语言模型

杨沐昀

语言技术研究中心
哈尔滨工业大学

语言模型

- 语言模型（Language Model，LM）
  - 描述一段自然语言的概率或给定上文时下一个词出现的概率
    - $P(w_1, \dots, w_l)$，$P(w_{l+1}|w_1, \dots, w_l)$
  - 以上两种定义等价（链式法则）
    - $P(w_1, \dots, w_l) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \dots P(w_l|w_1w_2 \dots w_{l-1}) = \prod_{i=1}^{l} P(w_i|w_{1:i-1})$
- 广泛应用于多种自然语言处理任务
  - 机器翻译（词排序）
    - P(the cat is small) > P(small the is cat)
  - 语音识别（词选择）
    - P(there are **four** cats) > P(there are **for** cats)

N元语言模型

- N元语言模型假设下一词出现的概率只与前n-1个词有关

$$P(w_t|w_1 w_2 \dots w_{t-1}) \approx P(w_t|w_{t-(n-1):t-1})$$

$$P(w_1 w_2 \dots w_l) = \prod_{t=1}^{l} P(w_t|w_{t-(n-1):t-1}) = \prod_{t=1}^{l} \frac{C(w_{t-(n-1):t})}{C(w_{t-(n-1):t-1})}$$

- 该假设被称为**马尔可夫假设**（Markov Assumption）
  - 当n=1时，下一个词的出现独立于其历史，相应的一元语法通常记作unigram
  - 当n=2时，下一个词只依赖于前1个词，对应的二元语法记作bigram。二元语法模型也被称为一阶马尔可夫链（Markov Chain）
  - 三元语法假设（n=3）也被称为二阶马尔可夫假设，相应的三元语法记作trigram

N元语言模型的平滑

- 当n过大，或者测试句子中的词不在词表中，会出现零概率问题
- 折扣法平滑（Smoothing）的思想是损有余而补不足，调整概率估计结果
  - 加1平滑（拉普拉斯平滑）

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)+1}{\sum_w \left(C(w_{i-1}w)+1\right)} = \frac{C(w_{i-1}w_i)+1}{C(w_{i-1})+|\mathbb{V}|}$$

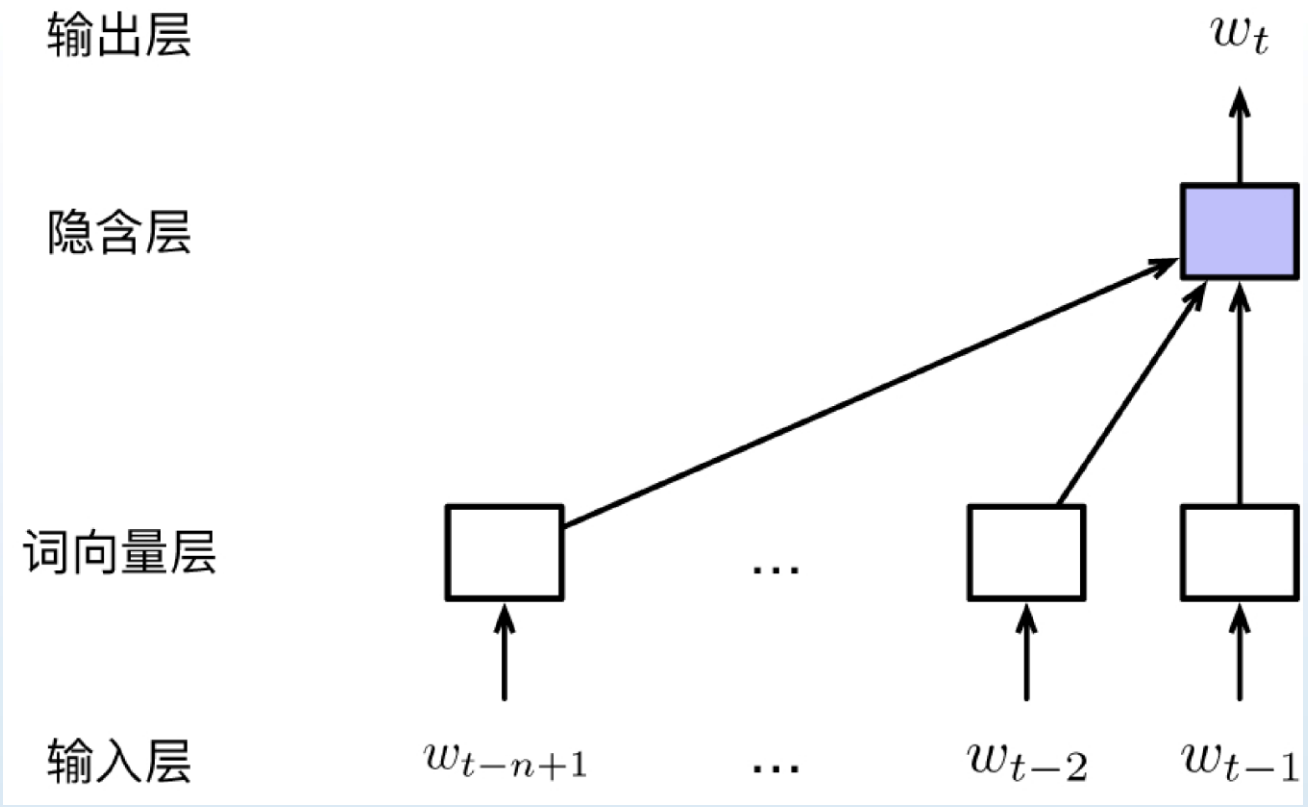
  - 加δ(0<δ<1)平滑(训练数据较小时，加1平滑会给出过高的估计），对于δ的选取，通过比较不同取值下模型的困惑度评价，选取最优取值

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)+\delta}{\sum_w \left(C(w_{i-1}w)+\delta\right)} = \frac{C(w_{i-1}w_i)+\delta}{C(w_{i-1})+\delta|\mathbb{V}|}$$

前馈神经网络语言模型

- 使用前馈神经网络（FFN）估计条件概率

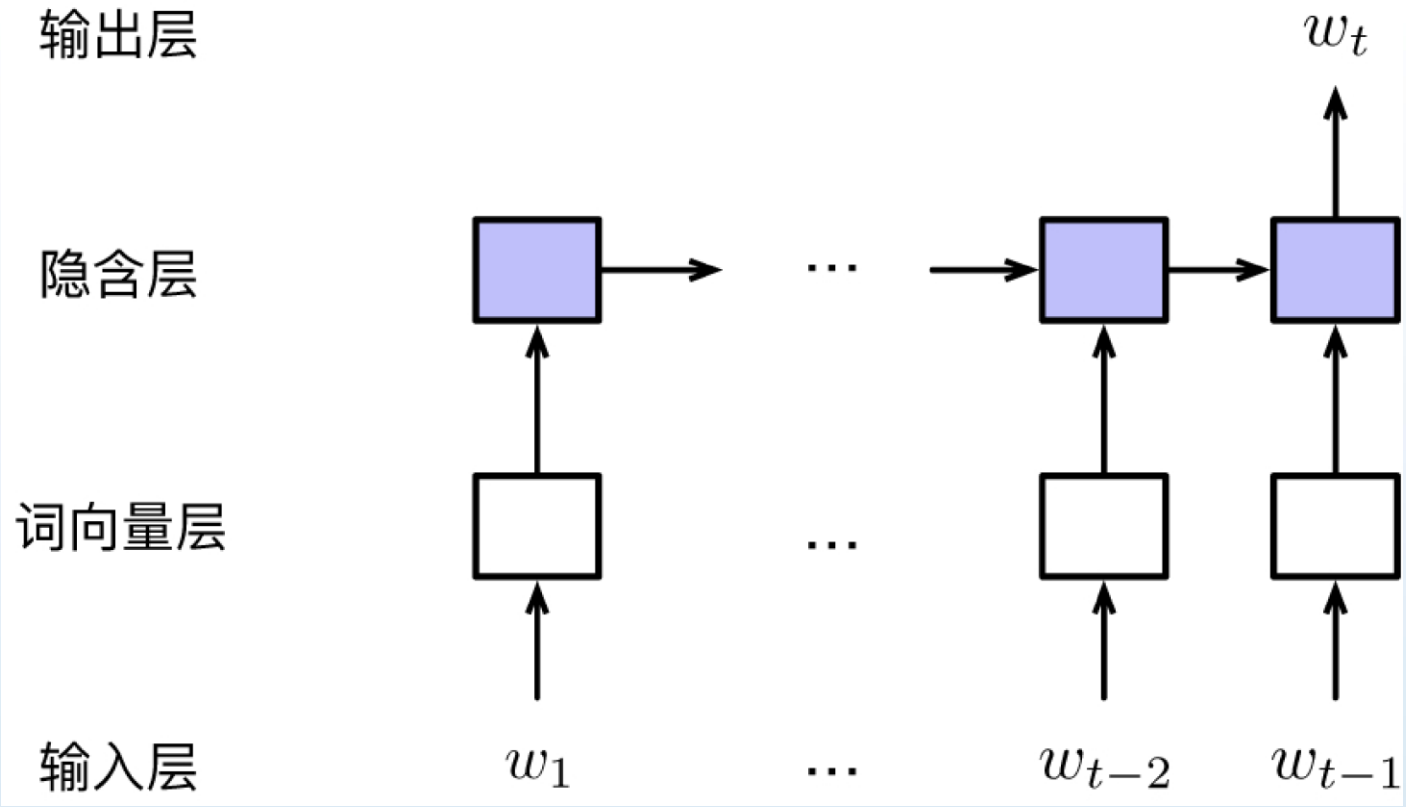
$$P(w_t|w_{1:t-1}) \approx P(w_t|w_{t-n+1:t-1})$$



循环神经网络语言模型

- 使用循环神经网络（RNN）估计条件概率

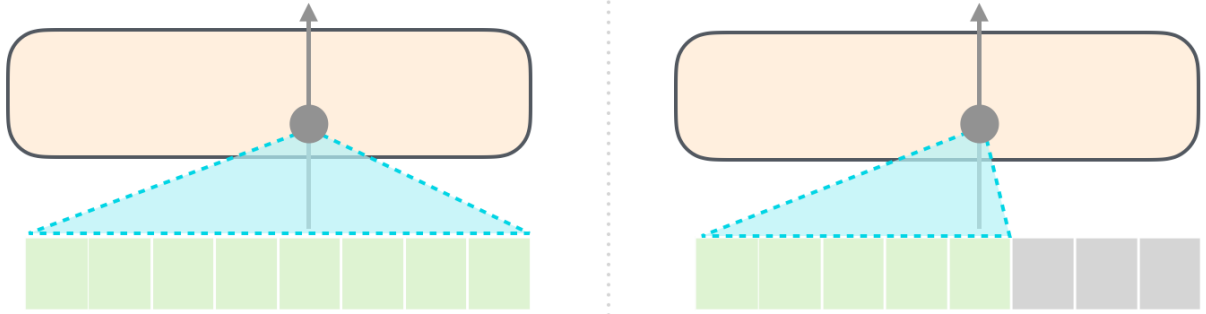
$$P(w_t|w_{1:t-1}) \approx P(w_t|w_{1:t-1})$$



Transformer语言模型

- 使用Transformer估计条件概率

$$P(w_t|w_{1:t-1}) \approx P(w_t|w_{t-n+1:t-1})$$



Features

Example:	position: 1	2	3	4	Labels
1	robot	must	obey	orders	must
2	robot	must	obey	orders	obey
3	robot	must	obey	orders	orders
4	robot	must	obey	orders	<eos>

参考： http://jalammar.github.io/illustrated-gpt2/

Transformer语言模型

Mask操作示例

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention Mask →

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

```
>>> mask = torch.tril(torch.ones(4, 4))
tensor([[1., 0., 0., 0.],
        [1., 1., 0., 0.],
        [1., 1., 1., 0.],
        [1., 1., 1., 1.]])
>>> attn = torch.rand(4, 4)
tensor([[0.6975, 0.3628, 0.9422, 0.6832],
        [0.2625, 0.9714, 0.1903, 0.2832],
        [0.8398, 0.2345, 0.4797, 0.5564],
        [0.4052, 0.7003, 0.9535, 0.5096]])
>>> attn = attn.masked_fill(mask == 0, float('-inf'))
tensor([[0.6975,   -inf,   -inf,   -inf],
        [0.2625, 0.9714,   -inf,   -inf],
        [0.8398, 0.2345, 0.4797,   -inf],
        [0.4052, 0.7003, 0.9535, 0.5096]])
```

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Softmax
(along rows) →

Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

```
>>> F.softmax(attn, dim=-1)
tensor([[1.0000, 0.0000, 0.0000, 0.0000],
        [0.3299, 0.6701, 0.0000, 0.0000],
        [0.4457, 0.2433, 0.3110, 0.0000],
        [0.1929, 0.2591, 0.3338, 0.2141]])
```

参考： http://jalammar.github.io/illustrated-gpt2/

语言模型的性能评价

- 基于具体应用的外部评价（最接近实际应用需求，但代价高）
- 基于困惑度（Perplexity, PPL）的内部评价
  - 将数据划分成两个不相交的数据集合，分别作为训练集和测试集，检测模型的泛化能力
- 测试集的概率

$$P(\mathbb{D}^{\text{test}}) = P(w_1 w_2 \dots w_N) = \prod_{i=1}^{N} P(w_i | w_{1:i-1})$$

- 困惑度：模型分配给测试集中每个词的概率的几何平均值的倒数

$$\text{PPL}(\mathbb{D}^{\text{test}}) = \left(\prod_{i=1}^{N} P(w_i | w_{1:i-1}) \right)^{-\frac{1}{N}}$$

- 为了避免下溢，转化为指数对数形式

$$\text{PPL}(\mathbb{D}^{\text{test}}) = 2^{-\frac{1}{N} \Sigma_{i=1}^{N} \log_2 P(w_i | w_{1:i-1})}$$

谢 谢!

语言技术紫丁香

微信扫描二维码，关注我的公众号