={健康情况下，正常：0.5，冷：0.4，头晕：0.1；发烧情况下，正常：0.1，冷：0.3，头晕：0.6}。列表如下

隐藏状态\感觉	正常	冷	头晕
健康	0.5	0.4	0.1
发烧	0.1	0.3	0.6



文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 3.问题描述
    - 现在，连续三天村民报告给月儿的感觉依次是正常、冷和头晕。
    - 如何根据这位村民的描述，推断这三天中他的身体健康状况的变化过程呢?



文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 4.问题的解决
    - 解决这个问题，就是要找出产生上述显式的感觉序列的隐藏的身体状态序列
      - 可以用Viterbi算法来解决
      - 根据Viterbi理论，后一天的状态仅依赖于前一天的状态和当前的可观察的状态
      - 那么，只要根据第一天的正常状态，依次推算，找出到达第三天头晕状态的最大的概率，就可以知道这三天的身体变化情况

文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 4.问题的解决
    - (a) 初始情况为：P(健康)=0.6，P(发烧)=0.4。
    - 求第一天的身体状况，计算该村民在感觉正常的情况下，最可能的身体状态
    - P(今天健康)=P(正常|健康)×P(健康|初始情况) = 0.5 × 0.6 = **0.3**
    - P(今天发烧)=P(正常|发烧)×P(发烧|初始情况) = 0.1 × 0.4 = 0.04
    - P(今天健康)更大，于是可以认为第一天最可能的身体状态是：**健康**

文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 4.问题的解决
    - (b) 接着，求第二天的身体状况，计算该村民在感觉冷的情况下，最可能的身体状态。第二天有四种情况，由第一天的发烧或者健康转换到第二天的发烧或者健康
    - P(前一天发烧，今天发烧) = P(发烧|前一天)×P(发烧→发烧)×P(冷|发烧) = 0.04 × 0.6 × 0.3 = 0.0072;
    - P(前一天发烧，今天健康) = P(发烧|前一天)×P(发烧→健康)×P(冷|健康) = 0.04 × 0.4 × 0.4 = 0.0064;
    - P(前一天健康，今天发烧) = P(健康|前一天)×P(健康→发烧)×P(冷|发烧) = 0.3 × 0.3 ×0.3 = 0.027。
    - P(前一天健康，今天健康) = P(健康|前一天)×P(健康→健康)×P(冷|健康) = 0.3 × 0.7 × 0.4 = **0.084**;
    - P(前一天健康，今天健康)最大，于是可以认为第二天最可能的状态是：**健康**

文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 4.问题的解决
    - (c) 最后，求第三天的身体状态，计算该村民在感觉头晕的情况下，最可能的身体状态
    - P(前一天发烧，今天发烧) = P(发烧|前一天)×P(发烧→发烧)×P(头晕|发烧) = 0.027 × 0.6 × 0.6 = 0.009 72;
    - P(前一天发烧，今天健康) = P(发烧|前一天)×P(发烧→健康)×P(头晕|健康) = 0.027 × 0.4 × 0.1 = 0.001 08;
    - P(前一天健康，今天发烧) = P(健康|前一天)×P(健康→发烧)×P(头晕|发烧) = 0.084 × 0.3 ×0.6 = **0.015 12。**
    - P(前一天健康，今天健康) = P(健康|前一天)×P(健康→健康)×P(头晕|健康) = 0.084 × 0.7 × 0.1 = 0.005 88;
    - P(前一天健康，今天发烧)最大，于是可以认为第三天最可能的状态是：**发烧**

文本分词（HMM）、词云

- HMM模型与维特比算法
  - 隐马尔可夫模型(Hidden Markov Model, HMM)是包含隐藏状态的马尔可夫模型。我们通过一个实例来介绍隐马尔可夫模型
    - 4.问题的解决
    - (5) 结论
    - 根据上述推导过程，月儿得出结论，这位村民这三天的身体健康状况分别是健康、健康、发烧

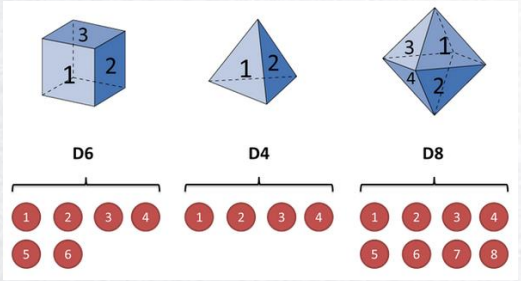
文本分词（HMM）、词云

文本分词（HMM）、词云

- 维特比算法效率分析
- 隐状态模型
  - 骰子D6：每个面（1，2，3，4，5，6）出现的概率是1/6
  - 骰子D4：每个面（1，2，3，4）出现的概率是1/4
  - 骰子D8：每个面（1，2，3，4，5，6，7，8）出现的概率是1/8

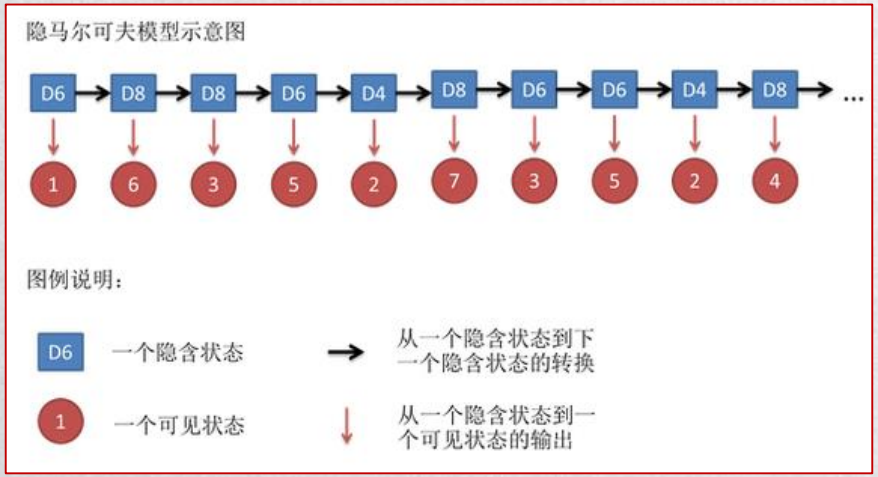
  - **采样**：先从三个骰子里随机挑一个，挑到每一个骰子的概率都是1/3；然后掷骰子，得到一个数字（1，2，3，4，5，6，7，8）中的一个。重复上述过程，得到一串数字：1 6 3 5 2 7 3 5 2 4
  - **问题**：每一次都掷哪个骰子？即**破解骰子序列**

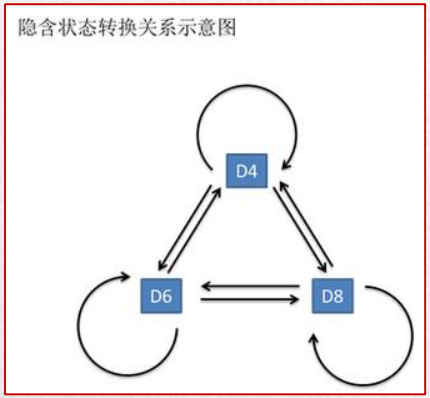
  - 观测：数字串1 6 3 5 2 7 3 5 2 4
  - 隐状态：每一次选到的骰子（编号为D4、D6或D8）



文本分词（HMM）、词云

- 维特比算法效率分析
- 加上限制→隐马尔科夫模型
  - 更加复杂一点：下一次选的骰子和上次选的骰子有关联
    - 隐状态之间的状态转换



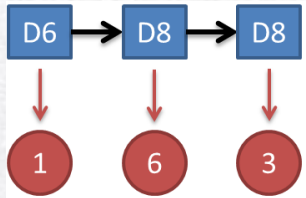


文本分词（HMM）、词云

- 维特比算法效率分析
- **观测**：1 6 3
- **求解**：掷骰子序列是什么？（破解骰子序列）
  - 换一个问题：假设掷骰子的序列为D6D8D8，出现观测序列1 6 3的概率是多大?

求解方法：概率最大化

$$P = P(D6) * P(D6 \to 1) * P(D6 \to D8) * P(D8 \to 6) * P(D8 \to D8) * P(D8 \to 3)$$
$$= \frac{1}{3} * \frac{1}{6} * \frac{1}{3} * \frac{1}{8} * \frac{1}{3} * \frac{1}{8}$$

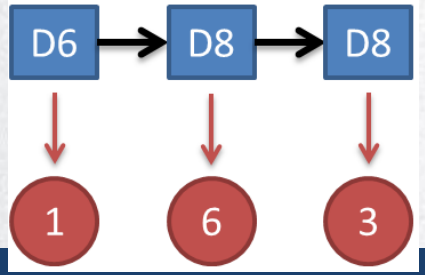


  - 枚举所有可能的骰子序列{D4D6D4, D4D6D6, D4D6D8, ...}
  - 依次计算这些序列出现的概率
  - 选取概率最大的掷骰子序列作为预测结果

文本分词（HMM）、词云

- 维特比算法效率分析
- 但是：枚举所有可能性复杂度太高
  - 长度为N，M个骰子，可能的骰子序列有M^N个（指数）
    - 对于长句子很难求解
  - 解决方案：动态规划（Viterbi算法）
    - 任意序列（如：D4D6D4）均为下表中的一个路径
    - 只要求解出概率最大的路径即可
    - 可利用的性质：
      - 如果路径D4D6D4最优，其前缀路径D4D6一定也最优（考虑马尔可夫性）
      - 假设最优路径在第K次观测时是DK，那么这条路径的前缀和后缀都是到达DK的所有路径中最优的

	第一次 观测1	第二次 观测6	第三次 观测3
D4			
D6			
D8			



文本分词（HMM）、词云

长度为N，有M个骰子

- 维特比算法效率分析：动态规划求解

该过程的时间复杂度是？
A: M*N
B: M + N
C: log(M^N)

发射1

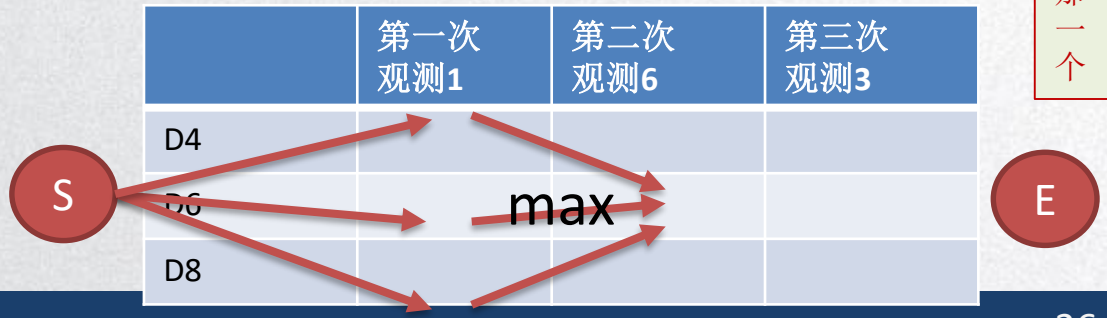
- T=1：首先假设序列长度为1，分别计算概率：
  - P14 = P(D4)P(D4->1)、P16 = P(D6)P(D6->1)、P18= P(D8)P(D8->1)
- T=2：
  - 假设最优路径在T=2时经过D6，那么此前缀路径的概率为
  - P14 * P(D4->D6)* P(D6->6) 、 P16 * P(D6->D6)* P(D6->6)、P18 * P(D8->D6)* P(D6->6)

发射6

  - 假设最优路径在T=2时经过D4，那么此前缀路径的概率为
  - P14 * P(D4->D4)* P(D4->6) 、 P16 * P(D6->D4)* P(D4->6)、P18 * P(D8->D4)* P(D4->6)
  - 假设最优路径在T=2时经过D8，那么此前缀路径的概率为
  - P14 * P(D4->D8)* P(D8->6) 、 P16 * P(D6->D8)* P(D8->6)、P18 * P(D8->D8)* P(D8->6)

三个数字中最大的那一个

- ……
- T=N：直到填满所有的空格
- 反向溯源每一次max所取的路径，
- 得到最优路径



文本分词（HMM）、词云

文本分词（HMM）、词云

- 在分词中采用HMM模型：如何与分词关联上？
  - 假设：每一个字符都是由标签所代表的类中随机产生的
    - **观测：**字串，每一个字符看成掷一次骰子的结果

      市 场 中 国 有 企 业 才 能 发 展
    - **隐状态：**标签序列，选取骰子的序列

      B E S B E B E B E B E
  - 和掷骰子相比
    - 隐状态选取，不是等概率
      - **硬性规则**：B后面不能立即再出现B
      - **概率规则**：对于中文大部分都是两字词，因此B后面出现E的概率比出现I的概率要大
    - 隐状态->观测的生成概率，也不是等概率

文本分词（HMM）、词云

- 在分词中采用HMM模型：如何表示并求解这些概率？
- 需要求解的参数
  - 隐状态选取（状态转移）概率
  - 隐状态->观测的生成概率
- 通常将这些概率表达为参数θ的函数
  - $P(s_1 \to s_2) = \frac{e^{\langle \theta_1, \phi(s_1,s_2) \rangle}}{Z_1}$
  - $P(s \to o) = \frac{e^{\langle \theta_2, \varphi(s,o) \rangle}}{Z_2}$
  - ϕ和φ为特征
- 机器学习对参数进行求解
  - 给定标注数据集
  - 获取每一个（观测X，标签Y）的概率
  - 最大似然估计：$\max_{\theta} P_1 P_2 \cdots P_N \Leftrightarrow \max_{(\theta_1,\theta_2)} \sum_{i=1}^{N} \log P_i$

通过训练数据计算参数

概率

P_1

P_2

P_N

找出产生这些文字序列的最优隐藏状态序列：怀特比算法

观测1：市场 中 国有 企业 才能 发展
标签1：B E S B E B E B E B E

观测2：南京市 长江 大桥
标签2：B I E B E B E

……

观测N：他 真 小气，象 个 铁公鸡
标签N：S S B E S S S B I E

文本分词（HMM）、词云

- 常用的基于统计的分词（序列标注）模型
- 常用的模型
  - 隐马尔科夫模型(Hidden Markov Model, HMM)
  - 最大熵马尔科夫模型 (Maximum Entropy Markov Model, MEMM)
  - 条件随机场(Conditional Random Fields, CRF)
- 优势
  - 只需提供**新训练数据和新特征**即可更新（重新训练）分词模型
  - 基于**全局（而非局部）的结果**判断一个句子整体分词的好坏
  - 特征可以包含更加丰富的信息：
    - 是否在字典中出现
    - 当前字的特征
    - 前后字的特征
    - 前一个字和后一个字的组合特征
    - 前一个字和后一个字的标签
    - ……

文本分词（HMM）、词云

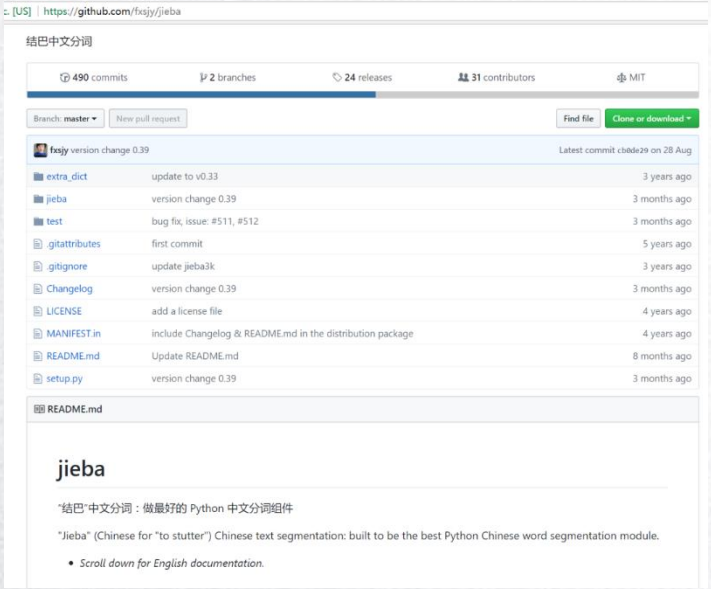
文本分词（HMM）、词云

- 分词实践：中文分词工具
  - **Jieba 结巴中文分词：(Python及大量其它编程语言衍生) 做最好的 Python 中文分词组件**
  - **北大中文分词工具：(Python) 高准确度中文分词工具，简单易用，跟现有开源工具相比大幅提高了分词的准确率。**
  - **kcws 深度学习中文分词：(Python) BiLSTM+CRF与IDCNN+CRF**
  - ID-CNN-CWS：(Python) Iterated Dilated（膨胀） Convolutions for Chinese Word Segmentation
  - Genius 中文分词：(Python) Genius是一个开源的python中文分词组件，采用CRF(Conditional Random Field)条件随机场算法。
  - loso 中文分词：(Python)
  - yaha "哑哈"中文分词 (Python)
  - ChineseWordSegmentation：(Python) Chinese word segmentation algorithm without corpus（无需语料库的中文分词）

资料来源：https://github.com/crownpku/Awesome-Chinese-NLP

文本分词（HMM）、词云

- 分词实践：结巴分词
  - https://github.com/fxsjy/jieba
  - 广为流传的Python中文处理工具
  - 最快捷的中文分词工具
  - 安装： pip install jieba



文本分词（HMM）、词云

- 分词实践：结巴分词
  - 支持三种分词模式：
    - **精确模式**，试图将句子最精确地切开，适合文本分析
    - **全模式**，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义
    - **搜索引擎模式**，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词
  - 支持繁体分词
  - 支持**自定义词典**
  - MIT 授权协议：免费使用和修改

文本分词（HMM）、词云

- 分词实践：Jieba分词→Python函数调用
  - 直接使用无需准备训练数据
  - 分词函数：
    - jieba.cut
      - 参数1：需要分词的字符串；
      - 参数2：cut_all 参数用来控制是否采用全模式
      - 参数3：HMM 参数用来控制是否使用 HMM 模型
    - jieba.cut_for_search
      - 参数1：需要分词的字符串；
      - 参数2：HMM 参数用来控制是否使用 HMM 模型
    - jieba.lcut 以及 jieba.lcut_for_search 直接返回 list

文本分词（HMM）、词云

- 普通分词

第一次运行：

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("我来到中国人民大学", cut_all=False)
print("Full Mode: " + "/ ".join(seg_list)) # 精确模式
```

```
Building prefix dict from the default dictionary ...
Dumping model to file cache /var/folders/67/81q9qf7d7cv3vkydc03wfjv80000gn/T/jieba.cache
Loading model cost 0.780 seconds.
Prefix dict has been built succesfully.
```

```
Full Mode: 我/ 来到/ 中国人民大学
```

返回generator：

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("我来到中国人民大学", cut_all=False)
print(seg_list)
for w in seg_list:
    print(w)
```

```
<generator object Tokenizer.cut at 0x10eeb5660>
我
来到
中国人民大学
```

文本分词（HMM）、词云

- 全模式分词：把句子中所有的可以成词的词语都扫描出来

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("我来到中国人民大学", cut_all=True)
print("精确模式: " + "/ ".join(seg_list))
```

```
精确模式: 我/ 来到/ 中国/ 中国人民大学/ 国人/ 人民/ 人民大学/ 大学
```

文本分词（HMM）、词云

- 搜索引擎模式分词：适合用于搜索引擎构建倒排索引的分词，粒度比较细

```
# encoding=utf-8
import jieba

seg_list = jieba.cut_for_search("我来到中国人民大学")
print(seg_list)
for w in seg_list:
    print(w)
```

```
<generator object Tokenizer.cut_for_search at 0x10ec26408>
我
来到
中国
国人
人民
大学
中国人民大学
```

文本分词（HMM）、词云

- lcut和lcut_for_search
  - jieba.lcut 以及 jieba.lcut_for_search 直接返回 list

```
# encoding=utf-8
import jieba

seg_list = jieba.lcut("我来到中国人民大学", cut_all=False)
print(seg_list)
```

```
['我', '来到', '中国人民大学']
```

文本分词（HMM）、词云

- 直接使用已有模型的局限
  - 分词模型快速，但是不可避免会出现错误

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("中区食堂和北区食堂都是我喜欢的中国人民大学的食堂", cut_all=False)
print("/".join(seg_list))
```

```
中/区/食堂/和/北区/食堂/都/是/我/喜欢/的/中国人民大学/的/食堂
```

- 未登录词：
  - 中区食堂
  - 北区食堂

文本分词（HMM）、词云

- 添加自定义词典
  - 开发者可以指定自己自定义的词典，以便包含 jieba 词库里没有的词
    - 虽然 jieba 有新词识别能力，但是自行添加新词可以保证更高的正确率
  - 用法： jieba.load_userdict(file_name) # file_name 为文件类对象或自定义词典的路径
    - 词典格式和 dict.txt 一样，一个词占一行；每一行分三部分：**词语**、词频（可省略）、词性（可省略），用空格隔开，顺序不可颠倒
    - file_name 若为路径或二进制方式打开的文件，则文件必须为 UTF-8 编码
  - 词频省略时使用自动计算的能保证分出该词的词频

文本分词（HMM）、词云

- 词典示例：增加dict.txt（必须为存为UTF-8格式）


```
import jieba
jieba.load_userdict("/Users/junxu/Documents/Work/上课/数据科学导论课程/程序/jieba分词/dict.txt")
seg_list = jieba.cut("中区食堂和北区食堂都是我喜欢的中国人民大学的食堂。", cut_all=False)
print("/".join(seg_list))
```

```
中区食堂/和/北区食堂/都/是/我/喜欢/的/中国人民大学/的/食堂/。
```

文本分词（HMM）、词云

- 动态增加单词
  - 用户词典文件适合提前批量增加
  - 在线动态增加单词：jieba.add_word

```
import jieba
#jieba.load_userdict("/Users/junxu/Documents/Work/上课/数据科学导论课程/程序/jieba分词/dict.txt")
seg_list = jieba.cut("中国人民大学数据科学导论本科生课程。", cut_all=False)
print("/".join(seg_list))
jieba.add_word("数据科学导论")
seg_list = jieba.cut("中国人民大学数据科学导论本科生课程。", cut_all=False)
print("/".join(seg_list))
```

```
中国人民大学/数据/科学/导论/本科生/课程/。
中国人民大学/数据科学导论/本科生/课程/。
```

文本分词（HMM）、词云

- 给文件分词

```
# encoding=utf-8
import jieba
fnInput = "/Users/junxu/Documents/Work/上课/数据科学导论课程/程序/jieba分词/不带标签短信.txt"
fnOutput = "/Users/junxu/Documents/Work/上课/数据科学导论课程/程序/jieba分词/不带标签短信-seg.txt"
fin = open(fnInput, "rb")
fout = open(fnOutput, "w+")
numline = 0
for line in fin:
    seg_list = jieba.cut(line, cut_all=False)
    fout.write(" ".join(seg_list))
    numline = numline + 1
fin.close()
fout.close()
print("Processed %d lines" % numline)
```

```
Processed 200000 lines
```

文本分词（HMM）、词云

- 分词结果

不带标签短信.txt

```
.x月xx日推出凭证式国债x年期x.xx.xx%，x年期x.xx%到期一次还本付息。真情邮政，为您竭诚服务！  咨询电话xxxx-xx
x强度等级水泥的必要性和可行性进行深入研究
Don’tSellaProduct
以上比赛规则由江苏科技大学教职工摄影协会负责解释
坐12个小时飞机身体已经疲惫不堪
为什么不能是你③以多数人的努力程度
地址位于天津市滨海新区响罗湾旷世国际大厦A座1801室
它是由AlexanderStepanov、MengLee和DavidRMusser在惠普实验室工作时所开发出来的
前首席执行官迪克·科斯特洛或将离开
zuzu气垫BB拍上去过几分钟后就会和皮肤越来越贴
年薪20万以上的工作岗位普遍较少
适当运用收纳设计把客厅改造成书房
被扭曲的独白拼凑折射出人性自私的阴暗面
命运永远会偏袒勇者…加油…
庆x'x节本会所优惠活动，为答谢新老顾客的支持与厚爱，，面部特卡:xxx元/xx次，身体活动，带脉减小肚腩:xxxx元/xx次，，肠胃
斯柯达对外发布了全新FabiaR5概念版
开头先夸一下自己：最近有不少人跟我说话都是这么开头的：哎呀
这样的ladybeard给吓坏了崩坏吧
```

不带标签短信-seg.txt

```
. x 月 xx 日 推出 凭证式 国债 x 年期 x . xx . xx% ， x 年期 x . xx% 到期 一次 还本付息 。 真情 邮政 ， 为 您 竭诚服务 ！  咨询电话 xxxx - xx
x 强度 等级 水泥 的 必要性 和 可行性 进行 深入研究
Don ’ tSellaProduct
以上 比赛规则 由 江苏 科技 大学 教职工 摄影 协会 负责 解释
坐 12 个 小时 飞机 身体 已经 疲惫不堪
为什么 不能 是 你 ③ 以 多数 人 的 努力 程度
地址 位于 天津市 滨海新区 响 罗湾 旷世 国际 大厦 A座 1801 室
它 是 由 AlexanderStepanov 、 MengLee 和 DavidRMusser 在 惠普 实验室 工作 时所 开发 出来 的
前 首席 执行官 迪克 · 科斯特 洛 或 将 离开
zuzu 气垫 BB 拍上去 过 几分钟 后 就 会 和 皮肤 越来越 贴
年薪 20 万 以上 的 工作岗位 普遍 较 少
适当 运用 收纳 设计 把 客厅 改造 成 书房
被 扭曲 的 独白 拼凑 折射出 人性 自私 的 阴暗面
命运 永远 会 偏袒 勇者 … 加油 …
庆 x ' x 节本 会 所 优惠活动 ， 为 答谢 新老 顾客 的 支持 与 厚爱 ， ， 面部 特卡 ： xxx 元 / xx 次 ， 身体 活动 ， 带脉 减 小肚腩 ： xxxx 元 / xx 次 ， ， 肠胃
斯柯达 对外 发布 了 全新 FabiaR5 概念 版
```

文本分词（HMM）、词云

文本分词（HMM）、词云

- 政府工作报告与词云

2020年政府工作报告.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

各位代表：现在，我代表国务院，向大会报告政府工作，请予审议，并请全国政协委员

这次新冠肺炎疫情，是新中国成立以来我国遭遇的传播速度最快、感染范围最广、防控
人民群众艰苦卓绝努力并付出牺牲，疫情防控取得重大战略成果。当前，疫情尚未结束
务。

一、2019年和今年以来工作回顾

去年，我国发展面临诸多困难挑战。世界经济增长低迷，国际经贸摩擦加剧，国内经济
目标任务，为全面建成小康社会打下决定性基础。

——经济运行总体平稳。国内生产总值达到99.1万亿元，增长6.1%。城镇新增就业1

——发展新动能不断增强。科技创新取得一批重大成果。新兴产业持续壮大，传统产

文本分词（HMM）、词云

- 政府工作报告与词云

```
1  import jieba
2  import wordcloud
3  #from scipy.misc import imread
4  from matplotlib.pyplot import imread
5
6  mask = imread("China.jpg")
7
8  f = open("2020年政府工作报告.txt", "r", encoding="utf-8")
9
10 exclude = {'我们','和','的','今年','万亿元'}
11 t = f.read()
12 f.close()
13 ls = jieba.lcut(t)
14
15 txt = " ".join(ls)
16 font='C:/Windows/Fonts/simfang.ttf'
17 w = wordcloud.WordCloud( \
18     width = 1000, height = 700,\
19     background_color = "white",\
20     font_path=font,\
21     mask=mask,\
22     stopwords=exclude
23     )
24 w.generate(txt)
25 w.to_file("政府工作报告词云.png")
```

https://zhuanlan.zhihu.com/p/143969768

文本分词（HMM）、词云

- 政府工作报告与词云

文本分词（HMM）、词云

- 微博情感数据集的分词与词云

文本分词（HMM）、词云

- 微博情感数据集的分词与词云

```
import jieba

def word_cut(texts):
    words_list = []
    word_generator = jieba.cut(texts, cut_all=False)
    exclude =['和','的']
    for word in word_generator:
        if (word.strip() not in exclude):
            words_list.append(word)
    return ' '.join(words_list)

pd_all['review_cut'] = pd_all.review.apply( word_cut)
pd_all.head()
```

	label	review	review_cut
0	1	更博了，爆照了，帅的呀，就是越来越爱你！生快傻缺[爱你][爱你][爱你]	更博 了 ， 爆照 了 ， 帅 呀 ， 就是 越来越 爱 你 ！ 生快 傻 缺 [爱 ...
1	1	@张晓鹏jonathan 土耳其的事要认真对待[哈哈]，否则直接开除。@丁丁看世界 很是细心...	@ 张晓鹏 jonathan 土耳其 事要 认真对待 [哈哈] ， 否则 直接 开除...
2	1	姑娘都羡慕你呢...还有招财猫高兴……//@爱在蔓延-JC:[哈哈]小学徒一枚，等着明天见您呢/...	姑娘 都 羡慕 你 呢 ... 还有 招财猫 高兴 ... ... / / @ 爱 在 蔓延 - JC ...
3	1	美~~~~~[爱你]	美 ~ ~ ~ ~ ~ [爱 你]
4	1	梦想有多大，舞台就有多大![鼓掌]	梦想 有 多 大 ， 舞台 就 有 多 大 ! [鼓掌]

文本分词（HMM）、词云

- 微博情感数据集的分词与词云
  - 绘制所有positive文档包含单词的词云

```
import wordcloud
import matplotlib.pyplot as plt

font='C:/Windows/Fonts/simfang.ttf'
exclude =['和','的']
def draw_wordcloud( words, color='white'):
    w = wordcloud.WordCloud( \
        width = 1000, height = 700,\
        background_color = color,\
        font_path=font,\
        stopwords=exclude
        ).generate(words)
    plt.figure(1, figsize=(13,13))
    plt.imshow(w)
    plt.axis('off')
    plt.show()

print("positive words")
pd_all_pos = pd_all[ pd_all['label']==1]
words_pos = ' '.join(pd_all_pos['review_cut'])
draw_wordcloud(words_pos)
```