

Chap2 正则表达式&文本规范化&编辑距离

2.1 正则表达式

2.1.1 基础正则表达式

RE	Example Patterns Matched
/woodchucks/	“interesting links to <u>woodchucks</u> and lemurs”
/a/	“ <u>Mary</u> Ann stopped by Mona’s”
/!/	“You’ve left the burglar behind again!” said Nori

Figure 2.1 Some simple regex searches.

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	“ <u>Woodchuck</u> ”
/[abc]/	‘a’, ‘b’, or ‘c’	“In uomini, in soldati”
/[1234567890]/	any digit	“plenty of <u>7</u> to 5”

Figure 2.2 The use of the brackets [] to specify a disjunction of characters.

RE	Match	Example Patterns
/[A-Z]/	an upper case letter	“we should call it ‘ <u>Drenched</u> Blossoms’ ”
/[a-z]/	a lower case letter	“ <u>my</u> beans were impatient to be hoed!”
/[0-9]/	a single digit	“Chapter <u>1</u> : Down the Rabbit Hole”

Figure 2.3 The use of the brackets [] plus the dash – to specify a range.

RE	Match (single characters)	Example Patterns
/[^A-Z]/	not an upper case letter	“Oyfn pripetchik”
/[^Ss]/	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
/[^.]/	not a period	“our resident Djinn”
/[e^]/	either ‘e’ or ‘^’	“look up ^ now”
/a^b/	the pattern ‘a^b’	“look up a^ b now”

Figure 2.4 The caret ^ for negation or just to mean ^. See below re: the backslash for escaping the period.

RE	Match	Example Patterns
/woodchucks?/	woodchuck or woodchucks	“ <u>woodchuck</u> ”
/colou?r/	color or colour	“color”

Figure 2.5 The question mark ? marks optionality of the previous expression.

RE	Match	Example Matches
/beg.n/	any character between beg and n	begin, <u>beg’n</u> , <u>begun</u>

Figure 2.6 The use of the period . to specify any character.

RE	Match
^	start of line
\\$	end of line
\b	word boundary
\B	non-word boundary

Figure 2.7 Anchors in regular expressions.

2.1.2 分隔、分组、符号优先级

Parenthesis	()
Counters	* + ? { }
Sequences and anchors	the ^my end\$
Disjunction	

正则表达式的匹配是贪心的，如 `[a-z]*` 会覆盖尽可能多的串，如once这个词，会完全覆盖

若需要非贪心匹配，使用 `*?` 和 `+?` 符号，尽可能少地匹配文本

2.1.3 更多操作符

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party_of_5
\D	[^0-9]	any non-digit	Blue_moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!
\s	[\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in_Concord

Figure 2.8 Aliases for common sets of characters.

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	n occurrences of the previous char or expression
{n,m}	from n to m occurrences of the previous char or expression
{n,}	at least n occurrences of the previous char or expression
{,m}	up to m occurrences of the previous char or expression

Figure 2.9 Regular expression operators for counting.

RE	Match	First Patterns Matched
*	an asterisk “*”	“K_A_P_L_A_N”
\.	a period “.”	“Dr. Livingston, I presume”
\?	a question mark	“Why don’t they come and lend a hand?”
\n	a newline	
\t	a tab	

Figure 2.10 Some characters that need to be backslashed.

2.1.4 替换，捕获分组，提前观察

替换: `s/([0-9]+)/<\1>/`

分组重现: `/the (.*)er they (*), the \1er we \2/`

`/(?:some|a few) (people|cats) like some \1/`

提前观察: `/^(?!Volcano)[A-Za-z]+/`

2.2 字词

语料集 (corpus / corpora)

- 文本语料集:
 - 标点符号 (punctuation) : 有利于区分语段 (句号、逗号)，识别特定含义 (引号、问号、冒号)
- 语音语料集:
 - 碎片 (fragment) : 说错话或说一个词卡壳
 - 停顿 (filled pause) : 语气词等、作为讲着思想的重启，可用于预测后续词，也可根据停顿习惯识别讲者身份

语言统计学三大定律:

- Zipf law: 在给定的语料中，对于任意一个词，其频度(freq)的排名 (rank) 和 freq 的乘积大致是一个常数
- Heaps law: 在给定的语料中，其词汇集 V 与语料大小 N ，存在大致关系 $|V| = kN^\beta$, $0 < \beta < 1$
- Benford law: 在 b 进位制中，以数 n 起头的数出现概率为 $\log_b(1 + \frac{1}{n})$ ，常用于检测数据是否造假

2.3 语料集

语料集说明文档

- 动机：为何建立这个语料集，谁建的，谁出钱支持
- 情况：文本的写/说是在何种环境下进行的，是专门的任务吗，文本是对话还是印刷稿/社交媒体通信/独白或对白
- 语言：语言种类，包括地域方言
- 讲者特征：性别/年龄等
- 收集过程：数据量大小，若是采样数据（如何采样？）数据采集被同意否，数据如何被预处理，是否有元数据
- 标注过程：标注是什么，标注者的特征是什么，标注者经过如何的训练，数据如何被标注
- 分发限制：是否有版权或存在私人财产限制

2.4 文本规范化

文本规范化 (text normalization)

- 标记化 (tokenization)：多语言、表情和标签
- 词形还原 (lemmatization)：判断词语的词根是否一致，如 sing (sang、sung、singing)
 - 词干提取 (stemming)：朴素的词形还原方法，通常剪去后缀
- 句子划分 (sentence segmentation)

2.4.1 使用Unix工具进行简易标记化

- 词频统计
 - Linux [tr](#) 命令用于转换或删除文件中的字符
 - Linux [sort](#) 命令用于将文本文件内容加以排序
 - Linux [uniq](#) 命令检查及删除文本文件中重复出现的行列，配合sort使用

2.4.2 词标记

- 命名实体识别 (NER)：处理价格，日期，链接，邮箱地址，专业长词时需要整体标记
- 缩写还原：what's = what is
- 词标记是自然语言处理的第一步，所以需要高效处理，标准的方法是使用正则表达式编译出的有限状态机去决定每个标记，python中可以使用nltk.regexp_tokenize()函数

```

>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)      # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...     | \w+(-\w+)*       # words with optional internal hyphens
...     | \$?\d+(\.\d+)?%? # currency and percentages, e.g. $12.40, 82%
...     | \.\.\.            # ellipsis
...     | [][.,;'"'?():-_'] # these are separate tokens; includes ], [
...     ''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']

```

Figure 2.12 A Python trace of regular expression tokenization in the NLTK Python-based natural language processing toolkit (Bird et al., 2009), commented for readability; the (?x) verbose flag tells Python to strip comments and whitespace. Figure from Chapter 3 of Bird et al. (2009).

- 对于汉字，使用单字作为标记单位更佳，因为单字是大多数场景下的语义单位能避免大量陌生词的出现，

2.4.3 Byte Pair Encoding (BPE) 算法

- tokenization模式
 - token learner: 从训练语料中归纳出词汇表 (token的集合)
 - token segmenter: 将测试语料标记化
- tokenization常用算法 ([参考](#))
 - byte-pair encoding (BPE,2016)

```

function BYTE-PAIR ENCODING(strings C, number of merges k) returns vocab V
    V ← all unique characters in C      # initial set of tokens is characters
    for i = 1 to k do                  # merge tokens til k times
        tL, tR ← Most frequent pair of adjacent tokens in C
        tNEW ← tL + tR           # make new token by concatenating
        V ← V + tNEW             # update the vocabulary
        Replace each occurrence of tL, tR in C with tNEW   # and update the corpus
    return V

```

Figure 2.13 The token learner part of the BPE algorithm for taking a corpus broken up into individual characters or bytes, and learning a vocabulary by iteratively merging tokens. Figure adapted from Bostrom and Durrett (2020).

- unigram language modeling(2018)
- WordPiece(2012)
- SentencePiece (2018)

2.4.4 word normalization, lemmatization, stemming

- Word normalization: 将 words/tokens 变成标准格式 (如 US, USA)
- Lemmatization: 将词的词干和词缀分离, 简易的方法叫stemming, 直接砍掉后缀
- [Porter stemmer](#) 是1980年提出的, 至今广泛使用的stemming方法, 此法基于重写规则, 较为简单, 但可能会遇到误判 (commission不同的判为相同, omission相同的判为不同) , 如下

Errors of Commission		Errors of Omission	
organization	organ	European	Europe
doing	doe	analysis	analyzes
numerical	numerous	noise	noisy
policy	police	sparse	sparsity

2.4.5 sentence segmentation

2.5 最小编辑距离

两个字符串的最小编辑距离: 一个字符串通过插入, 删除, 替换操作变成另一字符串的最小次数

动态递推式:

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) \end{cases}$$

例子 (记录backtrace)

#	#	e	x	e	c	u	t	i	o	n
#	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8	← 9
i	↑ 1	↖ ↗ 2	↖ ↗ 3	↖ ↗ 4	↖ ↗ 5	↖ ↗ 6	↖ ↗ 7	↖ 6	← 7	← 8
n	↑ 2	↖ ↗ 3	↖ ↗ 4	↖ ↗ 5	↖ ↗ 6	↖ ↗ 7	↖ ↗ 8	↑ 7	↖ ↗ 8	↖ 7
t	↑ 3	↖ ↗ 4	↖ ↗ 5	↖ ↗ 6	↖ ↗ 7	↖ ↗ 8	↖ 7	↔ 8	↖ ↗ 9	↑ 8
e	↑ 4	↖ 3	← 4	↖ 5	← 6	← 7	↔ 8	↖ ↗ 9	↖ ↗ 10	↑ 9
n	↑ 5	↑ 4	↖ ↗ 5	↖ ↗ 6	↖ ↗ 7	↖ ↗ 8	↖ ↗ 9	↖ ↗ 10	↖ ↗ 11	↖ ↗ 10
t	↑ 6	↑ 5	↖ ↗ 6	↖ ↗ 7	↖ ↗ 8	↖ ↗ 9	↖ 8	← 9	← 10	↖ ↗ 11
i	↑ 7	↑ 6	↖ ↗ 7	↖ ↗ 8	↖ ↗ 9	↖ ↗ 10	↑ 9	↖ 8	← 9	← 10
o	↑ 8	↑ 7	↖ ↗ 8	↖ ↗ 9	↖ ↗ 10	↖ ↗ 11	↑ 10	↑ 9	↖ 8	← 9
n	↑ 9	↑ 8	↖ ↗ 9	↖ ↗ 10	↖ ↗ 11	↖ ↗ 12	↑ 11	↑ 10	↑ 9	↖ 8

自然语言处理中的动态规划

- Viterbi算法：最小编辑距离的概率扩展，计算maximum probability alignment (<https://zhuanlan.zhihu.com/p/40208596>)
- CKY算法

Chap3 N-gram 语言模型

- 动机：如何预测下一个词
- 方法：通过模型为接下来可能出现的词分配概率
- 场景：语音识别中的纠错，文本语法纠错，机器翻译的正确表达，输入法提示
- 术语定义：
 - 语言模型（LM）：为词序列分配概率的模型
 - n-gram：一个包含n个单词的序列（n=2, bigram; n=3, trigram）

3.1 N-Grams

- 使用全部上文进行估计
 - 公式： $P(w_n | w_1 w_2 \dots w_{n-1}) = \frac{Count(w_1 w_2 \dots w_n)}{Count(w_1 w_2 \dots w_{n-1})}$
 - 问题：数据集中可能没有包含新奇句子的样本，导致分子或分母为0
- 使用历史距离最近的N-1个词，形成N个词的小组，来进行近似估计
 - 公式： $P(w_n | w_1 w_2 \dots w_{n-1}) \approx P(w_n | w_{n-N+1} \dots w_{n-1}) = \frac{Count(w_{n-N+1:n})}{Count(w_{n-N+1:n-1})}$
 - 链式法则： $P(w_{1:n}) = \prod_{k=1}^n P(w_k | w_{1:k-1}) \approx \prod_{k=1}^n P(w_k | w_{k-N+1:k-1})$
 - 问题：一直相乘会导致长句的概率极小，导致数值无法被表示（underflow），
 $\prod_{i=1}^n p_i = e^{\sum_{i=1}^n \log p_i}$
 - 改良方法：使用对数概率

3.2 评价语言模型

- 数据集切分：80%训练，10%验证，10%测试
- 评价指标：迷惑度（Perplexity, PP）

- 公式：

$$PP(W) = P(w_1 w_2, \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}} \approx \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-n+1} \dots w_{i-1})}}$$

- 思想：测试集中的句子都是正常的句子，那么训练好的模型在测试集上的概率越高越好，即最大化 $P(w_1 \dots w_N)$ ，最小化 PP

- 注意事项：
 - 1、词表相同的模型对比才有意义
 - 2、涉及句首词预测需要增加前置padding

3.3 泛化、零概率问题

- 假设一个语料库长度为 N , 有 V 个不同的词汇, n 较大时, 按概率生成的句子效果较好, 但也有一些问题
 - 存储空间占用较大, n -gram模型需要计算的概率有 V^n 个
 - 稀疏性问题 (较多的概率值为0), 当测试集出现训练集未曾出现过的词串模式时, 会导致整体预测概率变为0, PP无穷大
 - 训练数据需要与测试数据具有相同的体裁, 方言, 语言模型才会表现得好
- 稀疏性问题
 - 定义: 当测试集出现训练集未曾出现过的词串模式时, 会导致整体预测概率变为0
 - 原因及处理办法
 - 情况一: 测试集出现未知词汇
 - 处理办法: 使用固定的词典, 使用代表训练集中不在词典的词 (或出现次数小于阈值的词), 将当成一个普通词, 用同样的方法预测概率
 - 情况二: 测试集中出现了训练集不存在的上下文模式
 - 处理办法: 平滑化、减少 n -gram的 n 值 (backoff)

3.4 平滑

- k -平滑:
 - 公式: $\frac{C(w_{n-N+1:n})}{C(w_{n-N+1:n-1})} \rightarrow \frac{C(w_{n-N+1:n})+k}{C(w_{n-N+1:n-1})+kV}$ ($k=1$ 时又叫拉普拉斯平滑)
 - 评价: 对于语言模型而言效果不好, 概率分布的方差不好, 不合适的平滑
- 使用低阶 n -gram的概率做插值来达到高阶 n -gram的效果
 - 公式: $\hat{P}(w_n | w_{n-N+1} \dots w_{n-1}) = \sum_{i=1}^N \lambda_i P(w_n | w_{n-N+i} \dots w_{n-1}) \quad s.t. \quad \sum_i \lambda_i = 1$
 - 评价: 使用EM算法来学习 λ_i 的最优值, 有一种叫Katz backoff
- Kneser-Ney 平滑
 - 公式:
 - $P_{KN}(w_i | w_{i-1}) = \frac{\max(C(w_{i-1} w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$
 - $P_{CONTINUATION}(w) = \frac{|\{v: C(vw) > 0\}|}{\sum_{w'} |\{v: C(vw') > 0\}|}$
 - $\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}, v)} |\{w: C(w_{i-1} w) > 0\}|$
 - $P_{KN}(w_i | w_{i-n+1:i-1}) = \frac{\max(C_{KN}(w_{i-n+1:i}) - d, 0)}{\sum_v C_{KN}(w_{i-n+1:i-1}, v)} + \lambda(w_{i-n+1:i-1}) P_{KN}(w_i | w_{i-n+2:i-1})$

- $C_{KN}(.) = \begin{cases} count(.) & \text{for the highest order} \\ continuation - count(.) & \text{for the lower cases} \end{cases}$

- 评价：最常用、表现最好的n-gram平滑方法之一
- 例子：<https://medium.com/@dennyc/a-simple-numerical-example-for-kneser-ney-smoothing-nlp-4600addf38b8>
- 改良版：modified Kneser-Ney Smoothing（对于n-gram的不同n，使用不同的discount，而非原版固定的discount）

3.5 大语言模型和stupid backoff

- n-gram模型在语料库很大时需要考虑效率问题，有一些改良办法
 - 将单词本身的存储改成int64型的哈希值
 - 剪枝
 - 设置n-gram计数的阈值，仅存储大于阈值的n-gram
 - 用布隆过滤器设置近似的语言模型
- 大语言模型下，一个名为stupid backoff 的简单算法往往就足够了。

- $S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{count(w_{i-k+1}^i)}{count(w_{i-k+1}^{i-1})} & \text{if } count(w_{i-k+1}^i) > 0 \\ \lambda S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$
- 起始条件unigram: $S(w) = \frac{count(w)}{N}$
- 经验值: $\lambda = 0.4$

3.6 Perplexity 和 Entropy 的关系

- 熵 (Entropy) :
 - 给定一个随机变量 X ，其分布的集合为 χ ，概率函数为 $p(x)$ ，定义熵为

$$H(X) = - \sum_{x \in \chi} p(x) \log_2 p(x)$$
 - 将语言看作一个生成词序列的随机过程 L ，则

$$H(L) = - \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{W \in L} p(w_1, \dots, w_n) \log p(w_1, \dots, w_n)$$
 - 假设语言具有平稳性（概率与序列时刻无关，实际并不是的）和遍历性，可近似简化为

$$H(L) = \lim_{n \rightarrow \infty} -\frac{1}{n} \log p(w_1, \dots, w_n)$$
- 交叉熵 (Cross-Entropy) :
 - 在我们不知道数据生成的概率分布 p 时，依然有用，可以衡量简单模型 m 和复杂的概率分布 p 之间的交叉熵，并通过最小化这个值使模型 m 去拟合 p
 - $H(p) \leq H(p, m) = \lim_{n \rightarrow \infty} -\frac{1}{n} \log m(w_1 w_2 \dots w_n)$
- Perplexity 和 Cross-Entropy的关系
 - $H(W) = -\frac{1}{N} \log P(w_1 w_2 \dots w_N)$
 - $PP(W) = 2^{H(W)}$

Chap 4 朴素贝叶斯和情绪分类

4.1 贝叶斯分类器

- 多元贝叶斯分类器 (multinomial naive Bayes Classifier)

- 贝叶斯推断公式: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

- 两个假设
 - 词袋假设: 词的作用与词所处位置无关
 - 朴素贝叶斯假设: 特征 f_1, f_2, \dots, f_n 互相独立

- 给定类别集合 C , 对于由特征 f_1, f_2, \dots, f_n 表示的文档 d , 预估分类

$$\hat{c} = \arg \max_{c \in C} P(c|d) = \arg \max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg \max_{c \in C} P(d|c)P(c) = \arg \max_{c \in C} P(c) \prod_{f \in F} P(f|c)$$

- 若特征 f 即为词本身, 则 $c_{NB} = \arg \max_{c \in C} P(c) \prod_{i \in positions} P(w_i|c)$

- 实际运算时常采用对数形式以提速且避免溢出:

$$c_{NB} = \arg \max_{c \in C} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

4.2 训练朴素贝叶斯分类器

- 使用最大似然估计, 用频率估计概率

- 假设 N_c 为训练数据中类别为 c 的文档个数, N_{doc} 为训练集中文档总数, 则 $\hat{P}(c) = \frac{N_c}{N_{doc}}$

- 同理, 有 $\hat{P}(w_i|c) = \frac{\text{count}(w_i,c)}{\sum_{w \in V} \text{count}(w,c)}$

- 但为了避免0概率的问题, 使用 Laplace 平滑, 有 $\hat{P}(w_i|c) = \frac{1+\text{count}(w_i,c)}{|V|+\sum_{w \in V} \text{count}(w,c)}$

- 对于测试集中发现的, 训练集词表中没有的新词, 忽略之

- 对于无意义的高频词, 忽略之

```

function TRAIN NAIVE BAYES(D, C) returns log  $P(c)$  and log  $P(w|c)$ 

for each class  $c \in C$            # Calculate  $P(c)$  terms
     $N_{doc}$  = number of documents in D
     $N_c$  = number of documents from D in class c
     $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$ 
     $V \leftarrow$  vocabulary of D
     $bigdoc[c] \leftarrow \text{append}(d)$  for  $d \in D$  with class  $c$ 
        for each word  $w$  in  $V$            # Calculate  $P(w|c)$  terms
             $count(w,c) \leftarrow$  # of occurrences of  $w$  in  $bigdoc[c]$ 
             $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$ 
return  $logprior, loglikelihood, V$ 

function TEST NAIVE BAYES( $testdoc, logprior, loglikelihood, C, V$ ) returns best  $c$ 

for each class  $c \in C$ 
     $sum[c] \leftarrow logprior[c]$ 
    for each position  $i$  in  $testdoc$ 
         $word \leftarrow testdoc[i]$ 
        if  $word \in V$ 
             $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$ 
return  $\text{argmax}_c sum[c]$ 

```

4.3 情绪分类优化

- 改进1：一个词在文档中是否出现，比它出现多少次重要，故限制词频在每个文档的取值为0或1
- 改进2：否定词的出现可能会翻转整个句子的情感类别，故在词正则化阶段，对否定词后的词语添加NOT_前缀，直到出现标点符号
- 改进3：对于标注的训练数据不足时，可从专用的情绪词汇表中抽取正负倾向情绪特征来使用，四大流行情绪词表为 General Inquirer (1966) , LIWC (2007) , opinion lexicon (2004) , MPQA Subjectivity Lexicon (2005) ; MPQA中包含6885个词语，2718个正向、4912个负向；使用词表的时候，添加特征“该词出现在（正/负）向词表中”；当训练数据稀疏且与测试集分布差异较大时，可使用“出现在正负向词表中的词的个数”来代替单个词语的特征，以达到更好的泛化效果

4.4 朴素贝叶斯用于其他分类任务

- 对于垃圾邮件分类，使用预定义的短语作为特征，并结合一些非语言学特征，如HTML中的text部分占比，能达到更好的效果
- 对于语言种类分类，使用字母n-grams是比单个词汇更有效的特征
- 对于句子情感分类，使用基于单词的朴素贝叶斯即可

4.5 评价指标：准确率，召回率，F-指标

		gold standard labels		
		gold positive	gold negative	
system output labels	system positive	true positive	false positive	precision = $\frac{tp}{tp+fp}$
	system negative	false negative	true negative	
		recall = $\frac{tp}{tp+fn}$		accuracy = $\frac{tp+tn}{tp+fp+tn+fn}$

$F_\beta = \frac{(\beta^2+1)PR}{\beta^2P+R}$, 其中, 当 $\beta = 1$ 时, 对于准确率和召回率具有同等重要性, 故常用指标 F_1 作为衡量指标, 里面体现的是加权几何平均的思想, 多分类的混淆矩阵如下

		gold labels			
		urgent	normal	spam	
system output	urgent	8	10	1	precision _u = $\frac{8}{8+10+1}$
	normal	5	60	50	precision _n = $\frac{60}{5+60+50}$
	spam	3	30	200	precision _s = $\frac{200}{3+30+200}$
		recall _u = $\frac{8}{8+5+3}$	recall _n = $\frac{60}{10+60+30}$	recall _s = $\frac{200}{1+50+200}$	

MacroAveraging: 计算每个分类的性能指标, 再求平均作为总体性能指标 (类别重要性一致, 更合适)

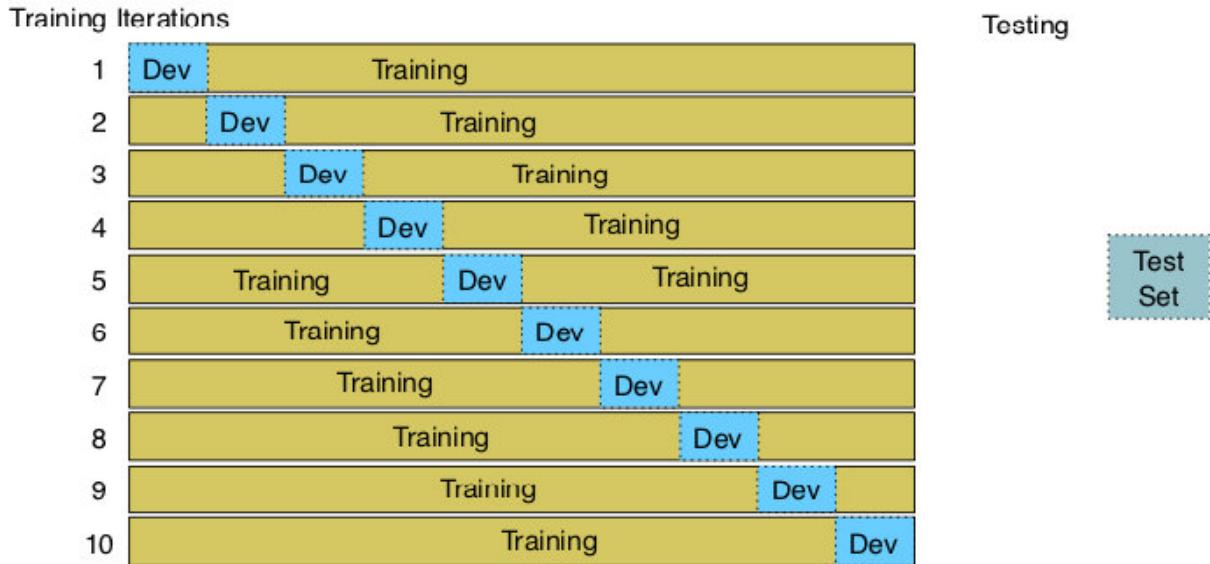
MicroAveraging: 收集所有类至一个混淆矩阵, 然后计算准确率和召回率 (受大类影响较大)

Class 1: Urgent		Class 2: Normal		Class 3: Spam		Pooled		
true	true	true	true	true	true	true	true	
urgent	not	normal	not	spam	not	yes	no	
system urgent	8	11	system normal	60	55	system spam	200	33
system not	8	340	system not	40	212	system not	51	83

$$\text{precision} = \frac{8}{8+11} = .42 \quad \text{precision} = \frac{60}{60+55} = .52 \quad \text{precision} = \frac{200}{200+33} = .86 \quad \text{microaverage precision} = \frac{268}{268+99} = .73$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60$$

使用训练-验证-测试集划分, 并使用交叉验证技术来衡量模型性能



4.6 统计显著性检测

- 在测试数据集 x 上，用评价指标 M ，比较两个分类器A和B的表现：

$$\delta(x) = M(A, x) - M(B, x)$$
- 形式化两个假设 $H_0 : \delta(x) \leq 0$ $H_1 : \delta(x) > 0$
- NLP领域有两个常用的非参数化检验：approximate randomization (1989) , bootstrap test (1993)

配对bootstrap test

- 考虑一个文档二分类问题，给定真实测试集 x 中包含10个文档，给定两个分类器A和B，A和B对于文档存在分类正确错误的4种组合情况，构造 b 个虚拟测试集的方法是每次从真实测试集中随机抽样并放回，如下表所示

	1	2	3	4	5	6	7	8	9	10	A%	B%	$\delta()$
x	AB	.70	.50	.20									
$x^{(1)}$	AB	.60	.60	.00									
$x^{(2)}$	AB	.60	.70	-.10									
...													
$x^{(b)}$													

- $p-value(x) = \sum_{i=1}^b \mathbb{I}(\delta(x^{(i)}) \geq 2\delta(x))$ ，若该值小于一定阈值，则表明 $\delta(x)$ 足够显著，可以说明两个分类器表现的好和坏

```

function BOOTSTRAP(test set  $x$ , num of samples  $b$ ) returns  $p\text{-value}(x)$ 

    Calculate  $\delta(x)$  # how much better does algorithm A do than B on  $x$ 
     $s = 0$ 
    for  $i = 1$  to  $b$  do
        for  $j = 1$  to  $n$  do # Draw a bootstrap sample  $x^{(i)}$  of size n
            Select a member of  $x$  at random and add it to  $x^{(i)}$ 
        Calculate  $\delta(x^{(i)})$  # how much better does algorithm A do than B on  $x^{(i)}$ 
         $s \leftarrow s + 1$  if  $\delta(x^{(i)}) > 2\delta(x)$ 
     $p\text{-value}(x) \approx \frac{s}{b}$  # on what % of the b samples did algorithm A beat expectations?
    return  $p\text{-value}(x)$  # if very few did, our observed  $\delta$  is probably not accidental

```

Chap 5 逻辑回归

- 逻辑回归是NLP分类问题中的baseline算法，是判别式的分类器，它包含两部分
 - 训练：使用随机梯度下降（SGD）算法和交叉熵损失函数（cross-entropy loss）来训练
 - 测试：给定测试样例 x ，模型计算 $P(y|x)$ ，并返回较高概率的类别 y

5.1 分类：sigmoid

- 输入样例 x 可以表示为一个特征向量 $x = [x_1, x_2, \dots, x_n]$
- 逻辑回归学习一个权重向量 $w = [w_1, w_2, \dots, w_n]$ 和截距 b
 - $z = w \cdot x + b$
 - 使用sigmoid函数将实数域输入映射到 $[0, 1]$ 区间的输出 $y = \sigma(z) = \frac{1}{1+e^{-z}}$
- 逻辑回归方法需要手工设计特征交叉，对于大多数任务，我们需要大量特征（从特征模板中构建出来），为了避免特征设计的人力浪费，NLP中有专门研究表示学习的方向，使用无监督方法自动学习输入的特征
- 逻辑回归对于特征相关性强的问题较为鲁棒，而贝叶斯方法假设特征互不相关；故对于大数据集，逻辑回归表现一般较好，对于小数据集，朴素贝叶斯一般表现较好；朴素贝叶斯易于实现且训练快速（无需优化步骤），在某些情况下，依然是合理的方法

5.2 交叉熵损失函数

- $L(\hat{y}, y)$ ：衡量模型估计值 \hat{y} 与真实值 y 的差异
- 通过学习权重，最大化正确标签的概率 $p(y|x)$ ，这是一个伯努利分布，

$$p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$
- $L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))]$

5.3 梯度下降

- 求解目标是一组参数，使得 $\hat{\theta} = \arg \min_{\theta} \frac{1}{m} \sum_{i=1}^m L_{CE}(f(x^{(i)}; \theta), y^{(i)})$
 - 对于逻辑回归，损失函数是凸的，仅有一个极小值点为全局最小值
 - 但对于神经网络，损失函数是非凸的，故容易陷入局部极小值点而无法找到全局最小值
 - 拓展到多维空间，学习过程为 $\theta^{t+1} = \theta^t - \eta \nabla_{\theta} L(f(x; \theta), y)$
- 梯度： $\nabla_{\theta} L(f(x; \theta), y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x; \theta), y) \\ \frac{\partial}{\partial w_2} L(f(x; \theta), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x; \theta), y) \end{bmatrix}$
- 学习率： η ，过大则训练阶段损失函数的值不收敛，过小则学习时间过长
- 使用mini-batch training，梯度下降方向为当前batch样例梯度的算术平均

逻辑回归的梯度

$$\frac{\partial L_{CE}(\hat{y}, y)}{\partial w_j} = [\sigma(w \cdot x + b) - y] x_j$$

随机梯度下降算法

```
function STOCHASTIC GRADIENT DESCENT(L(), f(), x, y) returns θ
    # where: L is the loss function
    #     f is a function parameterized by θ
    #     x is the set of training inputs x(1), x(2), ..., x(m)
    #     y is the set of training outputs (labels) y(1), y(2), ..., y(m)

    θ ← 0
    repeat til done  # see caption
        For each training tuple (x(i), y(i)) (in random order)
            1. Optional (for reporting):      # How are we doing on this tuple?
                Compute  $\hat{y}^{(i)} = f(x^{(i)}; \theta)$   # What is our estimated output  $\hat{y}$ ?
                Compute the loss  $L(\hat{y}^{(i)}, y^{(i)})$   # How far off is  $\hat{y}^{(i)}$  from the true output  $y^{(i)}$ ?
            2.  $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$   # How should we move  $\theta$  to maximize loss?
            3.  $\theta \leftarrow \theta - \eta g$   # Go the other way instead
    return θ
```

5.4 正则化

训练集中，若某特征仅恰巧在单一类别中出现，它在逻辑回归的训练过程中，会被赋予很大的权重，而这有可能是巧合，从而导致过拟合问题，为了避免过拟合，在目标函数中引入正则项 $R(\theta)$ ，用于惩罚大权重值

- $\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^m \log P(y^{(i)} | x^{(i)}) - \alpha R(\theta)$

正则项可以分为L2正则和L1正则，L2正则的逻辑回归又称岭回归，L1正则的逻辑回归又称Lasso回归；L2回归因为连续可导而易于优化，L1在零点不连续；L2的结果一般是多项小权重向量，L1的结果一般是一些权重值较大，一些权重值为0的权重向量

- L2正则： $R(\theta) = \|\theta\|_2^2 = \sum_{j=1}^n \theta_j^2$
- L1正则： $R(\theta) = \|\theta\|_1 = \sum_{i=1}^n |\theta_i|$

L1正则化可通过假设权重w的先验分布为拉普拉斯分布，由最大后验概率估计导出

- $P(w_j) = \frac{1}{\sqrt{2a}} \exp(-\frac{|w_j|}{a})$
- $\log P(w) = \log \prod_j P(w_j) = -\frac{1}{a} \sum_j |w_j| + C'$

L2正则化可通过假设权重w的先验分布为高斯分布，由最大后验概率估计导出

- $w_j \sim N(0, \sigma^2)$
- $\log P(w) = \log \prod_j P(w_j) = \log \prod_j [\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{w_j^2}{2\sigma^2})] = -\frac{1}{2\sigma^2} \sum_j w_j^2 + C'$

5.5 多元逻辑回归

- softmax操作：

- $\circ \text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^k \exp(z_j)} \quad 1 \leq i \leq k$

- 损失函数：

- $\circ L_{CE}(\hat{y}, y) = -\sum_{k=1}^K y_k \log \hat{y}_k = -\sum_{k=1}^K \mathbb{I}\{y=k\} \log \hat{p}(y=k|x) = -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^K \exp(w_j \cdot x + b_j)}$
(其中k是正确类别)

- 梯度：

- $\circ \frac{\partial L_{CE}}{\partial w_{k,i}} = -(\mathbb{I}\{y=k\} - p(y=k|x))x_i = -(\mathbb{I}\{y=k\} - \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^K \exp(w_j \cdot x + b_j)})x_i$

Chap 6 向量语义和嵌入表示

6.1 词汇语义

- 词相似度：SimLex-999数据集（2015）给词对赋予0-10的相似度分值
- 词相关性：两个词是否属于一个语义场，如医院（医生，护士，急救）
- 语义帧和角色：语义帧是某个特定活动的观点和参与者构成的词集合，语义帧中存在角色，句子中的词可以扮演这些角色，如A从B那里买了一本书（A是买家，B是卖家）

6.2 词向量语义学

- 词向量语义学的任务是在多维语义空间中找到一个点来表示词语，这个词语在多维空间中的向量表示叫做嵌入 (embeddings)



- 信息检索：给定文档集 D 和查询 q ，从文档集中返回最佳适配的文档 d ；

6.3 词和向量

- 词-文档矩阵：行数为词表大小 $|V|$ ，列数为文档个数 $|D|$ ， $M_{i,j}$ 表示词 i 在文档 j 中出现的次数

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- 从词语的角度而言，battle可以表示为向量[1,0,7,13]
- 从文档的角度而言，As you Like It 可以表示为向量 [1, 114, 36, 20]
- 词-词共现矩阵：行列数均为词表大小， $M_{i,j}$ 表示词 i 和词 j 在上下文（篇章、段落、句子等粒度皆可）中同时出现的次数

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

- 矩阵稀疏程度非常高

6.4 相似性衡量指标：余弦函数

- 内积： $(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^N v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_N w_N$
 - 向量越长，则内积越大，但为了衡量相似性，需要对其进行归一化
- 余弦： $\cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|}$

- 对于单位向量，它们的余弦相似度即为它们的内积

6.5 TF-IDF

- 算法

- 词频 (Term Frequency, TF) = 某个词在文档中出现的次数 / 文档的总词数 (在 Lucene 的实现中对 TF 开根号归一化)
- 逆文档频率 (Inverted Document Frequency, IDF) = $\log(\frac{\text{语料库文档总数}}{\text{包含该词的文档数} + 1})$
- $TF - IDF = TF * IDF$, 代表某个词在某个文档中的权重值 (比直接用词频效果好)

- 应用

- 在摘要场景下，根据文档中所有词语的TF-IDF值的大小，可以确定文档的关键词
- 在搜索场景下，可根据查询 q 涉及的关键词(k_1, \dots, k_n)在文档 d 中的TF-IDF值，建立相似度函数， $Similarity(q, d) = \sum_{i=1}^n TF - IDF(k_i, d)$
- 常用做NLP任务的baseline算法

6.6 Pointwise Mutual Information (PMI)

- 点互信息：衡量两个事件的相关性

- $PMI(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$
- 对于两个低概率事件，训练样本集不一定有同时出现的样本，容易导致 $I(x, y)$ 趋向负无穷，所以提出改进，正点互信息

- 正点互信息

- $PPMI(x, y) = \max(PMI(x, y), 0)$
- 例子：词-词共现矩阵（上面的是频率，下面的是PPMI）

■

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

■

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

■ 计算过程

- $P(w = \text{information}, c = \text{data}) = \frac{3982}{11716} = 0.3399$
 - $P(w = \text{information}) = \frac{7703}{11716} = 0.6575$
 - $P(c = \text{data}) = \frac{5673}{11716} = 0.4842$
 - $PPMI(\text{information}, \text{data}) = \log_2 \frac{0.3399}{0.6575 * 0.4842} = 0.0944$
 - 能反应较好的词-词关联性，但存在一定偏见，即对于罕见事件，会赋予较高的PPMI值

6.7 TF-IDF和PPMI向量模型的应用

- TF-IDF模型常用于衡量两个文档的相似度，将文档的所有词向量做算术平均得到文档向量，再使用余弦相似度衡量两个文档向量的相似度；用于信息检索、作弊检测、新闻推荐等下游任务
 - PPMI模型可以寻找语料集中的近义词

6.8 Word2vec

- embeddings是较短的，稠密的向量，而非之前提及的维度为词汇表大小或文档集基数的稀疏向量；但embeddings并没有直观的解释
 - 稠密向量需要学习较少的权重值，且对于同义的捕获更为鲁棒
 - word2vec是一个运用自监督方法的语言向量表示方法
 - skip-gram的思想：
 - 1、将目标词和它的邻居上下文词作为正样本(w, c)
 - 2、从词汇表中随机采样其他词作为负样本（下图正负样本采样比例为2）
 - 3、使用逻辑回归去训练一个二分类器
 - 4、使用学习好的权重作为embedding
 - Word2vec可以看作是PPMI矩阵的一个隐式优化版本

分类器

... lemon, a [tablespoon of apricot jam, a] pinch ...
c1 c2 w c3 c4

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

- skip-gram 模型：

- 假设：一个词 c 若经常出现在目标词 w 周围，则 c 的词嵌入和 w 的词嵌入是相似的
- 算法：给定一个目标词 w ，和它的上下文窗口里的 L 个词（该超参数需要调节，小了容易捕获句法相关，大了容易捕获话题相关）

$$\begin{aligned} \bullet \quad P(+|w, c) &= \sigma(c \cdot w) = \frac{1}{1+exp(-c \cdot w)} \\ \bullet \quad \log P(+|w, c_{1:L}) &= \sum_{i=1}^L \log \sigma(-c_i \cdot w) \end{aligned}$$

- 结果：得到两类embedding，一类是作为目标词的，一类是作为上下文词的；每一类embedding都包含 $|V|$ 个词，实际中，可以将两类embedding加起来，或者抛弃上下文词embedding，作为最终embedding

- 负采样

- 根据加权的unigram-frequency进行负采样： $p_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$
- 经验值 $\alpha = 0.75$ 表现较好，因为它能给罕见的词语赋予较高的概率， $P_\alpha(w) > P(w)$

- 训练目标：最大化正样本对的相似度，最小化负样本对的相似度

- 损失函数：

$$L_{CE} = -\log[P(+|w, c_{pos}) \prod_{i=1}^k P(-|w, c_{neg_i})] = -[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^k \log \sigma(-c_{neg_i} \cdot w)]$$

- 求解梯度：

$$\begin{aligned} \circ \quad \frac{\partial L_{CE}}{\partial c_{pos}} &= [\sigma(c_{pos} \cdot w) - 1]w & \frac{\partial L_{CE}}{\partial c_{neg}} &= [\sigma(c_{pos} \cdot w)]w \\ \circ \quad \frac{\partial L_{CE}}{\partial w} &= [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^k [\sigma(c_{neg_i} \cdot w)]c_{neg_i} \end{aligned}$$

其他静态词嵌入

- fasttext (2017) 使用subword models 去处理未知词汇和稀疏性问题，如where可以表示为自身和连续的n-gram组合 <wh, whe, her, ere, re>；对于每个连续的n-gram，通过skipgram学习它的embedding，最后，将该词所有n-gram的embedding加起来作为该词的embedding； fasttext 提供157种语言的预训练embeddings
- Glove (2014) ,Global Vectors的简称，是广泛使用的embedding模型，Glove是基于词词共现矩阵中的概率分布，结合如PPMI的count-based 模型，且能捕获线性结构

6.9 词嵌入可视化

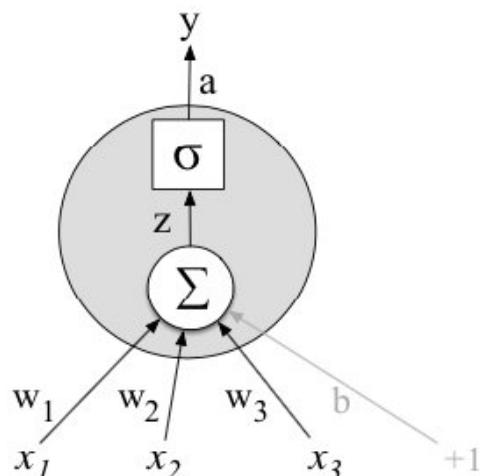
- 最简单的办法：根据词嵌入向量的相似性分数，列出最相近的词语列表
- 另一种办法：使用聚类算法反应词嵌入空间中的层次化表示
- 最常用的办法：使用t-SNE等投影算法，将高维向量投影到二维空间

6.10 评价词向量模型

- 相似度测试：WordSim-353数据集（2002），SimLex-999数据集（2015）
- 上下文测试：Stanford Contextual Word Similarity数据集（2012），Word-in-Context数据集（2019）
- 类比测试：给定 $a-b$, 求解 $a^* - b^*$ 的问题，如SemEval-2012 Task 2 dataset
- 考虑到词向量模型训练的随机性，可以通过在文档中采取bootstrap sampling训练多个 embeddings，并求平均的方法，以达到更好的效果

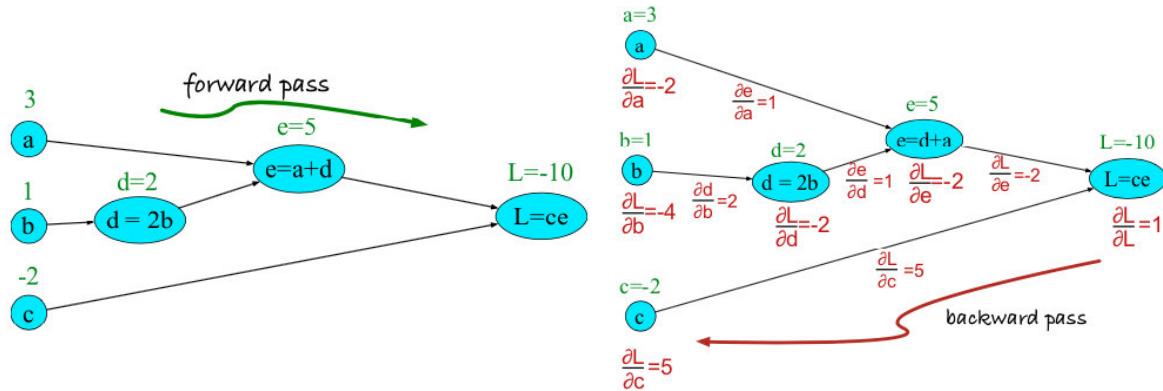
Chap 7 神经网络和神经语言模型

7.1 神经元和前向神经网络

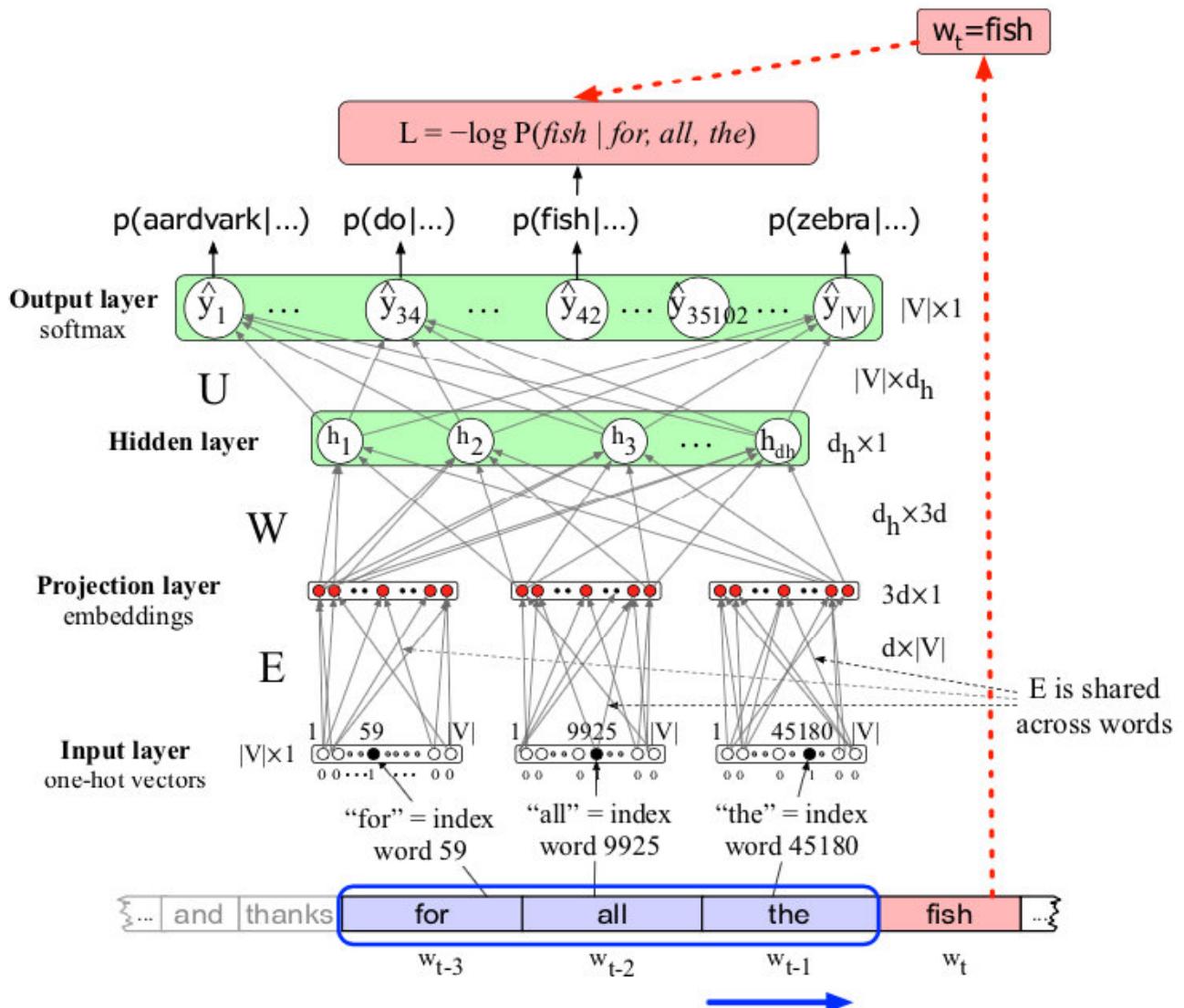


- $y = Activation(z) \quad z = w \cdot x + b$
- 激活函数
 - 常见种类：
 - Sigmoid: $\sigma z = \frac{1}{1+e^{-z}}$
 - Tanh: $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
 - Relu: $RELU(z) = \max(z, 0)$
 - 常见问题：
 - 关注斜率，太靠近0会导致梯度消失问题，如sigmoid和tanh都会有该问题
 - 异或问题是线性不可分的，单个神经元不能解决这个问题
- 前向神经网络 (feedforward network)：无环的多层网络，又名多层感知机 (MLP)

- 计算图：



7.2 神经语言模型



- 一般使用one-hot encoding作为原始输入，通过反向传播去train一个one hot encoding到特定维数embedding的矩阵E
- 训练过程为 $\theta^{s+1} = \theta^s - \eta \frac{\partial(-\log P(w_t | w_{t-1} \dots w_{t-n+1}))}{\partial \theta}$

Chap 8 词性标注和命名实体识别

8.1 英语词汇分类

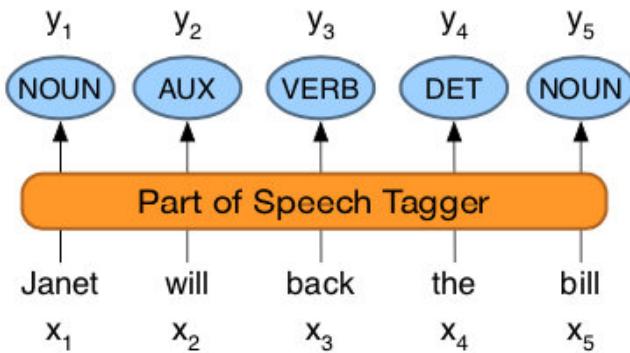
	Tag	Description	Example
Open Class	ADJ	Adjective: noun modifiers describing properties	<i>red, young, awesome</i>
	ADV	Adverb: verb modifiers of time, place, manner	<i>very, slowly, home, yesterday</i>
	NOUN	words for persons, places, things, etc.	<i>algorithm, cat, mango, beauty</i>
	VERB	words for actions and processes	<i>draw, provide, go</i>
	PROPN	Proper noun: name of a person, organization, place, etc..	<i>Regina, IBM, Colorado</i>
Closed Class Words	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	<i>oh, um, yes, hello</i>
	ADP	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	<i>in, on, by under</i>
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	<i>can, may, should, are</i>
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	<i>and, or, but</i>
	DET	Determiner: marks noun phrase properties	<i>a, an, the, this</i>
	NUM	Numeral	<i>one, two, first, second</i>
	PART	Particle: a preposition-like form used together with a verb	<i>up, down, on, off, in, out, at, by</i>
	PRON	Pronoun: a shorthand for referring to an entity or event	<i>she, who, I, others</i>
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	<i>that, which</i>
Other	PUNCT	Punctuation	<i>, , ()</i>
	SYM	Symbols like \$ or emoji	<i>\$, %</i>
	X	Other	<i>asdf, qwfg</i>

- closed class: 多为语法上的功能词汇, 较短, 高频出现
- open class: 能被持续创造或引申

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	<i>and, but, or</i>	NNP	proper noun, sing.	<i>IBM</i>	TO	"to"	<i>to</i>
CD	cardinal number	<i>one, two</i>	NNPS	proper noun, plu.	<i>Carolinas</i>	UH	interjection	<i>ah, oops</i>
DT	determiner	<i>a, the</i>	NNS	noun, plural	<i>llamas</i>	VB	verb base	<i>eat</i>
EX	existential 'there'	<i>there</i>	PDT	predeterminer	<i>all, both</i>	VBD	verb past tense	<i>ate</i>
FW	foreign word	<i>mea culpa</i>	POS	possessive ending	<i>'s</i>	VBG	verb gerund	<i>eating</i>
IN	preposition/ subordin-conj	<i>of, in, by</i>	PRP	personal pronoun	<i>I, you, he</i>	VBN	verb past participle	<i>eaten</i>
JJ	adjective	<i>yellow</i>	PRP\$	possess. pronoun	<i>your, one's</i>	VBP	verb non-3sg-pr	<i>eat</i>
JJR	comparative adj	<i>bigger</i>	RB	adverb	<i>quickly</i>	VBZ	verb 3sg pres	<i>eats</i>
JJS	superlative adj	<i>wildest</i>	RBR	comparative adv	<i>faster</i>	WDT	wh-determ.	<i>which, that</i>
LS	list item marker	<i>1, 2, One</i>	RBS	superlatv. adv	<i>fastest</i>	WP	wh-pronoun	<i>what, who</i>
MD	modal	<i>can, should</i>	RP	particle	<i>up, off</i>	WP\$	wh-possess.	<i>whose</i>
NN	sing or mass noun	<i>llama</i>	SYM	symbol	<i>+, %, &</i>	WRB	wh-adverb	<i>how, where</i>

- 另一种词性分类体系

8.2 词性标注



- 词性标注：给定序列 x_1, \dots, x_n ，和标签集，输出每个元素对应的标签 y_1, \dots, y_n
 - 考虑到某些词，如book，既可以是名词，也可以是动词，词性标注就是通过上下文来精确定性词汇，避免模棱两可的情况
 - 对于词性标注任务，HMMs, CRFs, BERT等算法和人类表现基本一致，准确率约为97%
 - 给定测试集中一个词，选择它在训练集中最频繁的标注类型，准确率能有92%（baseline）

8.3 命名实体和命名实体识别

[PER Jane Villanueva] of [ORG United], a unit of [ORG United Airlines Holding], said the fare applies to the [LOC Chicago] route.

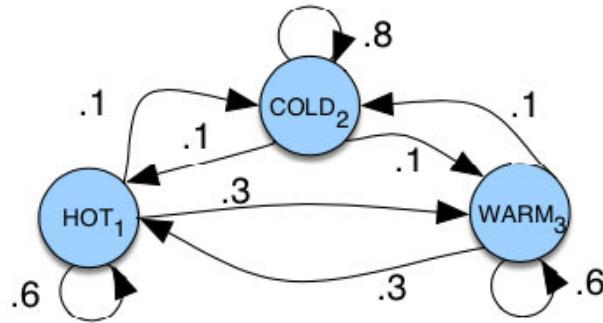
Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

- 命名实体识别（NER）：寻找特定类型实体
 - BIO 标注方法（1995）：将寻找离散实体段的任务转化为序列连续标注任务

8.4 隐马尔可夫模型（HMM）

- 马尔可夫假设：未来与过去无关，仅与当下相关
- $P(q_i = a | q_1 \dots q_{i-1}) = P(q_i = a | q_{i-1})$
- 马尔可夫链：包含状态集、转移概率矩阵、初始概率分布三个要素

•



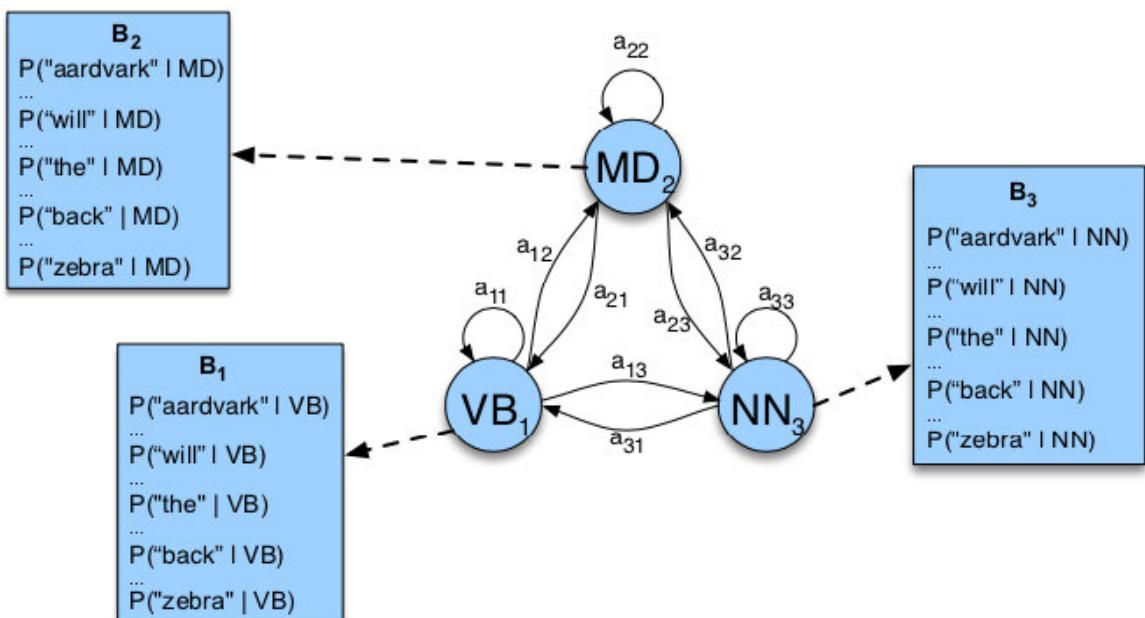
- 隐马尔可夫模型

- 很多情况下，我们关注的事件是隐藏的，隐马尔可夫模型允许我们将观测到的事件和隐藏的事件作为概率模型的作用因子，它是一个生成式模型
- 隐马尔可夫模型组成元素

$Q = q_1 q_2 \dots q_N$	a set of N states
$A = a_{11} \dots a_{ij} \dots a_{NN}$	a transition probability matrix A , each a_{ij} representing the probability of moving from state i to state j , s.t. $\sum_{j=1}^N a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	a sequence of T observations , each one drawn from a vocabulary $V = v_1, v_2, \dots, v_N$
$B = b_i(o_t)$	a sequence of observation likelihoods , also called emission probabilities , each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \dots, \pi_N$	an initial probability distribution over states. π_i is the probability that the Markov chain will start in state i . Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^n \pi_i = 1$

- 隐马尔可夫标签模型

- 标签tag转移概率 $P(t_i | t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$ (频率估计概率) , 常从大语料集中抽取出来
- HMM表示的两部分，A为标签转移概率矩阵，用于计算先验概率，B为与每个标签关联的词语观测概率



- 标签转移概率矩阵A (从大语料集中获得, 通用)

	NNP	MD	VB	JJ	NN	RB	DT
<S>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

- 标签关联词语观测概率B (从训练集中获得)

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

- 两个假设及序列估计公式

- 假设一：一个词出现的概率依赖于它自己的标签，和上下文的词和标签无关：

$$P(w_1 \dots w_n | t_1 \dots t_n) \approx \prod_{i=1}^n P(w_i | t_i)$$

- 假设二：一个词的标签仅依赖于上一个词的标签 (bigram)

$$P(t_1 \dots t_n) \approx \prod_{i=1}^n P(t_i | t_{i-1})$$

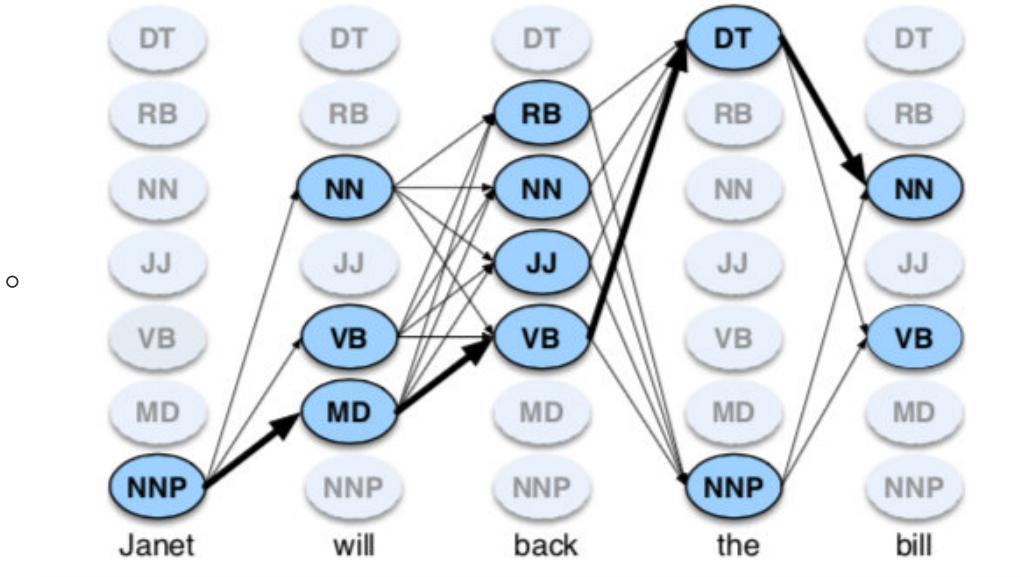
- 序列估计 (最后分别对应概率矩阵A和B) :

$$\begin{aligned} \hat{t}_{1:n} &= \arg \max_{t_1 \dots t_n} P(t_1 \dots t_n | w_1 \dots w_n) = \arg \max_{t_1 \dots t_n} \frac{P(w_1 \dots w_n | t_1 \dots t_n) P(t_1 \dots t_n)}{P(w_1 \dots w_n)} \\ &= \arg \max_{t_1 \dots t_n} P(w_1 \dots w_n | t_1 \dots t_n) P(t_1 \dots t_n) \\ &\approx \arg \max_{t_1 \dots t_n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1}) \end{aligned}$$

- 不足之处

- 需要对概率矩阵做平滑处理；难以融合其他特征；无法处理未知词汇

- 维特比算法：通过动态规划的方式寻找最优标签序列 (先将概率从左推到右，再回溯产生最大概率的标签路径)



8.5 条件随机场 (CRFs)

- 条件随机场：判别式模型，直接计算，概率是 K 个全局特征 F 的加权softmax值，每个全局特征是由对应的函数 f 在运用完整输入 X ，窗口为2的标签序列，标签位置求和得到的，这种又叫线性链随机场，局部特征函数 f 可以自由定义并组合多种人工特征

$$\circ F_K(X, Y) = \sum_{i=1}^n f_k(y_{i-1}, y_i, X, i)$$

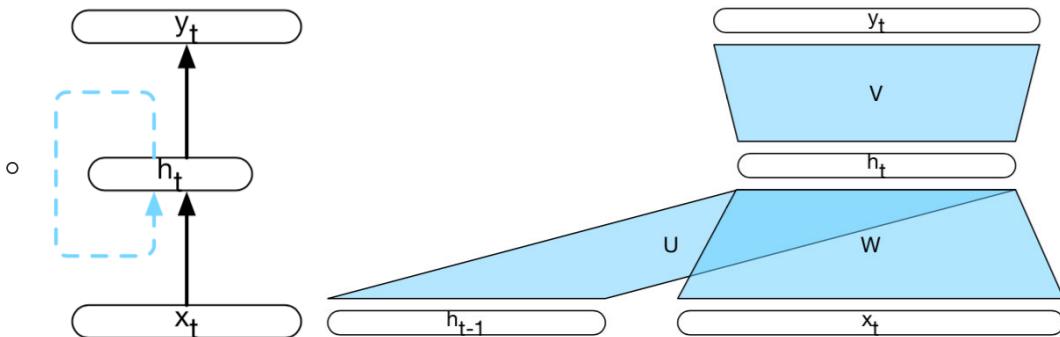
$$\circ P(Y|X) = \frac{\exp(\sum_{k=1}^K w_k F_k(X, Y))}{\sum_{Y' \in \mathcal{Y}} \exp(\sum_{k=1}^K w_k F_k(X, Y'))}$$

$$\circ \hat{Y} = \arg \max_{y \in \mathcal{Y}} P(Y|X) = \arg \max_{Y \in \mathcal{Y}} \sum_{i=1}^n \sum_{k=1}^K w_k f_k(y_{i-1}, y_i, X, i)$$

Chap 9 序列处理的深度学习架构

9.1 循环神经网络 (RNN)

- 结构特点：网络连接中有环，训练较难



- 数学表达：

- 给定输入维度 d_{in} ；隐藏层维度 d_h ；输出维度 d_{out} ；权重矩阵 $W \in \mathbb{R}^{d_h \times d_{in}}$ ；

$$U \in \mathbb{R}^{d_h \times d_h}; V \in \mathbb{R}^{d_{out} \times d_h}$$

- $h_t = g(Uh_{t-1} + Wx_t)$ $y_t = f(Vh_t)$
- 对于分类问题， f 通常是softmax函数