

word2vec

Tylerwang

2015.4.30

Word2vec是什么

- **word2vec**是一个将单词转换成向量形式的工具。通过转换，可以把对文本内容的处理简化为向量空间中的向量运算，计算出向量空间上的相似度，来表示文本语义上的相似度。

词向量

- 把自然语言中的一个词表示成一个向量

- One-hot Representation

例如：

- “话筒”表示为 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ...]
 - “麦克”表示为 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ...]

- 存在两个问题

- 维数灾难
 - 词汇鸿沟：任意两个词之间都是孤立的，不能体现词和词之间的关系

词向量

- **Distributional Representation**

- 将词表示为:

- 通过训练将每个词映射成**K**维实数向量，通过词之间的距离（比如**cosine**相似度、欧氏距离等）来判断它们之间的语义相似度。
 - 具体表示为如[0.792, -0.177, -0.107, 0.109, 0.542, ...]，常见维度几十到几百

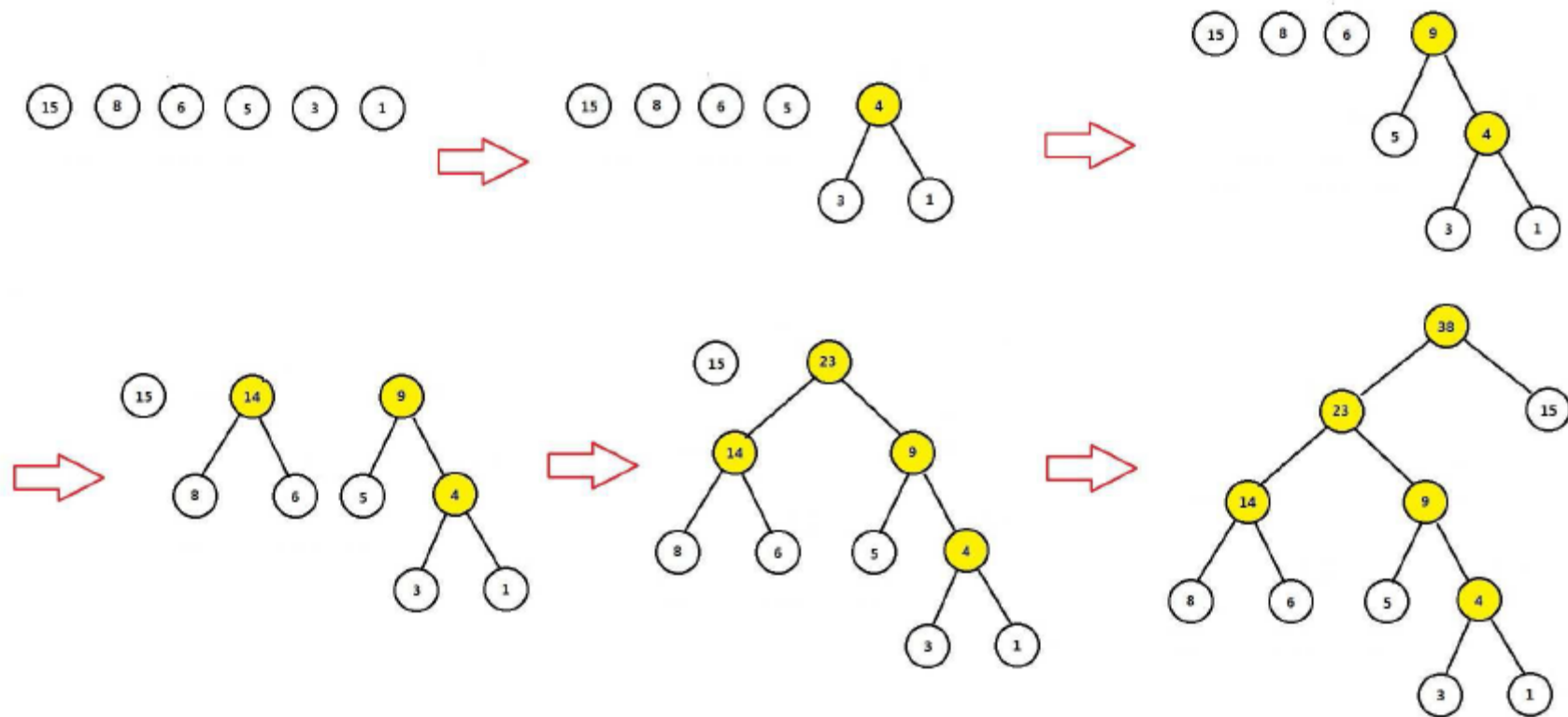
- 可解决“词汇鸿沟”问题

- 可以通过计算向量之间的距离（欧式距离、余弦距离等）来体现词与词的相似性
- 又叫**Word Representation**”或 **Word Embedding**

霍夫曼树

给定 n 个权值作为 n 个叶子结点, 构造一棵二叉树, 若它的带权路径长度达到最小, 则称这样的二叉树为**最优二叉树**, 也称为 **Huffman 树**.

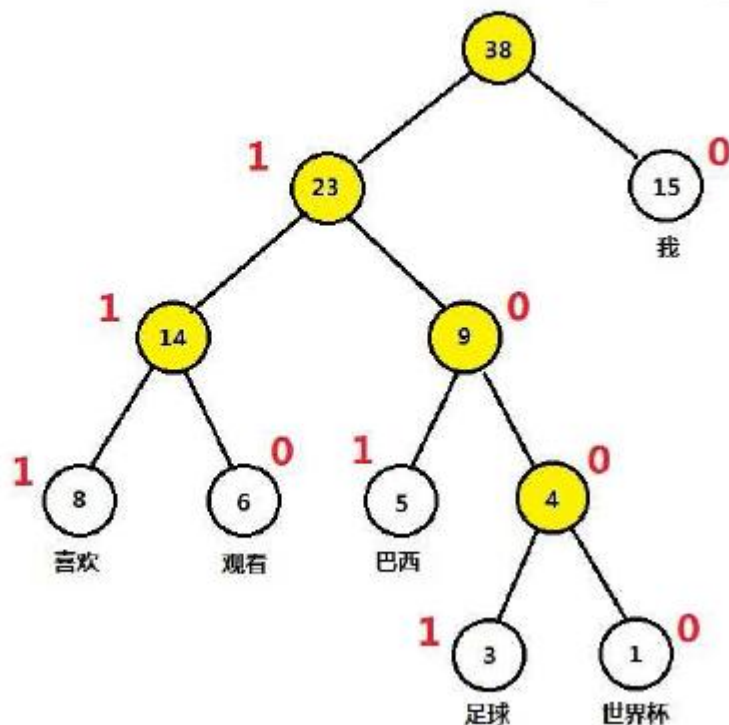
“我”、“喜欢”、“观看”、“巴西”、“足球”、“世界杯”这六个词出现的次数分别为 15, 8, 6, 5, 3, 1. 请以这 6 个词为叶子结点, 以相应词频当权值, 构造一棵 *Huffman* 树.



霍夫曼编码

利用 Huffman 树设计的二进制前缀编码, 称为 Huffman 编码

这六个词的 Huffman 编码分别为 0, 111, 110, 101, 1001 和 1000.



词频越高, 编码的长度越短, 从根节点到相应词的路径越短

语言模型

- 判断一句话是不是正常人说出来的，用数学符号描述为

- 给定一个字符串 " w_1, w_2, \dots, w_t ", 计算它是自然语言的概率 $p(w_1, w_2, \dots, w_t)$ ，一个很简单的推论是

$$p(w_1, w_2, \dots, w_t) = p(w_1) \cdot p(w_2 | w_1) \cdot p(w_3 | w_1, w_2) \cdot \dots \cdot p(w_t | w_1, w_2, \dots, w_{t-1})$$

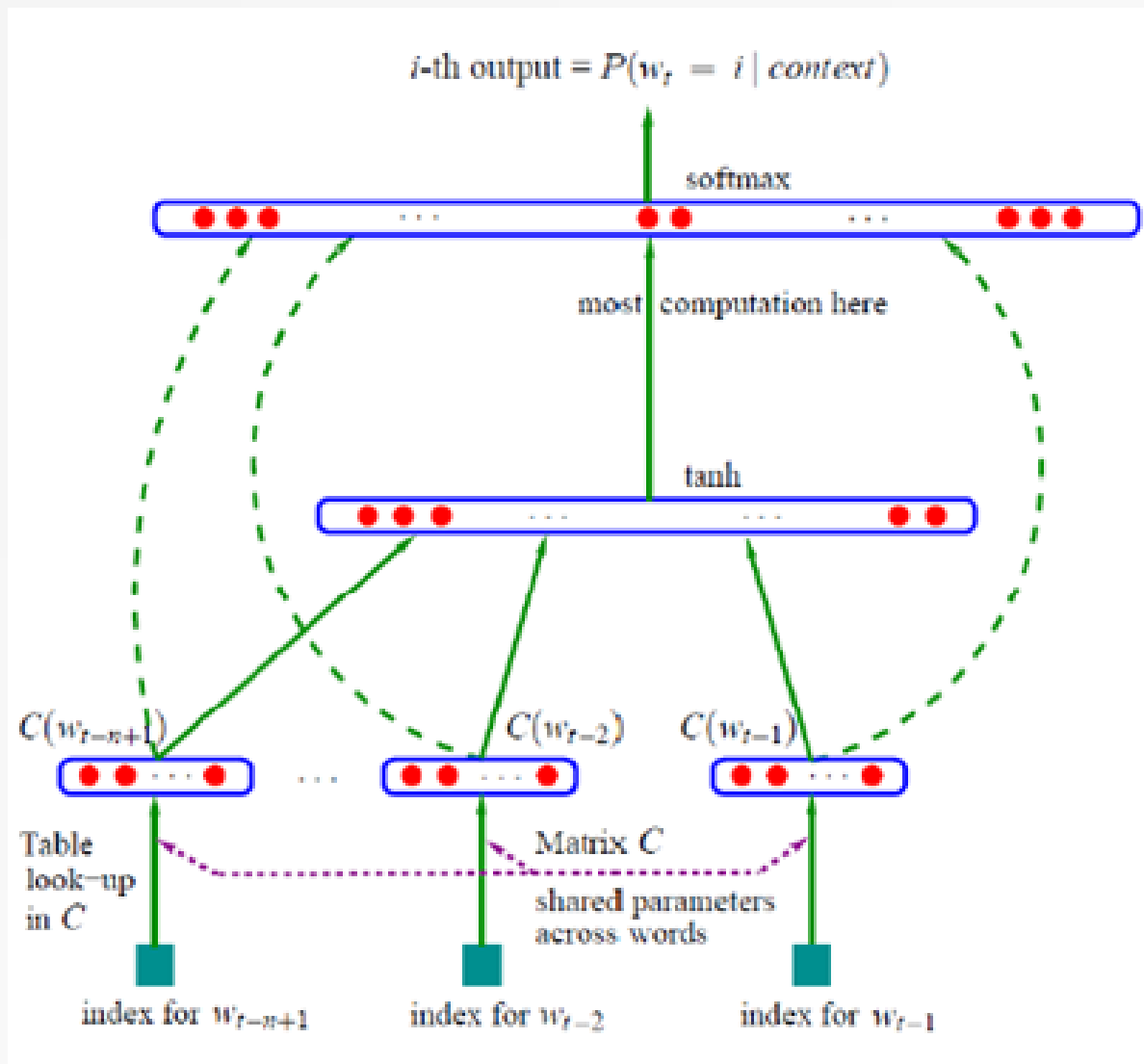
- 例如，有个句子 "大家, 喜欢, 吃, 苹果"

- $P(\text{大家, 喜欢, 吃, 苹果}) = p(\text{大家})p(\text{喜欢}|\text{大家})p(\text{吃}|\text{大家, 喜欢})p(\text{苹果}|\text{大家, 喜欢, 吃})$

- 简单表示为 $p(s) = p(w_1, w_2, \dots, w_T) = \prod_{i=1}^T p(w_i | \text{Context}_i)$

- 计算 $p(w_i | \text{Context}_i)$ 问题

神经网络语言模型



神经网络语言模型

- 输入是向量，也会被更新，每次查询C矩阵
- 论文中输入向量间用catenation, word2vec用sum
- 输入层和输出层之间构建一层传递关系，可更快传导输入word的变化
- 可以选择双曲正切或者sigmoid

神经网络语言模型

- N: 词语window大小，当前词前后多少个词
- D: 词向量维度
- H: hidden layer节点数
- V: 词汇表vocabulary大小
- 复杂度: $N * D + N * D * H + N * D * V + H * V$
- 第三第四项复杂度最高，word2vec去掉第三项，优化第四项(用hs或者neg)

word2vec原理

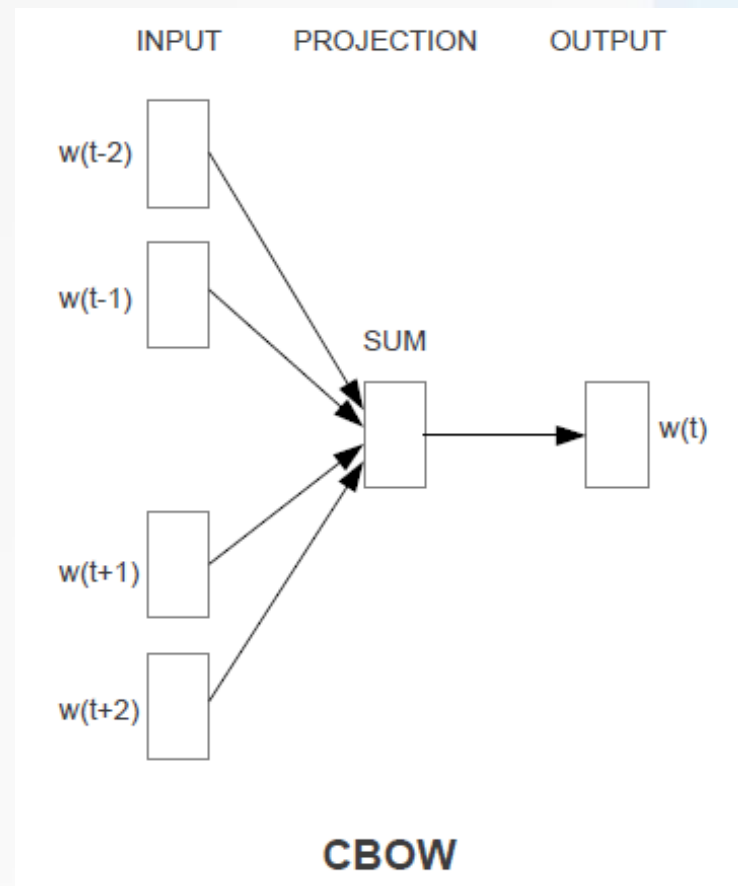
- 两种模型，两种加速策略

模型	CBOW		Skip-Gram	
方法	Hierarchical Softmax	Negative Sampling	Hierarchical Softmax	Negative Sampling

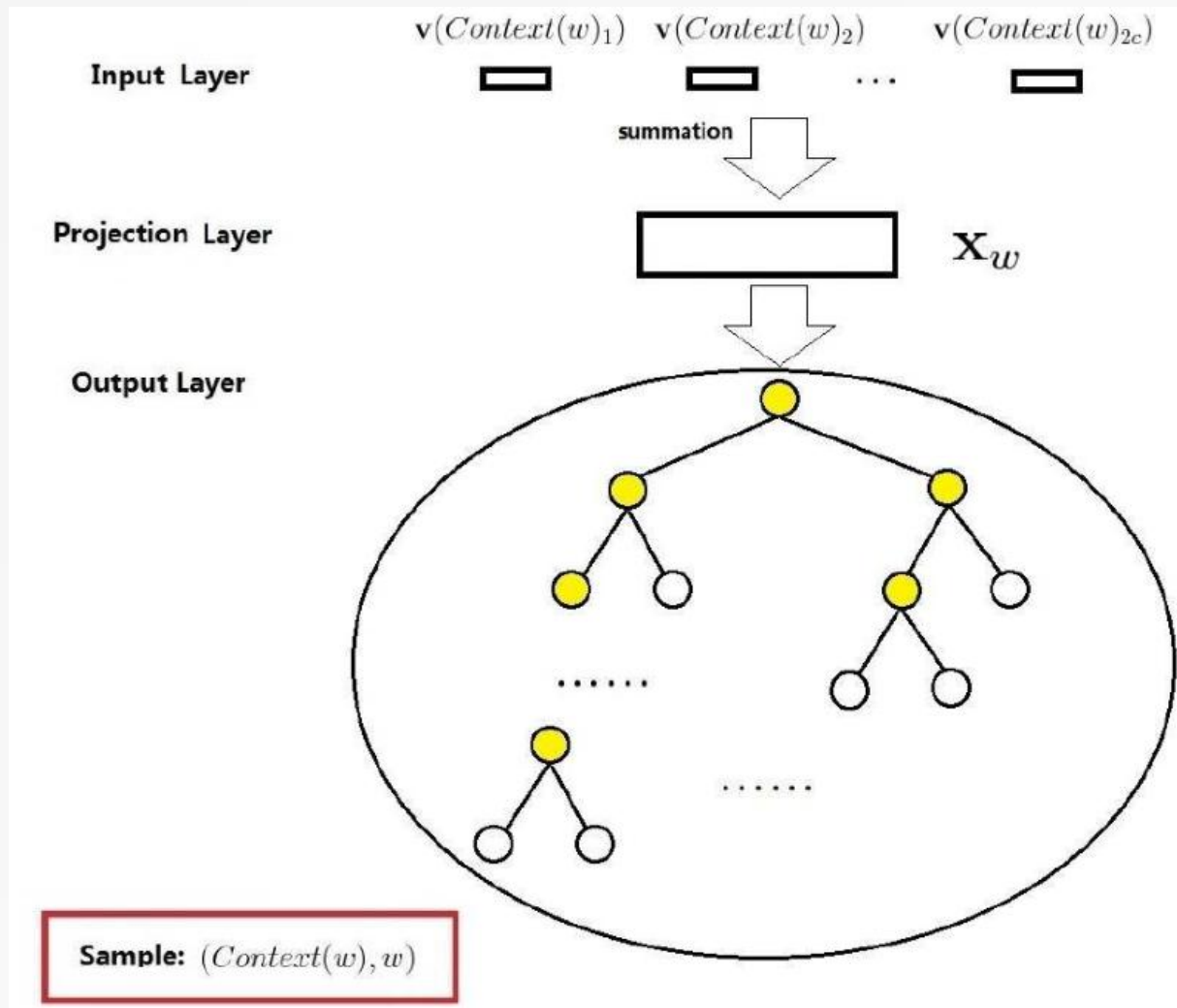
CBOW模型+Hierarchical Softmax方法

- CBOW模型
 - INPUT:输入层
 - PROJECTION:投影层
 - OUTPUT:输出层
 - $w(t)$:当前词语（向量）
 - $w(t-2), w(t-1), w(t+1), w(t+2)$:当前词语的上下文
 - SUM:context的累加和

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log p(\text{Context}(w)|w).$$



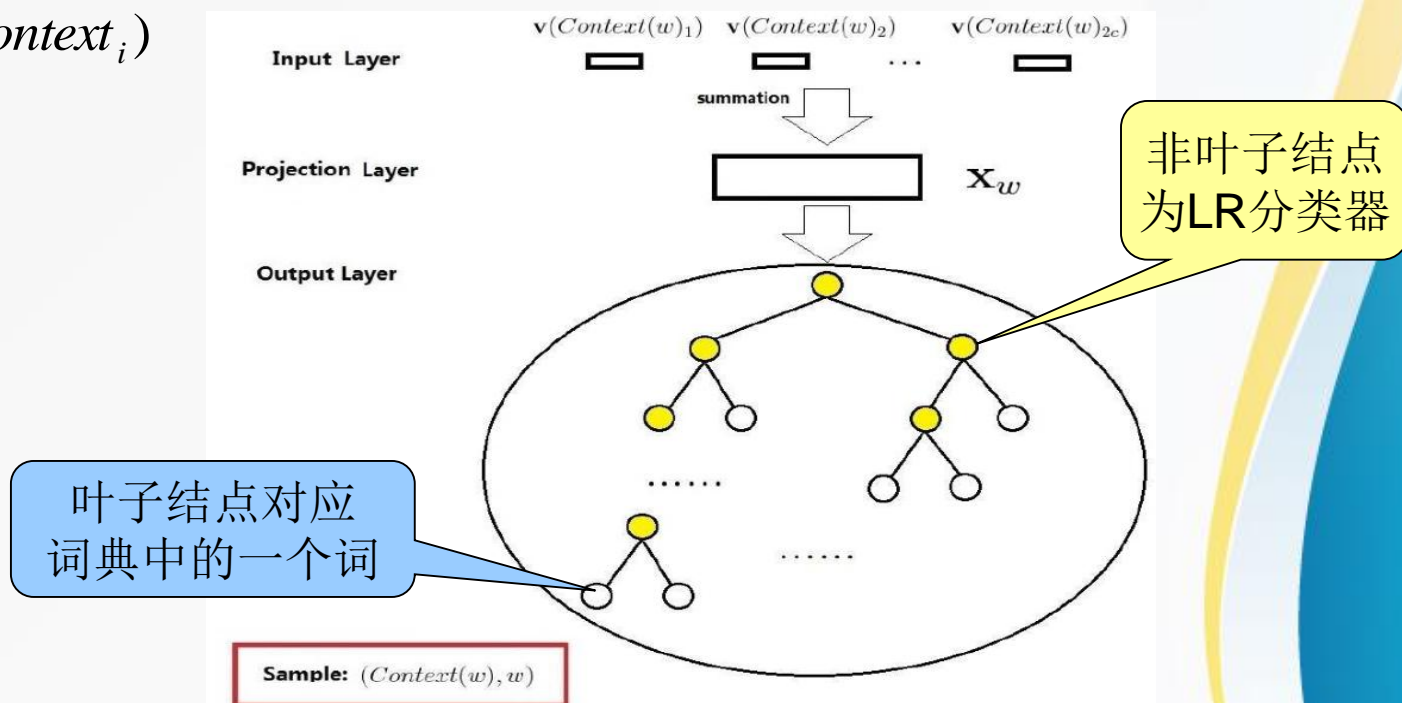
CBOW模型+Hierarchical Softmax方法（续）



CBOW模型+Hierarchical Softmax方法（续）

1. 输入层: 包含 $Context(w)$ 中 $2c$ 个词的词向量 $\mathbf{v}(Context(w)_1), \mathbf{v}(Context(w)_2), \dots, \mathbf{v}(Context(w)_{2c}) \in \mathbb{R}^m$. 这里, m 的含义同上表示词向量的长度.
2. 投影层: 将输入层的 $2c$ 个向量做求和累加, 即 $\mathbf{x}_w = \sum_{i=1}^{2c} \mathbf{v}(Context(w)_i) \in \mathbb{R}^m$.
3. 输出层: 输出层对应一棵二叉树, 它是以语料中出现过的词当叶子结点, 以各词在语料中出现的次数当权值构造出来的 Huffman 树. 在这棵 Huffman 树中, 叶子结点共 $N (= |\mathcal{D}|)$ 个, 分别对应词典 \mathcal{D} 中的词, 非叶子结点 $N - 1$ 个 (图中标成黄色的那些结点).

目标: $p(w_i | Context_i)$



CBOW模型+Hierarchical Softmax方法（续）

- 句子：我,喜欢,观看,巴西,足球,世界杯
- w =足球

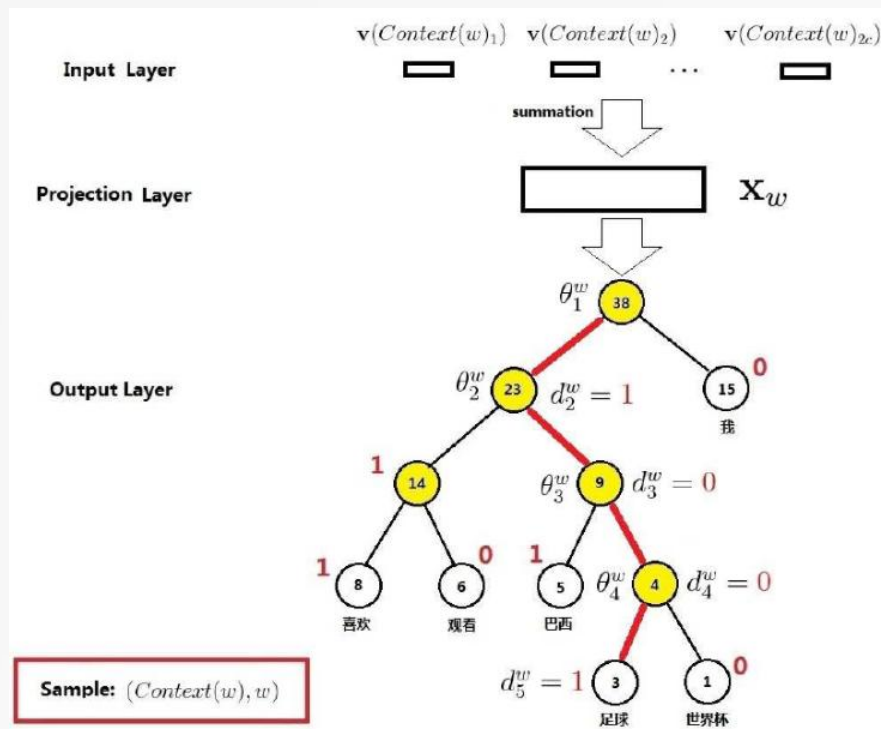
d_j^w : 编码 (1或0)

二分类

d_j^w : 正负类 (1 : 负类, 0 : 正类)

θ_j^w : 非叶子节点向量

θ_j^w : 类别向量



CBOW模型+Hierarchical Softmax方法（续）

- 正类概率: $\sigma(\mathbf{x}_w^\top \theta) = \frac{1}{1 + e^{-\mathbf{x}_w^\top \theta}},$
- 负类概率: $1 - \sigma(\mathbf{x}_w^\top \theta),$
- "足球" 叶子节点经过4次二分
- 每次分类结果对应的概率:

第1次: $p(d_2^w | \mathbf{x}_w, \theta_1^w) = 1 - \sigma(\mathbf{x}_w^\top \theta_1^w);$

第2次: $p(d_3^w | \mathbf{x}_w, \theta_2^w) = \sigma(\mathbf{x}_w^\top \theta_2^w);$

第3次: $p(d_4^w | \mathbf{x}_w, \theta_3^w) = \sigma(\mathbf{x}_w^\top \theta_3^w);$

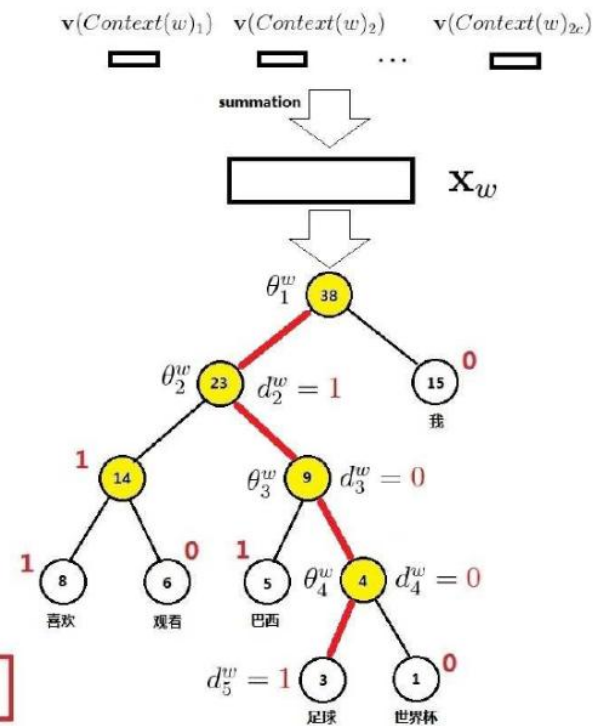
第4次: $p(d_5^w | \mathbf{x}_w, \theta_4^w) = 1 - \sigma(\mathbf{x}_w^\top \theta_4^w),$

Input Layer

Projection Layer

Output Layer

Sample: $(Context(w), w)$



- 由 $Context("足球")$ 预测"足球"出现的概率

$$p(\text{足球} | Context(\text{足球})) = \prod_{j=2}^5 p(d_j^w | \mathbf{x}_w, \theta_{j-1}^w).$$

CBOW模型+Hierarchical Softmax方法（续）

- 对于词典中的每个词 w 有, l^w 结点个数

$$p(w | Context(w)) = \prod_{j=2}^{l^w} p(d_j^w | X_w, \theta_{j-1}^w)$$

- 其中, $p(d_j^w | X_w, \theta_{j-1}^w) = \begin{cases} \sigma(X_x^T \cdot \theta_{j-1}^w), d_j^w = 0; \\ 1 - \sigma(X_x^T \cdot \theta_{j-1}^w), d_j^w = 1. \end{cases}$

- 或者表示为 $p(d_j^w | X_w, \theta_{j-1}^w) = [\sigma(X_w^T \cdot \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(X_w^T \cdot \theta_{j-1}^w)]^{d_j^w}$

- 对于由 S 个句子组成的语料库 C 有

$$L(X, \theta) = \prod_{s \in C} \prod_{w \in s} p(w | Context(w)) = \prod_{s \in C} \prod_{w \in s} \prod_{j=2}^{l^w} p(d_j^w | X_w, \theta_{j-1}^w)$$

- 取对数似然函数

$$\begin{aligned} \log L(X, \theta) &= \sum_{s \in C} \sum_{w \in s} \sum_{j=2}^{l^w} \log p(d_j^w | X_w, \theta_{j-1}^w) \\ &= \sum_{s \in C} \sum_{w \in s} \sum_{j=2}^{l^w} [(1 - d_j^w) \cdot \log \sigma(X_w^T \cdot \theta_{j-1}^w) + d_j^w \cdot \log(1 - \sigma(X_w^T \cdot \theta_{j-1}^w))] \end{aligned}$$

参数1

参数2

CBOW模型+Hierarchical Softmax方法（续）

- 梯度下降法进行求解

- 令 $f(w, j) = -(1 - d_j^w) \cdot \log \sigma(X_w^T \cdot \theta_{j-1}^w) - d_j^w \cdot \log(1 - \sigma(X_w^T \cdot \theta_{j-1}^w))$

- $f(w, j)$ 关于 θ_{j-1}^w 和 X_w 的梯度分别为

$$\frac{\partial f(w, j)}{\partial \theta_{j-1}^w} = -[1 - d_j^w - \sigma(X_w^T \cdot \theta_{j-1}^w)] \cdot X_w$$

$$\frac{\partial f(w, j)}{\partial X_w} = -[1 - d_j^w - \sigma(X_w^T \cdot \theta_{j-1}^w)] \cdot \theta_{j-1}^w$$

- 更新公式

$$\theta_{j-1}^w := \theta_{j-1}^w - \eta \cdot \frac{\partial f(w, j)}{\partial \theta_{j-1}^w} \quad V(\tilde{w}) := V(\tilde{w}) - \eta \cdot \sum_{j=2}^{l^w} \frac{\partial f(w, j)}{\partial X_w}, \tilde{w} \in \text{Context}(w)$$

- X_w 是 sum 得来的，有人提出应均分梯度到 context，即

$$\mathbf{v}(\tilde{w}) := \mathbf{v}(\tilde{w}) + \frac{\eta}{|\text{Context}(w)|} \sum_{j=2}^{l^w} \frac{\partial \mathcal{L}(w, j)}{\partial \mathbf{x}_w}, \tilde{w} \in \text{Context}(w)$$

为什么建huffman树

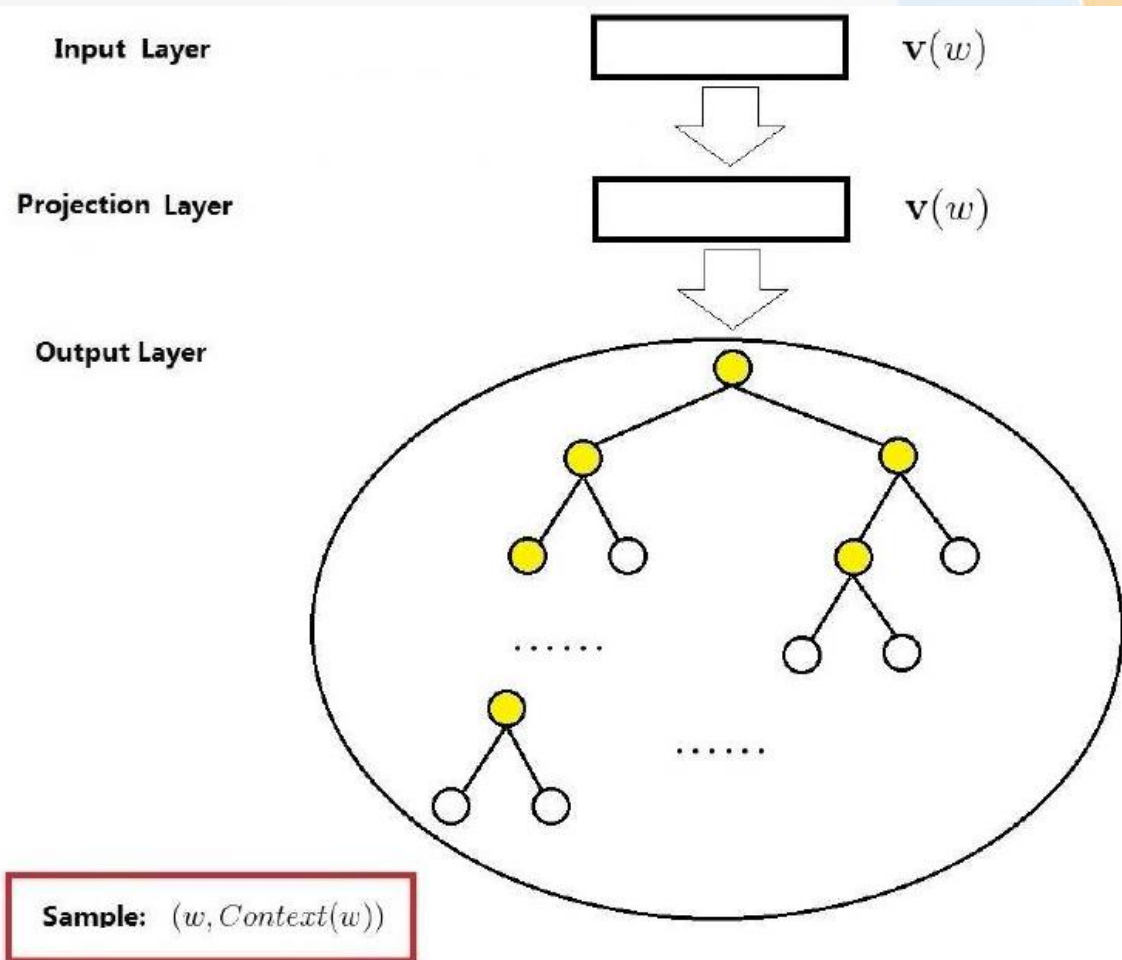
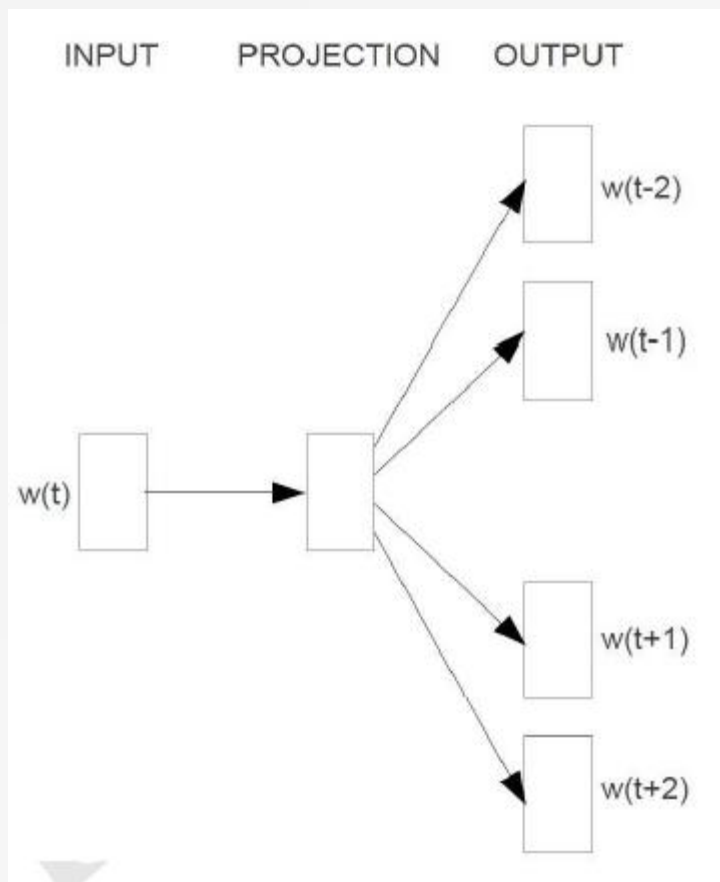
- 遇到词频高的词语时更快到达叶子节点，提高训练速度
- 复杂度从 $H*V$ 变为 $H*\log(V)$
- 输出即概率，不需归一化，神经网络语言模型的输出需要softmax归一化

经过上述两步计算得到的 $\mathbf{y}_w = (y_{w,1}, y_{w,2}, \dots, y_{w,N})^\top$ 只是一个长度为 N 的向量，其分量不能表示概率。如果想要 \mathbf{y}_w 的分量 $y_{w,i}$ 表示当上下文为 $Context(w)$ 时下一个词恰为词典 \mathcal{D} 中第 i 个词的概率，则还需要做一个 softmax 归一化，归一化后， $p(w|Context(w))$ 就可以表示为

$$p(w|Context(w)) = \frac{e^{y_{w,i_w}}}{\sum_{i=1}^N e^{y_{w,i}}}, \quad (3.6)$$

其中 i_w 表示词 w 在词典 \mathcal{D} 中的索引。

Skip-gram + HS



$$p(Context(w)|w) = \prod_{u \in Context(w)} p(u|w)$$

U和w可调换位置

$$p(u|w) = \prod_{j=2}^{l^u} p(d_j^u | v(w), \theta_{j-1}^u)$$

$$p(d_j^u | v(w), \theta_{j-1}^u) = [\sigma(v(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(v(w)^\top \theta_{j-1}^u)]^{d_j^u}$$

Negative sampling + CBOW

- 无严格理论证明，目的：提高训练速度并且提高训练所得词向量的质量
- 做法：按词频随机负采样

在 CBOW 模型中，已知词 w 的上下文 $Context(w)$ ，需要预测 w ，因此，对于给定的 $Context(w)$ ，词 w 就是一个正样本，其它词就是负样本了

假定现在已经选好了一个关于 w 的负样本子集 $NEG(w) \neq \emptyset$ 。且对 $\forall \tilde{w} \in \mathcal{D}$ ，定义

$$L^w(\tilde{w}) = \begin{cases} 1, & \tilde{w} = w; \\ 0, & \tilde{w} \neq w, \end{cases}$$

表示词 \tilde{w} 的标签，即正样本的标签为 1，负样本的标签为 0。

对于一个给定的正样本 $(Context(w), w)$ ，我们希望最大化

$$g(w) = \prod_{u \in \{w\} \cup NEG(w)} p(u|Context(w)),$$

Negative sampling + CBOW

$$p(u|Context(w)) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 1; \\ 1 - \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 0, \end{cases}$$

$$p(u|Context(w)) = [\sigma(\mathbf{x}_w^\top \theta^u)]^{L^w(u)} \cdot [1 - \sigma(\mathbf{x}_w^\top \theta^u)]^{1-L^w(u)}$$

$$g(w) = \sigma(\mathbf{x}_w^\top \theta^w) \prod_{u \in NEG(w)} [1 - \sigma(\mathbf{x}_w^\top \theta^u)]$$

$\sigma(\mathbf{x}_w^\top \theta^w)$, 同时最小化所有的 $\sigma(\mathbf{x}_w^\top \theta^u)$, $u \in NEG(w)$.
的概率同时降低负样本的概率.

最大化 $g(w)$, 相当于最大化

增大正样本

Negative sampling + skip-gram

$$\begin{aligned}\mathcal{L} &= \log G = \log \prod_{w \in \mathcal{C}} \prod_{u \in \text{Context}(w)} g(u) = \sum_{w \in \mathcal{C}} \sum_{u \in \text{Context}(w)} \log g(u) \\&= \sum_{w \in \mathcal{C}} \sum_{u \in \text{Context}(w)} \log \prod_{z \in \{u\} \cup \text{NEG}(u)} p(z|w) \\&= \sum_{w \in \mathcal{C}} \sum_{u \in \text{Context}(w)} \sum_{z \in \{u\} \cup \text{NEG}(u)} \log p(z|w) \\&= \sum_{w \in \mathcal{C}} \sum_{u \in \text{Context}(w)} \sum_{z \in \{u\} \cup \text{NEG}(u)} \log \left\{ [\sigma(\mathbf{v}(w)^\top \theta^z)]^{L^u(z)} \cdot [1 - \sigma(\mathbf{v}(w)^\top \theta^z)]^{1-L^u(z)} \right\} \\&= \sum_{w \in \mathcal{C}} \sum_{u \in \text{Context}(w)} \sum_{z \in \{u\} \cup \text{NEG}(u)} \left\{ L^u(z) \cdot \log [\sigma(\mathbf{v}(w)^\top \theta^z)] + [1 - L^u(z)] \cdot \log [1 - \sigma(\mathbf{v}(w)^\top \theta^z)] \right\}.\end{aligned}$$

word2vec实战（一）

- 训练数据集：经过分词后的新闻数据，大小184MB
 - 查看"中国", "钓鱼岛", "旅游", "苹果"几个词语的相似词语如下所示

请输入词语<exit退出>:中国

大陆	0.66763467
中共	0.57856727
共产党	0.56305367
解放军	0.55761635
台湾	0.5368497
反攻	0.5271177
日本	0.5103535
王文莹	0.49295437
内地	0.48557448
对岸	0.48428434

请输入词语<exit退出>:钓鱼岛

钓鱼台	0.6219264
钓岛	0.6123347
南海	0.6018163
领土	0.51753837
领海	0.4928774
岛屿	0.4853142
舰队	0.47854927
渔权	0.47229362
主权	0.46729872
东海	0.4613399

请输入词语<exit退出>:旅游

观光	0.65619475
景点	0.60212
陆客	0.59477097
旅行	0.5677106
游憩	0.557839
赏樱	0.5571045
游玩	0.52199984
观光客	0.51974636
行程	0.51943743
参观	0.5077874

请输入词语<exit退出>:苹果

三星	0.7224437
微软	0.7101249
Apple	0.66682446
iPhone5	0.62071097
Google	0.597368
iPadmini	0.5609188
新机	0.559093
库克	0.5589176
宏达电	0.555409
产品	0.55437565

word2vec实战（二）

- 向量加减法

- "中国+北京-日本", "中国+北京-法国", "家庭+孩子-学校"

请输入词语<exit退出>:中国,北京,日本

中国 + 北京 - 日本 =

东京 0.6188478

外海 0.4843038

广东 0.45942163

安倍 0.44709134

参拜 0.44158116

派出 0.43384194

釜山 0.4310615

海域 0.42759195

湖北 0.4272585

官邸 0.42487407

请输入词语<exit退出>:中国,北京,法国

中国 + 北京 - 法国 =

巴黎 0.7990697

伦敦 0.7614885

义大利 0.7026581

纽约 0.6534368

德国 0.6426668

首都 0.6354907

英国 0.6339937

西班牙 0.6326488

柏林 0.628184

加拿大 0.6255576

请输入词语<exit退出>:家庭,孩子,学校

家庭 + 孩子 - 学校 =

老师 0.948024

国中 0.76681674

学生 0.73207045

高中 0.7140167

小孩 0.70768154

小朋友 0.692751

上课 0.6889297

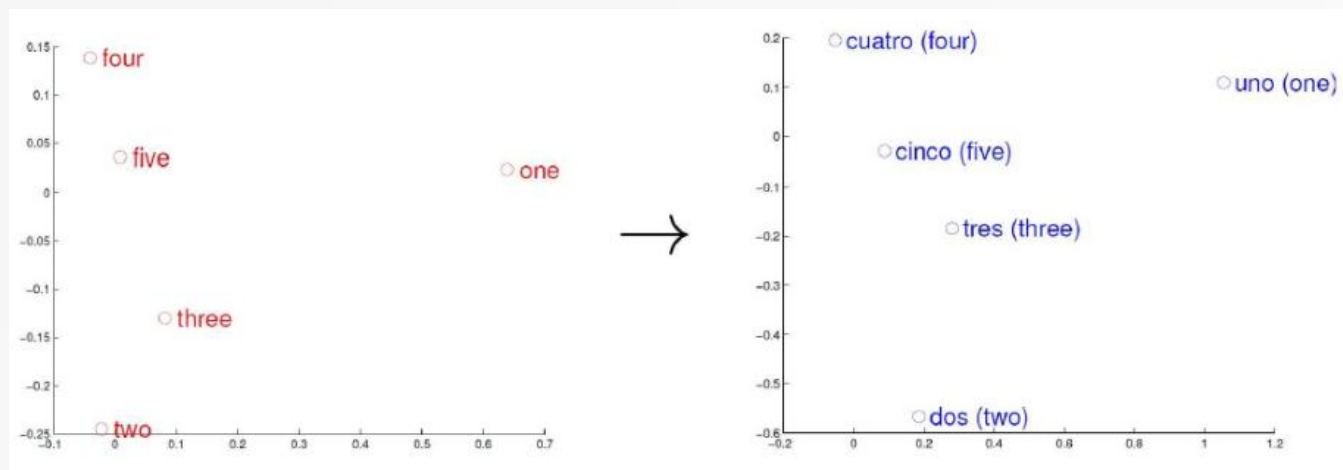
校方 0.687891

家长 0.66356415

高职 0.65407264

word2vec应用（三）

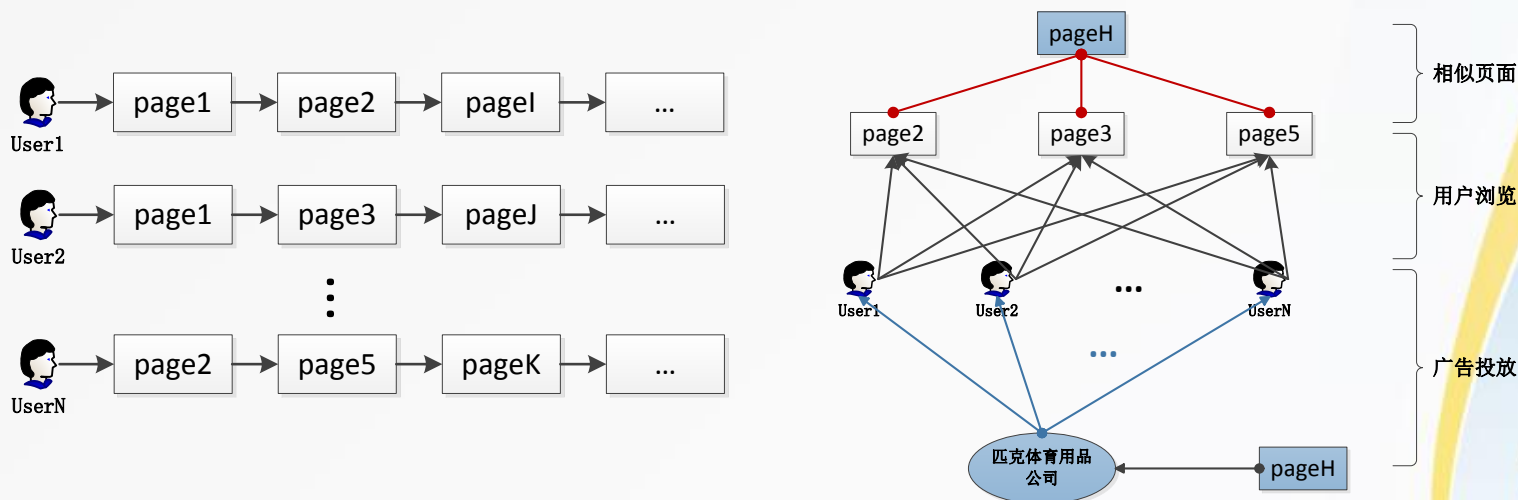
- 机器翻译
 - 语言词语的关系集合被表征为向量集合
 - 向量空间内，不同语言享有许多共性
 - 实现一个向量空间到另一个向量空间的映射和转换



- 图为英语和西班牙语的五个词在向量空间中的位置（已经过降维）
- 对英语和西班牙语之间的翻译准确率高达90%

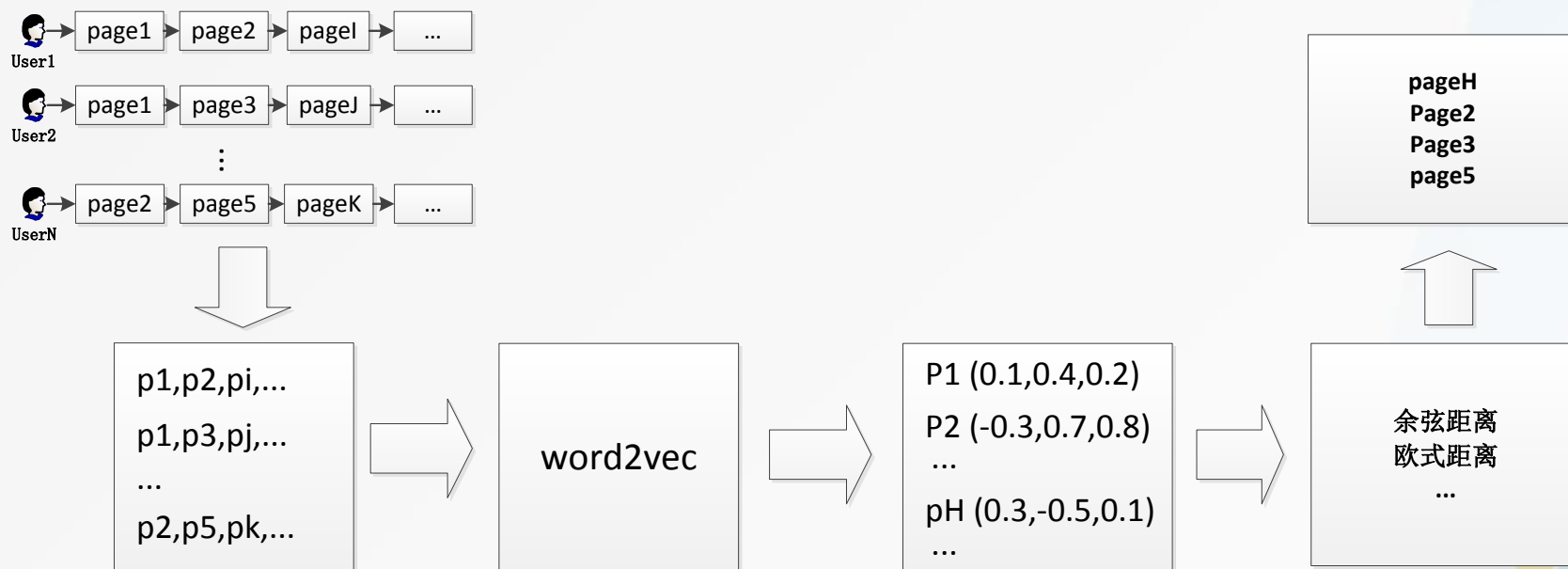
word2vec应用（四）

- 给广告主推荐用户
 - T媒体网站保存了用户浏览网页的记录
 - pageH是匹克体育用品公司在T上的官网
 - page2,page3,page5和pageH是比较相似的页面
 - 可给匹克体育用品公司推荐经常浏览page2,3,5这个几个页面的用户进行广告投放



word2vec应用（四）

- 相似的页面计算过程



也可用于挖掘同好用户：将一簇簇相似的用户作为doc（譬如QQ群），将单个用户作为word，我们则可以训练user distributed representation，可以借此挖掘相似用户。

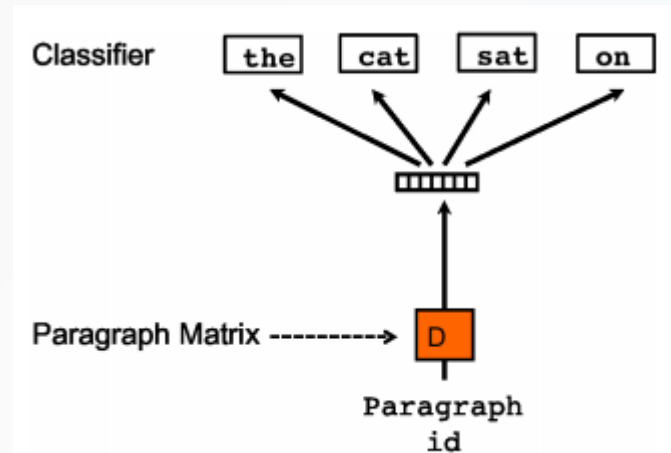
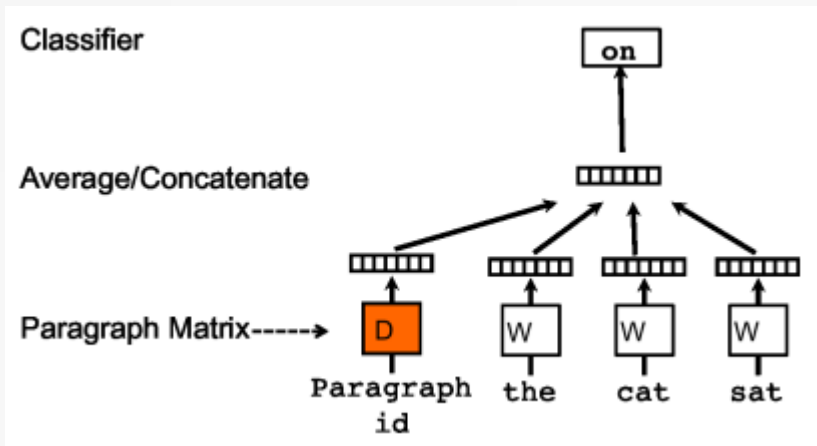
Paragraph2vec

- 训练过程中新增了paragraph id，即训练语料中每个句子都有一个唯一的id。paragraph id和普通的word一样，也是先映射成一个向量。paragraph vector与word vector的维数虽一样，但是来自于两个不同的向量空间。
- 对应于cbow，在计算里，paragraph vector和word vector累加或者连接起来，作为输出层softmax的输入。在一个句子或者文档的训练过程中，paragraph id保持不变，共享着同一个paragraph vector，相当于每次在预测单词的概率时，都利用了整个句子的语义。

Paragraph2vec

- 对应于skip-gram,

at each iteration of stochastic gradient descent, we sample a text window, then sample a random word from the text window and form a classification task given the Paragraph Vector.



Paragraph2vec

- 在预测阶段，给待预测的句子新分配一个 **paragraph id**，词向量和输出层**softmax**的参数保持训练阶段得到的参数不变，重新利用梯度下降训练待预测的句子。待收敛后，即得到待预测句子的**paragraph vector**。

Paragraph2vec

Sentence: 梁山伯与祝英台 Position in vocabulary: 7499	
Sentence	Cosine distance
梁山伯与朱丽叶	0.977390
梁山伯与朱丽叶	0.970993
梁山伯与祝英台电影	0.966587
梁山伯与祝英台新传	0.956935
越剧梁山伯与祝英台	0.951074
罗密欧与朱丽叶	0.945896
梁山伯与祝英台的故事	0.945522
罗密欧与朱丽叶	0.943463
梁山伯与朱丽叶歌词	0.938179
罗密欧与朱丽叶莱昂纳多	0.938168
罗密欧与朱丽叶1996	0.937464
梁山伯与祝英台主题曲	0.935705
吉诺密欧与朱丽叶	0.935683
梁思成与林徽因	0.935257
梁山伯与祝英台罗志祥	0.929949
罗密欧与朱丽叶电影	0.929613
王宝钏与薛平贵	0.928201
陈世美与秦香莲	0.928120
罗密欧与朱丽叶书籍	0.925813
薛平贵与王宝钏	0.925322
朱丽叶与罗密欧	0.925263
朗朗与杨嘉言	0.922493
安徒与黑仔	0.921022
傲慢与偏见	0.920723
蔡锷与小凤仙	0.920321
李后主与赵匡胤	0.918377
傲慢与偏见书籍	0.917086
罗密欧与朱丽叶的故事	0.914915
薛平贵与王宝	0.914541
唐三娘与杨柳虎王英	0.914000
罗密欧与朱丽叶剧本	0.913900
朗朗与杨嘉言	0.913703
艾尔文与花栗鼠	0.910418
傲慢与偏见读后感	0.910055
蔡锷与徐向前的恩怨	0.909936
罗密欧与朱丽叶书籍书译	0.907552
洛克与飞鸟	0.907524
罗密欧与朱丽叶歌曲	0.907324
薛平贵与王宝钏全集	0.906792
薛平贵与王宝钏插曲	0.906485
唐玄宗与杨贵妃	0.906347
罗密欧与朱丽叶钢琴曲	0.905614
魏靖与海瑞	0.904830
梁山伯与祝英台罗志祥版	0.904320
杨雄与石秀	0.904059
芬妮与亚历山大	0.903160
杨乃武与小白菜	0.902939
杨乃武与小白菜电影	0.901605
电磁场与电磁波	0.901341
斯滴滴与怒出出	0.900046
喜羊羊与	0.899474
方世玉与洪熙官	0.898899
顺治帝与董鄂妃	0.898492
罗密欧与朱丽叶读后感	0.898134
霍顿与无名氏	0.898078
王贵与安娜	0.897918

参考文献

- [1] <http://blog.csdn.net/mytestmy/article/details/26969149> 深度学习word2vec笔记之算法篇
- [2] <http://blog.csdn.net/itplus/article/details/37969979> word2vec 中的数学原理详解（四）基于 Hierarchical Softmax 的模型
- [3] <http://www.zhihu.com/question/21661274/answer/19331979> @杨超在知乎上的问答《Word2Vec的一些理解》
- [4] <http://xiaoquanzi.net/?p=156> hisen博客的博文
- [5] <http://blog.csdn.net/mytestmy/article/details/38612907> 深度学习word2vec笔记之应用篇
- [6] <http://techblog.youdao.com/?p=915> Deep Learning实战之word2vec, 网易有道的pdf
- [7] <http://blog.csdn.net/lingerlanlan/article/details/38232755> word2vec源码解析之word2vec.c
- [8] Hierarchical probabilistic neural network language model. Frederic Morin and Yoshua Bengio.
- [9] Distributed Representations of Words and Phrases and their Compositionality T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean.
- [10] A neural probabilistic language model Y. Bengio, R. Ducharme, P. Vincent.
- [11] Linguistic Regularities in Continuous Space Word Representations. Tomas Mikolov, Wen-tau Yih, Geoffrey Zweig.
- [12] Efficient Estimation of Word Representations in Vector Space. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean.
- [13] <http://licstar.net/archives/328> Deep Learning in NLP （一）词向量和语言模型

thank you !
Q&A