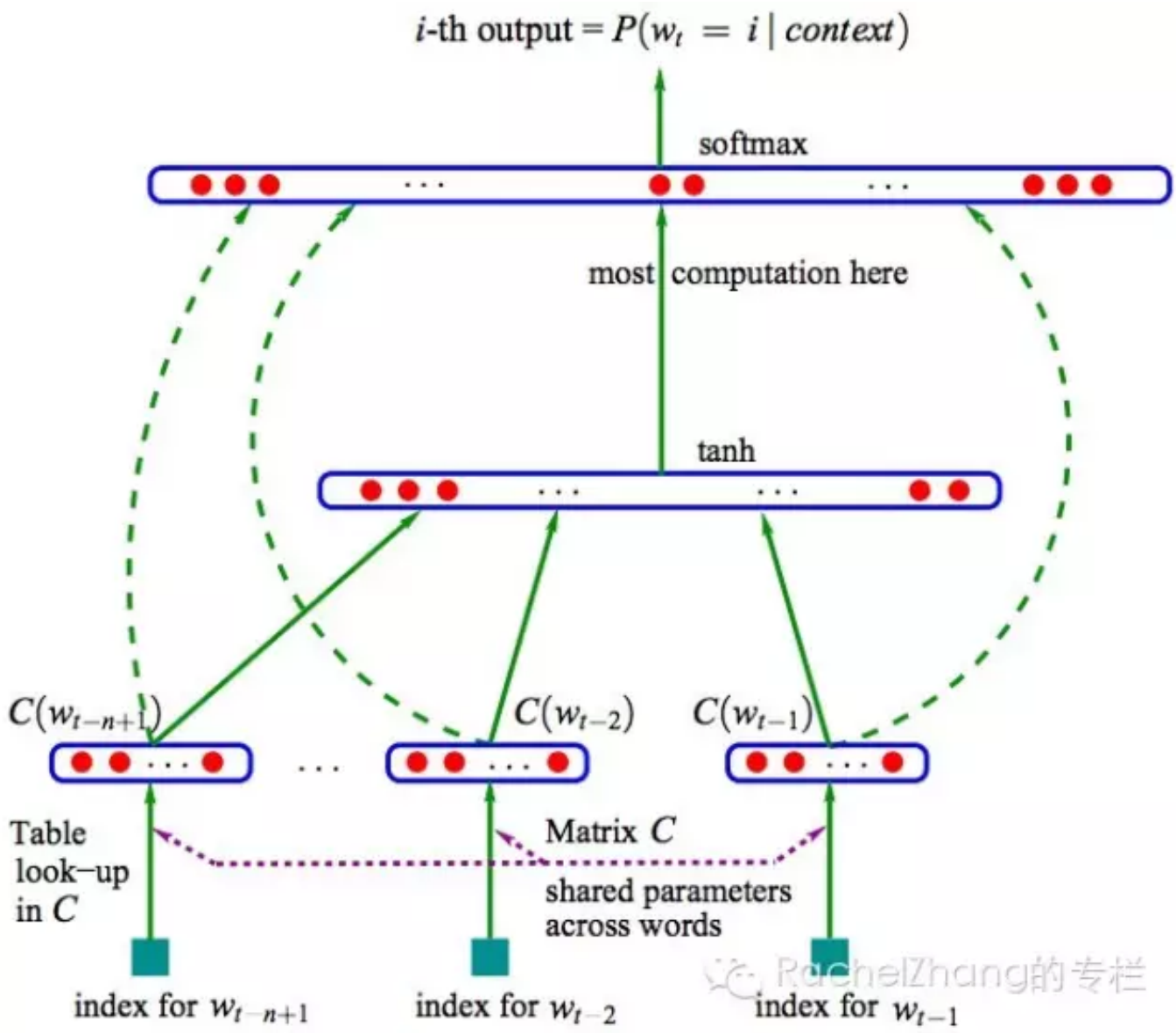
word2vec——高效word特征求取

Rachel RachelZhang的专栏 2015-06-07



继上次分享了经典统计语言模型，最近公众号中有很多做NLP朋友问到了关于word2vec的相关内容， 本文就在这里整理一下做以分享。 本文分为

- 概括word2vec
- 相关工作
- 模型结构
- Count-based方法 vs. Directly predict

几部分，暂时没有加实验章节，但其实感觉word2vec一文中实验还是做了很多工作的，希望大家有空最好还是看一下~

概括word2vec

要解决的问题： 在神经网络中学习将word映射成连续（高维）向量， 其实就是个词语特征求取。

特点：
1. 不同于之前的计算cooccurrence次数方法，减少计算量
2. 高效
3. 可以轻松将一个新句子/新词加入语料库
主要思想：神经网络语言模型可以用两步进行训练：1. 简单模型求取word vector; 在求取特征向量时，预测每个词周围的词作为cost 2. 在word vector之上搭建N-gram NNLM，以输出词语的概率为输出进行训练。

相关工作

在传统求取word的空间向量表征时， LSA 将词和文档映射到潜在语义空间，从而去除了原始向量空间中的一些“噪音”，但它无法保存词与词之间的linear regularities； LDA 是一个三层贝叶斯概率模型，包含词、主题和文档三层结构。文档到主题服从Dirichlet分布，主题到词服从多项式分布， 但是只要训练数据大了， 计算量就一下飚了。
基于神经网络的词语向量表征方法在[Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model, JMLR 2003]中就有提出， 名为NNLM， 它是一个前向网络， 同时学习词语表征和一个统计语言模型（后面具体讲）。
在Mikolov的硕士论文[1]和他在ICASSP 2009上发表的文章[2]中， 用一个单隐层网络训练词语表征， 然后将这个表征作为NNLM的输入进行训练。 Word2vec是训练词语表征工作的一个拓展。

模型结构

首先回顾NNLM，RNNLM，然后来看Word2Vec中提出的网络——CBOW，skip-gram Model。

1．NNLM[3]

NNLM的目标是在一个NN里，求第t个词的概率， 即

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

其中f是这个神经网络，包括 input，projection， hidden和output。将其分解为两个映射：C和g，C是word到word vector的特征映射(通过一个|V|*D的映射矩阵实现)，也称作look-up table， g是以word特征为输入，输出|V|个词语概率的映射：

$$f(i,w_{t-1},\cdots,w_{t-n+1}) = g(i,C(w_{t-1}),\cdots,C(w_{t-n+1}))$$

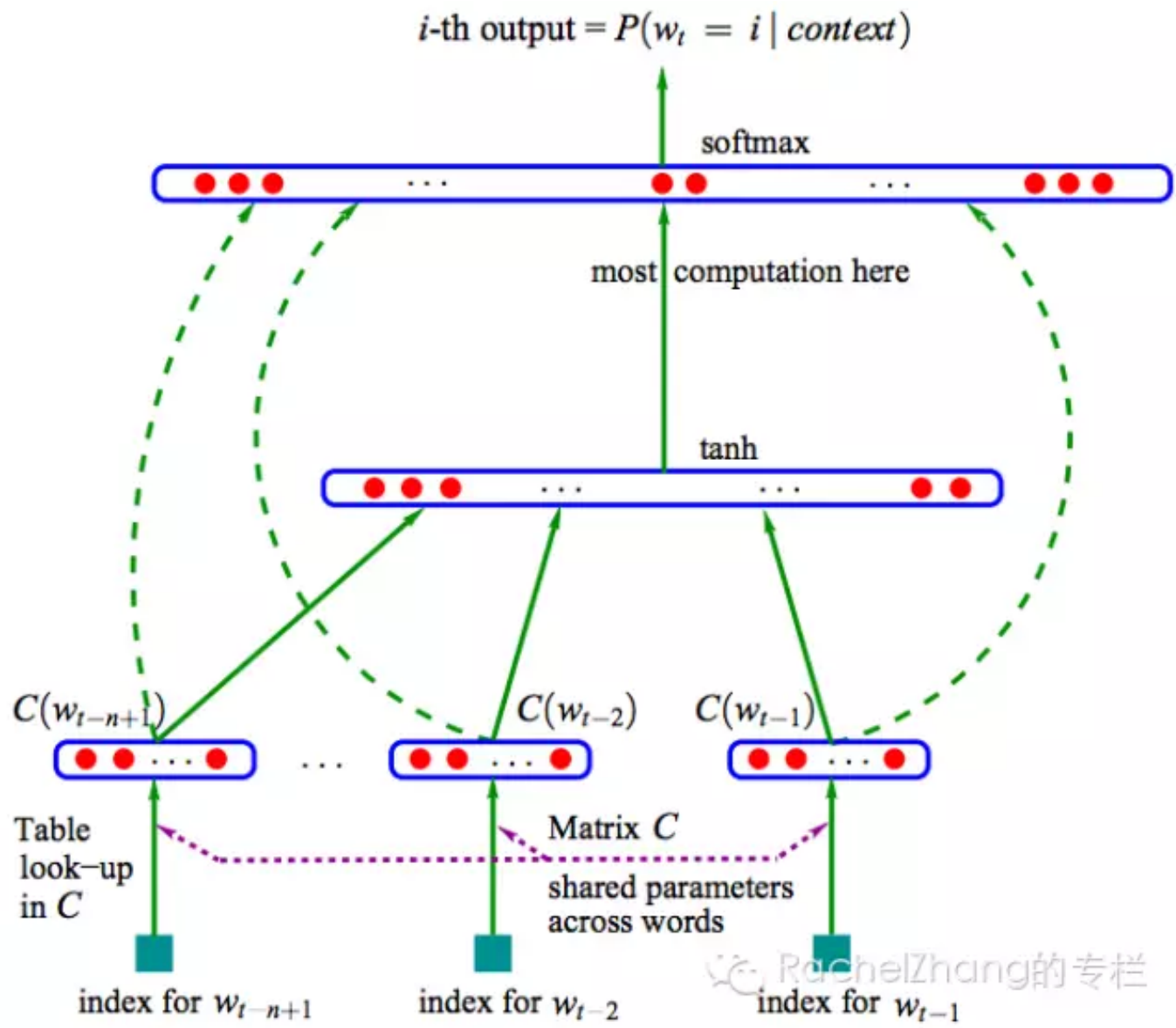
如下图所示：

输入： n个之前的word（其实是他们的在词库V中的index）

映射： 通过|V|*D的矩阵C映射到D维

隐层： 映射层连接大小为H的隐层

输出： 输出层大小为|V|，表示|V|个词语的概率



用parameter个数度量网络复杂度， 则这个网络的复杂度为：

$$O = N * D + N * D * H + H * V$$

其中复杂度最高的部分为H*V, 但通常可以通过hierarchical softmax或binary化词库编码

将|V|降至log2V， 这样计算瓶颈就在于隐层N*D*H了。在word2vec中，为了避免隐层带来的高计算复杂度而去掉了隐层。

2 . RNNLM

RNN在语言模型上优于其他神经网络，因为不用像上面NNLM中的输入要定死前N个词的N。（具体RNN的结构我会在下篇中讲）简单地说， RNN就是一个隐层自我相连的网络，隐层同时接收来自t时刻输入和t-1时刻的输出作为输入， 这使得RNN具有短期记忆能力， 所以RNNLM的复杂度为：

$$O = H * H + H * V$$

同样地，其中$H * V$也可以降至$log2\,V$， 瓶颈就在于$H * H$了。

由于复杂度最大的部分都在hidden layer, 而且我们的中级目标是提特征（而不是生成语言模型），文中就想能不能牺牲hidden layer的非线性部分， 从而高效训练。 这也是Word2vec中速度提升最多的部分。 这也就是一个Log linear model。所以本质上， word2vec并不是一个深度模型。文中提出了两种log linear model，如下面所述。

3 . Continuous Bag-of-Words(CBOW) Model

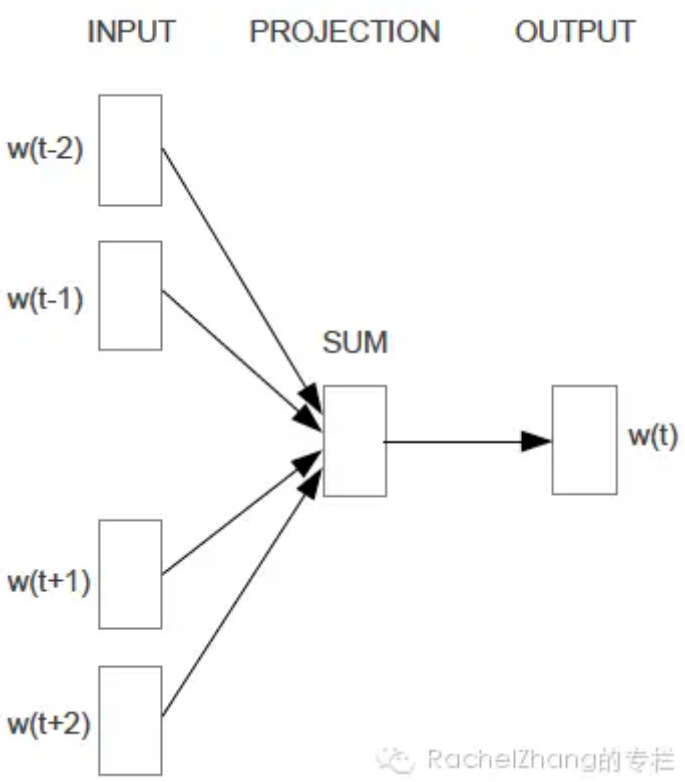
CBOW的网络结构和NNLM类似，变化：

1. CBOW去掉了NNLM的非线性部分
2. CBOW不考虑word之间的先后顺序， 一起放进bag，也就是在上面NNLM的projection层将映射后的结果求和/求平均（而非按照先后顺序连接起来）
3. 输入不止用了历史词语，还用了未来词语。 即， 用t-n+1...t-1,t+1,...t+n-1的word作为输入，目标是正确分类得到第t个word。
   PS: 实验中得到的best n=4

CBOW的复杂度为：

$$O = N * D + D * \log_2 V$$

CBOW结构图：

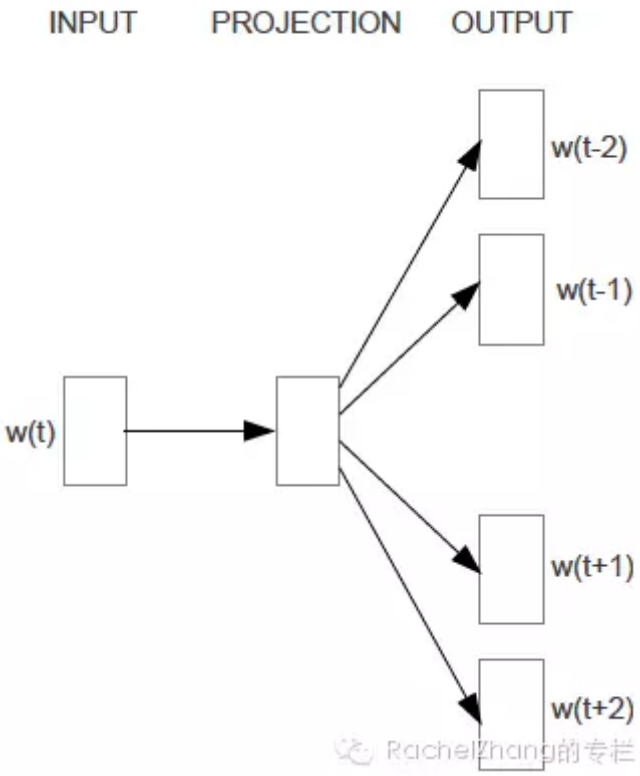


3 . Continuous Skip-gram Model

与CBOW相反，Continuous Skip-gram Model不利用上下文。 其输入为当前word，经过projection的特征提取去预测该word周围的c个词，其cost function为：

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

如下图所示。这里c增大有利于模型的完备性， 但过大的c可能造成很多无关词语相关联， 因此用随机采样方法，远的词少采， 近的多采。



比如定义最大周围距离为C，则对于每个词w(t)，就选择距离为R=range(1,C)， 选前后各R个词作为预测结果。
所以，Continuous Skip-gram Model的复杂度为：

$$O = 2C * (D + D * \log_2 V)$$

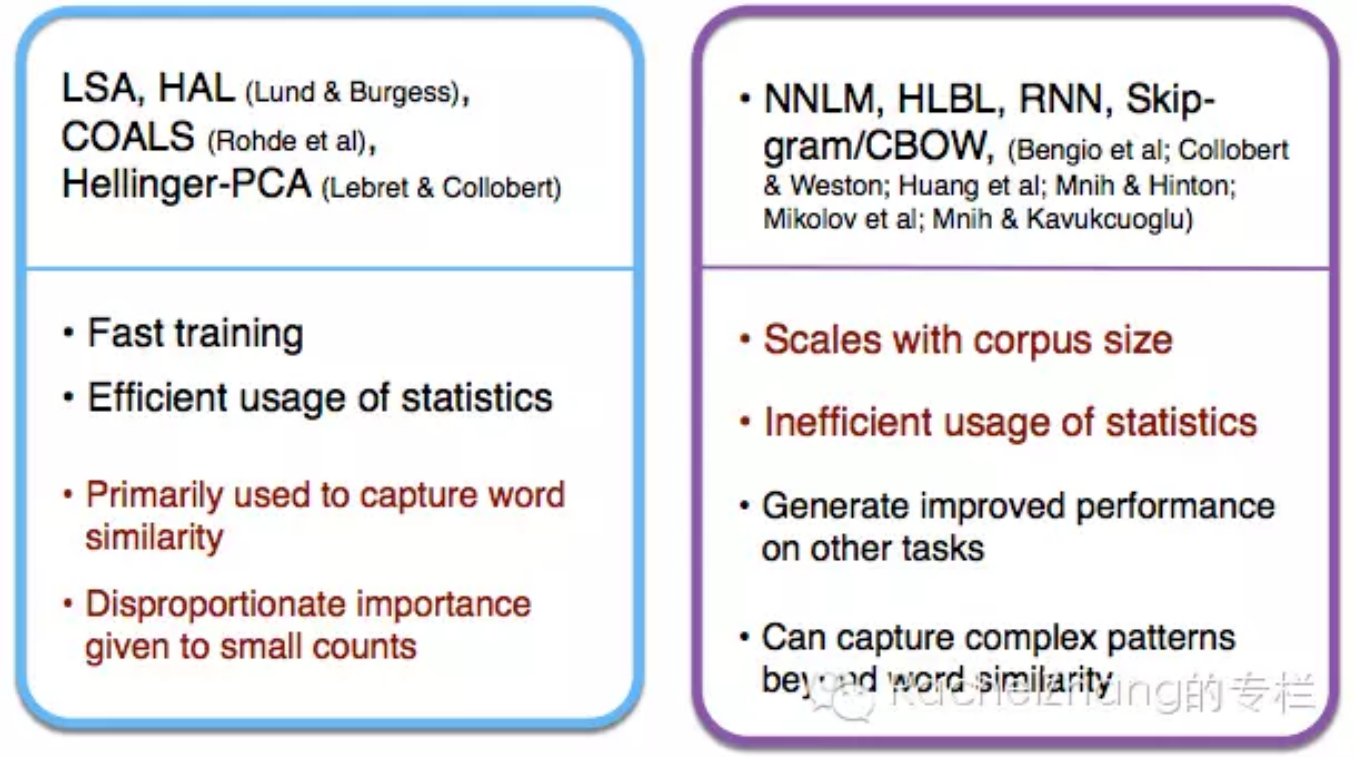
具体来说，最简单的情况下， $P(w_{t+j}|w_t)$ 的表达式可以为：

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

其中v和v'分别为输入和输出中的word特征向量。所以说， word2vec方法本质上是一个动态的逻辑回归。

Count-based方法 vs. Directly predict

最后我们看一下之前我们讲过的几个基于统计的传统语言模型与word2vec这种直接预测的方法的比较：



图片摘自Stanford CS244。

参考文献：

1. NNLM: Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model, JMLR 2003
2. 类似工作：T. Mikolov. Language Modeling for Speech Recognition in Czech, Masters thesis
3. 类似工作：T. Mikolov, J. Kopecky´, L. Burget, O. Glembek and J. Cˇ ernocky´. Neural network based language models for higly inflective languages, In: Proc. ICASSP 2009.]
4. 类似工作：Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation[J]. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 2014, 12.

阅读原文