词性标注和隐马尔科夫模型

Xinyu Dai

2020-11

概要

- 词性标注
- HMM模型
- HMM模型用于词性标注
- 相关问题讨论

词性标注

- 定义及任务描述
- 词性标注的问题 — 标注歧义 （兼类词）
- 词性标注之重要性
- 词性标注方法

词性标注任务描述

- 什么叫词性（Part-of-Speech）？
  - 词性又称词类，是指词汇基本的语法属性。
- 划分词类的依据
  - 词的形态、词的语法功能、词的语法意义
- 汉语的词类划分
  - 借用英文的词类体系
  - 缺乏词性的变化
- 词性标注：给某种语言的词标注上其所属的词类
  - The lead paint is unsafe.
  - The/Det lead/N paint/N is/V unsafe/Adj.
  - 他有较强的领导才能。
  - 他/代词 有/动词 较/副词 强/形容词 的/助词 领导/名词 才能/名词。

词性标注问题
— 词性标注歧义（兼类词）

- 一个词具有两个或者两个以上的词性
- 英文的Brown语料库中，10.4%的词是兼类词
  - The back door
  - On my back
  - Promise to back the bill
- 《现代汉语八百词》（吕叔湘），22.5%兼类词
  - 把门锁上，　买了一把锁
  - 他研究与自然语言处理相关的研究工作
- 对兼类词消歧— 词性标注的任务

词性标注的应用及重要性

- 句法分析的预处理
- 机器翻译
- Text – Speech （record）

词性标注常见方法

- 规则方法：
  - 词典提供候选词性
  - 人工整理标注规则
- 基于错误驱动的方法
  - 错误驱动学习规则
  - 利用规则重新标注词性
- 统计方法
  - 问题的形式化描述
  - 建立统计模型
    - HMM方法
    - 最大熵方法
    - 条件随机场方法
    - 结构化支持向量机方法

词性标注的性能指标

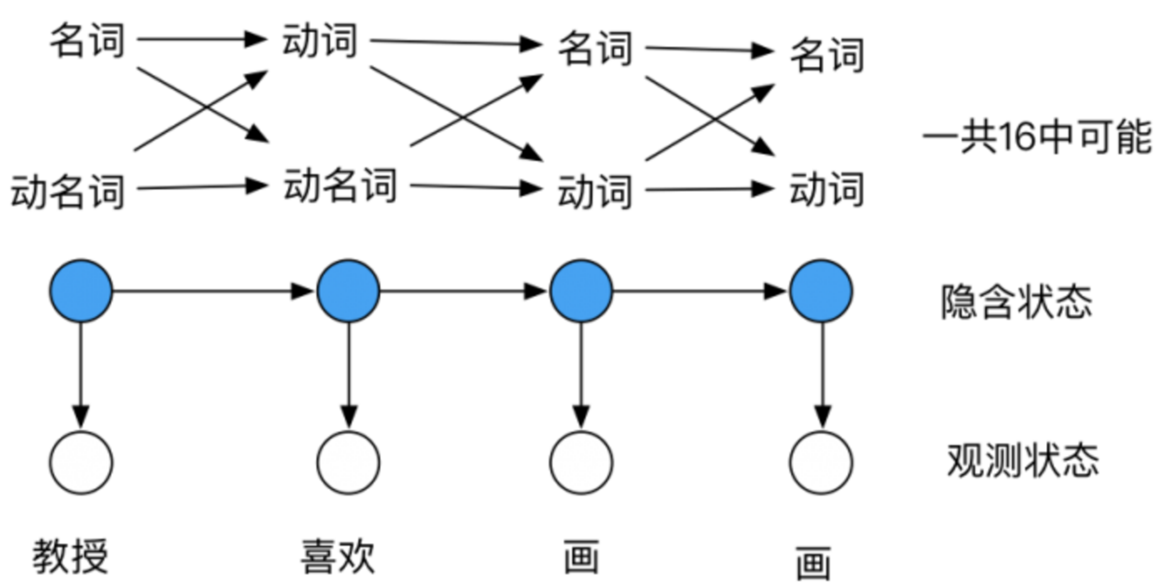
- 性能指标：标注准确率
- 当前方法正确率可以达到97%
- 正确率基线(Baseline)可以达到90%
  - 基线的做法：
    - 给每个词标上它最常见的词性
    - 所有的未登录词标上名词词性

形式化为一个分类问题

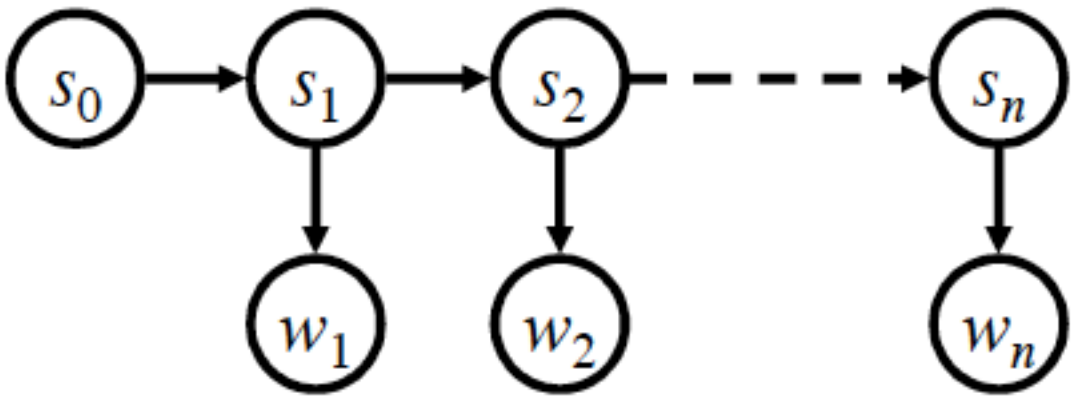
- 词串：$x_1x_2....x_n$; 词性串：$y_1y_2...y_n$
- Training data $(x^{(i)}, y^{(i)})$
- Learning a mapping function $f: X \rightarrow Y$
- ....

决定一个词词性的因素

- 从语言学角度：由词的用法以及在句中的语法功能决定
- 统计学角度：
  - 和上下文的词性（前后词的标注）相关
  - 和上下文单词（前后词）相关



- 词串*W*，词性串*S*
- *P*（*W,S*）?

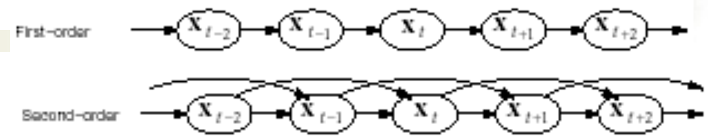


$$P(\mathbf{s},\mathbf{w}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)$$

隐马尔可夫模型 — 概要

- 马尔可夫模型:描述了一类随机过程
- 隐马尔可夫模型

马尔可夫模型



- 马尔科夫过程
  - 一个系统有N个有限状态$S=\{s_1,s_2,\ldots s_N\}$
  - $Q=(q_1,q_2,\ldots q_T)$是一个随机变量序列。随机变量的取值为状态集S中的某个状态。
  - $P(q_t=s_j|q_{t-1}=s_i,q_{t-2}=s_k,\ldots,q_1=s_h)$
- 假设1
  有限视野(Limited Horizon)

  $P(q_{t+1}=s_k|q_1,\ldots q_t) = P(q_{t+1}=s_k|q_{t-(n-1)},\ldots q_t)$

  $(n-1)^{th}$ 阶马尔可夫链 $\rightarrow$ n 元语言模型
- 假设2
  时间独立性(No change over time)

  $P(q_{t+1}=s_k|q_t=s_h) = P(q_{t+k+1}=s_k|q_{t+k}=s_h)$

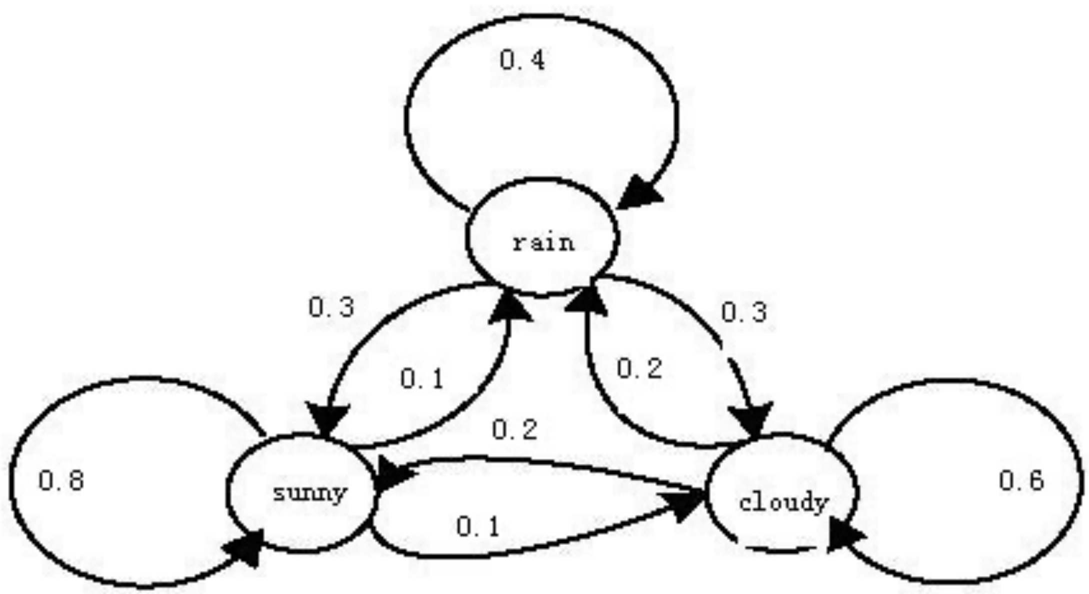
马尔可夫模型示例 — 天气预报

- 状态：雨、多云、晴
- 给定不同天气之间的转换概率，预测未来数天的天气
- 若是1阶马尔科夫链，则可以通过如右图所示的矩阵描述状态之间的转移概率

$$A=\{a_{ij}\}=\begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

马尔可夫模型示例
— 天气预报

- 通过有限状态自动机描述状态转移概率



预测
— 计算未来天气 （序列的概率）

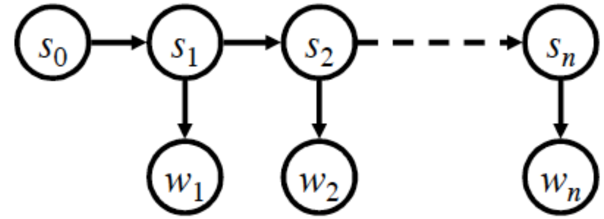
- 晴-雨-晴-雨-晴-多云-晴，未来七天天气是这种情况的概率

$$
\begin{aligned}
P(Q|Model) &= P(S_3,S_1,S_3,S_1,S_3,S_2,S_3|Model) \\
&= P(S_3|Begin)*P(S_1|S_3)*P(S_3|S_1)* \\
&\quad *P(S_1|S_3)*P(S_3|S_1)*P(S_2|S_3)*P(S_3|S_2) \\
&= \Pi_3*a_{31}*a_{13}*a_{31}*a_{13}*a_{32}*a_{23} \\
&= 0.33*0.1*0.3*0.1*0.3*0.1*0.2 = 5.94*10^{-6}
\end{aligned}
$$

隐马尔可夫模
— Hidden Markov Model

- 介绍
- 定义
- 隐马模型应用于词性标注

HMM模型的简单介绍



$$P(\mathbf{s}, \mathbf{w}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)$$

- HMM是一阶马尔可夫模型的扩展
  - 隐藏的状态序列满足一阶马尔可夫模型
  - 观察值与状态之间存在概率关系
- 何谓“隐”？
  - 状态（序列）是不可见的（隐藏的）
- 相对于markov模型的又一假设：输出独立性

$$P(O_1,\dots O_T \mid S_1,\dots S_T) = \prod_T \mathbf{P}(O_t \mid S_t)$$

HMM的定义

- 定义：一个HMM模型 $\lambda=(S,V,A,B,\pi)$
- S是状态集， $S=(S_1,S_2,\ldots S_N)$
- V是观察集， $V=(V_1,V_2,\ldots V_M)$
- 状态序列Q = $q_1q_2\ldots q_T$（隐藏），观察序列 O=$o_1o_2\ldots o_T$（可见）
- A是状态转移概率分布A=$[a_{ij}]$, $a_{ij}=P(q_t=s_j|q_{t-1}=s_i)$ *(满足假设1.)*
- B是观察值生成概率分布B=$[b_j(v_k)]$,
  $b_j(v_k)=P(o_t=v_k|q_t=s_i)$ *(满足假设2、3)*
- 初始状态值概率分布 $\Pi=[\pi_i]$, $\pi_i=P(q_1=s_i)$

词性标注的HMM模型定义

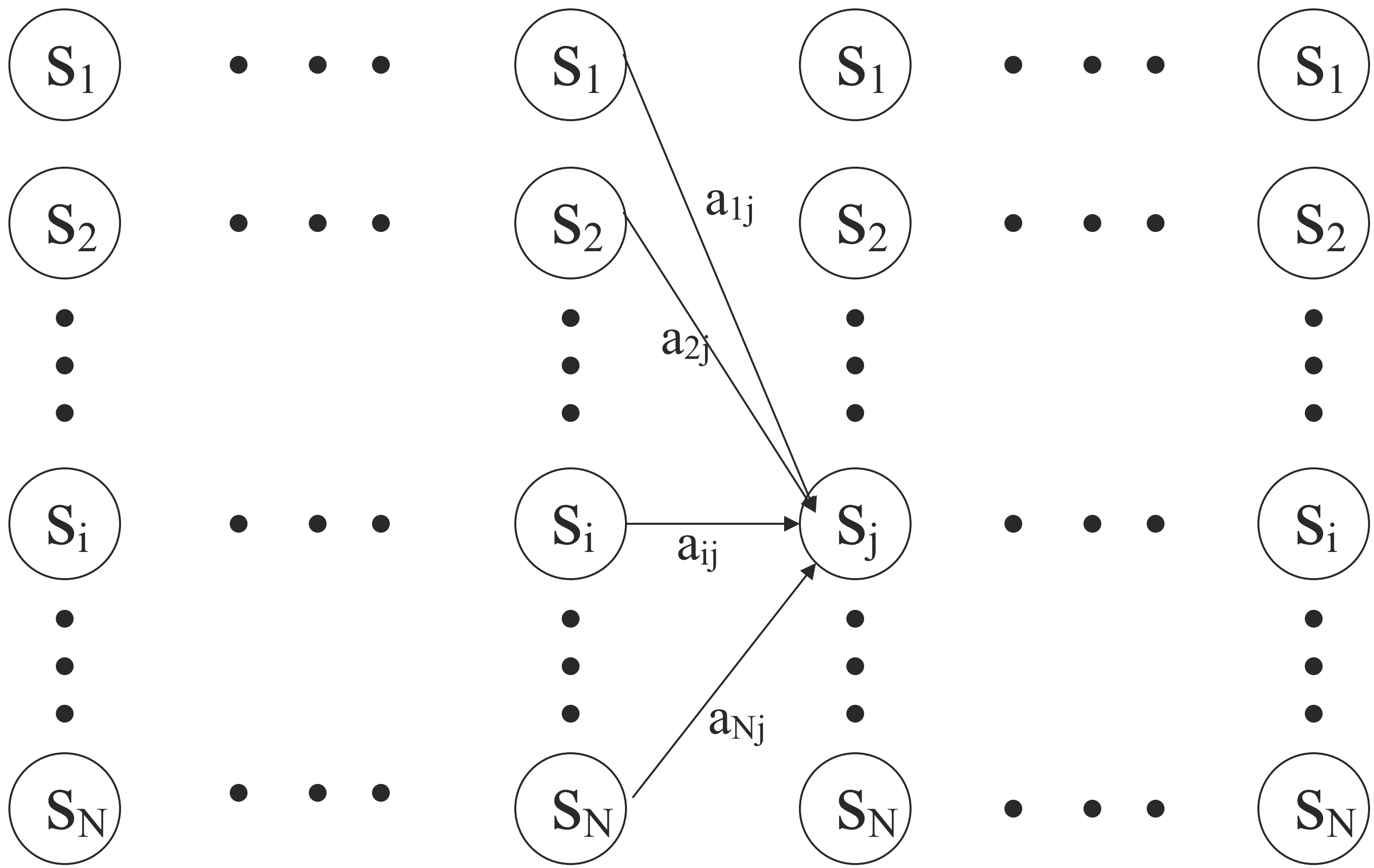
- HMM：S V A B π
- S：预先定义的词性标注集
- V：文本中的词汇
- A：词性之间的转移概率
- B：某个词性生成某个词的概率

  例，P(我|“代词”)
- π：初始概率
- 基于构建的HMM，利用某些算法，寻找一个最合适的词性标注序列，即为一个词串上的每个词标注上词性。
- 注：可见的观察序列为$w_1w_2...w_T$

POS tagging using HMM

- 模型解码(Decoding)
  - 给定模型和一个观测序列，寻求一个产生这个观测序列的可能性最大的状态序列
  - 给定词序列$w_1w_2...w_T$ （可见的观察序列），寻求产生这个词序列的最可能的词性标注序列$S_1S_2...S_T$ （隐藏的状态序列）
  - 如何发现“最优”状态序列能够“最好地解释”观察序列
  - 需要高效算法，Viterbi算法
- 模型参数学习(Learning)

Trellis representation of an HMM



Time= 1 w_1 t w_t t+1 w_{t+1} T w_T

计算观察序列对应某一状态序列的概率

- 模型λ，观察序列O，假设对应状态序列为Q，计算该状态序列存在的可能性：

$$P(O,Q|\lambda) = P(Q|\lambda)P(O|Q,\lambda)$$

$$= \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(o_t)$$

- 简单的方法，计算所有可能的状态序列，$O(N^T)$

Viterbi算法

- 一种更有效率的利用动态规划思想的算法
- 定义一个变量$\delta_t(i)$，指在时间t时，HMM沿着某一条路径到达S_i，并输出序列为$w_1w_2\ldots w_t$ 的最大概率

$$\delta_t(i) = \max P(Pos_1 \ldots Pos_{t-1}, Pos_t = s_i, w_1 \ldots w_t)$$

利用Viterbi算法的求解过程

- 初始化：$\delta_1(i) = \max P(Pos_1 = s_i, w_1) = \pi_i b_i(w_1), \quad 1 \le i \le N$
- 迭代：

$$\begin{aligned}\delta_{t+1}(j) &= \max P(Pos_1 \ldots Pos_t, Pos_{t+1} = s_j, w_1 w_2 \ldots w_{t+1}) \\ &= \max_i [a_{ij} b_j(w_{t+1}) \max P(Pos_1 \ldots Pos_{t-1}, Pos_t = s_i, w_1 w_2 \ldots w_t)] \\ &= \max_i [a_{ij} b_j(w_{t+1}) \delta_t(i)], \qquad 1 \le j \le N, 1 \le t \le T-1\end{aligned}$$

- 迭代结束
- 回溯记录最优路径：

$$\max_i [\delta_T(i)]$$

Viterbi算法时间复杂度

- 每计算一个 $\delta_t(i)$，必须考虑从t-1时刻所有的N个状态转移到状态的s_i概率，时间复杂度为*O*(*N*)，对应每个时刻t，要计算N个中间变量 $\delta_t(1)...\delta_t(N)$,时间复杂度为$O(N^2)$，又t从1…T，因此整个算法时间复杂度为$O(N^2T)$

模型参数学习

- 给定状态集S和观察集V，学习模型参数A、B、π
- 模型参数学习过程就是构建模型的过程
- 有指导的学习 － 最大似然估计
- 无指导的学习 － Welch-Baum

有指导学习模型参数 — 从标注语料中学习

- 最大似然估计

$$a_{ij} = P(s_j \mid s_i) = \frac{\text{Number of transitions from state } \mathrm{s_i} \text{ to stae } \mathrm{s_j}}{\text{Number of transtion out of stae } \mathrm{s_i}}$$

$$b_i(v_k) = P(v_k \mid s_i) = \frac{\text{Number of times observation } \mathrm{v_k} \text{ occurs in state } \mathrm{s_i}}{\text{Number of times in state } \mathrm{s_i}}$$

- 对于无标注的语料库（状态不可见）如何获取模型参数？

无指导学习模型参数 — Welch-Baum 算法

- 迭代估计参数，使得 $\arg\max_{\lambda} P(O_{training} \mid \lambda)$ 此时λ最能拟合训练数据

$$a_{ij} = P(s_j \mid s_i) = \frac{\text{Expected Number of transitions from state } s_i \text{ to stae } s_j}{\text{Expected Number of transtions out of state } s_i}$$

$$b_i(v_k) = P(v_k \mid s_i) = \frac{\text{Expected Number of times observation } v_k \text{ occurs in state } s_i}{\text{Expected Number of times in state } s_i}$$

$$\pi_i = P(s_i) = \text{Expected Frequency in state } s_i \text{ at time (t=1)}$$

- Baum证明：随着迭代过程， $P(\mathrm{O} \mid \hat{\lambda}) \geq P(\mathrm{O} \mid \lambda)$

无指导学习模型参数
－ Welch-Baum 算法

- **找到使得训练数据存在概率最大化的模型**
- 基本思想：随机给出模型参数的初始化值，得到最初的模型λ_0，然后利用初始模型λ_0得到某一状态转移到另一状态的期望次数，然后利用期望次数对模型进行重新估计，由此得到模型λ_1，如此循环迭代，重新估计，直至模型参数收敛（模型最优）。

HMM模型似然函数

- 给定一个HMM λ和一个观察序列，计算该序列存在的概率 — 对所有可能生成该序列的状态序列的概率求和

$$P(\sigma \mid \lambda) = \sum_{Q} P(O,Q \mid \lambda) = \sum_{Q} P(Q \mid \lambda) P(O \mid Q, \lambda)$$

$$= \sum_{Q} \left(\pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t} \right) \left(\prod_{t=1}^{T} b_{q_t}(o_t) \right)$$

$$= \sum_{Q} \left(\pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(o_t) \right)$$

$$|\log P(\sigma \mid \lambda_{i+1}) - \log P(\sigma \mid \lambda_i)| \leq \varepsilon$$

- 计算复杂度高

模型似然函数

- 类似Veterbi动态规划算法
- 前向算法：
  - 定义前向概率：观察值为$o_1o_2...o_t$,t时刻对应的状态值为s_i的概率

    $$\alpha_t(i) = P(o_1...o_t, q_t = s_i \mid \lambda)$$

  - 迭代 $$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}), \quad 1 \le j \le N \quad 1 \le t \le T-1$$
  - 模型似然 $$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$
- 后向算法：
  - 定义后向概率：观察值为$o_to_{t+1}...o_T$,t时刻对应的状态值为s_i的概率

    $$\beta_t(i) = P(o_t...o_T, q_t = s_i \mid \lambda)$$

  - 迭代 $$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), 1 \le i \le N;\ 1 \le t \le T-1$$
  - 模型似然

    $$P(O|\lambda)=\sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)$$

Welch-Baum算法的参数估计

- How to calculate Expected Number?
- 定义一个变量 $\xi_t(i,j)$ ，对应于观察序列$o_1o_2\ldots o_T$，假设在t时刻的状态是s_i，t+1时刻的状态是s_j的概率。

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid o_1 o_2 \ldots o_T)$$

$$= \frac{P(q_t = s_i,\ q_{t+1} = s_j,\ o_1\ o_2\ \ldots\ o_T)}{P(o_1\ o_2\ \ldots\ o_T)}$$

$$= \frac{P(q_t = s_i,\ o_1\ o_2\ \ldots\ o_t)\ a_{ij}\ b_j\ (o_{t+1})\ P(o_{t+2}\ \ldots\ o_T,\ q_{t+1} = s_j)}{P(o_1\ o_2\ \ldots\ o_T)}$$

$$= \frac{\alpha_t(i) a_{ij}\ b_j\ (o_{t+1}) \beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i) a_{ij}\ b_j\ (o_{t+1})\ \beta_{t+1}(j)}$$

Welch-Baum算法的参数估计（续）

- 定义一个变量 $\gamma_t(i)$ ，对应于观察序列 $o_1o_2\ldots o_T$，假设在t时刻的状态是s_i。

$$\gamma_t(i) = P(q_t = s_i \mid o_1o_2\ldots o_T)$$

$$= \frac{P(q_t = s_i, o_1 o_2 \ldots o_T)}{P(o_1 o_2 \ldots o_T)}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\, \beta_{t+1}(i)}$$

无指导学习模型参数 — Welch-Baum 算法（二）

- 赋a_{ij},$b_i(v_k)$的初始值

$$\overline{a_{ij}} = P(s_j \mid s_i) = \frac{\text{Expected Number of transitions from state } s_i \text{ to stae } s_j}{\text{Expected Number of transtions out of state } s_i} = \frac{\sum_t \xi_t(i,j)}{\sum_t \gamma_t(i)}$$

$$\overline{b_i(v_k)} = P(v_k \mid s_i) = \frac{\text{Expected Number of times observation } v_k \text{ occurs in state } s_i}{\text{Expected Number of times in state } s_i} = \frac{\sum_{t, o_t = v_k} \gamma_t(i)}{\sum_t \gamma_t(i)}$$

$$\overline{\pi_i} = P(s_i) = \text{Expected Frequency in state } s_i \text{ at time (t=1)} \quad = \gamma_1(i)$$

- 反复迭代，直到收敛 $|\log P(\sigma \mid \lambda_{i+1}) - \log P(\sigma \mid \lambda_i)| \leq \varepsilon$

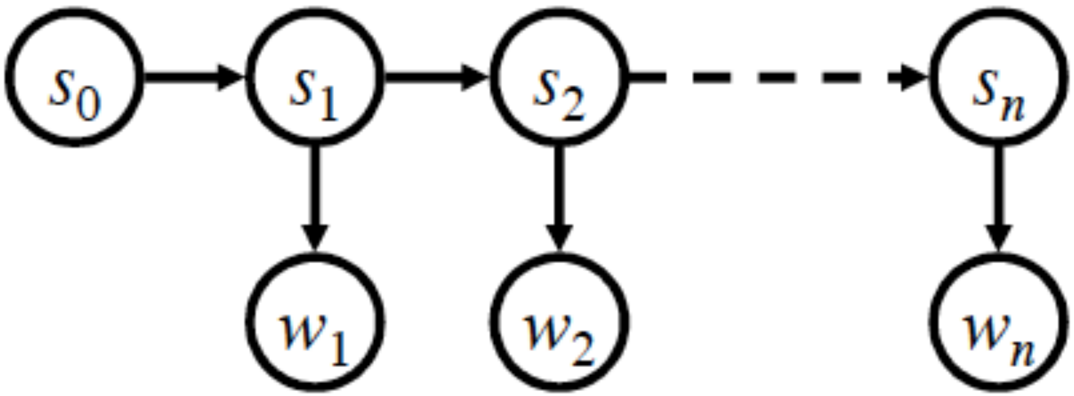
词性标注HMM的参数估计

- 状态转移概率（词性－词性的概率）

  e.g. P(N|V)
- 生成概率（词性－词的概率）

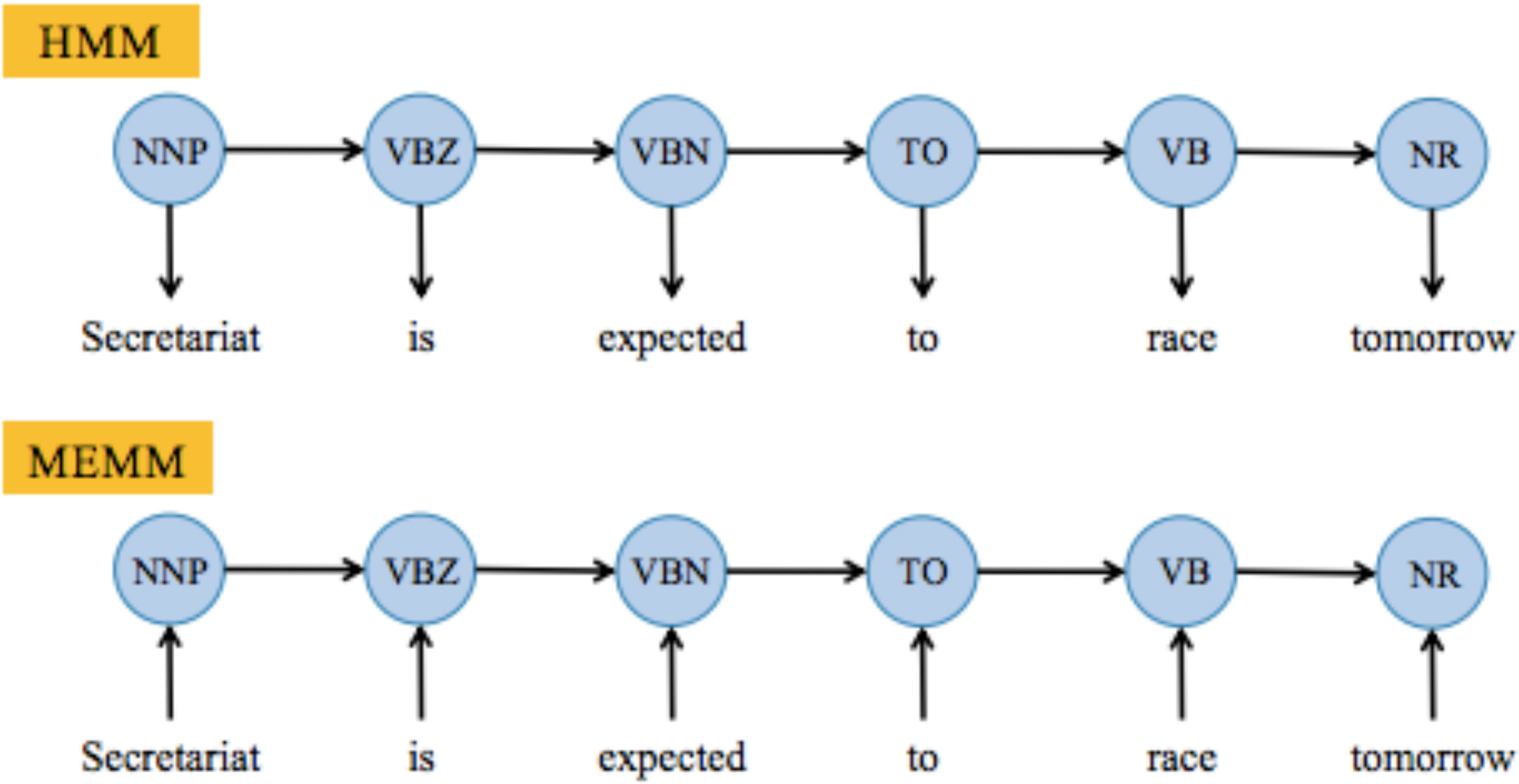
  e.g. P(“研究” |N)
- 利用词性标注语料库获取状态转移概率和生成概率（最大似然估计）
- 利用无标注语料库获取状态转移概率和生成概率（ Welch-Baum 算法）
- 平滑
- 未登录词处理

Revisit HMM

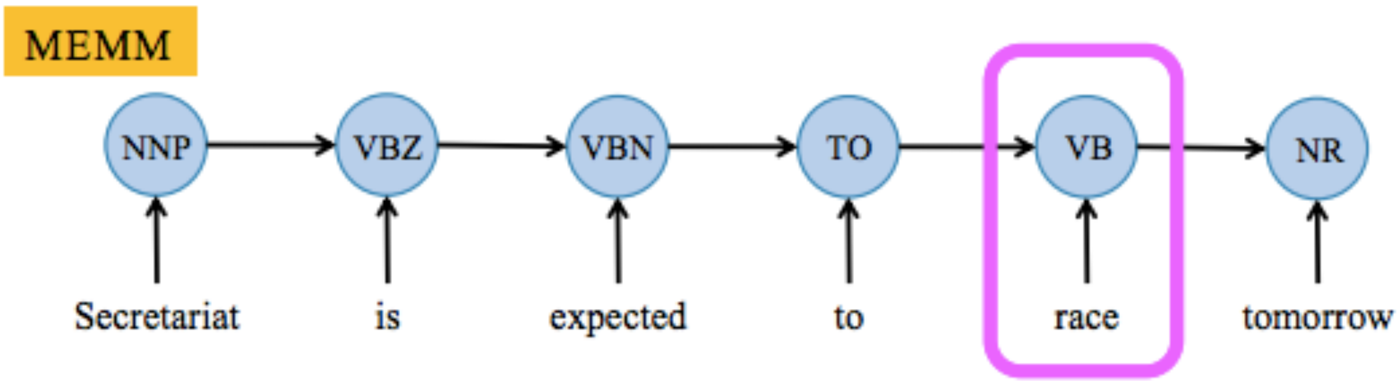


$$P(\mathbf{s}, \mathbf{w}) = \prod_i P(s_i | s_{i-1}) P(w_i | s_i)$$

Maximum Entropy Markov Model



MEMM: Conditional Model



$$P(Q \mid O) = \prod_{i=1}^{n} P(q_i \mid q_{i-1}, o_i)$$

$$P(q \mid q', o) = \frac{1}{Z(o, q')} \exp(\sum_i w_i f_i(o, q))$$

MEMM: Conditional Model

Given a conditional tagging model, the function from sentences $x_1 \ldots x_n$ to tag sequences $y_1 \ldots y_n$ is defined as

$$f(x_1 \ldots x_n) = \arg \max_{y_1 \ldots y_n \in \mathcal{Y}(n)} p(y_1 \ldots y_n | x_1 \ldots x_n)$$

We are left with the following three questions:

- How we define a conditional tagging model $p(y_1 \ldots y_n | x_1 \ldots x_n)$?
- How do we estimate the parameters of the model from training examples?
- How do we efficiently find

$$\arg \max_{y_1 \ldots y_n \in \mathcal{Y}(n)} p(y_1 \ldots y_n | x_1 \ldots x_n)$$

for any input $x_1 \ldots x_n$?

$$P(Y_1 = y_1 \dots Y_n = y_n | X_1 = x_1 \dots X_n = x_n)$$

$$\begin{aligned}
& P(Y_1 = y_1 \dots Y_n = y_n | X_1 = x_1 \dots X_n = x_n) \\
= & \prod_{i=1}^{n} P(Y_i = y_i | X_1 = x_1 \dots X_n = x_n, Y_1 = y_1 \dots Y_{i-1} = y_{i-1}) \\
= & \prod_{i=1}^{n} P(Y_i = y_i | X_1 = x_1 \dots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1})
\end{aligned}$$

For any pair of sequences $x_1 \dots x_n$ and $y_1 \dots y_n$, we define the i'th "history" h_i to be the four-tuple

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \dots x_n, i \rangle$$

Thus h_i captures the conditioning information for tag y_i in the sequence, in addition to the position i in the sequence. We assume that we have a feature-vector representation $f(h_i, y) \in \mathbb{R}^d$ for any history h_i paired with any tag $y \in \mathcal{K}$. The feature vector could potentially take into account any information in the history h_i and the tag y. As one example, we might have features

$$f_1(h_i, y) = \begin{cases} 1 & \text{if } x_i = \texttt{the} \text{ and } y = \texttt{DT} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(h_i, y) = \begin{cases} 1 & \text{if } y_{i-1} = \texttt{V} \text{ and } y = \texttt{DT} \\ 0 & \text{otherwise} \end{cases}$$

- *A set of words $\mathcal{V}$ (this set may be finite, countably infinite, or even uncountably infinite).*

- *A finite set of tags $\mathcal{K}$.*

- *Given $\mathcal{V}$ and $\mathcal{K}$, define $\mathcal{H}$ to be the set of all possible histories. The set $\mathcal{H}$ contains all four-tuples of the form $\langle y_{-2}, y_{-1}, x_1 \dots x_n, i\rangle$, where $y_{-2} \in \mathcal{K} \cup \{*\}$, $y_{-1} \in \mathcal{K} \cup \{*\}$, $n \geq 1$, $x_i \in \mathcal{V}$ for $i = 1 \dots n$, $i \in \{1 \dots n\}$. Here $*$ is a special "start" symbol.*

- *An integer d specifying the number of features in the model.*

- *A function $f : \mathcal{H} \times \mathcal{K} \to \mathbb{R}^d$ specifying the features in the model.*

- *A parameter vector $\theta \in \mathbb{R}^d$.*

Given these components we define the conditional tagging model

$$p(y_1 \dots y_n | x_1 \dots x_n) = \prod_{i=1}^{n} p(y_i | h_i; \theta)$$

where $h_i = \langle y_{i-2}, y_{i-1}, x_1 \dots x_n, i\rangle$, and

$$p(y_i | h_i; \theta) = \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}$$

Others

- Feature engineering:
  - https://github.com/Michael-Tu/ML-DL-NLP/tree/master/MEMM-POS-Tagger
- Training:
  - Maximum log-likelihood
- Decoding:
  - Still Viterbi

$$f_{104}(h,y) = \begin{cases} 1 & \text{if } \langle y_{-1}, y\rangle = \langle \text{JJ, VB}\rangle \\ 0 & \text{otherwise} \end{cases}$$

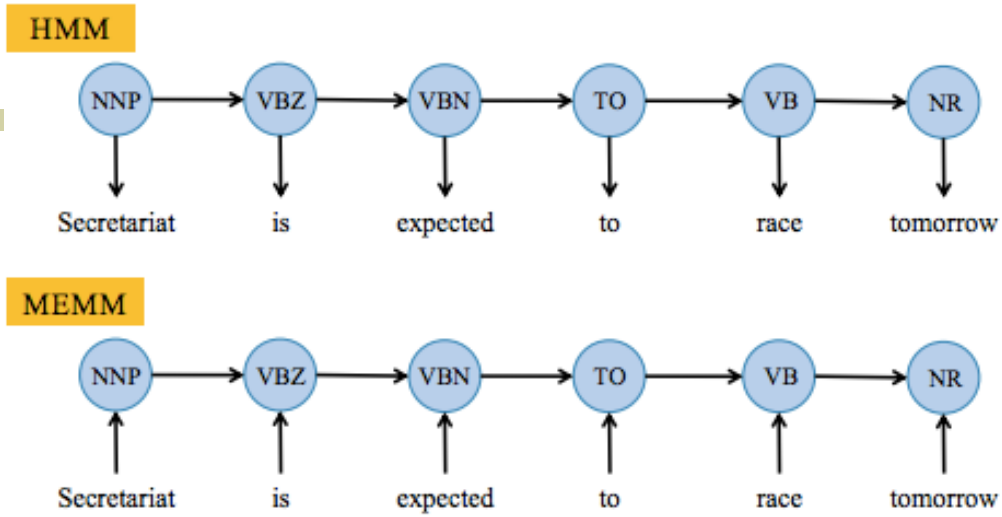
$$f_{105}(h,y) = \begin{cases} 1 & \text{if } \langle y\rangle = \langle \text{VB}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{103}(h,y) = \begin{cases} 1 & \text{if } \langle y_{-2}, y_{-1}, y\rangle = \langle \text{DT, JJ, VB}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{106}(h,y) = \begin{cases} 1 & \text{if previous word } x_{i-1} = \textit{the} \text{ and } y = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{107}(h,y) = \begin{cases} 1 & \text{if next word } x_{i+1} = \textit{the} \text{ and } y = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

HMM vs. MEMM



- HMM
- 生成式模型(Generative Model)
  - 计算联合概率 P(words, tags)
  - 除了生成tags，还生成words
  - 实际上，我们只需要预测tags
  - Probability of each slice = emission * transition =

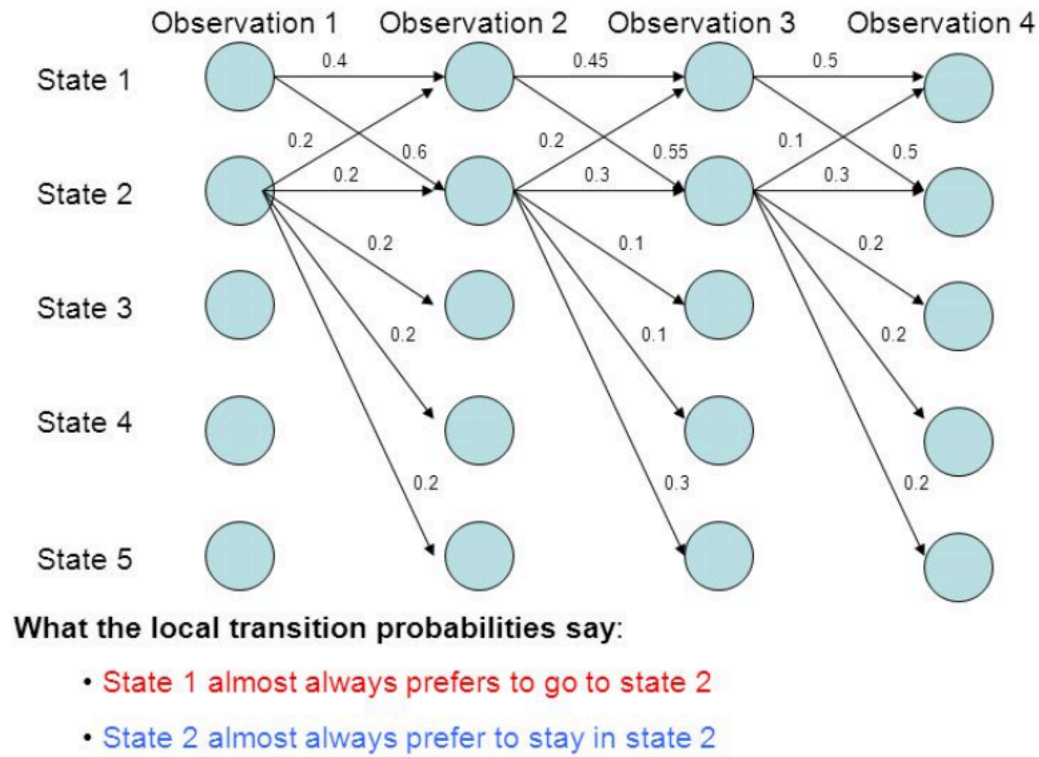
    $P(w_i|tag_i)*P(tag_i|tag_{i-1})$
  - ★ 很难结合更多的特征

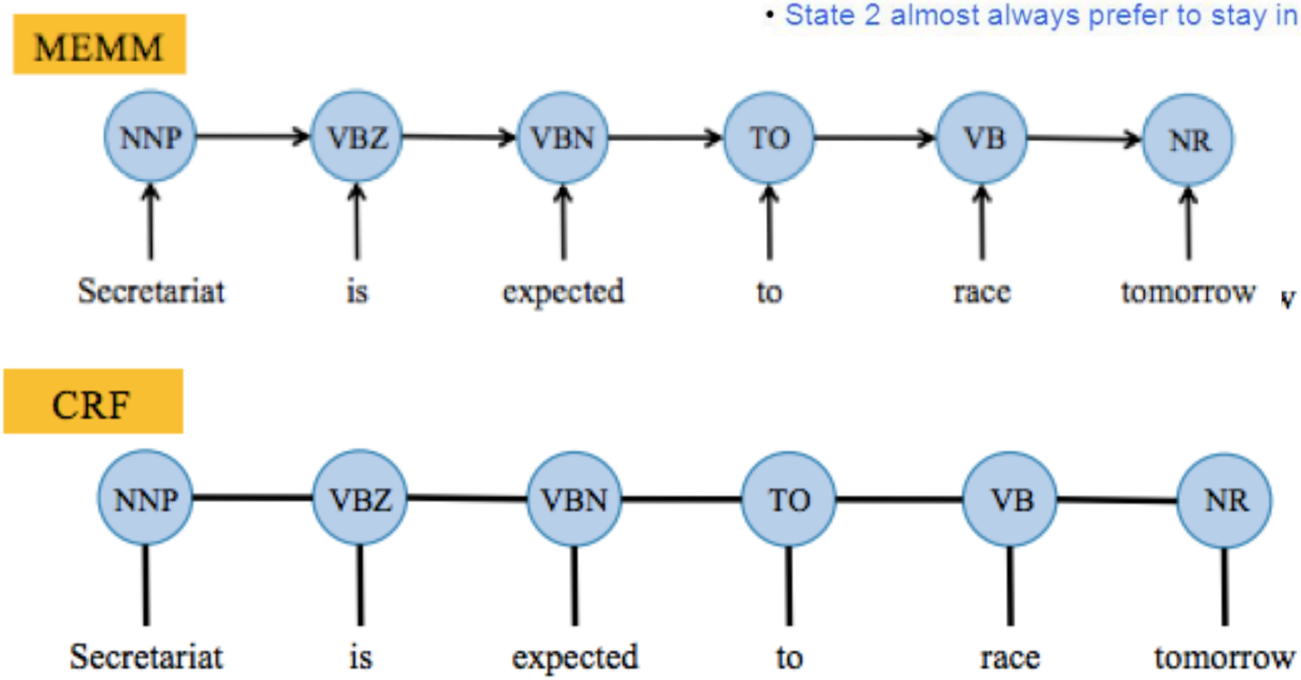
- MEMM
- 判别式模型(Discriminative Model)
  - 直接计算条件概率 P(tags|words)
  - 预测tags，不需要考虑input的分布
  - Probability of each slice = $P(tag_i|tag_{i-1},word_i)$

    or $P(tag_i|tag_{i-1}, \text{all words})$
  - ★ 容易引入各种特征

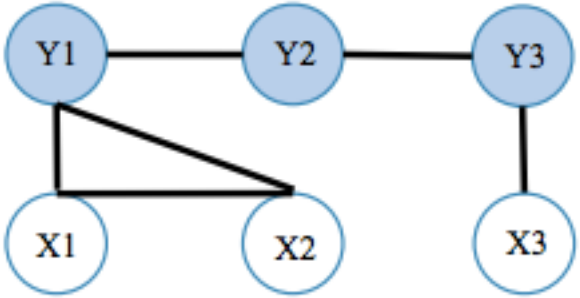
MEMM v

- 每个状态的归一化：自每
- 标注偏置：容易选择出边
- 条件随机场模型：克服M
  - Please refer a sampl
    http://www.cis.upenn





MEMM vs. CRF



- 是无向图，不再用概率模型去解释，而是一种称之为“团”或者“势”度量节点间的关系。

$$\phi(X1, X2, Y1) \text{ or } \phi(Y1, Y2)$$

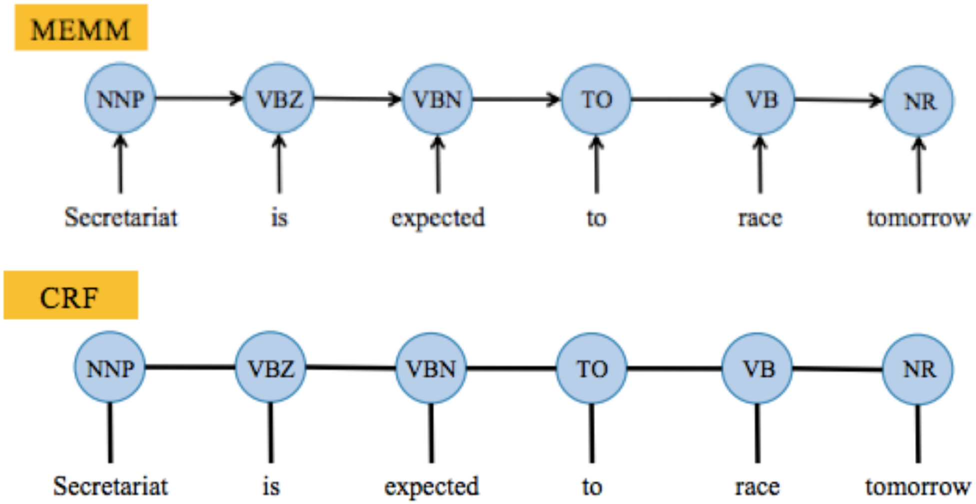
$$\phi(Y1, Y2) = \exp \sum_i w_i f_i(Y1, Y2)$$

- 仍然是条件概率模型，不是对每个状态做归一化，而是对整个串（图）做归一化，避免标注偏置。

$$P(Y \mid X) = \frac{1}{Z} \prod_{cliqueC} \varphi_C(X, Y)$$

$$Z = \sum_Y \prod_{cliqueC} \varphi_C(X, Y)$$

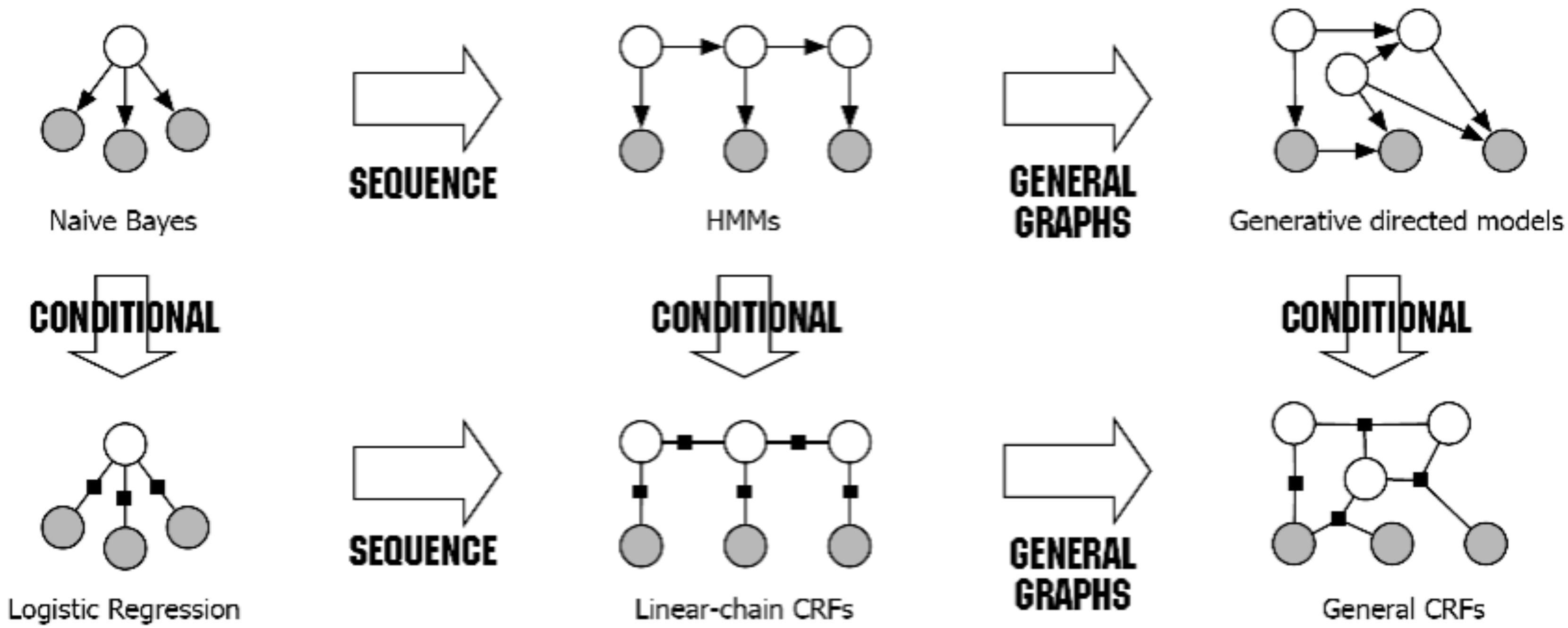
Linear Chain CRF



$$P(q \mid o) = \frac{1}{Z(o)} \prod_{t} \exp\left(\sum_{i=1} w_i f_i(q_t, q_{t-1}, o)\right)$$

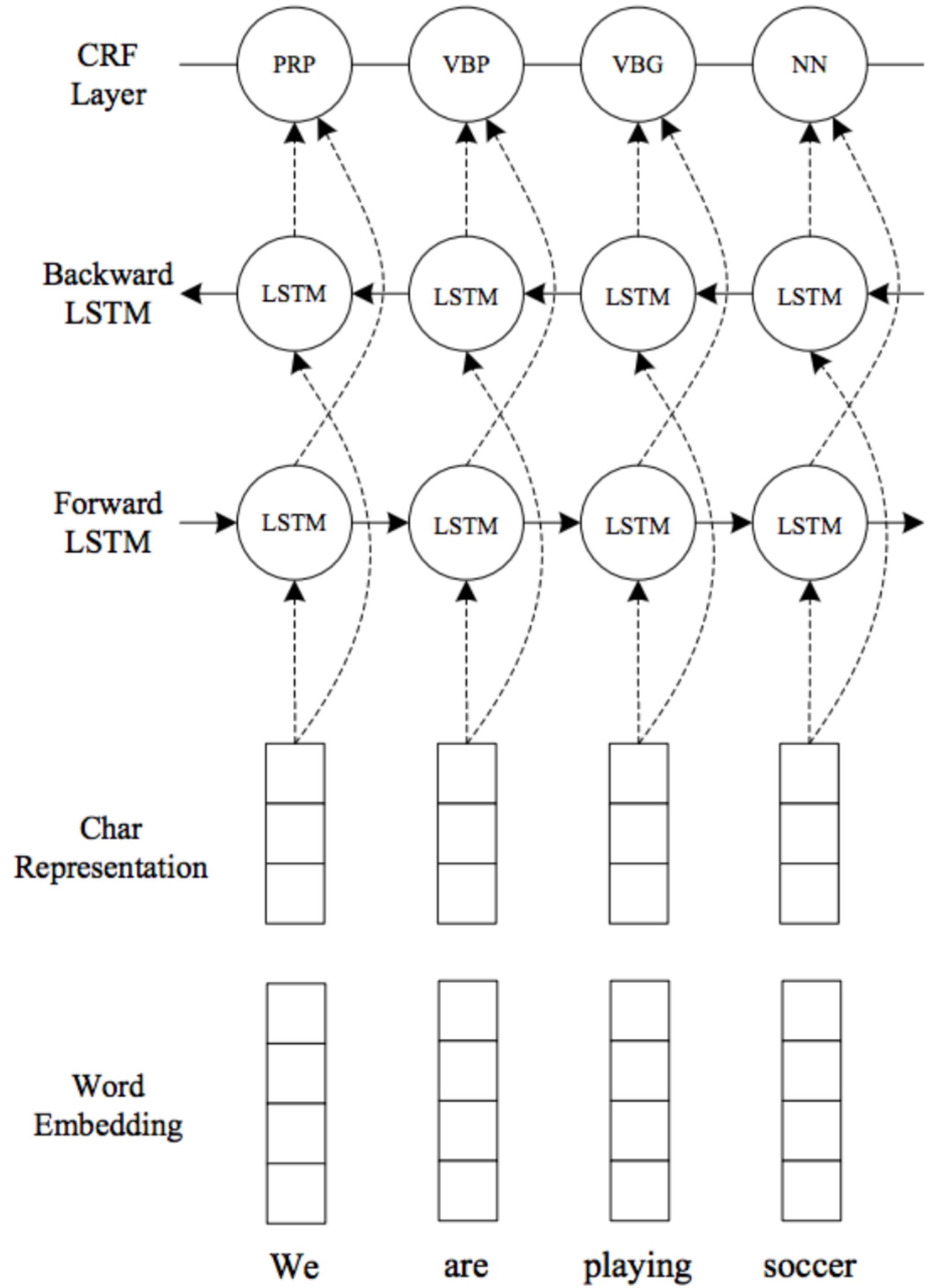
$$Z(o) = \sum_{q \subset Q} \prod_{t} \exp\left(\sum_{i=1} w_i f_i(q_t, q_{t-1}, o)\right)$$

方法的发展


Naive Bayes
SEQUENCE
HMMs
GENERAL GRAPHS
Generative directed models
CONDITIONAL
CONDITIONAL
CONDITIONAL
Logistic Regression
SEQUENCE
Linear-chain CRFs
GENERAL GRAPHS
General CRFs

Sequence Labeling
LSTM+CRF

End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF, Ma Xuezhe, Hovy, ACL2016



任务的扩展

- 词性标注
- 分词？
- 命名实体识别？
- 基因序列识别？
- ……
- 序列化标注任务：Sequence Labeling

Reference

- Chapter 6 of Speech and Language Processing: An introduction to natural language processing (http://www.mit.edu/~6.863/spring2009/jmnew/6.pdf)
- http://www.cis.upenn.edu/~pereira/papers/crf.pdf
- HMM, MEMM, CRF in wikipedia
- Some useful toolkit
  - Mallet
  - CRF++
  - …