论文｜万物皆可Vector之语言模型：从N-Gram到NNLM、RNNLM

原创 Thinkgamer 搜索与推荐Wiki 2020-12-01

收录于话题
#论文笔记

19个

Word2vec的出现改变了OneHot的高维稀疏的困境，自此之后各种xxx2vec如雨后春笋般冒了出来，用来解决各种嵌入式编码，包括后来的各种Embedding方式其实很多本质上都是Word2vec的延伸和优化。在本公众号「搜索与推荐Wiki」上也发布了不少Embedding相关的文章，后续也会持续的发布相关文章，欢迎关注。

本主题文章将会分为三部分介绍，每部分的主题为：

- word2vec的前奏-统计语言模型
- word2vec详解-风华不减
- 其他xxx2vec论文和应用介绍

后续会更新Embedding相关的文章，可能会单独成系列，也可能会放到《特征工程-Embedding系列中》，欢迎持续关注「**搜索与推荐Wiki**」

语言模型

a)：定义

语言模型（Language model）是自然语言处理的重要技术，自然语言处理中最常见的是文本数据，我们可以把一段自然语言文本看作是一段离散的时间序列，假设一段长度为T的文本中的词依次是$w_1, w_2, \ldots, w_T$，语言模型就是计算他的概率：

$$P(w_1, w_2, \ldots, w_T)$$

也就是说语言模型是对语句的概率分布的建模。

语言模型可以分为：统计语言模型和神经网络语言模型。

b）：概率表示

假设 S 表示一个有意义的句子，eg：今天天气晴朗，适合户外爬山，可以将这个句子表示为：$S=w_1,w_2,\ldots,w_n$，换成例子中的句子：w_1 =今天, w_2 =天气, w_3 =晴朗, w_4 =适合, w_5 =户外, w_6 =爬山 。

用$P(S)$ 表示这个句子出现的概率，展开为：

$$P(S)=P(w_1,w_2,\ldots,w_n)$$

利用条件概率可以转化为：

$$P(S)=P(w_1,w_2,\ldots,w_n)=P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\ldots P(w_n|w_1,w_2,\ldots,w$$

其中$P(w_1)$ 表示第一个词出现的概率，即「今天」在整个语料库中出现的概率，$P(w_2|w_1)$ 表示在给定第一个词的情况下，第二个词出现的概率，即在整个语料库中给定「今天」这个词，「天气」这个词也出现的概率，后边的依次类推。

其中的$P(w_1)$和$P(w_2|w_1)$很容易计算得到，但是$P(w_3|w_1,w_2)$及以后涉及变量较多，计算的复杂度也会变得更加复杂。

统计语言模型——N-gram模型

a）：马尔可夫假设

为了解决上面的过于复杂难以计算的问题，需要引入**马尔可夫假设**，马尔科夫假设中很重要的一点是**有限视野假设**，即每一个状态只与它前面的$n-1$个状态有关，这被称为 n 阶马尔可夫链

b）：n-gram

当应用在语言模型中时，就是指每一个词的概率只与前边的 $n-1$ 个词有关系，这就被称为n 元语言模型，当$n=2$ 时，被称为二元模型，此时上述公式展开为：

$$P(S)=P(w_1,w_2,\ldots,w_n)=P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|n_1)$$

经过马尔可夫假设的简化，计算$P(S)$的概率也会变得容易很多，当然随着n的增加，相应的计算复杂度也会增加，而 n 越大，越逼近数据的真实分布，n 通常取值为2、

3、4、5。

c）：概率估计

通过上面的描述，可以明确的是：

- 每一个句子都可以拆分成不同的词的全排列
- 每一个句子都可以通过条件概率公式计算得到一个表示该句子的合理性概率
- 通过引入马尔可夫假设，简化句子的计算概率

以二元模型为例，如何计算 $P(w_i|w_{i-1})$？从概率统计中可知：

$$P(w_i|w_{i-1}) = \frac{P(w_i, w_{i-1})}{P(w_i)}$$

在大语料的情况下，基于大数定理，词语$的共同出现次数除以w_i的出现次数可以近似等于P(w_i|w_{i-1})$，所以有：

$$P(w_i|w_{i-1}) = \frac{P(w_i, w_{i-1})}{P(w_i)} = \frac{N(w_i, w_{i-1})}{N(w_i)}$$

所以一般情况下，统计语言模型都要求语料足够大，这样得到的结果相对会准确一些。但这里边也存在一个问题，如果 $N(w_i, w_{i-1}) = N(w_i) = 1$ 或者都等于0的话，计算出来得结果显然是不合理的。因此引入了平滑技术。

d）：n-gram模型中的平滑技术

平滑技术就是为了解决 c）中描述的次数统计比值不合理的情况。常见的平滑技术有（这里不展开描述，感兴趣的可以自行搜索了解）：

- 加法平滑
- 古德-图灵(Good-Turing)估计法
- Katz平滑方法
- Jelinek-Mercer平滑方法
- Witten-Bell平滑方法
- 绝对减值法
- Kneser-Ney平滑方法

e）n-gram 语言模型的优缺点：

优点：

- （1）采用极大似然估计，参数易训练

- （2）完全包含了前 n-1 个词的全部信息
- （3）可解释性强，直观易理解。

缺点：

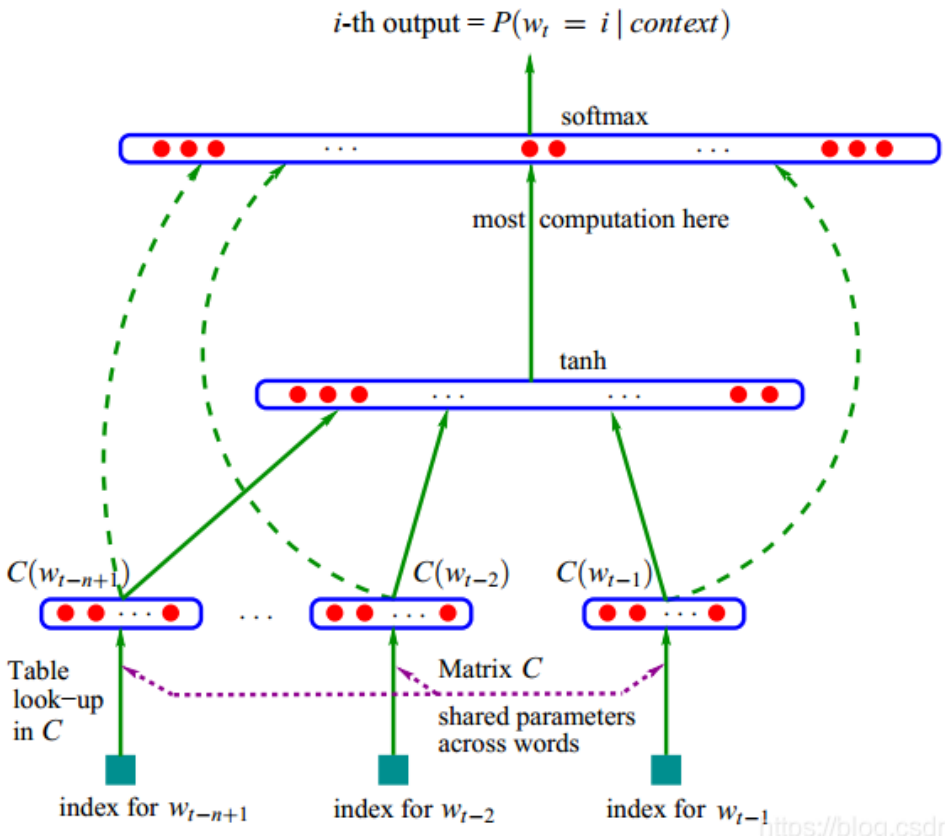
- （1）缺乏长期依赖，只能建模到前 n-1 个词
- （2）随着 n 的增大，参数空间呈指数增长
- （3）数据稀疏，难免会出现OOV的问题
- （4）单纯的基于统计频次，泛化能力差。

神经网络语言模型——NNLM

NNLM论文：《A Neural Probabilistic Language Model》

下载链接：http://www.ai.mit.edu/projects/jmlr/papers/volume3/bengio03a/bengio03a.pdf

NNLM模型的网络是一个三层的网络结构图，如下所示：



其中最下层为输出词前 $n-1$ 个词，NNLM模型的目标是急于这 $n-1$ 个词计算 第 t 个词 w_t 的概率。

$NNLM$的训练目标是：

$$f(w_t,\dots,w_{t-n+1}) = P(w_t|w_1^{t-1})$$

其中w_t 表示第 t 个单词，w_1^{t-1} 表示从第1个单词到第 $t-1$ 个词组成的子序列，模型需要满足的约束条件是：

- $f(w_t,\dots,w_{t-n+1}) > 0$
- $\sum_1^{|V|} f(w_t,\dots,w_{t-n+1}) = 1$

上面模型的意思是当给定一段序列时，由其前面的$t-1$个词预测第n个词的概率。

- 限制条件一：即是通过网络得到的每个概率值都要大于0。
- 而对于第二个限制条件：因为我们的神经网络模型最终得到的输出是针对每 $t-1$个词的输入来预测下一个，也即是第t个词是什么。因此模型的实际输出是一个向量，该向量的每一个分量依次对应下一个词为词典中某个词的概率。所以$|v|$维的概率值中必定有一个最大的概率，而其他的概率较小。

$NNLM$ 模型的训练目标可以分解为两部分：

- 1、特征映射：即将V 中的第i个单词映射成一个m维的向量$C(i)$，这里的映射可以是$OneHot$（可以是初始化的词向量，因为要同模型一起进行训练），然后将通过特征映射得到的$C(w_{t-n+1}),\dots,C(w_{t-1})$进行拼接合并成一个$m(n-1)$维的向量。也可以说是：一个从词汇表$V$到实数向量空间的映射$C$。通过这个映射得到每个单词的向量表示。
- 2、计算条件概率分布：通过一个函数 g 将输入的词向量序列 $(C(w_{t-n+1}),\dots,C(w_{t-1}))$ 转化为一个概率分布 $y \in R^{|V|}$，所以这里的输出是$|V|$维的，和词典的维度是相同的，y 中第i个词表示词序列中第n个词是V_i的概率，即：
  $f(i,w_{t-1},\dots,w_{t-n+1}) = g(i,C(w_{t-1}),\dots,C(w_{t-n+1}$

$NNLM$ 模型的输出为一个$softmax$ 函数，形式如下：

$$P(w_t|w_{t-1},\dots,w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_{w_t}}}$$

其中y_i 表示的是第i 个单词没有进行归一化的概率，其计算公式为：

$$y = b + Wx + U\,tanh(d + Hx)$$

其中模型参数为：$\theta = (b,d,W,U,H,C)$

- x 表示的神经网络的输入矩阵：$x = (C(w_{t-1}),\dots,C(w_{t-n+1}))$
- h 表示隐藏层神经元个数
- d 表示隐藏层的偏置参数
- m 表示每个单词对应的向量维度

- W 是可选参数，如果输入层和输出层之间没有直接相连，可令$W=0$
- H 表示的是输入层到隐藏层的权重矩阵
- U 表示的时隐藏层到输出层的权重矩阵

一般的神经网络不需要对输入进行训练，但是在该模型中的输入x是词向量，也是需要训练的参数，由此可见模型的权重参数和词向量是同时训练的，模型训练完成后同时得到网络的权重参数和词向量

$NNLM$ 的训练目标是最大化似然函数，即：

$$L=\frac{1}{T}\sum_{i}logf(w_t,w_{t-1},\ldots,w_{t-n+1},;\theta)+R(\theta)$$

其中θ为所有模型参数，$R(\theta)$为正则化项（在论文对应的实验中，R 表示的是权重衰减参数，仅适用于神经网络和单词对应的向量矩阵）。

然后使用梯度下降法更新参数：

$$\theta\leftarrow\theta+\epsilon\frac{\vartheta logp(w_t|w_{t-1},\ldots,w_{t-n+1})}{\vartheta\theta}$$

其中ϵ为学习率（步长）。

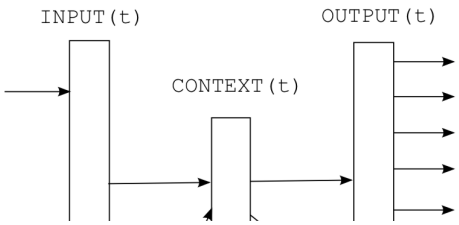
基于Pytorch和Tf实现的代码参考：
https://www.jianshu.com/p/be242ed3f314

神经网络语言模型——RNNLM

RNNLM 论文：
《Recurrent neural network based language model》

下载连接：
https://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

RNNLM模型的思想比较简单，主要改进的是NNLM中的前馈神经网络，其主要的结构图如下所示：



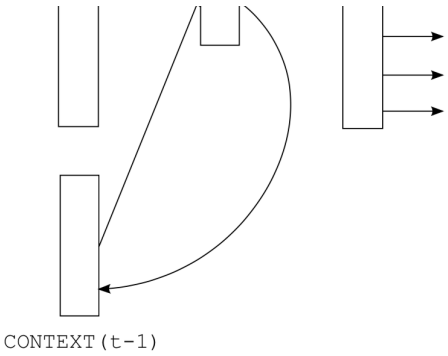
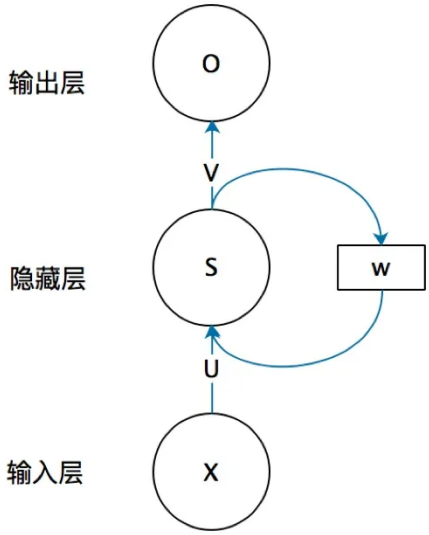


Figure 1: *Simple recurrent neural network.*

RNNLM

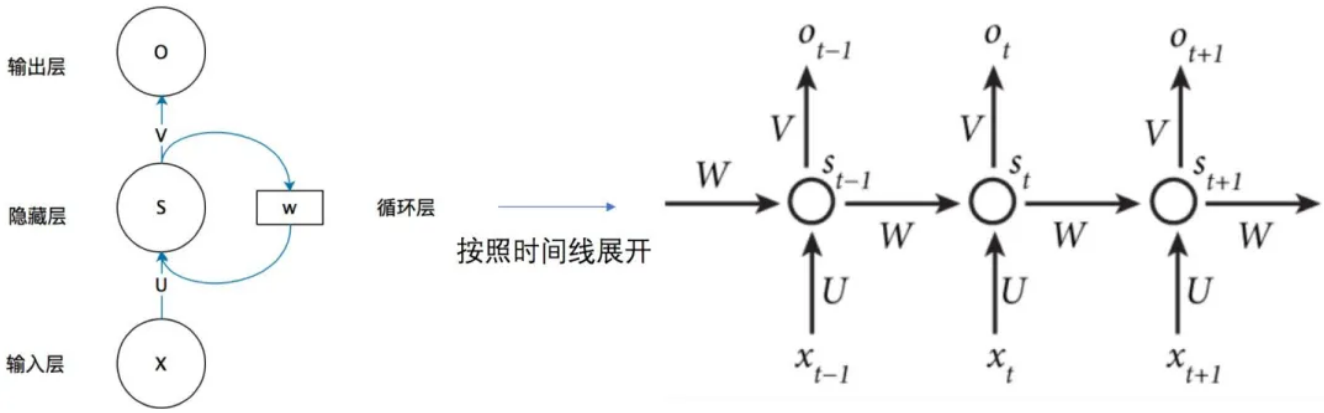
读者乍看可能不知道这个描述的是什么，不着急，先来补一下简单的RNN知识。

一个简答的RNN结构如下图所示：



其包含输入层、隐藏层和输出层，X 表示的是输入层的向量，U 表示输入层到隐藏层的权重矩阵，S 表示隐藏层的向量值，V 表示的是隐藏层到输出层的权重矩阵，O 表示的是输出层的向量值，w 表示隐藏层上一次的值。

将上面的图展开为：



RNN时间线展开图

现在看上去就比较清楚了，这个网络在t时刻接收到输入 x_t 之后，隐藏层的值是 s_t ，输出值是 o_t 。关键一点是，s_t 的值不仅仅取决于 x_t ，还取决于 s_{t-1}。

接着看RNNLM模型的结构图，$INPUT(t)$ 表示的就是 t时刻的输入x_t，$CONTEXT(t)$ 表示的就是 t时刻的隐藏层（s_t），$CONTEXT(t-1)$ 则表示$t-1$时刻的隐藏层的值（s_{t-1}），$OUTPUT(t)$ 表示的就是t时刻的输出（o_t）。

其中：

$$x(t)=w(t)+s(t-1)s_j(t)=f(\sum_i x_i(t)u_{ji})y_k(t)=g(\sum_i s_j(t)v_{kj})$$

其中$f(z)$ 表示的是$sigmoid$ 函数：

$$f(z)=\frac{1}{1+e^{-z}}$$

$g(z)$表示的是$softmax$函数：

$$g(z_m)=\frac{e^{z_m}}{\sum_k e^{z_k}}$$

其中有一些细节需要注意：

- $s(0)$ 通常设置为比较小的向量值，如0.1
- 隐藏层单元数一般为30-500，因为输入的词库特别大
- 权重使用带噪声的高斯函数进行初始化
- 优化算法使用的是随机梯度下降
- 学习率初始值设置为0.1，后续随着迭代的进行，如果没有明显改善，学习率修改为原先的一半
- 通常在10-20次迭代之后开始收敛

每一epoch结束之后，向量的误差基于交叉熵准则进行计算：

$$error(t)=desired(t)-y(t)$$

其中$desired(t)$ 表示预测值，$y(t)$为真实值。

最终单词出现概率计算公式为：

$$P(w_i(t+1)|wi,s(t-1))=\begin{cases}\frac{y_{rare}(t)}{C_{rare}} & if\ w_i(t+1) israre\\ y_i(t) & otherwise\end{cases}$$

其中C_{rare}表示的是词汇表中单词出现次数小于阈值的单词数。文中提到在$Browncorpus$数据集中（有80万单词），阈值设置的为5，隐藏层单元数为100。

关于RNNLM代码实现可以参考：https://www.jianshu.com/p/f53f606944c6

如果觉得文章不错，点个赞、在看，或者分享给更多人看到吧（戳【**阅读原文**】触达更多精彩内容）！

收录于话题 #论文笔记·19个

上一篇
论文｜万物皆可Vector之Word2vec：2个模型、2个优化及实战使用

下一篇
论文｜Airbnb Embedding的实践和思考

阅读原文

喜欢此内容的人还喜欢

独家｜利用Python实现主题建模和LDA 算法（附链接）
数据派THU