

hanLP：网址：github.com/hankcs/HanLP 上有包

有很多种包，斯坦福的，哈工大的，斯图福的，还有处理英文的包。这里讲主要处理中文的包。  
[关键词提取] 短语提取。

《自然语言处理入门 Python》可以作为教材。

分词，词性标注，命名实体识别，依存句法分析，语义依存图。

输出的是单字实体和类别

底层 java

词法分析是核心任务

在 python 语言的 jpyppel 的 whl 文件，pip install JPype1-0.7.2-cp37m-win\_amd64.whl

用 pip install pyhanlp，https://pypi.tuna.tsinghua.edu.cn/simple 清华镜像，在命令中  
pip install -I 清华镜像 pyhanlp，再找 data 文件，下到安装的 pyhanlp 里面。!

pip show pip 可显示 pip 安装位置。在新的 jdk 15 中，只生成了 jdk 脚本，进入 jdk 安装目录，输入命令：

bin\jlink.exe --module-path jmods --add-modules java.desktop --output jre

pyhanlp 应配 jdk 8. 而非其他的。词：表达完整意义的最小单位。→ 是一个合适的粒度。) 文档可看

机器学习将很多问题转为数学问题，才可解决。所以要分词。

dictionary 文件夹里也包含了人为的分词 (Core Nature Dictionary) 分词。

分词是 NLP 基础，现在不讲。要看会看论文  
Gensim

关键词提取，命名实体识别。(分词方式不同，中文更难)

easy ai.tech  
ai-definition  
tokenization

nltk 工具，对英文处理 HanLP.segment("中国科学技术大学") 井结果是对的。提高匹配率基于词典匹配

三大难点(中文) 没有统一标准，各词组保存各语言。NP 没有图像成熟。

3 种典型分词 基于统计

歧义如何切分，

新词如何识别 (比如 蓝瘦香菇)

HMM  
CRF  
深度学习

基于深度学习  
提取特征或提高  
LSTM + CRF

HanLP 基于 CRF 算法

适应性强  
速度慢，成本高

词典分词，速度快，成本低

适应性不强，领域不同，效果不同

常见： 机器学习算法和词典相结合。

逆向最大匹配，学算支 → 要主动学习，建议买教材。

forward-segment 前向分词

forward\_segment 前向分词 (从读北京大学).dict) 其中 dic = load\_dictionary()  
text[i:j] → j 取到 i 遍历  
from utility import load\_dictionary

def forward\_segment(text, dict)

word\_list = []

i = 0

while i < len(text):

longest\_word = text[i]

for j in range(i+1, len(text)+1):

word = text[i:j]

if word in dict:

if len(word) > len(longest\_word):

longest\_word = word

word\_list.append(longest\_word)

it = len(longest\_word)

return word\_list

比如研究生命起源用前向的，或为研究物种起源，而后向的是对的

backward\_segment 后向匹配：其中 work\_list.insert(0, longest\_word) 插到前面

统计得出，一般的后向匹配准确率高于前向匹配。

双向匹配，单字更优先级更高，相等则正向匹配优先级更高。

长度相同时 单字成词 假如已知能匹配的最大长度，则减少很多匹配次数  
数量更少就是哪个。 n-gram 字典匹配

精确率：在预测为正的样本中，正确的概率召回率：在所有正的样本中，被预测为正的概率。

hapl properties 配置文件 加 .mhi 的小的字典。

字典树，一般称为trie树，搜索提交（添加查询）叶子节点是单词，也是字母。

查询效率比哈希高，空间换时间，利用公共前缀。字典树构建复杂度为 O(NL) → 上限长度

可以看这篇博客：[https://www.sohu.com/a/1300621285\\_115128](https://www.sohu.com/a/1300621285_115128) (字典树) N 单词

找以五元组开头的字符串，将构造字典树。搜索带 提交包含的，甚至前缀是相似的。

contains 对应 in，丁没有的调 { setitem > 魔术方法  
getitem

assert '自然' in trie 声明 不是 true 判报错。

\_\_init\_\_ 没有返回值 没有 return 语句。(3. 用魔术方法，自己写一样功能的代码可以加深理解)

import sys  
print(sys.getsizeof(combined)) 打得大。

前向匹配和逆向匹配时间基本一致，双向则是两倍。比原生 Java 虽然慢，调用了 JPIPE。

dat.py 为 (DoubleArrayTrie) 的双字典树。先了解再深度了解。

sys.path.append(..) 加入之前的为搜索路径 print(sys.path) 看看。print(\_\_file\_\_) 显示 这路径

sys.path.append('具体路径') 看看

评价指标: cnblogs.com/liyuewei5game/p/13270346.html → 混淆矩阵 → 分别统计数量

		真实值		预测值		
		+ (P)	- (N)	假设10个同学，8个成绩好，2个成绩差。		
预测值	+	TP	FP	预测值1	预测值2	预测值3
	-	FN	TN	1 2 3 4 5 6 7 8 9 10	好 差 好 差 好 差 好 差 好 差 好 差	好 差 好 差 好 差 好 差 好 差 好 差

→ true

混淆矩阵1: TP → positive 真实为 预测为好的数量

FP: 真实错 预测为好

模型	P	N
预测P	TP(5)	FP(2)
预测N	FN(3)	TN(8)
准确率	$acc = \frac{TP+TN}{10} = 0.5$	(P+N)
精确率	$P = \frac{TP}{TP+FP} = \frac{5}{7} = 0.71$	
灵敏度/召回率	$recall = \frac{TP}{TP+FN} = \frac{5}{8} = 0.625$	$F_1 = \frac{2 \cdot PR}{P+R}$
		调和平均值 = $\frac{2 \cdot P \cdot R}{\frac{P}{R} + \frac{R}{P}} = 0.67$

通常用  $F_1$  来比,  $F_1$  越高 越好

假设总样本为 10000 模型 1

PPD 正 1000 预测为正

准确率  $PPD / 10000 = 0.999$

精确率  $PPD / PPD + FN = 0.999$

召回率(要提高)  $PPD / PPD + F1 = 1$

模型 2

在正类别 PPD 预测对, 负类 P 对

0.999

→ 不靠谱(准确率)

精确率  $PPD / PPD + FN = 0.999$

$PPD / PPD + F1 = 0.998$

工作中可能涉及多个类别, 也是一样的。demo\_stopwords.py 停词

去停用词: segment = DoubleArray Trie Segment()

termList = segment.segment(text)

print("分词结果是:", termList)

print("分词结果去除停用词:", remove\_stopwords(termList, trie))

trie = load\_from\_words("的", "相对而言", "吧")

print("3. 分词去掉停用词", replace\_stopwords\_text(text, "xx", trie))

公司多数基于词典。少数大公司, 需单独设计分词。有中英文混台的(很少), { ① 转换语言  
② 去掉其他语言  
③ 在字典里识别, 外则子集 }  
自然语言处理入门中的窗口

evaluate-aws.py 评判准确率, 召回率等。OOV-R (范化能力): IV-R: 在字典中识别成功概率  
有监督 成本高, 一般用基于字典的。模型预训练方法

序列标注? 标注字符。即 NLP 中语言 V EN, GPT, BERT (面试中讲半小时) 自己看些论文了解前沿趋势。



模型

~~填空~~ 需要解决普通程序员无法解决的问题。

BERT 结合了 ELMo 和 GPT。 $P(w_1, w_2, \dots, w_n)$  是对于语言序列计算概率，一语言

机器学习角度：对语句概率分布的建模。通俗：判断题是人话。

任何基于语言模型的，与训练数据有关（在某些背景）。培训出来一般讲很绝对。

$P(w_1, w_2, \dots, w_n) = P(w_1) \cdot P(w_2 | w_1) \cdots P(w_n | w_1, w_2, \dots, w_{n-1}) \xrightarrow{\text{条件概率}} \frac{C(w_1, w_2, \dots, w_n)}{\sum C(w_1, w_2, \dots, w_{n-1}, w_n)}$  链式法则

极大似然估计计算概率， $P(w_i | w_1, w_2, \dots, w_{i-1}) = \frac{C(w_1, w_2, \dots, w_i)}{\sum C(w_1, w_2, \dots, w_{i-1}, w_i)}$  次数

先造语料库：我们今天去打篮球 我们去吃饭 我们去玩

$P(\text{我们今天去打篮球}) = \frac{1}{3}$ ,  $P(\text{我们去吃饭}) = \frac{1}{3}$ , ...,  $P(\text{其他}) = 0$

马尔可夫假设： $P(w_i | w_1, w_2, \dots, w_{i-1}) = P(w_i | w_{i-n+1}, \dots, w_{i-1})$

假设只依赖于前  $n-1$  个词。n-gram 语言模型定义 → 一般取 2, 3.

$n=1$  unigram:  $P(w_1, w_2, \dots, w_n) = \prod P(w_i)$

$n=2$  bigram  $P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i | w_{i-1})$ .

trigram 语言模型理解 <https://zhuanlan.zhihu.com/p/52061158>

OOV：指超出样本。前馈神经网络，循环神经网络。

my\_cws\_corpus 造语料库：load\_cws\_corpus 读数中，return  
CorpusLoader, convert2SentenceList (corpus\_path).

ViterbiSegment() 维特比算法。链接：cngblog.com/pinard/p/  
CRF 条件随机场：hmm-cws.py。隐写科夫模型 (HMM) → 解决序列标注问题，全局分词（标注命名实体识别）。crf.

### 画出状态预测

隐形，代表状态观测不到，叫做预测。天气状观察状态序列，行为属于观测序列。

HMM要素：初始概率分布  $\pi$ ，隐含状态转移矩阵  $A$ ，隐含状态到可观测状态转移概率分布矩阵  $B$ ，隐含状态  $H$ ，可观测状态  $O$ ，开始隐含状态  $P_1$ 。

predict 预测问题 解码问题，learning 问题

$\downarrow$  encode 知  $O, H$  ( $p_i, A, B$  已知) 根据  $O$  估计  $A$  和  $B$ 。  
 $H, A, B, P_1$  已知，计算某一个观察序列出现概率。

(向前推)  $H, A, B, P_1$  已知，已知可观测状态序列， $\{O\}$  背后最可能对应隐含状态 (惟一化)。 $\rightarrow$  属于可观测状态 序列  $P_1$

序列标注问题：我们今天打篮球 (规则序列)  
 $S S B E S S S S \rightarrow$  对应问题二  
BES 规则 (序列标注)：B：词语开始，E：代表词语中间或者后面 S：单独是一个词。  
 $B E B E S B E E$  规则打标签

问题3：Baum-Welch 算法。解码  $\rightarrow$  惟一化  $\rightarrow$  分词。 $\rightarrow$  的形态规则。  
 $\rightarrow$  通用求解最短路径  $\checkmark$

$P_1 S S B E S S S S$  我们今天打篮球) =  $P_1, \dots, P_n$ ，概率大的就是。

隐含状态序列  $I^* = \{i_1^*, \dots, i_t^*\}, \{o_{t+1}, o_{t+2}, \dots\}$  为观测序列。

$\lambda = (A, B, \pi)$  为模型。维特比不用在工程上。

HMM 模型定义： $Q = \{q_1, \dots, q_N\} \rightarrow$  可能隐藏状态集合， $V$  是所有可能观测状态集合， $V = \{v_1, v_2, \dots, v_M\}$ ， $i_t \in Q, o_t \in V$ ，

HMM 模型假设：① 马尔可夫假设， $i_t = q_{t+1} | i_t = q_j$

从时刻  $t$  到  $t+1$  的 HMM 状态转移概率 =  $Q \ p(i_{t+1} = q_j | i_t = q_i)$

$a_{ij}$ ：组成马尔可夫链状态转移矩阵  $A$ ： $A = [a_{ij}]_{N \times N}$

② 观测独立性：观察状态依赖于当前时刻的隐藏状态。当  $i_t = q_j$ ，  
且  $o_t = v_k$ ，该时刻  $v_k$  在  $q_j$  下生成概率  $b_{jk}$  满足。

$b_{jk} = P(o_t = v_k | i_t = q_j)$ ， $B = [b_{jk}]_{N \times M}$ .

$T=1$  隐马尔科夫模型分布  $\Pi = [\pi_{ij}]_N$ , 其中  $\pi_{ij} = P(v_i = o_j)$ , HMM 模型由  $\Pi, A, B$ ,  $\lambda = (A, B, \Pi)$

状态转移矩阵

观测矩阵

状态转移矩阵：指隐含状态之间的转换概率。

发射矩阵：指隐含状态到观测状态的概率。机器学习推荐 李航的《统计学习方法》。西瓜书有些难懂。

$f_{ti}(i) = \max_{1 \leq j \leq N} [f_{t-1}(j) a_{ji} b_{ij}(o_{ti})]$ ,  $t+1$  时刻观测值情况下从  $j$  转  $i$  的概率。

$\pi(i) = \arg \max_{1 \leq j \leq N} [f_{t-1}(j) a_{ji}]$ , 从时刻  $0 \rightarrow T$  用  $\pi(i)$  记录前一个最可能的状态返回值。

$f_{ti}(i) = \max_{j_1, j_2, \dots, j_{t-1}} P(j_t=i | v_1, v_2, \dots, v_{t-1}, o_1, o_2, \dots, o_t | \lambda) \quad i=1, 2, \dots, N$

$f_{ti}(i) = \pi_{ri} b_{ri}(o_i) \quad r=1, 2, \dots, N, i=1, 2, \dots, N$

训练，获得  $A, B$  矩阵，预测：通过  $A, B$  矩阵分词、实体识别、词性标注（序列标注）

我们  
拿  
来  
看  
就  
能  
解  
决  
这  
个  
问  
题  
三

往精细化算法：有监督的机器学习

前向传播，反向传播

鲍特-韦尔奇算法（比较难，可不看）

CRF 条件随机场：Seq2Seq 模型  $\rightarrow$  词性标注，chunking，命名实体识别，  
命令实体识别， $X$ : observation,  $Y$ : predict, conditional random field 是一个判别式模型

HMM: a generative model,

条件概率

随机变量  $X$

$Pr(X, Y)$  HMM models joint probability 联合概率

$Pr(W, T) = Pr(W|T) * Pr(T) = \prod_{i=1}^n Pr(w_i|T) * Pr(T)$ ,  $W$  输入， $T$  输出，前提是独立假设。

条件独立

展开成每个单词

$= \prod_{i=1}^n Pr(w_i|T) * Pr(T)$  由于自己相碰  
单词概率依赖于使用 bigram, 只依赖于前一个相碰

$= \prod_{i=1}^n Pr(w_i|t_i) * Pr(t_i|t_{i-1})$

只与当前词性与上一个词性

} 当前词语与当前词性

当前 state

input segment

前半 state

CRF 用 feature function 更抽象表达特征 不局限于两步修正，可以表示为  $f(x, y_i, y_{i-1})$ ，  
特征函数可以表示当前 state 与任意一个 observation state (而且和位置有关) 当前位置。

$y_{i-1}$  可以是  $y_{i-1}(1, \dots, n)$ , 用  $y_{i-1}$  表示概率分布关系。

在CRF中， $y$ 在时间*i*的状态与 $y(i-1), y(i)$ 相关， $P(y|X)$ 是随机游走给定 $X$ 下条件概率。

$$P(y|X) = \sum_{\{x\}} \exp \left( \sum_k \text{atk}(y_{i-1}, y_i, x, i) + \sum_l \text{cls}(y_i, x, i) \right)$$

↓ 权值  
 状态转移权值  
 表示状态中差4种情况

其中  $Z(X) = \sum_{\{y\}} \exp \left( \sum_k \text{atk}(y_{i-1}, y_i, x, i) + \sum_l \text{cls}(y_i, x, i) \right)$

$$P(y|X) = \frac{\exp \left( \sum_k \text{atk}(y_{i-1}, y_i, x, i) + \sum_l \text{cls}(y_i, x, i) \right)}{Z(X)}$$

给定 $X$ 之后，输出 $y$ 的分布  
 ↓ 特征权值  
 ↓ 策略权值

$X$ : 特征信息集合， $Z(X) = \sum_y \exp \left( \sum_k \text{atk}(y_{i-1}, y_i, x, i) \right)$  称为归一化因子。

$$Y = \begin{cases} P(\text{负}) \\ P(\text{良}) \\ P(\text{中}) \\ P(\text{差}) \end{cases}$$

$\text{Y(负)} \leftarrow \exp \left( \sum_k \text{atk}(X, Y\text{负}) \right)$   
 $\text{Y(良)} \leftarrow \exp \left( \sum_k \text{atk}(X, Y\text{良}) \right)$

181819286

TF-IDF转化为矩阵形式，链接CRF: [https://blog.csdn.net/echo\\_kang/article/details/113145286](https://blog.csdn.net/echo_kang/article/details/113145286)

$f_i(x, y)$ 表示某个特征信息对某一个输出是否影响的事实。 $f_i(x, y) = 1$  表示有影响， $f_i(x, y) = 0$  表示无影响。

综合各种信息， $t_j(y_{i-1}, y_i, X, i)$  表示特征 $j$ ，  
 单 良 差 动  $\rightarrow \begin{cases} 1 & (\text{有影响}) \\ 0 & (\text{无影响}) \end{cases}$

$S_k(y_i, x, i)$  值占得越大。

关键词提取TF-IDF算法: (`demo_extract_word.py`): 从多篇文章中提取每篇文章的关键词。

$$\text{TF-IDF}(t, d) = \frac{\text{TF}(t, d)}{\text{DF}(t)} = \text{TF}(t, d) \cdot \text{IDF}(t)$$

单词  
 文档  
 → 出现频数  
 多数文档包含

textrank: pagerank。

从一篇文章中提取关键词，提取摘要。为 pagerank 在文本上应用。  
 pagerank 计算公式：初始权重  $S(V_i) = 1$ ，则  $S(V_i) = (1-d) + d \times \sum_{V_j \in \text{In}(V_i)} \frac{1}{\text{out}(V_j)} S(V_j)$

成熟度  
 外链权重与外链数  
 与提高外链权重有关

与垃圾网站往往也是垃圾网站。

IDF-IDF:  $TF_w = \frac{W_w}{N} \rightarrow W_w$  为词频,  $IDF_w = \log \frac{N}{W_w + 1} \rightarrow N$  为语料库文档数, 防止分母为0  $\rightarrow$  越常见的词, ID越小.

textrank: 自然语言处理入门非常有趣  
权重: 高频词  
投票者的权重  
短语提取一般过滤停用词.

关键词提取: 引入 BM25 衡量句子相似度, 改进链接权重计算, 相似句子得到较高投票.

$$BM25(D, Q) = \sum_{i=1}^n IDF(q_i) \cdot \frac{TF(q_i, D) \cdot (k_1 + 1)}{TF(q_i, D) + k_1(1 - b + b \cdot \frac{|D|}{avgL})},$$

改进算法, 称为 textrank!

$$WS(V_i) = (1-d) + d \cdot \sum_{V_j \in n(V_i)} \frac{BM25(V_i, V_j)}{\sum_{V_k \in out(V_i)} BM25(V_k, V_j)}$$

提取 keyword Extra for  
FeatureRank Keyword TfIdfCounter TermFrequency Counter

修改 tfidf.py: from pyhanlp import \*

```
from pyhanlp import data_
def Stopwords():
    path = HanLP.Config.CoreStopWordDictionaryPath
    map = JClass('java.util.TreeMap')()
```

操作中直删一段

class TfIdfCounter1:

Stopwords = Stopwords()

segment = DoubleArrayTrieSegment1()

def \_\_init\_\_(self): < self.word\_tf\_idf\_dict = {}

self.document = []

def add(self, text\_name, text):

(text)

self.document.append([text\_name, [term.word for term in self.segment seg

if not self.stopwords.contains(key(term.word))]]

```

def compute(self): → for text in self.document:
    pass
    if __name__ == '__main__':
        counter = TfIdfCounter()
        counter.add(' ', ',')
        counter.add(' ', '.')
        counter.compute()
        print(counter.document)
        回答相似度 + 括号文本识别
        回答系统：文本相似度。
        只左右信息熵 → 左右词频因词
        单信息 互信息大，关系密切。
         $I(x,y) := \frac{p(x,y)}{p(x)p(y)} = \frac{p(x,y)}{\log p(x) \cdot p(y)}$ 
        E(p(x,y))
        将两个指标（左右信息熵 互信息）合并
        一定阈值过滤掉，剩下降序排列。
        截取最高N次即完成词语提取。
        文本相似度可用tf_idf,
        elastic search 相似度计算被 tf_idf bms5
        解决 { 用规则语言去做
        相关 { 再用深度学习 <机器学习>
            ↓ 基础 { 应用原因 → 用英文写搜索查询
            ↓ 资料 (文档) → 全有些调研的文档
                } 数据筛选: 多数过语言之外。

```

$W_B = \text{BOS}$  (begin of sentence)  $\rightarrow <\text{s}>$

$W_E = \text{EOS}$  (end ..)  $</\text{s}>$

语义图谱？ 揭示实体之间关系的意义网络

print(counter.cal\_ssim('')) 计算相似度

```

def cal_sim(self, text):
    text_list = []
    for term in text:
        if term in self.segment.seg(text) or term in self.containsKey(term.word)

```

```

            for key, value in self.word_dict.items():
                if key == term:
                    score += value

```

```

for word in text_list:

```

```

    if word in value:
        score += value

```

```

    .text_list[key] = score

```

```

return self.sim_dict

```

一篇文档 textrank 多篇文档 tfidf

```

8 | 第3章 二元语法与中文分词
tests/book/ch03/ngram_segment.generate_wordnet():

def generate_wordnet(sent, trie):
    """
    生成词网
    :param sent: 句子
    :param trie: 词典(一元语法)
    :return: 词网
    """
    searcher = trie.getSearcher(JSONObject(sent), 0)
    wordnet = WordNet(sent)
    while searcher.hasNext():
        wordnet.add(searcher.begin + 1, Vertex(sent[searcher.begin:searcher.begin + searcher.length], searcher.value, searcher.index))
    # 原子分词，保证图连通
    vertexes = wordnet.getVertexes()
    i = 0
    while i < len(vertexes):
        if len(vertexes[i]) == 0: # 空白行
            j = i + 1
            for j in range(i + 1, len(vertexes) - 1): # 寻找第一个非空白行
                if len(vertexes[j]):
                    break
            # 填充[i, j]之间的空白行
            wordnet.add(i, Vertex.newPunctuationInstance(sent[i - 1: j - 1]))
            i = j
        else:
            i += len(vertexes[i] - 1).realWord)
    return wordnet

```

值得注意的是，词网必须保证从起点出发的所有路径都会连通到终点。细心的读者可以发现，行 5 的务是一元语法中没有的词语，它就是为了保证词网连通而添加的单字词语。

在 HanLP 的 Java 代码中，词网的实现是 WordNet，里面每个词语存储为 vertex，如下（详见 com.hankcs.hanlp.net.Wo

与词先模型缓解了部分数据稀疏的问题  
同时可以利用滑窗路一线性插值

$$p(w_t | w_{t-1}) = \lambda p_{ml}(w_t | w_{t-1}) + (1-\lambda) p(w_t)$$

一元语法：

$$p(w_t) = \lambda p_{ml}(w_t) + (1-\lambda) \frac{1}{N} \sum_{i=1}^N p_{ml}(w_i)$$

语料库总词频

书中选用 MSR 为语料库。

语料库统计 词频  
总词数  
总词频

生成词网：有点懵 → 不晓得如何实现

$In(v_j)$  指向网页的邻集集合

$Out(v_i)$  是网页链接出去的邻集集合。经验公式

$$p(w_t | w_{t-1}) = \lambda \left[ u \frac{c(w_{t-1}, w_t)}{c(w_{t-1})} + (1-u) \right] + (1-\lambda) \frac{c(w_t)}{N}$$

基于距离计算，用MLE(极大似然估计)

加  $u$ ,  $u$  为平滑策略

加平滑 拉普拉斯平滑

$\frac{1}{k+1} p(w_t | w_{t-1}) \rightarrow - \sum_{k=1}^{k+1} \log p(w_t | w_k)$  防止将值连乘之后向下溢出，取对数  
OOV: (out of vocabulary) 在字典外  $\downarrow$  pagerank 有向图  
文本生成也可以生成摘要 不过要进行语句检测。  $\downarrow$  textrank 无向图

textrank 也可以进行关键词提取。  $\downarrow$  先筛选

文本聚类 (demo-text-clustering.py): 没有标签的聚类。 k-means (k 均值聚类)  
距离  $\rightarrow$  相似性，迭代求解，随机选取 k 个对象为初始的聚类中心。  $\downarrow$  microsoft edge 可以画图。  
腾讯会议可以开批注。  
把点划分最近的那一类。  $\downarrow$  其中哪些 根据簇间距离判断，停止迭代。  $\downarrow$  目标函数

k-means: jianshu.com/p/fc81fed8c77b, 簇内尽量小，簇间尽量大。  
可计算文本相似度 文本聚类  $\rightarrow$  一般用主题模型，  
 $\downarrow$  协同过滤用词，文档向量

代码: cnblogs.com/bianding/p/8878098.html

sigmoid 分类, demo-word2vec 词向量, word2vec 词向量, 一个词用一个向量表示 (3年谷歌)  
跳跃 (skip-gram) 和词袋模型 (CBOW), 其中 skip-gram 用中心词预测周围词。  
 $\downarrow$  用周围词预测中心词。 softmax 多分类。

丰度: 是指一种化学元素在某个自然体中重量占这个自然体总重量的相对份额。

命名实体识别，使用BMES 标注集，不构成命名实体的单词，统一标注为 O (outside)  
 $\downarrow$  现实存在的实体，  $\uparrow$  命名实体识别