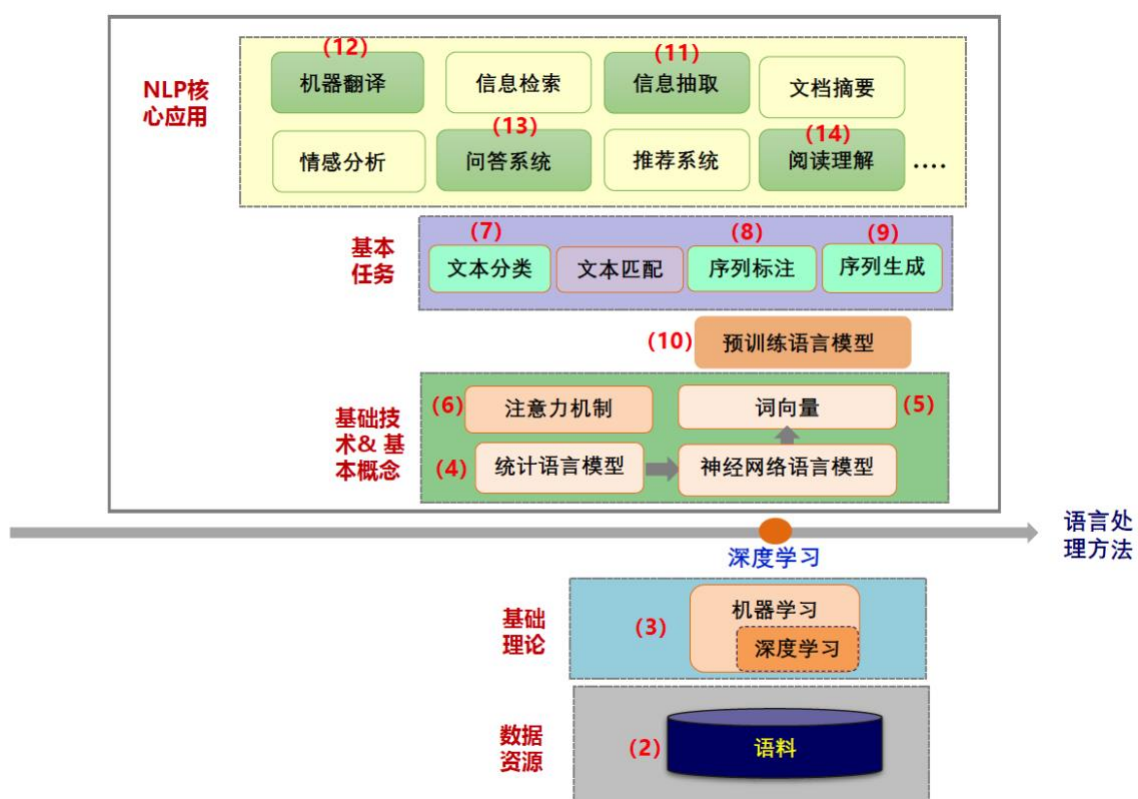


国科大-自然语言处理-网安秋季学期

——陈若愚

——2021.12.31



大纲

第三章复习 DNN, CNN 与 RNN.....	3
3.1 DNN	3
3.2 CNN.....	4
3.3 RNN.....	5
第四章复习 语言模型+词向量.....	7
4.1 统计语言模型.....	7
4.2 神经语言模型.....	10
4.3 词向量.....	12
第五章复习 注意力机制	18
5.1 注意力机制概述.....	18
5.2 传统注意力机制.....	18
5.3 注意力编码机制.....	20
第六章复习 序列标注	22
第六章复习 序列生成	31
第七章复习 预训练语言模型	38
7.1 第三范式相关.....	39
7.2 第四范式相关.....	41

第三章复习 DNN, CNN 与 RNN

3.1 DNN

激活函数： 为了增强网络的表达能力，需要引入连续的非线性激活函数。

激活函数性质：

- 连续并可导（允许少数点上不可导）的非线性函数。（方便优化）
- 激活函数及其导函数要尽可能的简单，为了提高网络计算效率。
- 激活函数的导函数的值域要在一个合适的区间内（太大或太小容易影响训练的效率和稳定性）

Sigmoid 函数：

Logistic 函数 $f(x) = \frac{1}{1+\exp(-x)}$ $f'(x) = f(x)(1 - f(x))$

人工神经网络： 由多个神经元组成的具有并行分布结构的神经网络模型。

迭代调参法： 通过调整参数，让模型输出递归性地逼近标准输出。

迭代调参法步骤： 1.定义目标函数（即损失函数）； 2.优化目标函数。

交叉熵损失函数也是负对数似然损失函数：

◆ **交叉熵损失函数：**

$$L(Y, f(X, \theta)) = - \sum_{i=1}^C y_i \log f_i(X, \theta)$$

梯度下降法需注意问题：

1. 参数初值应尽量随机；
2. 学习率不要设置过大或者过小。

前馈神经网络的训练过程：

1. 先计算前向传播各层数值
2. 计算误差，即目标函数
3. 反向传播，计算每一层参数的偏导数，并更新。

注意，梯度下降是参数更新方法，而反向传播是误差传递方法。

梯度消失问题：

激活函数角度，其导数值域常小于 1，在反向传播链式法则时多次乘激活函数的导数会使误差经过每一层大小都不断衰减。

解决梯度消失方法：

选择合适的激活函数，推荐 ReLU；用复杂的门结构代替激活函数；残差结构。

解决过拟合：

1. Dropout
2. Batch Normalization
3. 损失函数加入正则项，例如对参数进行 L2 惩罚。

3.2 CNN

参数更少，共享一组参数权重。

池化层作用：降低网络规模，因为卷积连接在 `stride=1` 时无法减少节点数量。

卷积神经网络的特性：

1. 局部连接
 2. 权重共享
 3. 空间或时间上的次采用
- 平移，缩放和扭曲不变性。

3.3 RNN

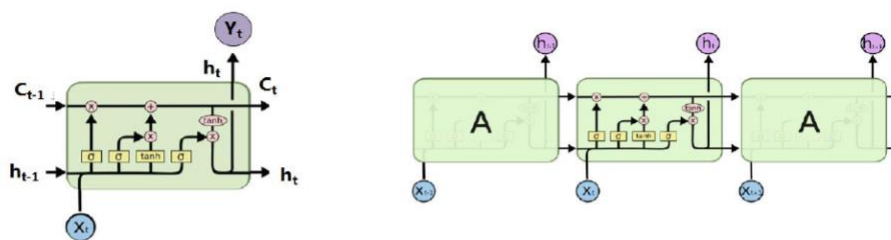
问题：DNN、CNN 输入、输出定长；处理输入、输出变长问题效率不高。而自然语言处理中的语句通常其长度不固定。

BPTT (Backpropagation through time)

梯度消失问题，和 DNN 类似，主要是在时间传播参数 h 上。

LSTM 基本思想：需建立一个机制（维持一个细胞状态 C_t ）能保留前面远处结点信息在长距离传播中不会被丢失。

LSTM 单元结构：



输入： C_{t-1}, h_{t-1}, x_t

输出： C_t, h_t, y^t

细胞状态： $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

新信息： $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

输入门： $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

遗忘门： $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

隐状态输出 $h_t = o_t * \tanh(C_t)$

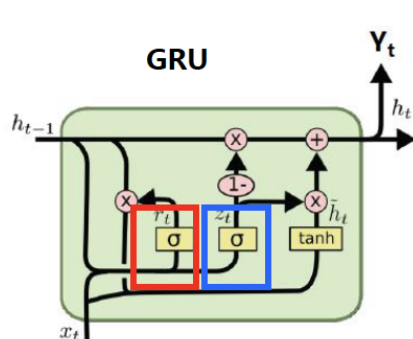
输出门： $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

输出 $y^t = \sigma(W' h^t)$

参数： W_f, W_i, W_o, W_C, W'

GRU: 输入门和遗忘门合并为更新门（更新门决定隐状态保留放弃部分）

输入门和遗忘门合并为更新门（更新门决定隐状态保留放弃部分）



输入: h_{t-1}, x_t

输出: h_t, y^t

隐状态输出 $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

新信息: $\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$

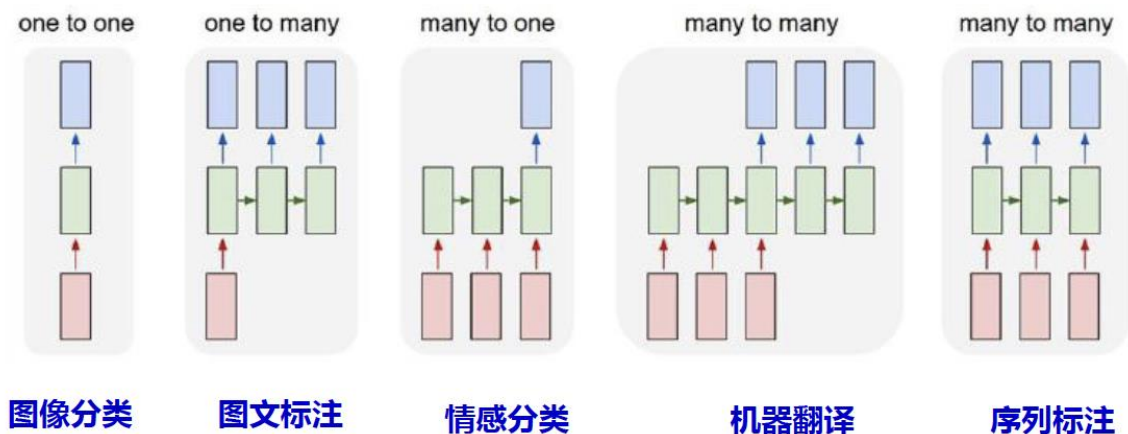
重置门: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$

更新门: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$

输出 $y^t = \sigma(W' h^t)$

参数: W_z, W_r, W, W'

RNN/LSTM 建模的序列问题:



第四章复习 语言模型+词向量

4.1 统计语言模型

语言模型：提出了用数学的方法描述语言规律；

语言模型的基本思想：用句子 $S=w_1, w_2, \dots, w_n$ 的概率 $p(S)$ 刻画句子的合理性

对语句合理性判断：

规则法：判断是否合乎语法、语义（语言学定性分析）

统计法：通过可能性（概率）的大小来判断（数学定量计算）

语言模型：

$$p(S) = p(w_1)p(w_2|w_1)\dots p(w_n|w_1, \dots, w_{n-1})$$

$$= \prod_{i=1}^n p(w_i | w_1 \dots w_{i-1})$$

n-gram 语言模型：

模型假设一个词的出现概率只与它前面的 $n-1$ 个词相关，距离大于等于 n 的上文词会被忽略。理论上 n 越大越好，但是再高阶模型也无法覆盖所有的语言现象； n 越大，需要估计的参数就越多。3-gram 用的最多，4-gram 以上需要的参数太多，少用。

例： 给定句子：John read a book 求 概率

解： 增加标记：<BOS> John read a book <EOS>

基于1元文法的概率为：

$$p(\text{John read a book}) = p(\text{John}) \times p(\text{read}) \times p(a) \times p(\text{book})$$

基于2元文法的概率为：

$$p(\text{John read a book}) = p(\text{John}|\text{<BOS>}) \times p(\text{read}|\text{John}) \times p(\text{a}|\text{read}) \times p(\text{book}|\text{a}) \times p(\text{<EOS>}|\text{book})$$

语言模型参数估计：获得语言模型中所有词的条件概率（语言模型参数，即：词表中词 W_i 的条件概率 $p(w_i|w_1, \dots, w_{i-1})$ ）。

(1) 训练语料：

- 训练语料应尽量和应用领域一致
- 语料尽量足够大
- 训练前应预处理

(2) 参数学习的方法：最大似然估计

对于 n-gram，参数 $p(w_i | w_{i-n+1}^{i-1})$

$$p(w_i | w_{i-n+1}^{i-1}) = f(w_i | w_{i-n+1}^{i-1}) = \frac{\sum_{w_i} c(w_{i-n+1}^i)}{\sum_{w_j} c(w_{i-n+1}^j)}$$

其中：

$\sum_{w_i} c(w_{i-n+1}^{i-1})$ 是历史串 w_{i-n+1}^{i-1} 在给定语料中出现的次数

$\sum_{w_i} c(w_{i-n+1}^i)$ ，为 w_{i-n+1}^{i-1} 与 w_i 同现的次数

例2：求句子 *John read a book* 的概率（2元文法）

解：

$$p(\text{John read a book}) = p(\text{John}|\text{<BOS>}) \times p(\text{read}|\text{John}) \times p(\text{a}|\text{read}) \times p(\text{book}|\text{a}) \times p(\text{<EOS>}|\text{book})$$

$$P(\text{John read a book}) = \frac{1}{3} \times 1 \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} \approx 0.06$$

参数的数据平滑：调整最大似然估计的概率值，使零概率增值，使非零概率下调。

数据平滑方法：

➤ 加 1 法(Additive smoothing):

基本思想：每一种情况出现的次数加1。

如，对于2-gram 有：

$$p(w_i | w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i} [1 + c(w_{i-1}w_i)]} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}$$

其中，V 为被考虑语料的词汇量（全部可能的基元数）

➤ 减值法、折扣法（Discounting）：

基本思想：修改训练样本中事件的实际计数，使样本中(实际出现的)不同事件的概率之和小于1，剩余的概率量分配给未见概率。

1) Good-Turing 2) Back-off (Katz)

3) 绝对减值(H. Ney) 4) 线性减值

➤ 删除插值法(Deleted interpolation):

基本思想：用低阶语法估计高阶语法，即当 3-gram 的值不能从训练数据中准确估计时，用 2-gram 来替代，同样，当 2-gram 的值不能从训练语料中准确估计时，可以用 1-gram 的值来代替。插值公式：

$$p(w_3 | w_1 w_2) = \lambda_3 p'(w_3 | w_1 w_2) + \lambda_2 p'(w_3 | w_2) + \lambda_1 p'(w_3)$$

$$\text{其中, } \lambda_1 + \lambda_2 + \lambda_3 = 1$$

(详情略)

语言模型的性能评价：

1. 实用方法：

通过查看该模型在实际应用（如拼写检查、机器翻译）中的表现来评价，优点是直观、实用，缺点是缺乏针对性、不够客观。

2. 理论方法:

用模型的 迷惑度/困惑度/混乱度 (preplexity) 衡量。其基本思想是能给测试集赋予较高概率值 (低困惑度) 的语言模型较好

困惑度定义

平滑的 n-gram 模型句子的概率:
$$p(s) = \prod_{i=1}^{m+1} p(w_i | w_{i-n+1}^{i-1})$$

假定测试语料 T 由 l_T 个句子构成 (t_1, \dots, t_{l_T})

则整个测试集的概率为:
$$p(T) = \prod_{i=1}^{l_T} p(t_i)$$

模型 $p(w_i | w_{i-n+1}^{i-1})$ 对于测试语料的交叉熵:

$$H_p(T) = -\frac{1}{W_T} \log_2 p(T)$$

其中, W_T 是测试文本 T 的词数。

模型 p 的 困惑度 $PP_p(T)$ 定义为:
$$PP_p(T) = 2^{H_p(T)}$$

4.2 神经语言模型

神经语言模型概念:

语言模型

$$p(S) = \prod_{i=1}^n p(w_i | w_1 \dots w_{i-1})$$

输入: 句子 S

输出: 句子概率 $p(S)$

参数: $p(w_i | w_1, \dots, w_{i-1})$

函数关系: $p(S) = \prod_{i=1}^n p(w_i | w_1 \dots w_{i-1})$

对于语言模型参数 $p(w_i | w_1, \dots, w_{i-1})$

统计语言模型: 用概率统计法学习参数

神经语言模型: 用神经网络学习参数

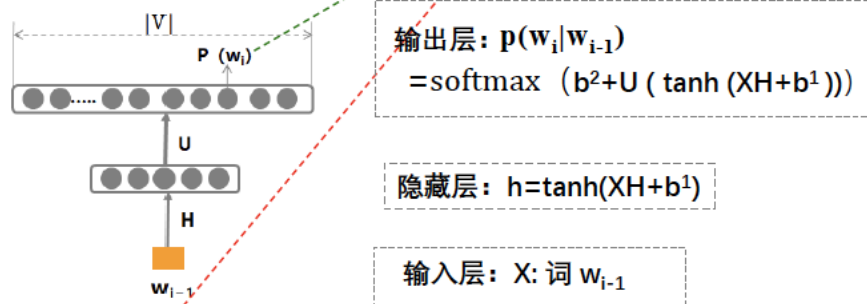
DNN 语言模型 (NNLM) :

(1) NNLM模型结构

语言模型参数

2-gram:

$$p(w_1, \dots, w_m) = \prod_{i=1}^m p(w_i | w_{i-1})$$



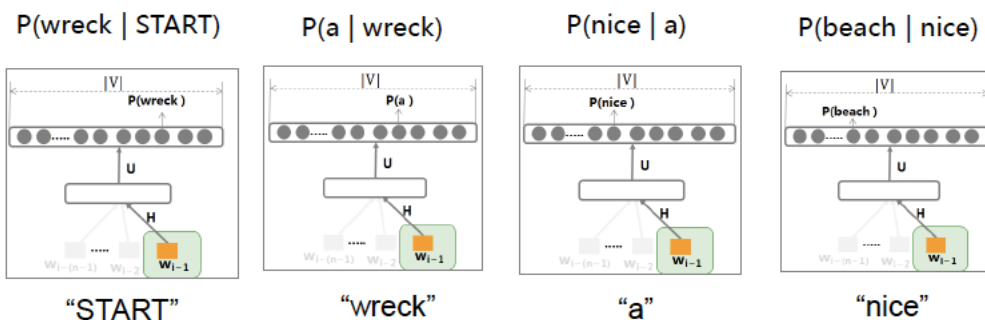
参数: $\theta = \{H, U, b^1, b^2\}$

神经网络参数

(3) NNLM模型预测

例: $P(\text{"wreck a nice beach"})$

$$= P(\text{wreck} | \text{START})P(a | \text{wreck})P(\text{nice} | a)P(\text{beach} | \text{nice})$$

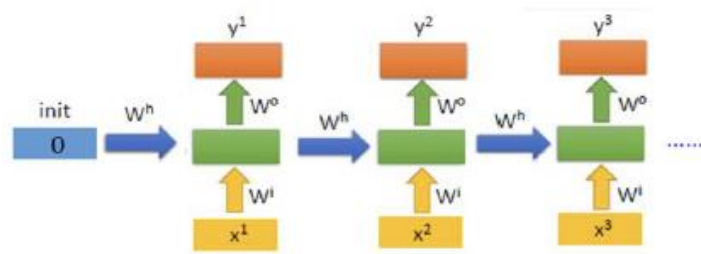


每求一个参数用一遍神经网络

RNN 语言模型 (RNNLM) :

目标: 用神经网络RNN 学习语言模型 $p(S) = \prod_{i=1}^n p(w_i | w_1 \dots w_{i-1})$

模型参数 $p(w_i | w_{i-(n-1)} \dots w_{i-1})$



RNNLM 优点:

- RNNLM 模型可以保留每个词的全部历史信息，不需简化为 n-gram
- 引入词向量作为输入后不需要数据平滑

神经语言模型对比概率语言模型的优势:

概率语言模型存在问题

1. 由于参数数量问题需要对词 i 的历史简化 n-gram
2. 需要数据平滑

神经网络语言模型

利用RNN 语言模型可以解决以上概率语言模型问题

当语言模型训练好后，模型网络参数固定，这时给

任意的 W_{i-1} $P(W_i)$ 不会为 0 ➡ 无需数据平滑

4.3 词向量

词的表示:

1. 离散表示 One-hot:

优势: 系数方式存储简洁

缺点: 词汇鸿沟，维数灾难

2. 离散表示 词袋模型:

每个数表示该词在文档中出现的次数（One-hot 的加和）

3. TF_IDF: 每个数代表该词在整个文档中的占比

词的分布式表示:

核心思想: 用一个词附近的其他词来表示该词

分布式假设: 在相同上下文中出现的词倾向于具有相同的含义 [Harris ,1954]

如, Marco saw a hairy little wampinuk Crouching behind a tree .
wampinuk 的含义可由其上下文推断。

分布式语义学: 根据词语在大型文本语料中的分布特性量化词语及词语语义相似性。

词向量特性:

➤ 语义相似的词, 其词向量空间距离更相近 (分布假说)

优点: 降维, 消除词汇鸿沟其语言模型自带平滑功能

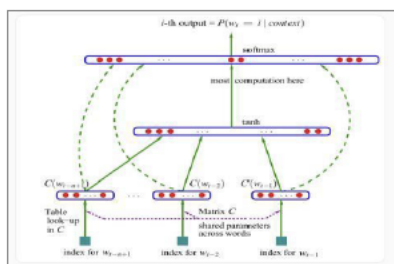
应用: 同义词检测、单词类比等

➤ 相似关系词对的词向量之差也相似

应用: 直接使用词向量的加减法进行推理

(1) NNLM模型词向量

NNLM模型-输入表示



词的 one-hot 表示

张: 0000100...00

三: 0010000...00

李: 00000010...00

四: 0100000...00

.....
|V|

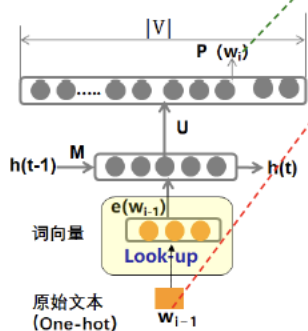
问题: one-hot 表示维度太高

(2) RNNLM模型词向量

■ RNNLM模型结构(词向量)

语言模型参数

2-gram: $p(w_1, \dots, w_m) = \prod_{i=1}^m p(w_i | w_{i-1})$



输出层: $p(w_i | w_{i-1})$
 $= \text{softmax}(b^2 + U h(t))$

隐藏层: $h(t) = \tanh(X + M h(t-1) + b^1)$

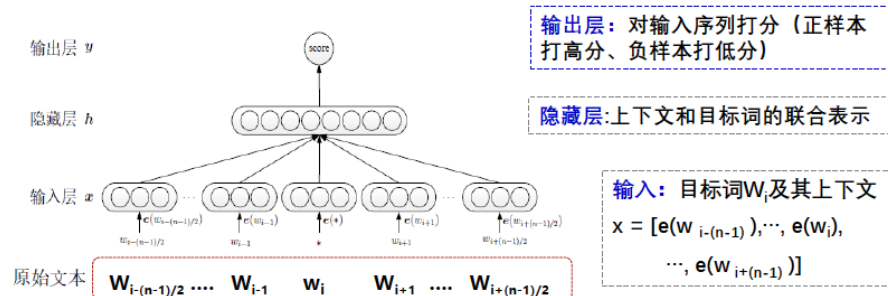
输入层: X: 词 w_{i-1} 的词向量 $X = e(w_{i-1})$ 初值 $h(t-1)$

参数: $\theta = \{U, M, b^1, b^2, \text{词向量}\}$

神经网络参数

(3) C&W 模型词向量

■ C&W模型结构



为从语料中选出的一个 n 元短语 $w_{i-(n-1)/2}, \dots, w_i, \dots, w_{i+(n-1)/2}$ 一般 n 为奇数, 以保证上文和下文的词数一致; w_i 为目标词(序列中间的词) $x = [e(w_{i-(n-1)/2}), \dots, e(w_i), \dots, e(w_{i+(n-1)/2})]$

■ C&W模型学习

优化目标:

对于整个语料最小化:

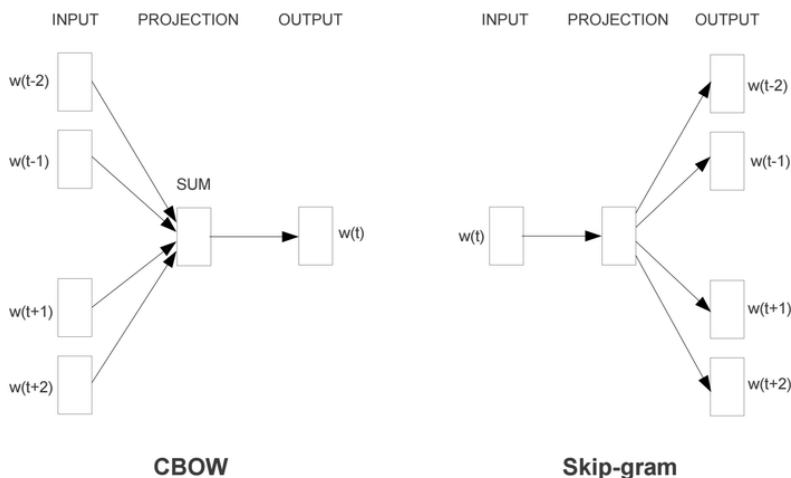
$$\sum_{(w,c) \in D} \sum_{w' \in V} \max(0, 1 - \text{score}(w, c) + \text{score}(w', c))$$

其中, (w, c) : c 表示目标词 w 的上下文

- 正样本 (w, c) 来自语料
- 而负样本 (w', c) 则是将正样本序列中的中间词替换成其它词

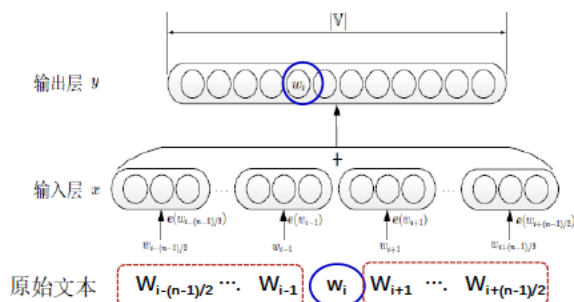
参数训练:

采用 pairwise 的方式对文本片段进行优化, 即可得词向量



(4) CBOW 模型

■ CBOW 模型结构



输出层: $p(w_i | C)$

$$= \frac{\exp(e'(w_i)^T x)}{\sum_{w' \in V} \exp(e'(w')^T x)}$$

输入层: x : 词 w_i 的上下文词向量平均值

$$x = \frac{1}{n-1} \sum_{w_j \in c} e(w_j)$$

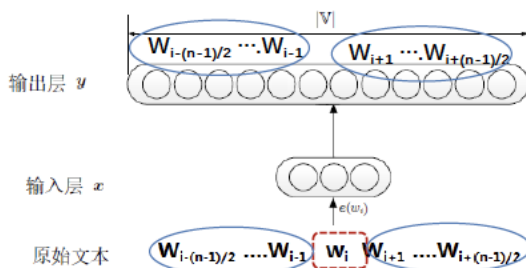
为从语料中选出的一个 n 元短语 $w_{i-(n-1)/2}, \dots, w_i, \dots, w_{i+(n-1)/2}$ 一般 n 为奇数, 以保证上文和下文的词数一致; w_i 为目标词, w_i 上下文 C 为不包括 w_i 的 $n-1$ 元短语

■ 模型学习

- 优化目标: 最大化: $\sum_{(w,c) \in D} \log P(w|c)$
- 参数训练: 梯度下降法

(5) Skip-gram 模型

■ Skip-gram 模型



输出层: $P(w_j | w_i)$

$$= \frac{\exp(e'(w_j)^T x)}{\sum_{w' \in V} \exp(e'(w')^T x)}$$

输入层: x : 词 w_i 的词向量

将目标词 w_i 的词向量作为输入, 每次从 w_i 的上下文 C 中选一个词作为预测词进行预测。目标词 w_i 及上下文 C 定义同 CBOW 模型

■ 模型学习

- 优化目标: 最大化 $\sum_{(w,c) \in D} \sum_{w_j \in c} \log P(w_j | w)$
- 参数训练: 梯度下降法

模型特点

模型	目标词与上下文位置 (n-gram)	模型输入	模型输出	目标词与上下文 词之间的关系
NNLM	(上文)(目标词)	上文词向量拼接	目标词概率	上文在输入层、 目标词在输出层， 优化预测关系
C&W	(上文)(目标词)(下文)	上下文及目标 词词向量拼接	上下文及目 标词联合打 分	上下文和目标词 都在输入层，优 化组合关系
CBOW	(上文)(目标词)(下文)	上下文各词词 向量平均值	目标词概率	上下文在输入层、 目标词在输出层， 优化预测关系
Skip-gram	(上文)(目标词)(下文)	目标词词向量	上下文词概 率	目标词在输入层、 上下文在输出层， 优化预测关系

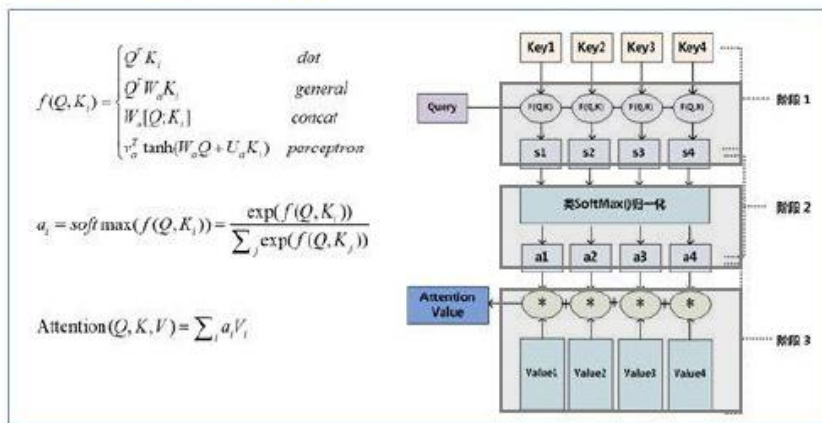
第五章复习 注意力机制

5.1 注意力机制概述

注意力机制： 加权求和机制/模块

5.2 传统注意力机制

输入 → 输出 函数关系：



步骤1： 计算 $f(Q, K_i)$

步骤2： softmax ($f(Q, K_i)$) (计算对于Q 各个 K_i 的权重)

步骤3： 计算输出 (各 K_i 乘以自己的权重，然后求和)

注： 内积相当于计算向量到另一个向量的投影，也就是求相似度，这是 $K \times Q$ 步骤；然后需要对其归一化，因为 K 和 Q 可能没经过 L2 正则化，数值不定；相似度和数值乘机作为总体，相当于对数值进行加权，数值本身的大小也有意义。这里图中的 query 为单个，可以视为单个单词。所以最终输出的注意力值是关于此 query 单词的。

注意考点，键值对模式就是 QKV，普通模式是 QKK。

普通模式

K1	K2	K3	K4	K5
----	----	----	----	----

$$\text{Att-V} = a1 \times K1 + a2 \times K2 + a3 \times K3 + a4 \times K4 + a5 \times K5$$

键值对模式

K1	K2	K3	K4	K5
V1	V2	V3	V4	V5

$$\text{Att-V} = a1 \times V1 + a2 \times V2 + a3 \times V3 + a4 \times V4 + a5 \times V5$$

注意力模块训练：

将模块放到整体模型中，不需要额外的训练数据。

注意力模块评价：

放到各个任务中检验，通过任务指标的提升证明模块的效果。

了解软注意力（概率）、硬注意力（只有 0，1）、全局注意力和局部注意力。

注意力机制优势：

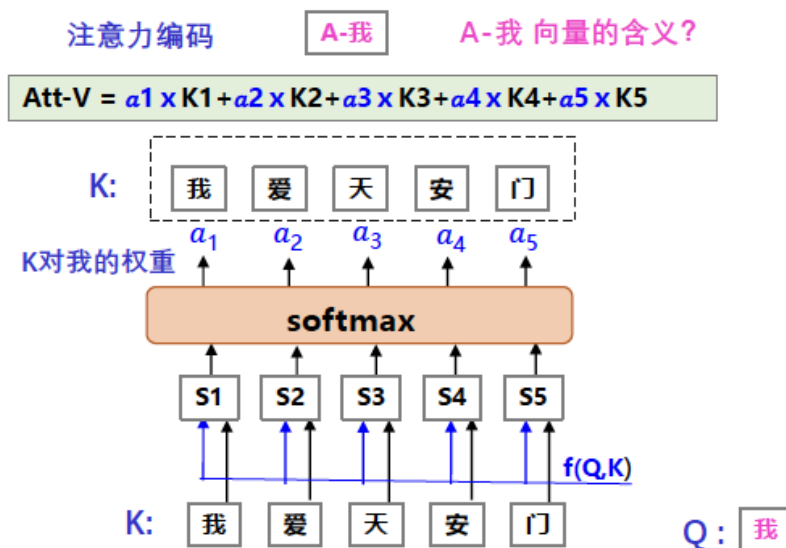
- 让任务处理系统找到与当前任务相关显著的输入信息，并按重要性进行处理，从而提高输出的质量。
- 不需要监督信号，可推理多种不同模态数据之间的难以解释、隐蔽性强、复杂映射关系，对于先验认知少的问题，极为有效。
- 解决长距离依赖问题，提升任务性能。

传统注意力机制存在问题：对 RNN 有注意力偏置问题

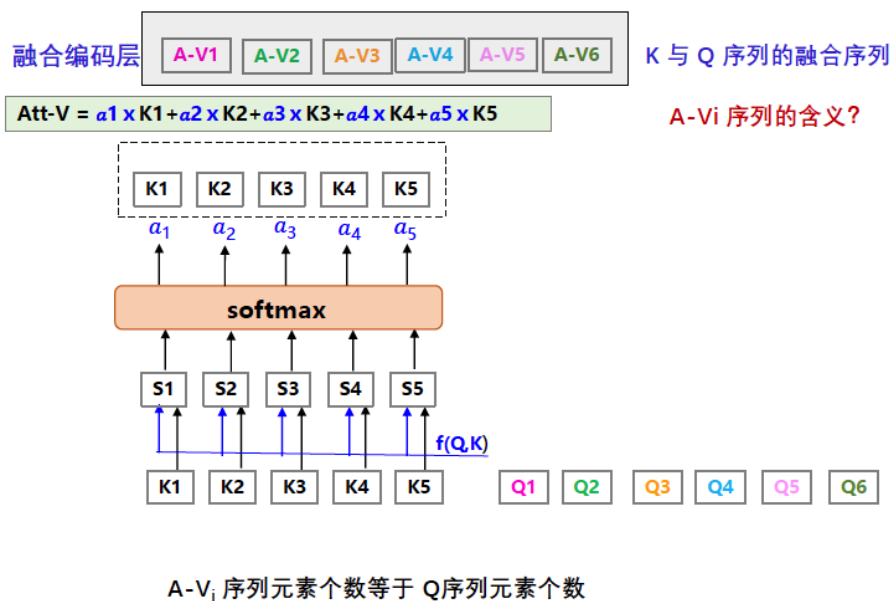
解觉：加入 Coverage 机制可以缓解，就是过往时间步 attention 分布的和。

5.3 注意力编码机制

单一向量编码：将输入序列按规则编码成单一向量表示。

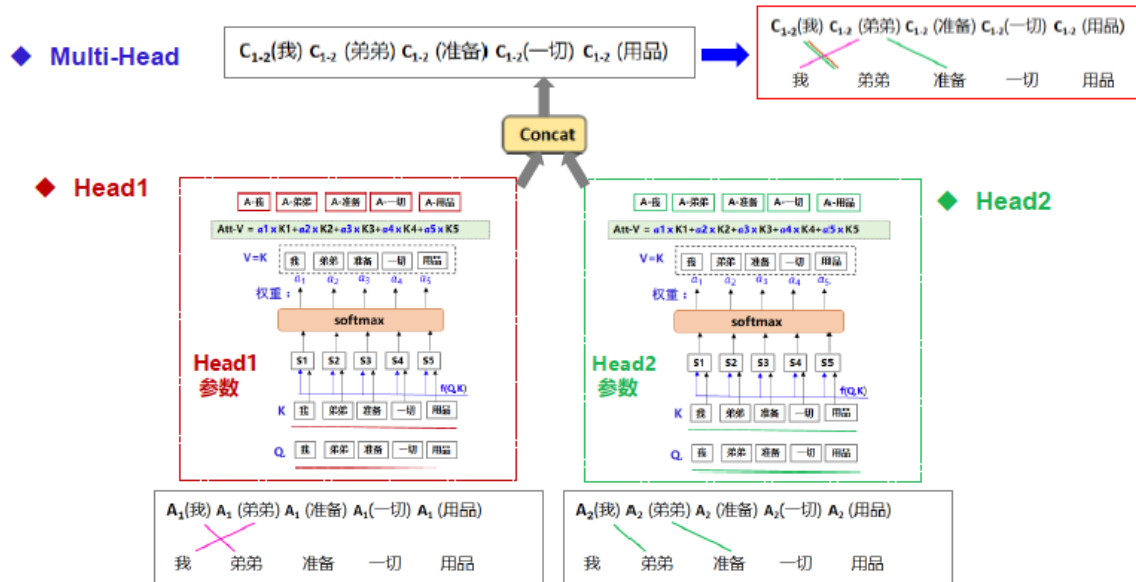


不同序列间编码：将 2 个序列编码成二者的融合的代表序列。匹配任务和阅读理解任务常用的融合层表示。



同一序列自编码：利用多头自注意力编码对一个句子编码可以起到类似句法分析器的作用。

多头注意力机制：多个相同结构，但参数不同不共享，拼接结果。



第六章复习 序列标注

序列标注问题举例:

命名实体识别（人名识别）

命名实体识别（组织机构名识别）

信息抽取（实体识别）

词性序列标注 (POS)

图例：

如：输入序列：新任总裁**罗建国**宣布了对部门经理**邓奇**的任免通知
↓
输出序列：0 0 0 0 **B | E** 0 0 0 0 0 0 0 0 0 **B E** 0 0 0 0 0

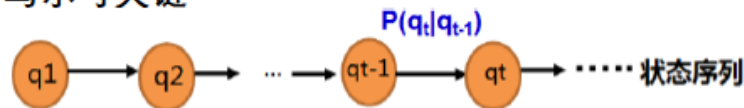
$\{B | EO\}$ 或 $\{B | O\}$

B - 词首
I - 词中
E - 词尾
O - 单个词

马尔科夫模型 MM:

$$p(S_0, S_1, \dots, S_T) = \prod_{t=1}^T p(S_t | S_{t-1}) p(S_0)$$

马尔可夫链




独立于时间 t 的随机过程:

$$P(q_t = S_j \mid q_{t-1} = S_i) = a_{i,j}, 1 \leq i, j \leq N$$

其中：状态转移概率 a_{ij} 必须满足 $a_{ij} \geq 0$ ，

$$\text{且} \quad \sum_{j=1}^N a_{i,j} = 1$$

通常给定三元组，**S**：状态集合；**A**：状态转移矩阵； π ：初始状态向量。

$$S = \{s_1 \ s_2 \ s_3 \ \dots \ s_n\}$$


$$A = [a_{ij}] = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 & \dots & s_n \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{matrix} & \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \end{matrix}$$

其中：状态转移概率 a_{ij}

$$P(q_i = S_j | q_{i-1} = S_i) = a_{ij}, 1 \leq i, j \leq N$$

$$\text{满足 } a_{ij} \geq 0, \text{ 且 } \sum_{j=1}^N a_{ij} = 1$$

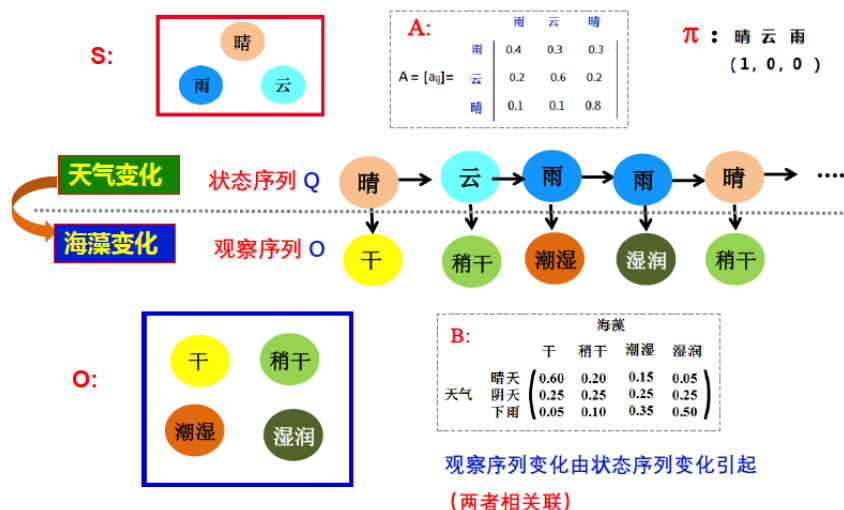
π 初始状态向量

注意：通常初始化的状态概率 $p(S_0) = 1$

隐马尔科夫模型 **HMM**：

HMM 是一个双重随机过程，我们不知道具体的状态序列，只知道状态转移的概率，即模型的状态转换过程是不可观察的（隐蔽的）而可观察事件的随机过程是隐蔽状态转换过程的随机函数。

马尔可夫模型： \longrightarrow 隐马尔可夫模型 HMM



HMM 特点:

- HMM 的状态是不确定或不可见的只有通过观测序列的随机过程才能表现出来
- 观察到的事件与状态 并不是一一对应，而是通过一组概率分布相联系
- HMM 是一个双重随机过程，两个组成部分：
 - 马尔可夫链：描述状态的转移 用转移概率描述。
 - 一般随机函数：描述状态与观察序列间的关系用观察值概率描述。

HMM五元组说明：

1. 隐藏状态s：一个系统的(真实)状态，可以由一个马尔科夫过程进行描述（如, 天气）

3. 观察状态 o：在这个过程中‘可视’的状态（例如，海藻的湿度）

3. 状态转移概率矩阵 $A = a_{ij}$ ：包含了一个隐藏状态到另一个隐藏状态的概率。其中，

$$\begin{cases} a_{ij} = p(q_{t+1} = S_j | q_t = S_i), & 1 \leq i, j \leq N \\ a_{ij} \geq 0 \\ \sum_{j=1}^N a_{ij} = 1 \end{cases}$$

4. 观察概率矩阵 $B = b_j(k)$ ：从隐藏状态 S_j 观察到某一特定符号 v_k 的概率分布矩阵。

其中，

$$\begin{cases} b_j(k) = p(O_t = v_k | q_t = S_j), & 1 \leq j \leq N, \quad 1 \leq k \leq M \\ b_j(k) \geq 0 \\ \sum_{k=1}^M b_j(k) = 1 \end{cases}$$

5. 初始状态的概率分布为： $\pi = \pi_i$ ，其中，

$$\begin{cases} \pi_i = p(q_1 = S_i), & 1 \leq i \leq N \\ \pi_i \geq 0 \\ \sum_{i=1}^N \pi_i = 1 \end{cases}$$

HMM 三个假设：

假设 1：马尔可夫性假设（状态构成一阶马尔可夫链）

假设 2：不动性假设（状态与具体时间无关）

假设 3：输出独立性假设（输出仅与当前状态有关）

函数关系：

观察序列的概率值为 HMM 评估问题；

隐状态序列为 HMM 解码问题。

评估问题：

对于给定观察序列 $O = O_1, O_2, \dots, O_T$ ，以及模型 $\lambda = (A, B, \pi)$

评估-前向算法（非计算题考点）：

使用递归来降低计算复杂度

前向算法 (The forward procedure)

(1) 初始化： $\alpha_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N$

(2) 循环计算：

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1$$

(3) 结束，输出：

$$p(O|\mu) = \sum_{i=1}^N \alpha_T(i)$$

与神经网络类似，每个节点表示一个状态，数值代表各个状态在当前时刻 T 给定观测值 O 的条件下的概率。

评估-后向算法（非计算题考点）：

后向算法与前向算法的思想差不多，每个节点表示一个状态，数值代表各个状态在后一时刻的观测值 O 的条件下的概率，在最后一个时刻 T ，各个节点的概率为 1。

后向算法 (The backward procedure)

(1) 初始化: $\beta_T(i) = 1, 1 \leq i \leq N$

(2) 循环计算:

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad T-1 \geq t \geq 1, 1 \leq i \leq N$$

(3) 输出结果:

$$P(O | \mu) = \sum_{i=1}^N \pi_i b_i(O_1) \beta_1(i)$$

HMM 解码问题：

对于给定观察序列 $O = O_1, O_2, \dots, O_T$ ，以及模型 $\lambda = (A, B, \pi)$ ，如何选择 一个对应的状态序列 $S = S_1, S_2, \dots, S_T$ ，使 S 能够最为合理地解释观察序列 O 。



求：状态序列 $S = S_1, S_2, \dots, S_T$

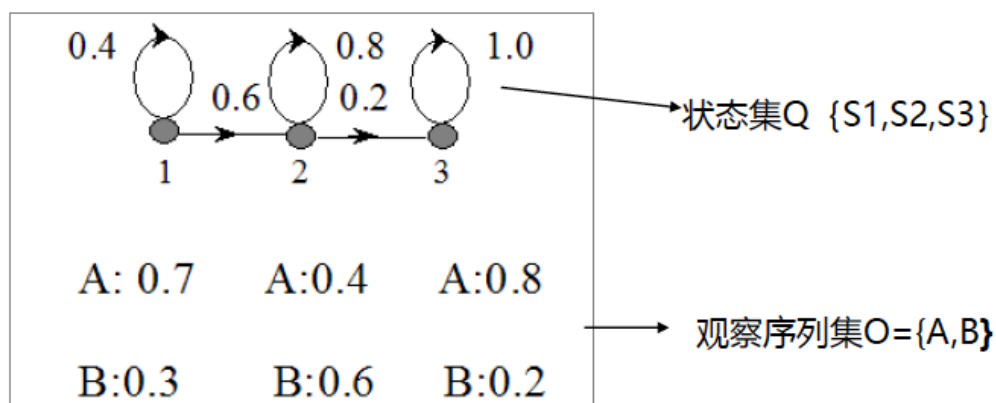
Viterbi 搜索算法

1. 初始化： $\delta_1(i) = \pi_i b_i(O_1)$, $\varphi_1(i) = 0$, $1 \leq i \leq N$
2. 递归： $\delta_t(j) = [\max_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij}] b_j(O_t)$, $2 \leq t \leq T, 1 \leq j \leq N$
 $\varphi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t)$, $2 \leq t \leq T, 1 \leq j \leq N$
3. 终结： $p^* = \max_{1 \leq i \leq N} [\delta_T(i)]$, $q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$
4. 路径回溯： $q_t^* = \varphi_{t+1}(q_{t+1}^*)$, $t = T-1, T-2, \dots, 1$

例题：

例2： HMM模型如下，试根据Viterbi算法计算产生观察符号序列

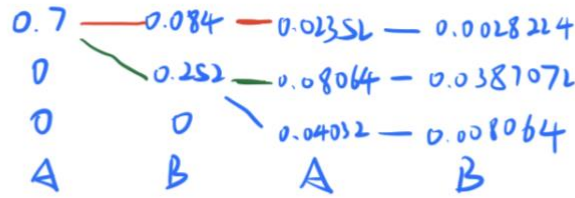
$O = \{A B A B\}$ 的最优状态序列 Q



具体解题步骤如下：

$$a = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0 & 0.8 & 0.2 \\ 0 & 0 & 1 \end{bmatrix} \quad b = \begin{matrix} A & \begin{bmatrix} 0.7 & 0.4 & 0.8 \end{bmatrix} \\ B & \begin{bmatrix} 0.3 & 0.6 & 0.2 \end{bmatrix} \\ S_1 & S_2 & S_3 \end{matrix}$$

$$O = [A \ B \ A \ B]$$



初始概率矩阵: $\pi = [1, 0, 0]$

$$\delta_1(1) = 0.7 \quad \delta_1(2) = 0.4 \quad \delta_1(3) = 0.8$$

迭代:

$$\begin{aligned} \delta_2(1) &= \max_i [\delta_1(i) \cdot a_{i1} \cdot b_{s1}] \\ &= \max [0.7 \times 0.4 \times 0.3, 0 \times 0 \times 0.3, 0 \times 0 \times 0.3] \\ &= 0.084 \\ \delta_2(2) &= \max [0.7 \times 0.6 \times 0.6, 0 \times 0.8 \times 0.6, 0 \times 0 \times 0.6] \\ &= 0.252 \\ \delta_2(3) &= \max [0.7 \times 0 \times 0.2, 0, 0] \\ &= 0 \end{aligned}$$

迭代:

$$\begin{aligned} \delta_3(1) &= \max_i [\delta_2(i) \cdot a_{i1} \cdot b_{s1}] \\ &= \max [0.084 \times 0.4 \times 0.7, 0.252 \times 0 \times 0.7] \\ &= 0.02352 \\ \delta_3(2) &= \max_i [0.084 \times 0.6 \times 0.4, 0.252 \times 0.8 \times 0.4] \\ &= 0.08064 \\ \delta_3(3) &= \max_i [0.084 \times 0, 0.252 \times 0.2 \times 0.8] \\ &= 0.04032 \end{aligned}$$

迭代:

$$\begin{aligned} \delta_4(1) &= \max [0.02352 \times 0.4 \times 0.3, 0, 0] \\ &= 0.0028224 \\ \delta_4(2) &= \max [0.02352 \times 0.6 \times 0.6, 0.08064 \times 0.8 \times 0.6, 0] \\ &= 0.0387072 \\ \delta_4(3) &= \max [0, 0.08064 \times 0.2 \times 0.2, 0.04032 \times 0.2] \\ &= 0.008064 \end{aligned}$$

故最短路径终结于状态 S_2

估算路径: $S_1 - S_2 - S_2 - S_2$
(回溯)

HMM 的参数:

状态转移矩阵 **A** 和观测值分布矩阵 **B** 需要估测:

$$P(S_t | S_{t-1}) = \frac{P(S_{t-1} S_t)}{P(S_{t-1})} \quad P(O_t | S_t) = \frac{P(O_t S_t)}{P(S_t)}$$

方法: 最大似然估计 (有监督)

A 的估计:

$$\bar{a}_{ij} = \frac{\text{Q中从状态 } q_i \text{ 转移到 } q_j \text{ 的次数}}{\text{Q中所有从状态 } q_i \text{ 转移到另一状态(包括 } q_j \text{ 自身)的总数}} = \frac{\sum_{t=1}^{T-1} \delta(q_t, S_i) \times \delta(q_{t+1}, S_j)}{\sum_{t=1}^{T-1} \delta(q_t, S_i)}$$

B 的估计:

$$\bar{b}_j(k) = \frac{\text{Q中从状态 } q_j \text{ 输出符号 } v_k \text{ 的次数}}{\text{Q到达 } q_j \text{ 的总次数}} = \frac{\sum_{t=1}^T \delta(q_t, S_j) \times \delta(O_t, v_k)}{\sum_{t=1}^T \delta(q_t, S_j)}$$

若状态序列 **Q** 无法观测, 则用 **EM** 算法 (无监督)

HMM 参数与 NLP 关系:

观察序列 $O = O_1 O_2 \cdots O_T$: 处理的语言单位, 一般为 **词**

状态序列 $S = S_1 S_2 \cdots S_T$: 与语言单位对应的句法信息, 一般为 **词类**

模型参数: 初始状态概率、状态转移概率、发射概率 需要学习获得

情况1：有大规模分词和词性标注语料 用最大似然估计方法计算各概率

$$\bar{\pi}_{pos_i} = \frac{\text{POS}_i \text{出现在句首的次数}}{\text{所有句首的个数}}$$

$$\bar{a}_{ij} = \frac{\text{从词类POS}_i \text{转移到词类POS}_j \text{的次数}}{\text{所有从状态POS}_i \text{转移到另一POS (包括POS}_j) \text{的总数}}$$

$$\bar{b}_{j(k)} = \frac{\text{从状态POS}_j \text{输出词} w_k \text{的次数}}{\text{状态POS}_j \text{出现的总次数}}$$

情况2： 无标注语料：

◆ 需要一部有词性标注的词典

- 获取词类个数(POS状态数)；
- 获取对应每种词类的词汇数(观察符号数)

◆ 利用无监督EM迭代算法获取初始 状态概率、
状态转移概率 和 发射概率

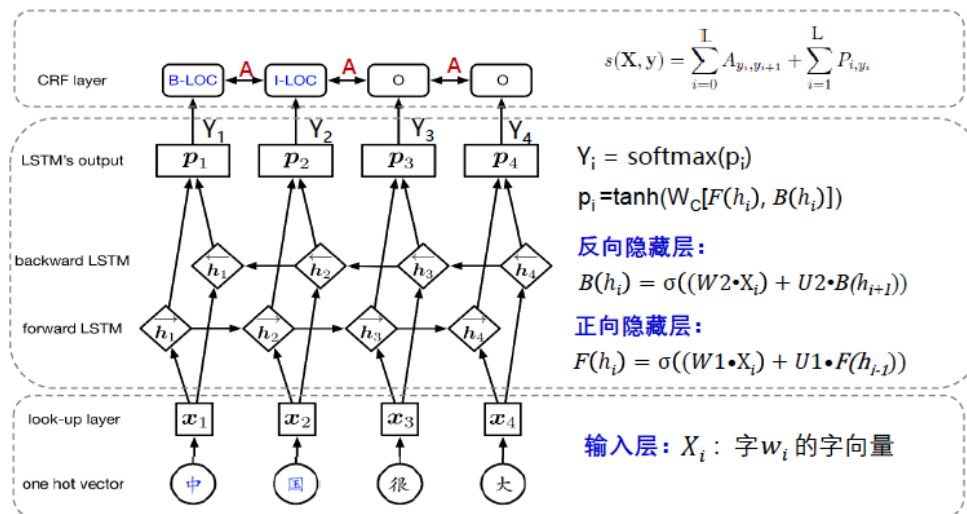
神经网络序列标注（额外学习，考试不考）：

4. 神经网络序列标注模型

(2) 用双向RNN+CRF 模型：

• 模型结构：

输出层： $y^* = \operatorname{argmax}_{\tilde{y} \in Y_X} s(X, \tilde{y})$.



参数： $W_1, U_1, W_2, U_2, W_c, A$

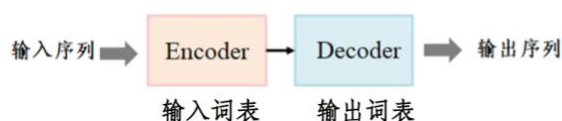
第六章复习 序列生成

本章主要围绕序列生成问题：

1. 三类典型的生成模型：基于编码-解码框架的序列生成模型；基于选择式框架的序列生成模型和基于混合模式的序列生成模型。
2. 序列生成模型存在的问题。
3. 序列生成模型的常用评价指标。

深度学习中建模序列生成问题方法：

构建一个联合的神经网络，以端到端的方式，将一个序列化数据映射成另一个序列化数据。简称 **Sequence-to-Sequence Generation (Seq2Seq)** 模型。主流的 Seq2Seq 模型通常基于 **Encoder-Decoder** 框架实现。



Encoder: 将输入序列进行编码形成后继处理需要的输入表示形式

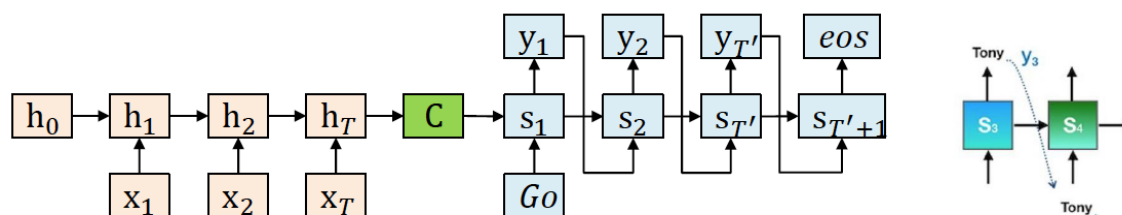
Seq2Seq 模型按输出生成方式分为三类：①生成式、②选择式、③选择-生成式。

- **生成式模型 Decoder:** 根据编码端形成的输入表示和先前时刻产生的输出 **tokens**（就是隐参数），生成当前输出 **token**（编码端和解码端有各自词表，二者可相同或不同。解码端需处理集外词 **OOV**，一般用 **UNK** 代替）

- **选择式模型 Decoder:** 根据编码端形成的输入表示和先前时刻产生的输出 tokens，从输入端选择一个 token 作为输出 token（解码端和编码端词表相同）
- **选择-生成式模型 Decoder:** 根据编码端形成的输入表示和先前时刻产生的输出 tokens，生成或从输入端选择当前输出 token 编码端和解码端有各自词表，二者可相同或不同。解码端需处理集外词 OOV，一般用 UNK 代替，该方法可有效的处理输出端的 OOV 问题）

生成式：

基本 RNN 架构：

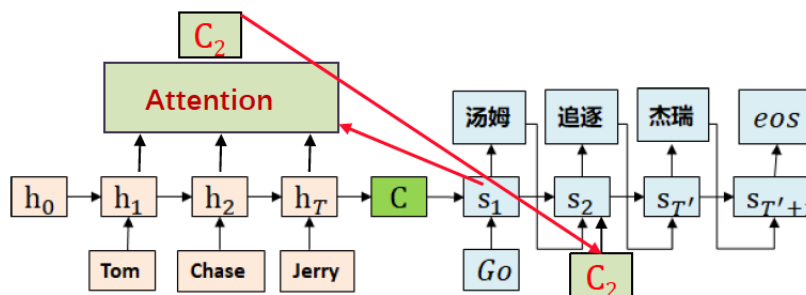


函数关系：

$$P(y_i | y_1 \dots y_{i-1}, x) = g(y_{i-1}, s_i)$$

$$s_i = f(s_{i-1}, y_{i-1})$$

RNN + Attention 架构：



$$Q : \boxed{s_{i-1}}$$

$$K = V : \boxed{h_1} \rightarrow \boxed{h_2} \rightarrow \boxed{h_T}$$

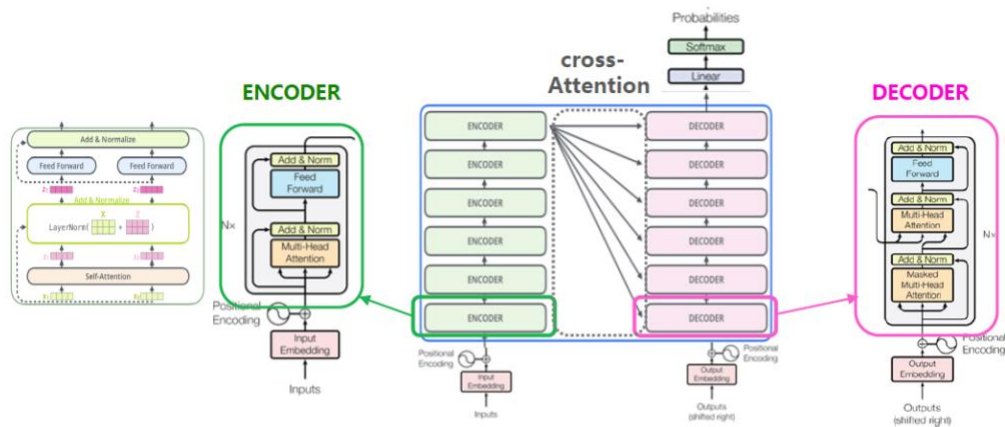
$$\text{Attention}(Q, K, V) : \boxed{C_i}$$

Transformer 架构:

Transformer 模型特点:

- 全部采用 Attention 机制;
- 克服了 RNN 无法并行计算的缺点;
- 有捕捉长距离计算的能力。

结构:



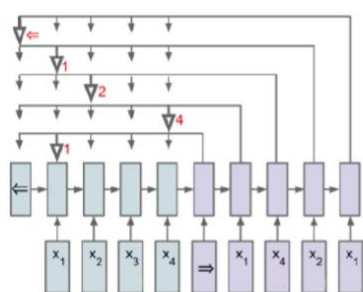
- **编码端:** 6层Attention堆叠, 包含2个子层 (Multi-head attention 和Feed Forward Network)
- **解码端:** 6层Attention堆叠, 包含3个子层 (Multi-head attention , cross-attention和 Feed Forward Network)
- **交叉注意力部分:** 解码端的每一层与编码端的最后输出层做 cross-attention

选择式：

问题：在预测输出端词表的大小是固定的，输出 **token** 是输出词表中概率最大的。这样就无法解决输出词表需按输入情况动态变化的问题，如凸包问题，旅行商问题。例如：如果输入为 4 个点，输出词表宽度为 4 即可，但输入是不确定的，输出词表的宽度如何确定？

■ 指针网络 (Point Networks)

模型结构



输出生成过程：将 $\text{Softmax}(e_i)$ 概率最大元素输出，同时将其作为输出端下个输入产生下个 s_{i+1} ，计算 $e_{i+1,j}$ 重复执行本步骤，直至 $\text{Softmax}(e_i)$ 最大值对应“结束”符号。

输入：X 序列 (x_1, x_2, x_3, x_4)

输出：从输入序列中选出的序列 $(x_1, x_4, x_2, x_1, \text{<Eos>})$

函数关系：

$$e_{ij} = a(s_i, h_j) = V^T \tanh(Ws_i + Uh_j), \quad j \in (1, \dots, n)$$

s_i 表示解码端第 i 时刻的隐状态

h_j 表示编码端第 j 时刻的隐状态

经过 embedding layer 和 RNN 得到隐状态

$$p(y_i | y_1, \dots, y_{i-1}, X) = \text{softmax}(e_i)$$

参数：RNN 参数 + Attention 参数 (W, U, V^T)

特点： y_i 从 X 的标识词典中产生

指针网络在 NLP 领域有广泛的用途，如文本摘要，阅读理解等从输入序列选输出序列的一系列复制类型的任务。

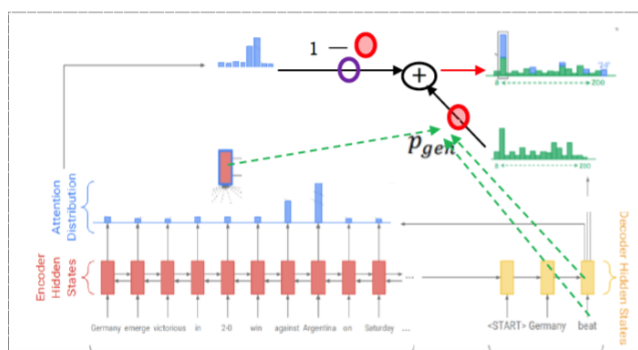
选择-生成式：

指针网络特点：输出直接从输入中选择，输出词表与输入词表相同，无法处理输出需要产生输出词表以外词的情况。例如摘要技术：分为两种：1. 抽取式 **extractive** 2. 生成式 **astractive**，单纯抽取式很难生成高

质量的摘要，单纯生成式摘要难以准确复述原文细节也无法处理原文中的未登录词(OOV)。

2. 指针生成器网络模型 (pointer-generator network)

问题：如何将生成网的词概率分布与指针网词的概率分布融合



融合方法

在时刻 t 由Att向量 h_t^* ，decoder状态向量 s_t 和decoder输入 x_t 共同计算生成概率门控 p_{gen} (软门)

$$p_{gen} \in [0,1]$$

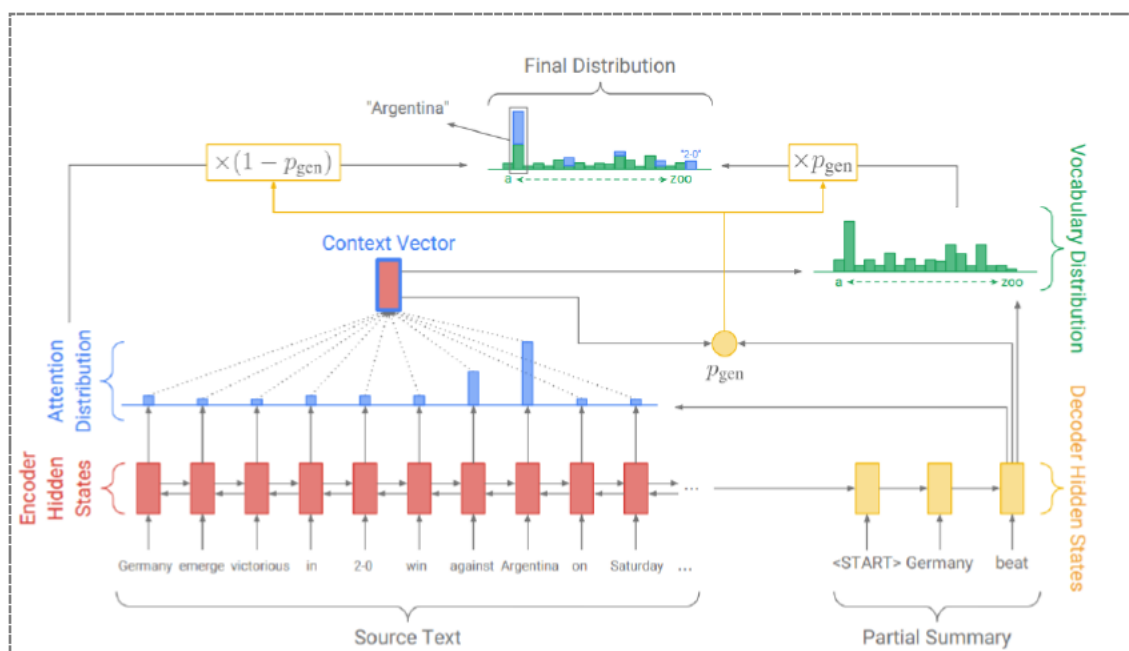
$$p_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr})$$

其中, w_h^* , w_s , w_x , b_{ptr} 是参数

融合后得到最终概率分布

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t$$

注: “+” 是进行维度的扩充并加和



■ 拷贝网络 (Copy-Network)

问题引入：传统 seq2seq 存在的问题：高度的依赖词的表征，当遇到 OOV (Out-of-vocabulary) 会出现表达不准确。而在实际应用中，有些场合仅仅需对该生僻词逐字保留即可

如：对话 hello, my name is Tony Jebara.
 hi, Tony Jebara.

Copy-Net 通过 Copy mechanism 可以将输入序列中的词拷贝到输出序列中，较好的处理 OOV 问题 (测试-文本摘要)

评价指标：

正确率 (Precision)：全部预测为正类中，正确的比例。

$$P = \frac{TP}{TP + FP}$$

召回率 (Recall)：全部预测对的情况下，预测是 Positive 的比例。

$$R = \frac{TP}{TP + FN}$$

BLEU:

$$P_n(\mathbf{x}) = \frac{\sum_{w \in \mathcal{W}} \min(c_w(\mathbf{x}), \max_{k=1}^K c_w(\mathbf{s}^{(k)}))}{\sum_{w \in \mathcal{W}} c_w(\mathbf{x})},$$

ROUGE:

$$\text{ROUGE-N}(\mathbf{x}) = \frac{\sum_{k=1}^K \sum_{w \in \mathcal{W}} \min(c_w(\mathbf{x}), c_w(\mathbf{s}^{(k)}))}{\sum_{k=1}^K \sum_{w \in \mathcal{W}} c_w(\mathbf{s}^{(k)})},$$

其中, $c_w(\mathbf{x})$ 是 n 元组 w 在候选序列 \mathbf{x} 中出现的次数, $c_w(\mathbf{s}^k)$ 是 n 元组 w 在参考序列 \mathbf{s}^k 中出现的次数。

序列生成存在问题：

1. 曝光偏差问题：模型生成的分布与真实的数据分布并不严格一致，也就是 **OOD** 问题。解决办法为混合使用真实数据和模型生成数据。
2. 训练-评价目标不一致问题。序列生成模型一般采用和任务相关的指标来进行评价，比如 **BLEU**、**GOUGE** 等，而训练时使用最大似然估计，这导致训练目标和评价方法不一致。而这些评价指标通常不可微，需要采用强化学习的策略进行模型的训练。

第七章复习 预训练语言模型

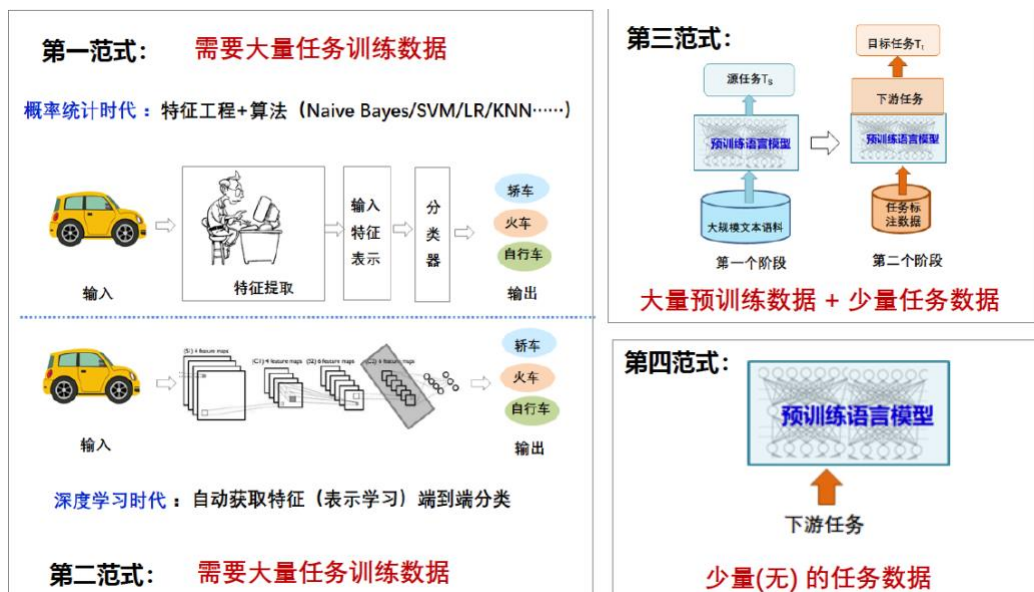
NLP 范式：

P1. 非神经网络时代的完全监督学习 特征工程；特点：人工进行大量的特征模版定义。

P2. 基于神经网络的完全监督学习 架构工程；特点：人工设计各种网络结构。

P3. 预训练，精调范式——目标工程（Pre-train，Fine-tune）；特点：引入各种辅助任务 loss，将其添加到预训练模型中，然后继续 pre-training，以便让其适配下游任务，之后，通过引入额外的参数，用特定任务的目标函数对模型进行微调，使其更适配下游任务。研究重点转向了目标工程，设计在预训练和微调阶段使用的训练目标。

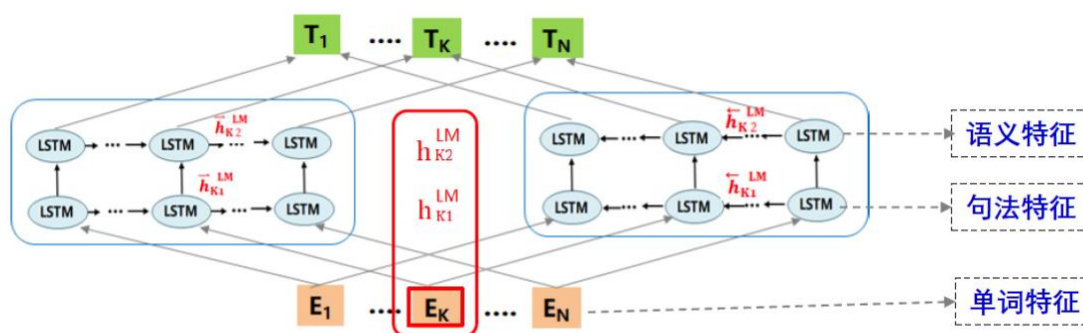
P4. 预训练，提示，预测范式——prompt 挖掘工程；特点：不通过目标工程使预训练的语言模型（LM）适应下游任务，而是将下游任务建模的方式重新定义（Reformulate），通过利用合适 prompt 实现不对预训练语言模型改动太多，尽量在原始 LM 上解决任务的问题。



7.1 第三范式相关

EMLO（了解即可）：

ELMo-BiLM 模型结构



对每个 t_k L层 BiLM 计算 $2L+1$ 个表示

$$R_k = \{x_k^{LM}, \vec{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \mid j = 1, \dots, L\} = \{h_{k,j}^{LM} \mid j = 0, \dots, L\},$$

其中： $h_{k,0}^{LM}$: token 层 ; $h_{k,j}^{LM} = [\vec{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM}]$

$$\text{ELMo}_k = E(R_k; \Theta_e)$$

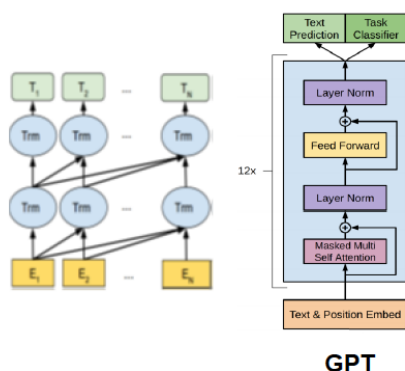
第一个阶段预训练（Pre training）阶段，用大规模无标注数据预训练双向语言模型，然后将 BiLM 网络参数固定得到 ELMo_K

第二阶段任务模型训练 Fine tuning，将 $\text{ELMo}_K^{\text{task}}$ 加入下游任务，用任务标注数据微调任务模型参数及 $\text{ELMo}_K^{\text{task}}$ 中的 S_j^{task} 和 y^{task}

GPT:

GPT 采用了 Transformer 的 **Decoder** 部分，并且每个子层只有一个 Masked Multi Self Attention 768 维向量和 12 个 Attention Head 和一个 Feed Forward（无普通 transformer 解码器层的编码器-解码器注意力子层），模型共叠加使用了 12 层的 Decoder。

■ 模型结构:



输入: $U = \{u_{-k}, \dots, u_{-1}\}$

$$h_0 = UW_e + W_p$$

$$h_l = \text{transformer_block}(h_{l-1}) \quad \forall l \in [1, n]$$

输出: $P(u) = \text{softmax}(h_n W_w^T)$

其中, $U = \{u_{-k}, \dots, u_{-1}\}$ 是当前词的前 k 个 Token, n 为神经网络的层数, W_e 是 Token 的 Embedding 矩阵, W_p 是位置编码的 Embedding 矩阵。

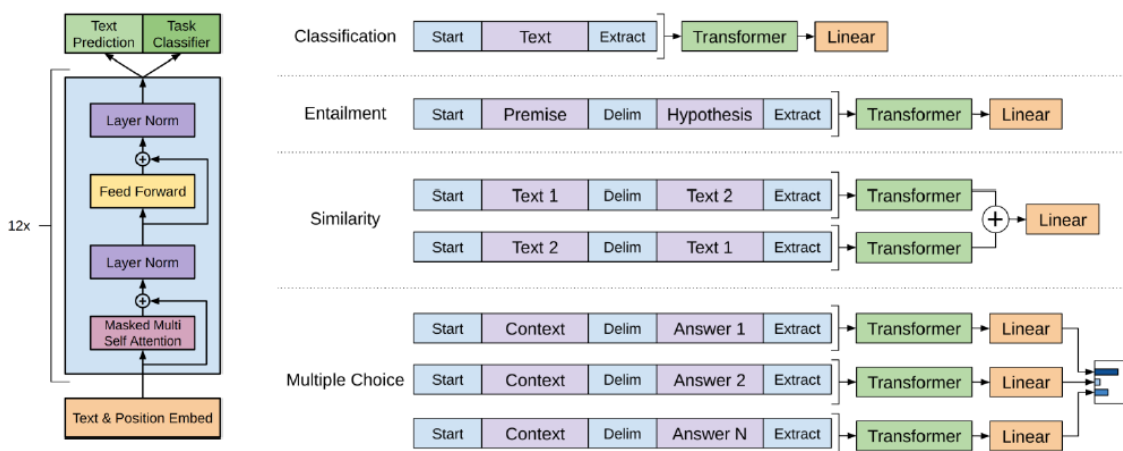
训练过程分为两个阶段:

第一个阶段 Pre-training 阶段, 主要利用大型语料库完成非监督学习;

第二阶段 Fine-tuning, 针对特定任务在相应数据集中进行监督学习, 通过 Fine-tuning 技术来适配具体任务。

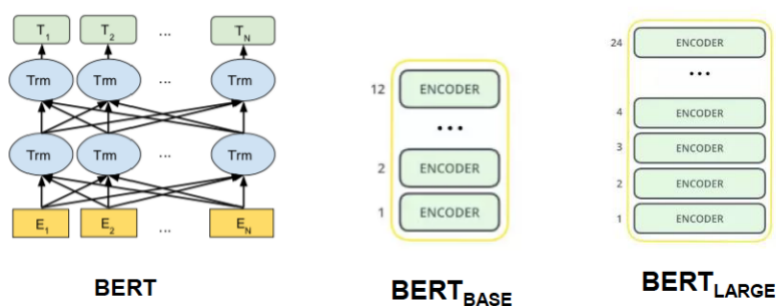
注意: 微调可以与预训练模型参数一起调, 也可以只调任务参数。

模型对接问题:



BERT:

使用堆叠的双向Transformer Encoder，在所有层中共同依赖于左右上下文基础版是12个Encoder (12层)；高级版24个Encoder (24层)



分类任务和匹配任务利用[CLS]位置的 embedding，序列标注任务利用每个 token 的输出 embedding 序列生成任务（选择式生成）第二句 token 输出 embedding。

7.2 第四范式相关

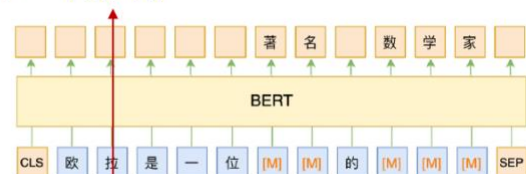
核心思想：改变任务形式利用预训练模型完成任务（用于小样本学习或半监督学习，某些场景下甚至能做到零样本学习。）

二分类：

如：给句子“这个餐厅的服务真不错。”补充描述（补充模板），构建如下的完形填空形式任务：

Prompt-任务输入：_____满意。这个餐厅的服务真不错。

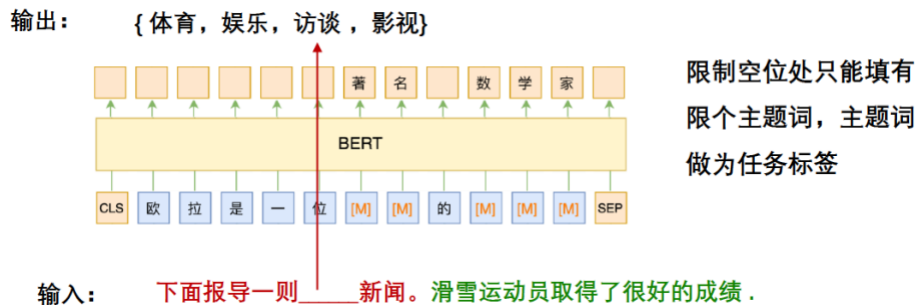
输出：{ 很, 不 }



输入：_____满意。这个餐厅的服务真不错。

构建预测Token的候选空间,限制空位处只能填一个“很”或“不”，即要根据上下文一致性判断是否满意如果“很”的概率大于“不”的概率，是正面情感倾向，否则就是负面情感

多分类：



匹配任务：

目标：判断这两个句子“**我去了北京**”跟“**我去了上海**”是否相容



方法：将两个句子拼接起来输入到模型做，作为一个二分类任务

- 补充描述（补充模板），构建如下的完形填空形式任务：
我去了北京？____，我去了上海。
- 其中空位之处的候选词为 {是的, 不是}

基于 MLM 模型的局限性：MLM 所使用的独立假设限制了它对更长文本的预测能力（空位处的文字不能太长），无法预测不定长的答案也约束了它的场景（所以当前只能用于做选择题，不能做生成）

Prompt 方法：将原输入附加一段补充描述语句，通过这段补充描述语句实现任务转换和对任务求解，这段描述语句与原始输入一起要形成一段语义合理的语句作为 Prompt 的输入。

1. 完形填空 (cloze) prompts: 用在如BERT-MLM式预训练模型上，如情感分

类任务可以输入「**这个饼不错**」，太 [Z]，Z 输出「棒」。一般 [Z]在句中

2. 前缀提示 (prefix) prompts: 用在如GPT2-3单向LM预训练模型上, 输入「好好学习」, 翻译成英文: [Z], [Z] 输出「good good study」.一般 [Z]在句末

Prompt 主要问题:

1. 输入端: 怎么样选取合适的 Prompt, 适配不同任务, 同时把模型潜能激发出来 (怎么把 Prompt 结合输入的句子构成模型的输入)
2. 输出端: 模型的输出 (Answer) 可能与标签不同, 拿到 answer 后如何往标准化的 Y (标签空间) 映射
3. 训练: 怎样利用 Prompt 机制精调模型