6.1 Probabilities

首先介绍概率对自然语言处理的重要性，在于很多情况下我们直接得到的可能是一系列候选结果，需要从概率的意义上做出更合理的选择。概率的角度也使得我们更容易将不同类型的证据结合以获得更好的结果。

接下来介绍了关于概率的基础知识，离散（连续）概率分布、条件分布、联合分布，等等，复习了一遍无需赘述。

6.2 Bayes' Theorem

本节介绍贝叶斯理论，回顾一下这一经典方法即可。

p(A,B) = p(B|A)p(A), and p(A,B) = p(A|B)p(B), so p(B|A)=p(A|B)p(B)/p(A).

不要忘记实际使用时，分母 p(A)常常是累加求得的。

6.3-6.4 Language models 1/3

所谓概率语言模型（Probabilistic Language Model），是计算给定句子概率的方法（assign a probability to a sentence）。其应用广泛，如 predicting the next word、Speech recognition、Speech recognition、Speech recognition、Machine translation 等，该模型在这些应用中发挥的重要作用不言而喻。

我们知道语句的精确概率模型是单词序列的联合概率分布，形如 P(S)=P(w1,w2,w3…wn)，或者写成条件概率 P(S)= P(w1) P(w2|w1) … P(wn|w1,w2…wn-1)。但是训练集总是相对稀疏的，目标语句可能无法利用样本频率直接估计。N-gram Model 是一种折衷（sacrifice some of the accuracy of the prediction but get very good performance and deal properly with sparse training data）。该模型限定了决定当前单词概率的单词个数为 N-1（加上当前单词所以是 N），其他单词将是不相关的（irrelevant）。显然，N 不能过大。

应当对真实数据中的 N-gram 有一种直观的印象。例如在 Shakespeare 的作品集中，unigrams 对应 29,524 types（种类数目），约 900K tokens（样本数目）；而 bigrams 对应 346,097

types，约 900K tokens。这说明平均每个 unigram type（即单词）对应约 10 个 bigram type，可见 collocation（搭配）的稀疏性。

N-gram 模型中的参数（即一系列条件概率）可以在训练集上用最大似然估计得到。为便于统一处理，一个技巧是对句子添加开头标记（如<S>）和结尾标记（如</S>），然后像估计 P(will|He)那样估计 P(He|<S>)与 P(</S>|sleep)。实际计算出的概率值一般很小（如 10^{-6}），必要时可在计算中使用对数运算。

另外，一个理论知识是：N-grams 模型都可以表示为 HMM（隐马尔科夫模型），而 HMM 与有限状态转移是等价的（HMM is equivalent to weighted finite state transitions），因此 N-grams 与正则语言（Regular Languages）是相似的，或者说具有正则语言的特点。

6.5 Language models 2/3

本节讨论概率语言模型中参数估计的平滑（smoothing），这一策略的出发点是对训练数据中未出现的样本赋予概率（reassigning some probability mass to unseen data）。

一种简单的平滑方式称为 Add-one (Laplace) smoothing。例如对于 Bigrams，$P(w_i|w_{i-1}) = (c(w_{i-1},w_i)+1)/(c(w_i-1)+V)$，其中 V 是整个词汇表单词数目（不是样本数目，相当于样本种类数目）。这种方式可以保证每个条件概率都不为 0，且满足概率的归一化条件（ensure that the bigram probability is still a valid probability and add up to 1）。但是这种方式对未出现的单词（或 n 元组）赋予的概率其实显得过大，因此效果较差，实际中很少应用。

实际中常用更为“先进”的方式（Advanced Smoothing），例如 Good-Turing、Kneser-Ney（未讲述）、Class-based n-grams 等。其中，的基本思路是，对训练集上未出现的单词/元组，利用同类单词/元组对其概率进行估计，所谓同类可能是基于词性或语义等。

对 Good-Turing 作了一定介绍。先利用公式计算修正后的频数 $c^* = (c+1)N_{c+1}/N_c$，再利用修正后频数计算概率。这里 c 是单词/元组真实的频数，N_c 是频数为 c 的单词/元组数目（即统计多少个单词/元组出现了 c 次），并规定 N_0 等于全体单词/元组数目。该方法直观上可获得两点认识：

1. 一般而言 $c^*<c$，因为实际数据一般 $N_c>N_{c+1}$，即只有少数单词出现次数较多；

2. 从未出现过的单词频数也大于 1，即 $c^*=N_1/N_0$，注意 N_0 是很大的数。

另外两种 smoothing 方法是 Backoff 和 Interpolation（添写、插补）。Backoff 是指对过于

稀疏的样本使用更低阶的 N-gram（lower-order n-gram），或者赋予默认概率值，这里判断阶数涉及到的阈值参数以及默认概率均可在 development(validation) data set 上得到。

Interpolation 相对 Backoff 更为有效，例如若 $P'(w_i|w_{i-1},w_{i-2})$稀疏，则使用多阶的线性组合 $\lambda_1P'(w_i|w_{i-1},w_{i-2})+\lambda_2P'(w_i|w_{i-1})+\lambda_3P'(w_i)$来平滑。Stanley Chen and Josh Goodman 1998 的文献中可以学习更多关于 Backoff 和 Interpolation 的细节。

对于 OOV（Out of vocabulary words）问题，即测试集出现新的单词的情况，6.6 节提到一种方法，即将训练样本分为 part1 和 part2，并将 part2 独有的单词标记为<UNK>（即 unknown），这样训练出的模型可以在必要的时候生成<UNK>，即待定单词。

6.6 Language models 3/3

本节首先介绍模型的评价。一个指标是混乱度（perplexity），定义为

$$Per = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

或

$$Per = 2^{-(1/N)\sum \log_2 P(w_i)}$$

显然当 p(wi) = 1/k 均匀分布时，Per = (1/k)^N^(-1/N) = k。而分布越不均匀，Per 越小，表示模型质量越高。

上面的只是 perplexity 评价了模型区别于均匀分布的程度。为了验证所得分布与目标分布是否一致，可以使用交叉熵（类似于 log(perplexity)）。交叉熵越小，两个分布越相近。

$$\mathrm{H}(p,q) = -\sum_x p(x) \log q(x)$$

应用 perplexity 的一个实际例子是对于 Wall Street Journal corpus，包含 38 M words 和 20 K types。在 1.5M 的样本上进行评价，Unigram、Bigram、Trigram 的 perplexity 依次为 962、170、109，显著地低于 20K。

另一个评价指标是之前介绍的 Levenshtein Edit Distance（Number of insertions, deletions, and substitutions），在这里称为 Word Error Rate，用于衡量所得语句与目标语句之间的差异。

下面，讨论 N-grams 存在的一些问题。首先是它无法捕捉 long distance dependencies，这是它固有的缺陷。一种可能的解决方法是首先利用 syntacitc language model 分析句子各部

分的语法相关性，尤其是从属关系（dependency condition），然后将具有语法关系的词构成元组进行训练。此外还有一些语句本身以外的策略，如 Caching models（贮藏模型），它针对罕见单词突然频繁出现的情况，例如一篇文章中已经出现某个人名的情况下，该文章后面出现该人名的概率会显著增大。

最后列出了一些链接，包括 N-gram 相关的工具包、数据集、文字生成应用网页等。

6.7 Word Sense Disambiguation

正如之前介绍的，自然语言中涉及到多种歧义。本节 WSD 关注的是一词多义（Words have multiple senses），特别指出只关注同一词性的不同含义，因为不同词性的区分更多涉及词性标注。例如 bar 作为名词具有很多种含义。该任务可描述为：

- given a word
- and its context
- determine which sense it is

WSD 在自然语言处理中应用广泛，机器翻译、对话生成等。机器翻译中一个有趣的例子是，英语中 play 是多义的，翻译为 Spanish（西班牙语）时需要区分，play the violin = tocar el violin, play tennis = jugar al tenis。即 if a word is ambiguous in one language, that doesn't mean that it is necessarily ambiguous in another language.

接下来简要介绍了几种 WSD 的方法。Michael Lesk 的 Dictionary Method 是一种古老的方法，基本思路是根据字典得到每个单词的含义（可能多个含义），然后找到重叠度最高的含义组合，以决定每个单词的含义（Match sentences to dictionary definitions, and find the pair of meanings that have the most overlapping definitions）。

Decision Lists Method 是 David Yarowsky(Garofsky)在 1994 年提出的方法，基本的思路是 two senses per word, one sense per collocation（词的搭配），即每个单词最多只考虑两种含义，并根据搭配确定其唯一含义。需要注意，这里的 collocation 是广义的，给定长度的上下文窗口中出现的其他单词都与当前单词形成 collocation。利用训练集，得到每种搭配下单词的两种含义各自的 score，score 定义为

$$\log\left(\frac{p(sense_A|collocation_i)}{p(sense_B|collocation_i)}\right)$$

随后的测试语句中，获得上下文窗口内每个 collocation 对应的 score，取最高 score 对应的词义。注意到，这里按 score 排序后的规则其实构成了一棵决策树，例如 1. fish within window -> bass1, 2. striped bass -> bass1, 3. guitar within window -> bass2, 4. bass player -> bass2, 5. Play/V bass -> bass2。

当然，这种方法可以考虑多种特征（不仅是 collocation），只要构建合理的规则排序标准（即定义恰当的 score）。常用的特征如 collocation、position、syntactic information、topic of the text 等。这里 position 是指 plant pesticide vs. pesticide plant 的情况，syntactic information 是根据词汇在句子中的语法性质（词性、句子成分等）。

既然提到特征，自然引入第三类方法，即 classification，如 K-nearest neighbor 等等。

Bootstrapping 也是 David Yarowsky 在 90 年代中期提出的方法，是一种半监督方法（semi-supervised learning），可以看做是 Decision Lists Method 的半监督版本。对某个两个可能含义的单词如 plant，人工标记一对可靠的 collocation，如 plant1:leaf, plant2:factory。注意需要对每个重要的单词进行标记，但每个单词只需一对。然后，按照一定规则自动完成进一步的标记工作，例如根据 plant1:leaf、leaf1:live，可以推出 plant1:live，类似于这样的规则。这样，就可以有效地获得大量标记样本。

WSD 的常用评价指标是 precision-recall，即

- A = number of assigned senses
- C = number of words assigned correct senses
- T = total number of test words
- Precision = C/A; Recall = C/T

目前的 WSD 水平是：

- best recall around 77P/77R
- human lexicographer 97P/96R
- most common sense 57P/50R (decent but depends on domain)，即为每个单词赋予其出现频率最高的含义。

最后 Radev 列出了一些 WSD 样本数据获取的渠道。