



BERT相关——（1）语言模型

📅 发表于 2021-08-20 | 🔄 更新于 2021-11-09 | 📁 深度学习
| 📖 字数总计: 1,602 | ⌚ 阅读时长: 6分钟 | 👁 阅读量: 954

BERT相关——（1）语言模型

引言

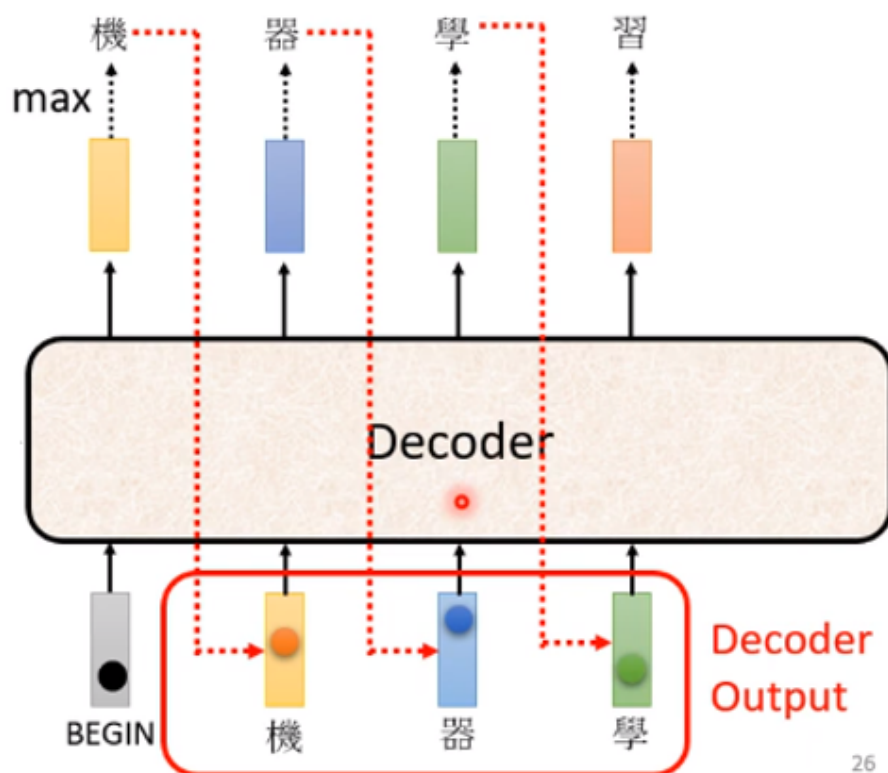
BERT模型的全称是**Bidirectional Encoder Representations from Transformers**，它是一种基于Transformer中Encoder结构的新型语言模型，18年底由谷歌提出。自它提出以后，NLP逐步进入了BERT时代，基于BERT改进的模型相继霸榜。

这一篇开始的“BERT相关”的博客会总结BERT触及的相关知识，帮助更好地理解BERT家族的模型。

想深入了解BERT模型和BERT家族中的模型，我们需要知道语言模型是什么和Transformer结构，后者在“Transformer相关”的系列中做了介绍，在这个系列中就不再详细解释。

语言模型Language Model-LM

语言模型是对一段文本的概率进行估计即针对文本 Y ，计算 $P(Y)$ 的概率。但语言模型仅仅对句子出现的概率进行建模，并不尝试去理解句子的内容含义。其对信息检索，机器翻译，语音识别等任务有着重要的作用。语言模型可以根据句子的一部分预测下一个词，就像下面这张图一样。



26

语言模型分为统计语言模型和神经网络语言模型。

统计语言模型

统计语言模型的基本思想是计算条件概率。

句子: $Y = (y_1, y_2, \dots, y_m)$

输出: $P(Y)$ 的概率

$$P(Y) = P(y_1, y_2, y_3, \dots, y_m) = P(y_2|y_1)P(y_3|y_1, y_2) \dots P(y_m|y_1, y_2, \dots, y_{m-1})$$

$$P(y_6|y_1, y_2, y_3, y_4, y_5) = \frac{\text{count}(y_1, y_2, y_3, y_4, y_5, y_6)}{\text{count}(y_1, y_2, y_3, y_4, y_5)}$$

这个公式有2个问题:

1 自由参数数目

假定字符串中字符全部来自与大小为 V 的词典，上述例子中我们需要计算所有的条件概率，对于所有的条件概率，这里的 y_i 都有 V 种取值，那么实际上这个模型的自由参数数目量级是 V^6 ，6为字符串的长度。从上面可以看出，模型的自由参数是随着字符串长度的增加而指数级暴增的，这使我们几乎不可能正确的估计出这些参数。

2 数据稀疏性

从上面可以看到，每一个 y 都具有 V 种取值，这样构造出了非常多的词对，但实际中训练语料是不会出现这么多种组合的，那么依据最大似然估计，最终得到的概率实际是很可能是0。

N-gram

一个词会和它附近的词联系比较紧密，我们假设一个词的概率和它前面的 k 个词相关，可以对上面的式子进行简化，公式改写如下：

$$P(y_i | y_1, y_2, \dots, y_{i-1}) = P(y_i | y_{i-(k-1)}, \dots, y_{i-1}) \quad (4)$$

$$P(Y) = \prod_{i=1}^m P(y_i | y_{i-(k-1)}, \dots, y_{i-1}) \quad (5)$$

2-gram就是统计两个词汇计算token sequence的几率，**3-gram**就是统计三个词汇等等。

从模型的效果来看，理论上 n 的取值越大，效果越好。但随着 n 取值的增加，效果提升的幅度是在下降的。同时还涉及到一个可靠性和可区别性的问题，参数越多，可区别性越好，但同时单个参数的实例变少从而降低了可靠性。

N-gram比较好解决了自由参数多的问题，但是数据稀疏的问题还是没有被解决。

N-gram Language Model必须要搭配一个**language model smoothing**的技巧。

language model smoothing 平滑化

方法1:

假设有一个词组在训练语料中没有出现过，那么它的频次就为0。但实际上我们无法保证训练语料的完备性，不能认为它出现的概率为0。

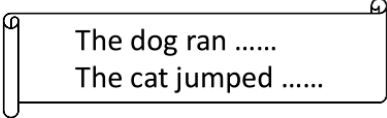
如果我们默认每一个词组都出现1次呢，无论词组出现的频次是多少，都往上加1，这就能够解决概率为0的问题了。

$$P(y_6|y_1, y_2, y_3, y_4, y_5) = \frac{\text{count}(y_1, y_2, y_3, y_4, y_5, y_6) + 1}{\text{count}(y_1, y_2, y_3, y_4, y_5) + 1}$$

或者是给训练数据得到的矩阵为0的地方一个很小的值：

Challenge of N-gram

- The estimated probability is not accurate.
 - Especially when we consider n-gram with large n
 - Because of data sparsity (many n-grams never appear in training data)

Training Data: 

$P(\text{jumped} | \text{the, dog}) = \cancel{0} \text{ } 0.0001$

$P(\text{ran} | \text{the, cat}) = \cancel{0} \text{ } 0.0001$

Give some small probability

This is called **language model smoothing**.

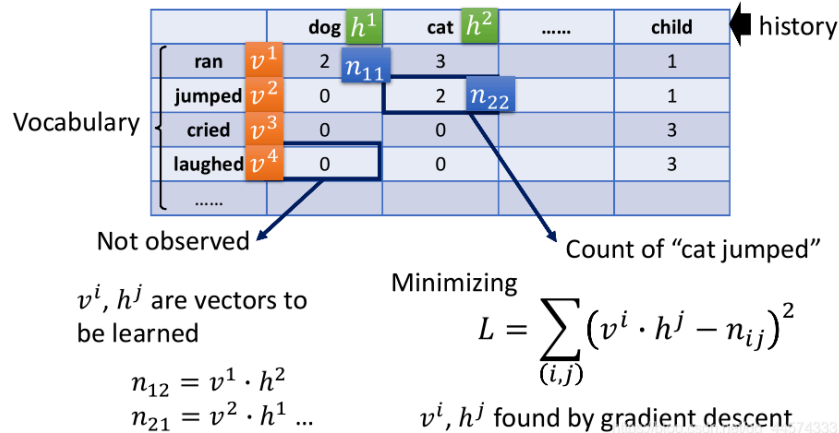
https://blog.csdn.net/qq_44574333

方法2: Continuous LM

Continuous LM引入了推荐系统中的**Matrix Factorization**方法，用到Language Model上来补全0概率的gram。

Borrowing the idea from recommendation system

Continuous LM



上表，横纵轴都代表了Vocabulary的token，表中的数字代表横轴后接纵轴的统计次数。

仿照推荐系统做一个同样的表格，估测出表中0的较小值应该填什么。我们就可以用Matrix Factorization技术来解这个问题：

- 1 首先，将每一个词汇都对应一个向量（词向量） h 和 v ，而这个词向量就是我们要训练的；
- 2 其次，我们将表格中非零数值 n 的部分当作 Y ，将 h 和 v 当作 x ；
- 3 最终，最小化 h 和 v 的点积与 n 的差为训练目标。

至此我们将得到所有词汇的词向量，而表中数值为0的部分将替换为词向量的点积，这样便解决了smoothing的问题。

神经网络语言模型

NN-based LM

Continuous LM 可以看作是只有一个hidden layer的简单的deep network，输入的是token的1-of-N encoding。

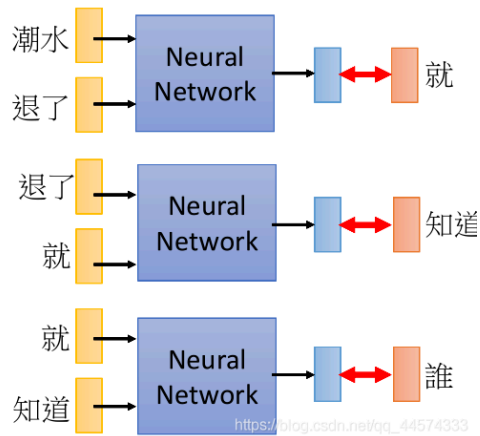
NN-based LM

• Training:

Collect data:

潮水 退了 就 知道 誰 ...
不爽 不要 買 ...
公道價 八萬 一 ...
.....

**Learn to predict
the next word**



因此，一般的NN-based的Language Model 类似，它最早是想要取代N-gram的LM，比如我们想训练一个NN-based LM去取代3-gram。

- 1 首先，收集一定的资料
- 2 然后，给模型输入两个词汇，让模型去预测下一个词汇

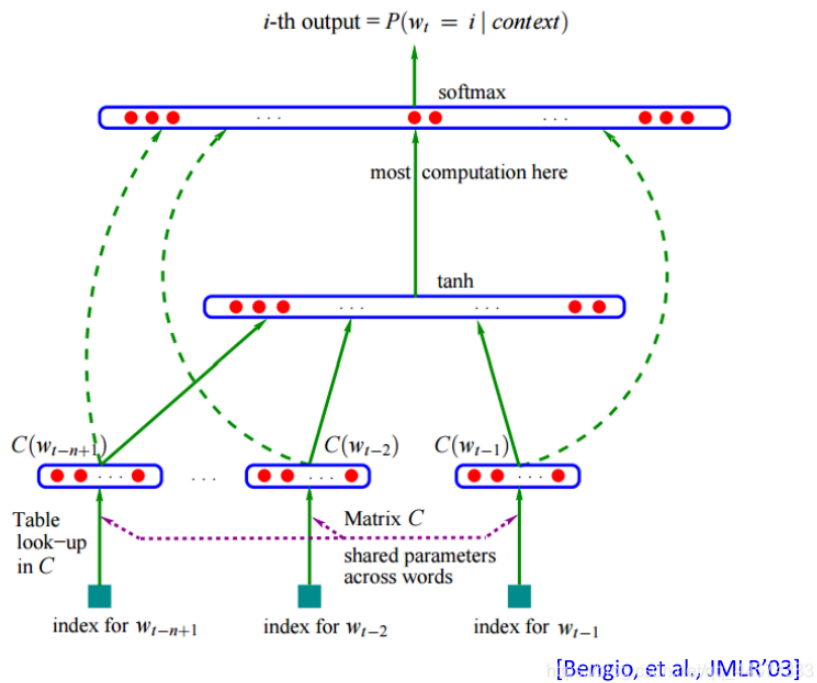
将

$$P(y_1, y_2, \dots, y_n) = P(y_1 | BOS)P(y_2 | y_1) \dots P(y_n | y_{n-1}) -$$

n-gram里面每一项的几率换成neural network的输出即可。

NN-based LM起源

可以找到的最早的nn-based LM的文献2003年Bengio发表的，用network来取代n-gram，自动做smoothing，跨时代的是，在这篇论文中，它里面提到了word embedding的概念(2013年发明)。



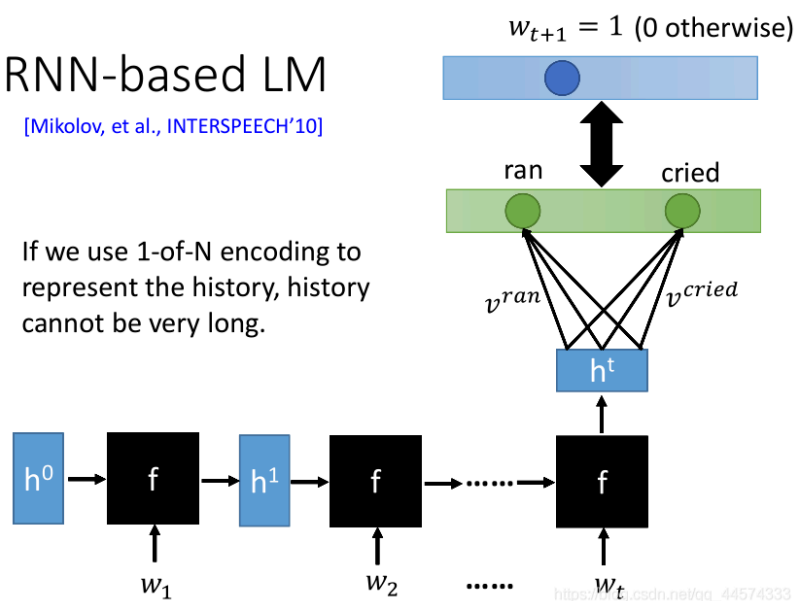
RNN-based LM

如果想看非常长的history决定下一个token出现的几率，如果只是一般的NN的话，输入将变得非常的长。因此，我们使用RNN-based的 Language Model，利用RNN将history读进去，最后一个hidden layer输出的隐藏层向量 h_t 与每一个token的向量 v 相乘得到 $t + 1$ 应该出现的token。

RNN-based LM

[Mikolov, et al., INTERSPEECH'10]

If we use 1-of-N encoding to represent the history, history cannot be very long.



参考文献

1. 语言模型

课程向：深度学习与人类语言处理 ——李宏毅，2020 (P9)