Word2vec

1. 词向量基础

One-hot-representation

distributed representation

2. CBOW (continuous Bag of words,) skip-gram

用于神经网络语言模型

采用三层神经网络结构(可以多层)

- 输入
- 隐藏
- softmax 输出

CBOW：输入：词的上下文相关词的词向量

输出：特定词的词向量

词袋模型：输入词平等的，未考虑位置关系

输入8个词向量，输出所有词：softmax概率

期望目标词概率最大

skipgram：输入：特定词词向量

输出：softmax概率排前8的8个词

词表过大时有问题

3. word2vec Huffman树

用霍夫曼树来代替隐藏层和输出层神经元

霍夫曼树保证高权重节点路径较短，编码值较短，反之较长，常用词编码更短

word2vec中约定左子树编码为1，右子树为0，同时约定左子树权重不小于右子树

4. 基于Hierarchical Softmax的模型概述

① 输入层到隐藏层：对所有输入词向量求和取平均，得到一个词向量

② H→Output：采用霍夫曼树

沿着Huffman树走来路径到叶子根点，叶子结点就是词

③ 采用二元逻辑回归：

沿左：负类(编码1)：

沿右：正类(编码0)

判断正负：

$$p(+1)=\sigma(x_w^T\theta)=\frac{1}{1+e^{-x_w\theta}}$$

x_w是当前内部节点词向量 θ为模型参数

*优点：1)计算量 $V\to \log_2 V$

2)高频词靠近根，更快被找到，会优化

5. 基于Hierarchical Softmax的模型梯度计算

① 能化描述：

输入词w，相当点词向量x_w，相到叶子路径节点总数l_w，w经过i节点表示为p_i^w，对应Huffman编码$d_i^w\in\{0,1\}$ $i=2,3,\cdots l_w$

节点参数θ_i^w $i=1,2,\cdots l_{w-1}$

w经过Huffman树某个节点j的逻辑回归概率

$$P(d_j^w\mid x_w,\theta_{j-1}^w)=\begin{cases}\sigma(x_w^T\theta_{j-1}^w) & d_j^w=0\\ 1-\sigma(x_w^T\theta_{j-1}^w) & d_j^w=1\end{cases}$$

对某个目标输出词w，最大似然

$$\prod_{j=2}^{l_w}P(d_j^w\mid x_w,\theta_{j-1}^w)=\prod_{j=2}^{l_w}[\sigma(x_w^T\theta_{j-1}^w)]^{1-d_j^w}[1-\sigma(x_w\theta_{j-1}^w)]^{d_j^w}$$

每次只使用一个样本更新梯度(随机梯度上升)

$$L=\log\prod_{j=2}^{l_w}P(d_j^w\mid x_w,\theta_{j-1}^w)=\sum_{j=2}^{l_w}\left((1-d_j^w)\log\sigma(x_w^T\theta_{j-1}^w)+d_j^w\log[1-\sigma(x_w^T\theta_{j-1}^w)]\right)$$

($\sigma'(x)=\sigma(x)(1-\sigma(x))$)

$$\frac{\partial L}{\partial\theta_{j-1}^w}=(1-d_j^w)\frac{\sigma(x_w^T\theta_{j-1}^w)(1-\sigma(x_w^T\theta_{j-1}^w))}{\sigma(x_w^T\theta_{j-1}^w)}x_w^T-d_j^w\frac{\sigma(x_w^T\theta_{j-1}^w)(1-\sigma(x_w^T\theta_{j-1}^w))}{1-\sigma(x_w^T\theta_{j-1}^w)}x_w$$

$$=\left[1-d_j^w-\sigma(x_w^T\theta_{j-1}^w)\right]x_w$$

同理

$$\frac{\partial L}{\partial x_w}=\sum_{j=2}^{l_w}(1-d_j^w-\sigma(x_w^T\theta_{j-1}^w))\theta_{j-1}^w$$

② 基于该方法CBOW

1) 先建立Huffman树

2) 随机初始θ，x_w

3) a) 求$x_w=\frac{1}{2c}\sum_{i=1}^{2c}x_i$

b) for j=2 to l_w 计算

$f=\sigma(x_w^T\theta_{j-1}^w)$

$g=(1-d_j^w-f)\eta$

$e=e+g\theta_{j-1}^w$

$\theta_{j-1}^w=\theta_{j-1}^w+gx_w$

$x_i=x_i+e$

5、基于负采样的skip-gram模型

我们期望 $p(x_i|x_w)$ $i=1,2,\cdots 2c$ 最大，上下文相互的，在期望 $p(x_i|x_w)$ 最大时，$p(x_w|x_i)$ 也最大。

选择后者，在一个迭代窗口内，不只更新1个词，而2c个词，整体迭代更加均衡。

所以 skip-gram 不对输入更新，而是2c个输出

a) 建立 Huffman 树

b) 随机初始化

c) 1) for i=1 to 2c

ii) e=0

对1个词 iii) for j=2 to l_w

解释：

$$f=\sigma(x_w^T\theta_{j-1}^w)$$
$$g=(1-d_j^w-f)\eta$$
$$e=e+g\theta_{j-1}^w$$
$$\theta_{j-1}^w=\theta_{j-1}^w+gx_i$$

iii) $x_i=x_i+e$

缺点：虽然提高了效率，但对于生僻词，还要很多次迭代。

6、基于负采样的模型

① 中心词是w，context(w)有2c个，w是一个正例，负采样得到neg个和w不同中心词，利用一个正neg个负二元逻辑回归。

② 如何通过一个正例多个负例二元逻辑回归

正：$P(context(w),w_i)=\sigma(x_w^T\theta^{(w_i)})$ $y_i=1, i=0$

负：$P(context(w_0),w_i)=1-\sigma(x_w^T\theta^{(w_i)})$ $y_i=0, i=1,\cdots neg$

$$\prod_{i=0}^{neg}P(context(w_0),w_i)=\sigma(x_{w_0}^T\theta^{(w_0)})\prod_{i=1}^{neg}(1-\sigma(x_{w_0}^T\theta^{(w_i)}))$$

似然函数：

$$\prod_{i=0}^{neg}\sigma(x_{w_0}^T\theta^{(w_i)})^{y_i}(1-\sigma(x_{w_0}^T\theta^{(w_i)}))^{1-y_i}$$

对数似然：

$$L=\sum_{i=0}^{neg}y_i\log(\sigma(x_{w_0}^T\theta^{(w_i)}))+(1-y_i)\log(1-\sigma(x_{w_0}^T\theta^{(w_i)}))$$

$$\frac{\partial L}{\partial\theta^{w_i}}=(y_i-\sigma(x_{w_0}^T\theta^{(w_i)}))x_{w_0}\quad\frac{\partial L}{\partial x^{w_0}}=\sum_{i=0}^{neg}(y_i-\sigma(x_{w_0}^T\theta^{(w_i)}))\theta^{w_i}$$

③ 如何进行负采样？

$$len(w)=\frac{count(w)^{\frac{3}{4}}}{\sum count(w)^{\frac{3}{4}}}$$

将长度为1的线段等分为 $M>>V$

从M个位置中采样 neg 个位置

每个位置所在的词就是采样的词（负例）

$M(default)=10^8$