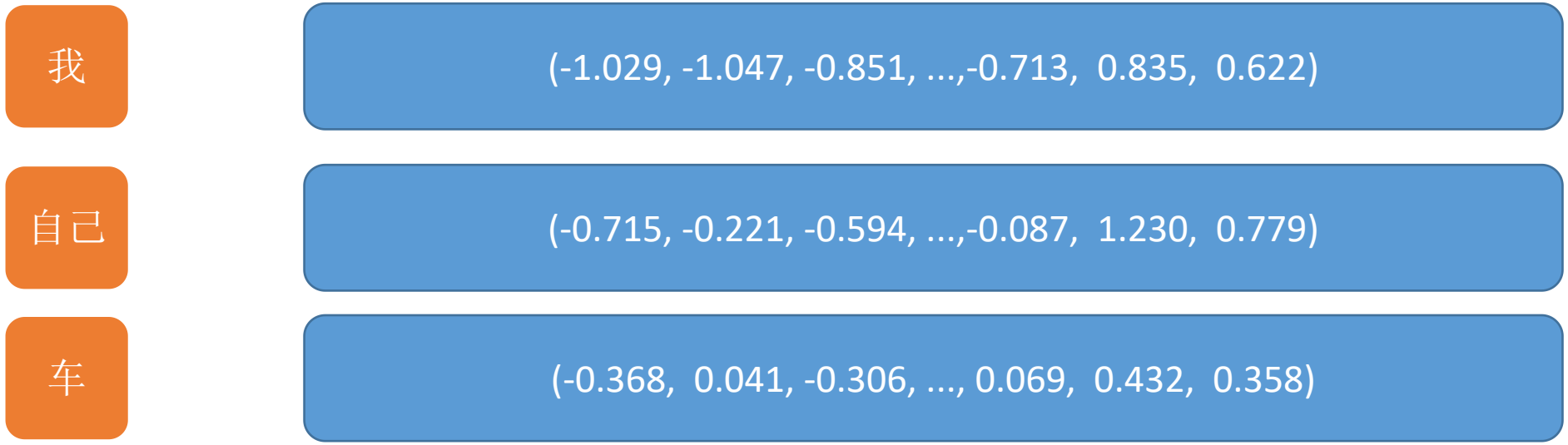
NLP项目
词向量

NLP基础_词向量

- 在分词之后，对于文本类型的特征属性，需要进行文本数据转换，也就是需要将文本/Token数据转换为数值型数据。常用方式如下：
  - **序号化**、哑编码(One-Hot)、词袋法(BOW/TF)
  - TF-IDF(Term frequency-inverse document frequency)
  - 主题模型(LSA、LDA等)
  - **Word2Vec、Char2Vec**
  - Doc2Vec
  - FastText、cw2vec
- 常见词向量转换工具：
  - gensim
    - https://github.com/RaRe-Technologies/gensim
    - https://radimrehurek.com/gensim/index.html

NLP基础_词向量

- 词向量转换也就是将文本数据转换为数值型数据的一种方式，在人工智能中，这种方式统称为**Word Embedding**；**Word Embedding**实际上就是一种**映射**，将文本空间中的某个**word**，通过一定的方法，**映射**或者**嵌入(Embedding)**到另外一个数值向量空间。之所以称为embedding，是因为这种映射往往伴随着降维的思想。



NLP基础-词向量-序号化

- 将单词按照**词典映射**，给定从0或者1或者2开始的序号即可，一般情况有几个特征的单词: <PAD>、<UNK>；PAD表示填充字符，UNK表示未知字符。
  - 假设存在三个文本/样本:[我 是 小明，我 来自 湖南 长沙，我 喜欢 辣椒]

单词	序号
<PAD>	0
<UNK>	1
我	2
湖南	3
长沙	4
是	5
来自	6
喜欢	7
辣椒	8
…	

	文本1	文本2	文本3
Token1	2	2	2
Token2	5	6	7
Token3	1	3	8
Token4	0	4	0

NLP基础_词向量_One-Hot

- 使用一个**非常稀疏**的向量来表示**单词**的特征向量信息，假设现在有n个单词(词表大小为n)，那么转换的特征向量就是n维，仅在对应位置为1，其它位置全部为0。

单词的OneHot结果

我	1,0,0,0,0,0,0,0,0,0,0,...,0
来	0,1,0,0,0,0,0,0,0,0,0,...,0
自	0,0,1,0,0,0,0,0,0,0,0,...,0
湖	0,0,0,1,0,0,0,0,0,0,0,...,0
南	0,0,0,0,1,0,0,0,0,0,0,...,0
张	0,0,0,0,0,1,0,0,0,0,0,...,0
家	0,0,0,0,0,0,1,0,0,0,0,...,0
界	0,0,0,0,0,0,0,1,0,0,0,...,0
文本向量	1,1,1,1,1,1,1,1,0,0,0,...,0

文本向量 --> 词袋法
单词向量求和

NLP基础_词向量_词袋法

- 词袋法(Bag of words, BOW)是最早应用于NLP和IR领域的一种文本处理模型，该模型**忽略文本的语法和语序**，用一组无序的单词(words)来表达一段文字或者一个文档，词袋法中**使用单词在文档中出现的频数/频率来表示文档，使用所有文档中出现的单词作为特征属性**。

d1:this is a sample is a sample

d2:this is anthoer example anthoer example

dict(词典、字典、特征属性): this sample anthoer example

	this	sample	another	example
d_1	1	2	0	0
d_2	1	0	2	2

NLP基础_词向量_TFIDF

- 在词袋法的基础上加入单词重要性的影响系数：
  - **单词的重要性随着它在文本(当前)中出现的次数成正比增加，也就是单词的出现次数越多，该单词对于文本的重要性就越高。** --> TF
  - **同时单词的重要性会随着在语料库/训练数据中出现的频率成反比下降，也就是单词在语料库中出现的频率越高，表示该单词越常见，也就是该单词对于文本的重要性越低。** --> IDF

Variants of term frequency (TF) weight

weighting scheme	TF weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of inverse document frequency (IDF) weight

weighting scheme	IDF weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(1 + \frac{N}{n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

NLP基础_词向量_TFIDF

N表示总的文档数目，n_t表示单词t在所有文档中出现的文档数目

- 有两个文档，单词统计如下，请分别计算各个单词在文档中的TF-IDF值以及这些文档使用单词表示的特征向量。

$$idf = \log\left(\frac{N+1}{n_t+1}\right)$$

单词	单词次数
this	1
is	1
a	2
sample	1

单词	单词次数
this	2
is	1
another	2
example	3

$$tf("this", d_1) = \frac{1}{5} \quad tf("this", d_2) = \frac{2}{8}$$

$$idf("this", D) = \log\left(\frac{2+1}{2+1}\right) = 0$$

$$tfidf("this", d_1) = tf("this", d_1) * idf("this", D) = 0$$

$$tfidf("this", d_2) = 0$$

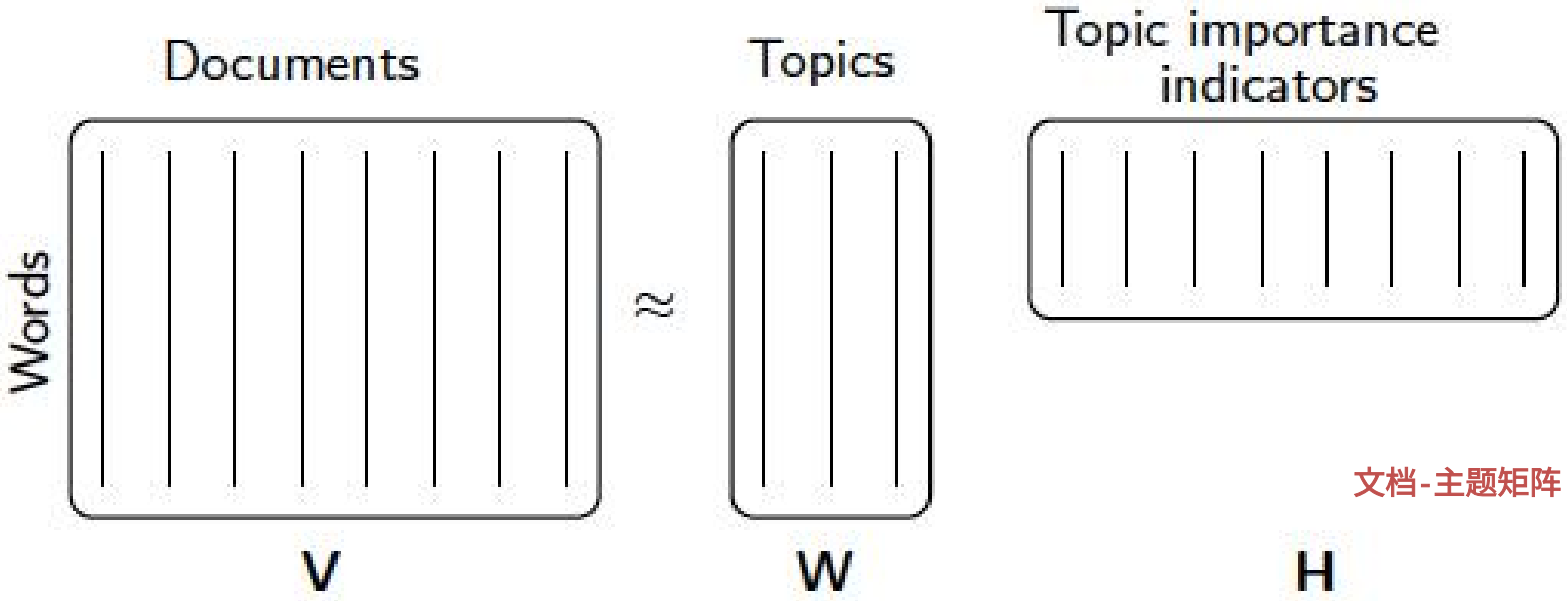
	this	is	a	another	sample	example
d_1	0	0	0.191	0	0.095	0
d_2	0	0	0	0.119	0	0.179

NLP基础_词向量

- 两个词向量之间的距离（例如，任意两个向量之间的L2范式距离或更常用的余弦距离）一定程度上表征了的词之间的语义关系；
- 例如，“椰子”和“北极熊”是语义上完全不同的词，所以它们的词向量在一个合理的嵌入空间的距离将会非常遥远。但“厨房”和“晚餐”是相关的话，所以它们的词向量之间的距离会相对小。
- 但是哑编码、词袋法以及TF-IDF这些方式都不能达到这个效果，故提出了基于矩阵分解的**主题模型**算法以及基于“神经网络”的**Word2Vec**的词向量转换方式。

NLP基础_词向量_主题模型

- 直接使用机器学习的主题模型相关算法将词袋法/TF-IDF转换的文本单词特征属性句子转换为文本主题特征属性矩阵以及单词主题特征属性矩阵即可得到单词对应的特征向量以及文本对应的特征向量。
- V矩阵就是词袋法/TFIDF转换出来的文本-单词特征向量矩阵，W和H是V分解后的两个矩阵。



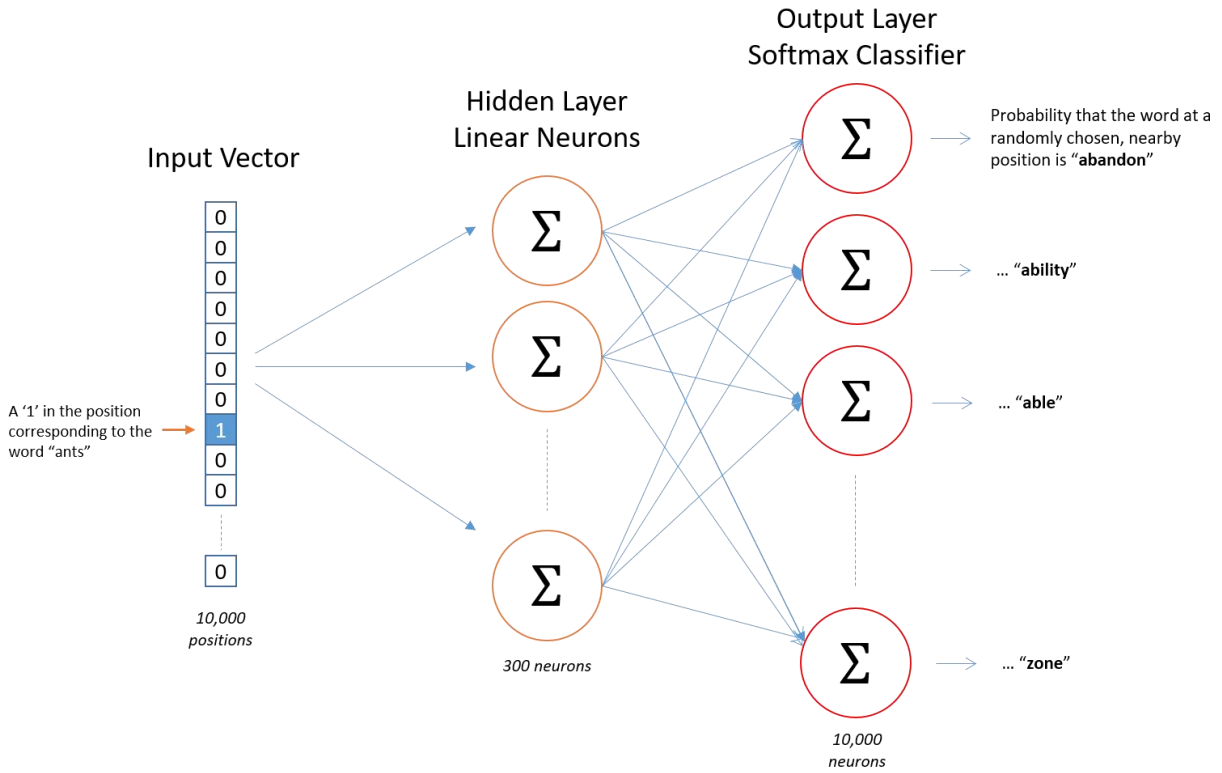
NLP基础_词向量_Word2Vec

- 词向量方法是“无监督学习”的少数几个成功应用之一，优点在于不需要人工进行预料的标注，直接使用未标注的文本训练集作为输入，**输出的词向量**可以用于下游的业务处理。
- 词向量真正的推广是开始于2013年Mikolov开发出来的Word2Vec，2014年Pennington发布了GloVe，进一步让让词向量的应用成为NLP领域中的主流。
- NOTE: Word2Vec可以认为是应用最广泛的词向量转换技术。

NLP基础_词向量_Word2Vec

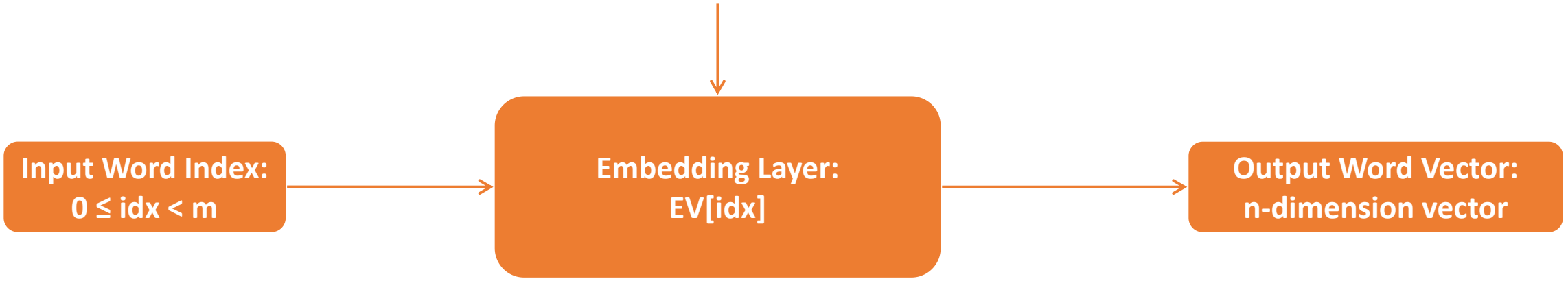
- 神经网络将词表中的词语作为输入(一般输入哑编码的单词)，输出一个低维度的向量表示这个词语，然后用反向传播的方法不断优化参数。输出的低维向量是神经网络第一层的输出，这一层通常也称作**Embedding Layer**。

$$\begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix} \times \begin{bmatrix}8 & 2 & 1 & 9\\ 6 & 5 & 4 & 0\\ 7 & 1 & 6 & 2\\ 1 & 3 & 5 & 8\\ 0 & 4 & 9 & 1\end{bmatrix} = \begin{bmatrix}1 & 3 & 5 & 8\end{bmatrix}$$



NLP基础_词向量_Word2Vec

$$EV_{m,n} = \begin{bmatrix} -1.0152 & .... & .... & 0.1254 \\ .... & .... & .... & .... \\ .... & .... & .... & .... \\ 1.0251 & .... & .... & 0.2535 \end{bmatrix}$$

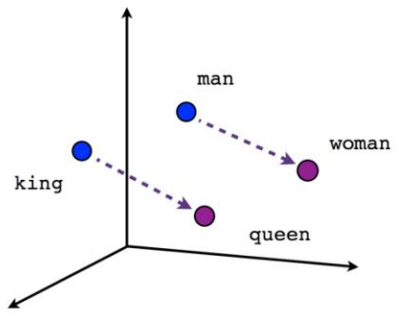


NLP基础_词向量_Word2Vec

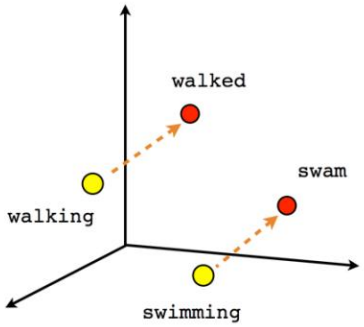
- 生成词向量的神经网络模型分为两种，一种是类似Word2Vec的方式，这类模型的目的就是生成词向量，另一种是将词向量作为“副产品”产生，两者的区别在于计算量不同。若词表非常庞大，用深层结构的模型训练词向量需要许多计算资源。
- Word2Vec和GloVe(Global Vectors for Word Representation)的目的是训练可以表示语义关系的词向量，它们能被用于后续的任务中；如果后续任务不需要用到语义关系，则按照此方式生成的词向量并没有什么用。另一种模型则根据特定任务需要训练词向量。当然，若特定的任务就是对语言建模，那么两种模型生成的词向量非常相似了。

NLP基础_词向量_Word2Vec

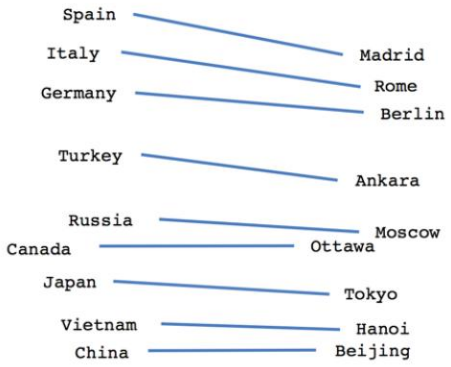
- Word2Vec(Efficient Estimation of Word Representations in Vector Space)
  - CBOW
    - predicts the current word based on the context;
  - Skip-Gram
    - predicts surrounding words given the current word;



Male-Female

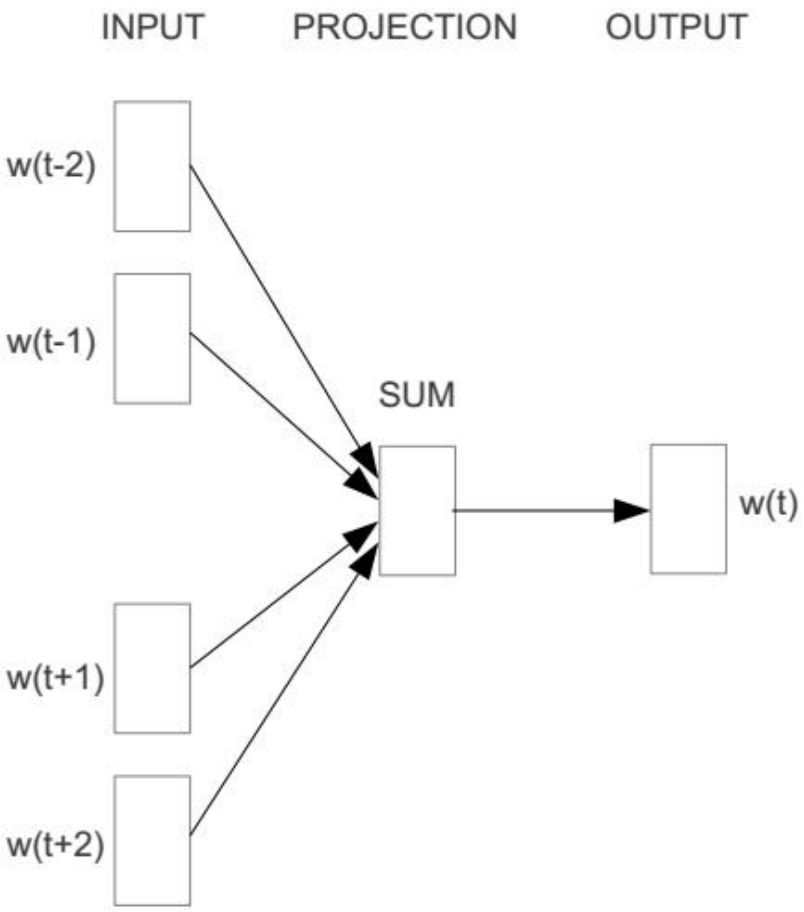


Verb tense

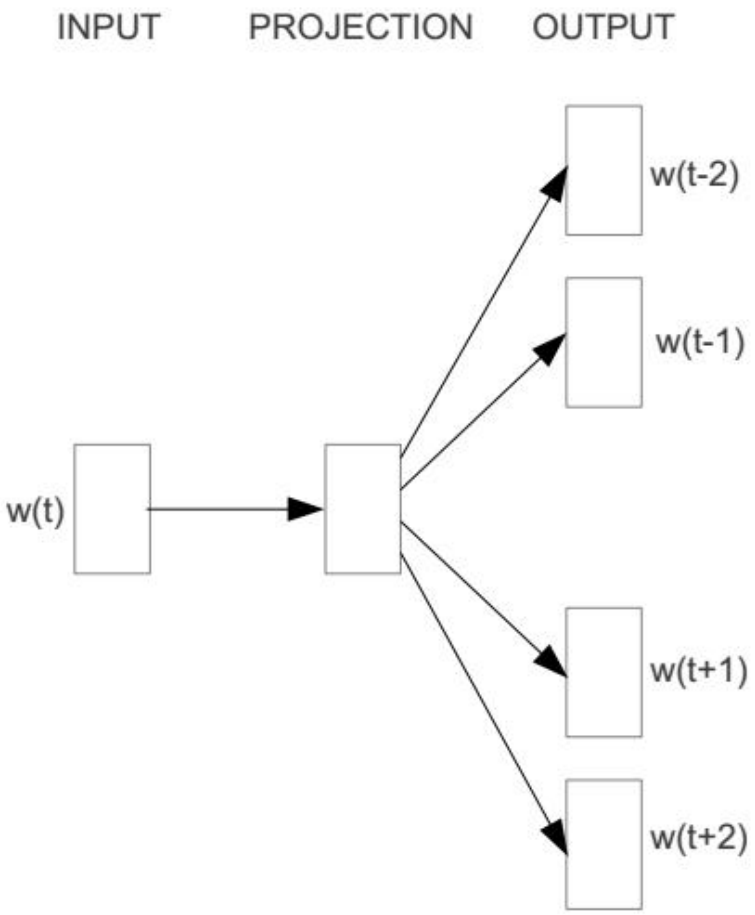


Country-Capital

NLP基础_词向量_Word2Vec



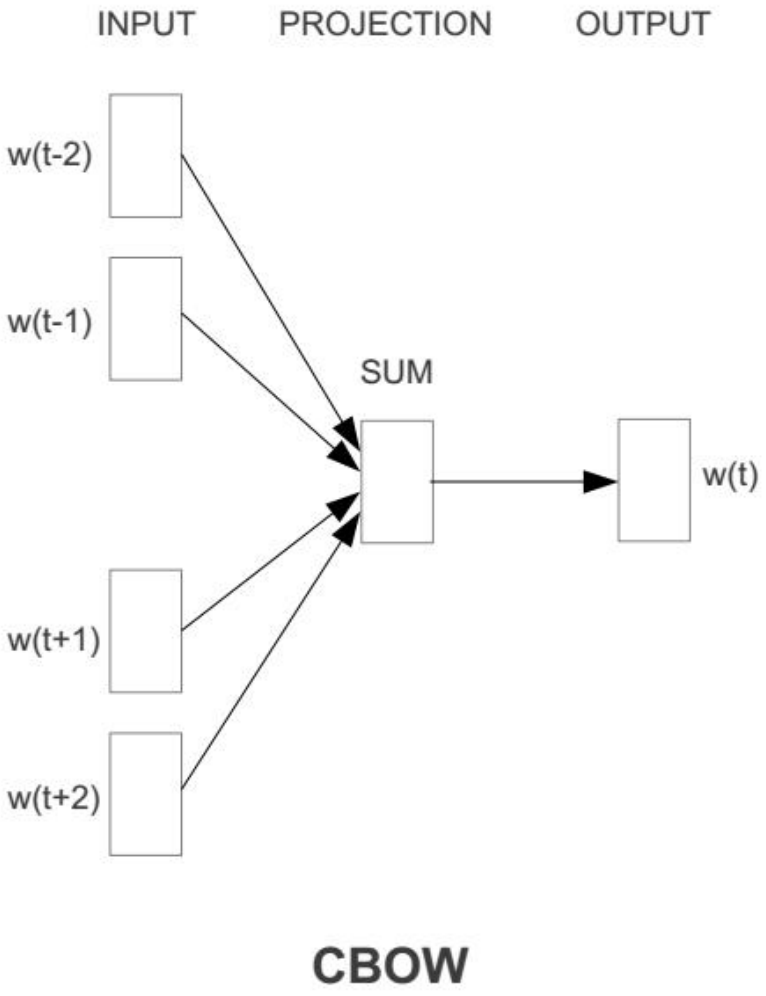
CBOW



Skip-gram

NLP基础_词向量_Word2Vec_CBOW

- CBOW(Continuous Bag of Words)
  - 给定上下文预测目标词的概率分布，例如，给定｛The，cat，()，over，the，puddle｝预测中心词是jumped的概率，模型的结构如下:
  - 比如：”一只猫坐在地上”，中间词为”猫”，那么上下文词为: ["<PAD>"，“一只”，“坐在”，“地上”]



NLP基础_词向量_Word2Vec_CBOW

词向量转换

$$v_{c-k} = Uw_{c-k}$$

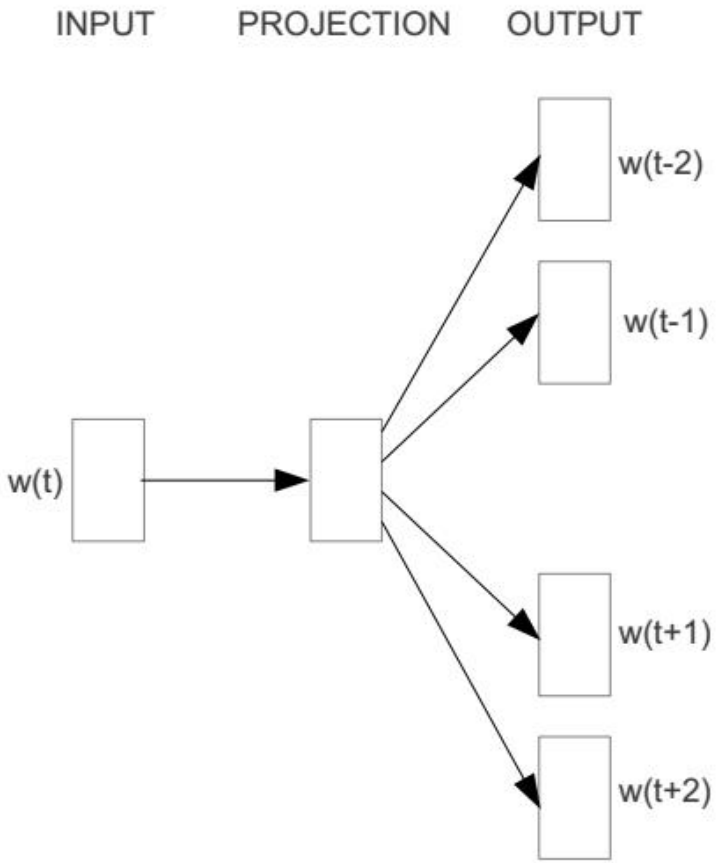
$$v_c = v_{c-m} + \ldots + v_{c-1} + v_{c+1} + \ldots + v_{c+m}$$

$$z_c = Vv_c$$

$$\begin{aligned}
\text{minimize } J &= -\log P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c | \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}$$

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram
  - 给定中心词预测上下文各个单词的概率分布，例如，给定jumped，预测上下文单词｛The，cat，()，over，the，puddle｝的概率，模型的结构如下:
  - 比如：”一只猫坐在地上”，中间词为”猫”，那么上下文词为: ["<PAD>",“一只”，“坐在”，“地上”]



Skip-gram

NLP基础_词向量_Word2Vec_Skip-gram

词向量转换

$$v_c = Uw_c$$

$$z_{2m} = V_{2m} v_c$$

$$\begin{aligned}
\text{minimize } J &= -\log P(w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(w_{c-m+j}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(u_{c-m+j}|v_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} \frac{\exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|}\exp(u_k^T v_c)} \\
&= -\sum_{j=0,j\neq m}^{2m} u_{c-m+j}^T v_c + 2m\log\sum_{k=1}^{|V|}\exp(u_k^T v_c)
\end{aligned}$$

NLP基础_词向量_Word2Vec

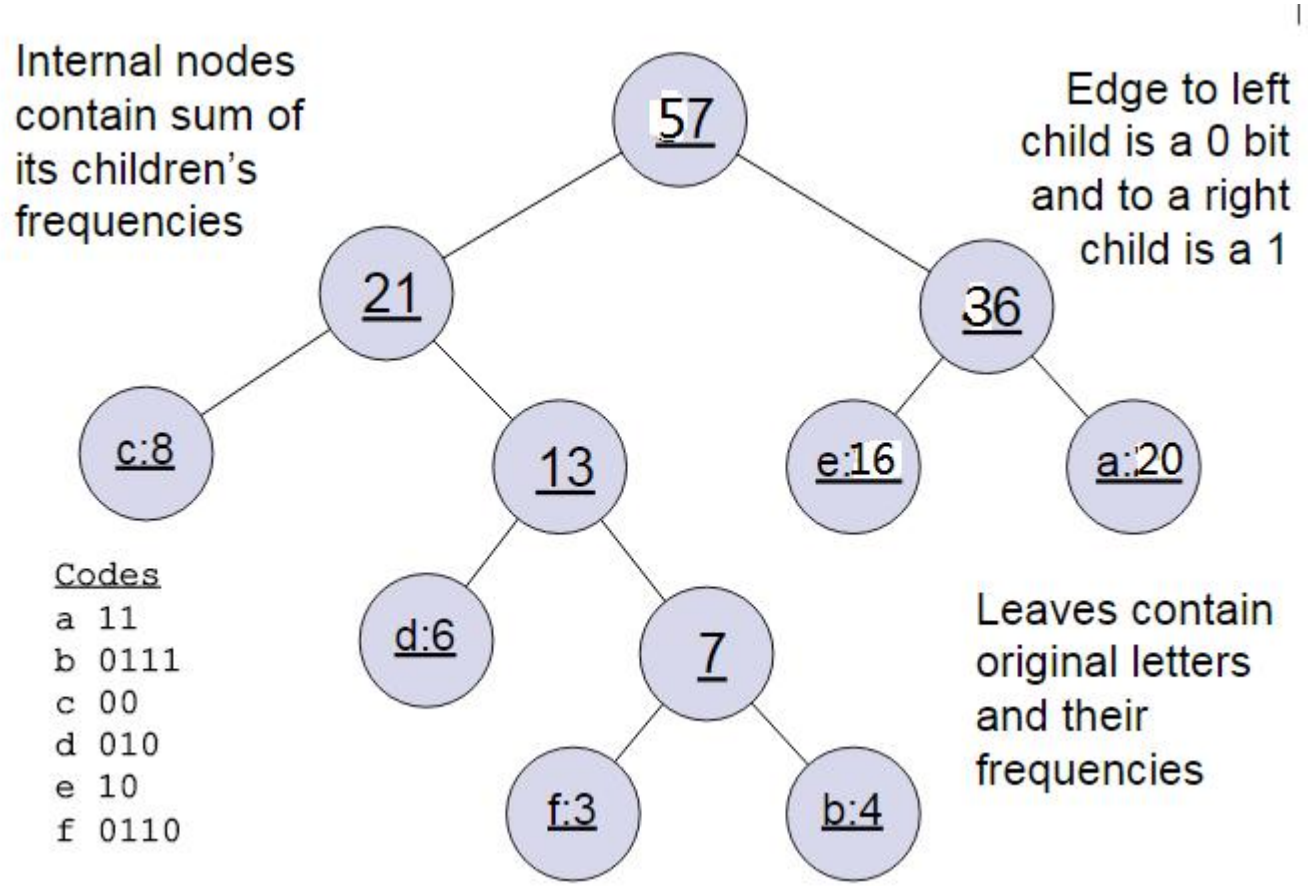
- 上述所说的Word2Vec是最原始的结构，在计算过程中需要计算所有单词(2m)和中心词之间的概率，**比较耗时**，所以在Word2Vec的论文(Efficient Estimation of Word Representations in Vector Space)中，作者Mikolov提出**Hierarchical softmax(霍夫曼树,HS)**和**Negative sampling(负样本采样)**两种方法对Word2Vec的模型训练进行优化。这也是Word2Vec能够大量真正应用于NLP领域的主要原因(不需要依赖神经网络训练模型)。

NLP基础_词向量_Word2Vec

- CBOW和Skip-Gram的训练可以使用霍夫曼树加速，霍夫曼树的构建如下所示：
  - 输入： 权重值为(w1, w2, ..., wn，单词出现的次数)为n个节点(单词)
  - 输出：对应的霍夫曼树
  - 步骤：
    - 将(w1, w2, ..., wn)看成有n棵树的森林，每个树仅有一个节点；
    - 在森林中选择根节点权重最小的两棵树进行合并，得到一个新的树，这两颗树分布作为新树的左右子树，新树的根节点权重为左右子树的根节点权重之和。
    - 将之前的根节点权重最小的两棵树从森林删除，并将新树加入森林；
    - 重复步骤2和步骤3直到森林只有一个树为止。

NLP基础_词向量_Word2Vec

- 六个节点: (a, b, c, d, e, f); 节点权重为: (20, 4, 8, 6, 16, 3)



NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

1. **输入层**: 包含 $Context(w)$ 中 $2c$ 个词的词向量 $\mathbf{v}(Context(w)_1), \mathbf{v}(Context(w)_2), \cdots, \mathbf{v}(Context(w)_{2c}) \in \mathbb{R}^m$. 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 $\mathbf{x}_w = \sum_{i=1}^{2c} \mathbf{v}(Context(w)_i) \in \mathbb{R}^m$.

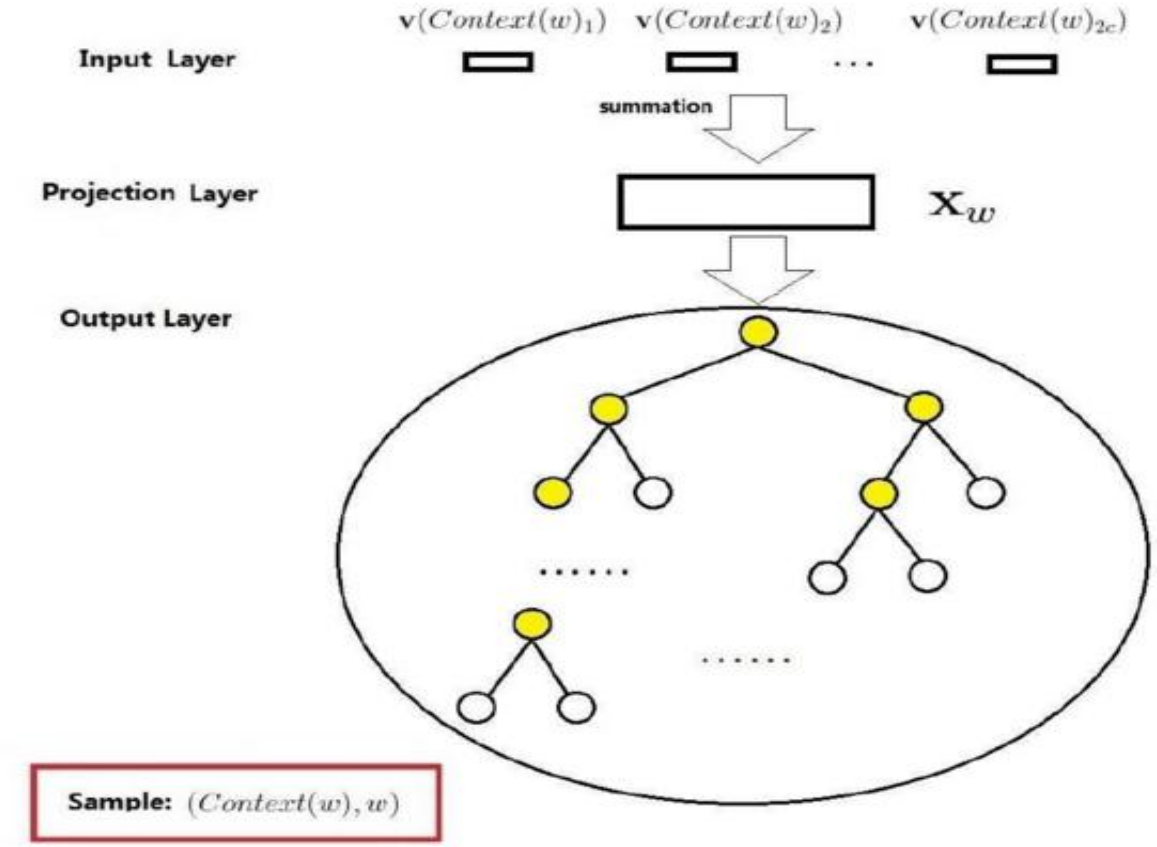


图 10 CBOW 模型的网络结构示意图

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

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).

对比 §3.3 中神经概率语言模型的网络图 (见图 4) 和 CBOW 模型的结构图 (见图 10), 易知它们主要有以下三处不同:

1. (从输入层到投影层的操作) 前者是通过拼接, 后者通过**累加求和**.

2. (隐藏层) 前者有隐藏层, 后者**无隐藏层**.

3. (输出层) 前者是线性结构, 后者是**树形结构**.

在 §3.3 介绍的神经概率语言模型中, 我们指出, 模型的大部分计算集中在隐藏层和输出层之间的矩阵向量运算, 以及输出层上的 softmax 归一化运算. 而从上面的对比中可见, CBOW 模型对这些计算复杂度高的地方有针对性地进行了改变, 首先, 去掉了隐藏层, 其次, 输出层改用了 Huffman 树, 从而为利用 Hierarchical softmax 技术奠定了基础.

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

§4.1.2 梯度计算

Hierarchical Softmax 是 wor2vec 中用于提高性能的一项关键技术. 为描述方便起见, 在具体介绍这个技术之前, 先引入若干相关记号. 考虑 Huffman 树中的某个叶子结点, 假设它对应词典 $\mathcal{D}$ 中的词 w, 记

1. p^w: 从根结点出发到达 w 对应叶子结点的路径.
2. l^w: 路径 p^w 中包含结点的个数.
3. $p_1^w, p_2^w, \cdots, p_{l^w}^w$: 路径 p^w 中的 l^w 个结点, 其中 p_1^w 表示根结点, $p_{l^w}^w$ 表示词 w 对应的结点.
4. $d_2^w, d_3^w, \cdots, d_{l^w}^w \in \{0,1\}$: 词 w 的 Huffman 编码, 它由 $l^w - 1$ 位编码构成, d_j^w 表示路径 p^w 中第 j 个结点对应的编码 (根结点不对应编码).
5. $\theta_1^w, \theta_2^w, \cdots, \theta_{l^w-1}^w \in \mathbb{R}^m$: 路径 p^w 中**非叶子结点**对应的向量, θ_j^w 表示路径 p^w 中第 j 个非叶子结点对应的向量.

注 4.1 按理说, 我们要求的是词典 $\mathcal{D}$ 中每个词 (即 *Huffman* 树中所有叶子节点) 的向量, 为什么这里还要为 *Huffman* 树中每一个非叶子结点也定义一个同长的向量呢? 事实上, 它们只是算法中的辅助向量, 具体用途在下文中将会为大家解释清楚.

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

好了，引入了这么一大堆抽象的记号，接下来，我们还是通过一个简单的例子把它们落到实处吧，看图 11，仍以预备知识中例 2.1 为例，考虑词 $w =$ “足球”的情形.

图 11 中由 4 条红色边串起来的 5 个节点就构成路径 p^w，其长度 $l^w = 5$. $p_1^w, p_2^w, p_3^w, p_4^w, p_5^w$ 为路径 p^w 上的 5 个结点，其中 p_1^w 对应根结点. $d_2^w, d_3^w, d_4^w, d_5^w$ 分别为 1, 0, 0, 1，即“足球”的 Huffman 编码为 1001. 此外，$\theta_1^w, \theta_2^w, \theta_3^w, \theta_4^w$ 分别表示路径 p^w 上 4 个非叶子结点对应的向量.

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

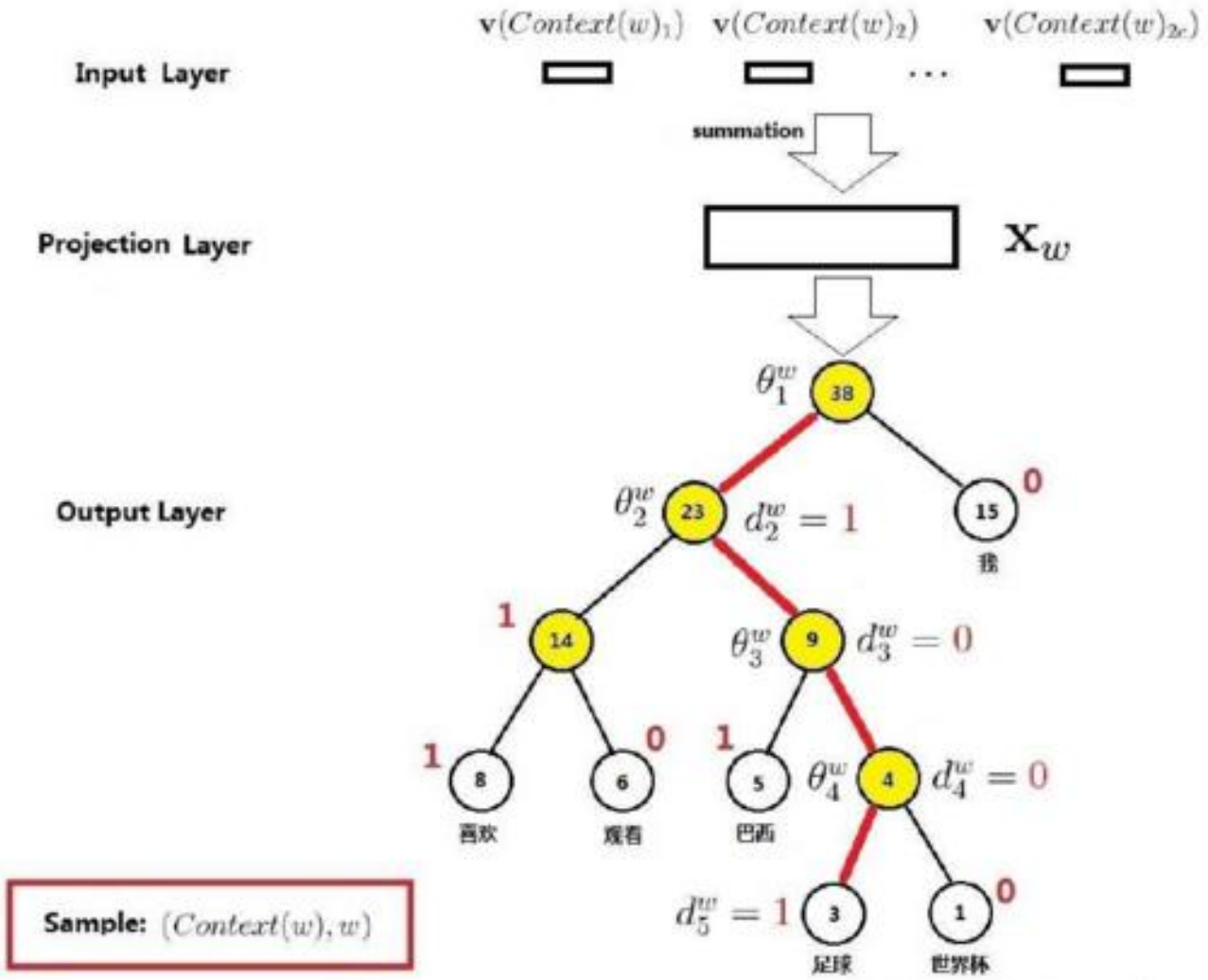


图 11 $w =$ “足球” 时的相关记号示意图

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

那么，在如图 10 所示的网络结构下，如何定义条件概率函数 $p(w|Contex(w))$ 呢？更具体地说，就是如何利用向量 $\mathbf{x}_w \in \mathbb{R}^m$ 以及 Huffman 树来定义函数 $p(w|Contex(w))$ 呢？

以图 11 中词 $w =$“足球”为例，从根结点出发到达“足球”这个叶子节点，中间共经历了 4 次分支 (每条红色的边对应一次分支)，而每一次分支都可视为进行了一次**二分类**.

既然是从二分类的角度来考虑问题，那么对于每一个非叶子结点，就需要为其左右孩子结点指定一个类别，即哪个是正类 (标签为 1)，哪个是负类 (标签为 0)．碰巧，除根结点以外，树中每个结点都对应了一个取值为 0 或 1 的 Huffman 编码．因此，一种最自然的做法就是将 Huffman 编码为 1 的结点定义为正类，编码为 0 的结点定义为负类．当然，这只是个约定而已，你也可以将编码为 1 的结点定义为负类，而将编码为 0 的结点定义为正类．事实上，word2vec 选用的就是后者，为方便读者对照着文档看源码，下文中统一采用后者，即约定

$$Label(p_i^w) = 1 - d_i^w, i = 2, 3, \cdots, l^w.$$

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

简言之就是，**将一个结点进行分类时，分到左边就是负类，分到右边就是正类.**

根据预备知识 §2.2 中介绍的逻辑回归，易知，一个结点被分为正类的概率是

$$\sigma(\mathbf{x}_w^\top\theta)=\frac{1}{1+e^{-\mathbf{x}_w^\top\theta}},$$

被分为负类的概率当然就等于

$$1-\sigma(\mathbf{x}_w^\top\theta),$$

注意，上式中有个叫 θ 的向量，它是待定参数，显然，在这里非叶子结点对应的那些向量 θ_i^w 就可以扮演参数 θ 的角色 (这也是为什么将它们取名为 θ_i^w 的原因).

对于从根结点出发到达“足球”这个叶子节点所经历的 4 次二分类，将每次分类结果的概率写出来就是

1. 第 1 次: $p(d_2^w|\mathbf{x}_w,\theta_1^w)=1-\sigma(\mathbf{x}_w^\top\theta_1^w)$;
2. 第 2 次: $p(d_3^w|\mathbf{x}_w,\theta_2^w)=\sigma(\mathbf{x}_w^\top\theta_2^w)$;
3. 第 3 次: $p(d_4^w|\mathbf{x}_w,\theta_3^w)=\sigma(\mathbf{x}_w^\top\theta_3^w))$;
4. 第 4 次: $p(d_5^w|\mathbf{x}_w,\theta_4^w)=1-\sigma(\mathbf{x}_w^\top\theta_4^w)$,

但是，我们要求的是 $p(\text{足球}|Contex(\text{足球}))$，它跟这 4 个概率值有什么关系呢? 关系就是

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

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

至此，通过 $w =$ “足球”的小例子，Hierarchical Softmax 的基本思想其实就已经介绍完了. 小结一下：**对于词典 $\mathcal{D}$ 中的任意词 w，Huffman 树中必存在一条从根结点到词 w 对应结点的路径 p^w（且这条路径是唯一的）. 路径 p^w 上存在 $l^w - 1$ 个分支，将每个分支看做一次二分类，每一次分类就产生一个概率，将这些概率乘起来，就是所需的 $p(w|Context(w))$.**

条件概率 $p(w|Context(w))$ 的一般公式可写为

$$p(w|Context(w)) = \prod_{j=2}^{l^w} p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w), \tag{4.3}$$

其中

$$p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta_{j-1}^w), & d_j^w = 0; \\ 1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w), & d_j^w = 1, \end{cases}$$

或者写成整体表达式

$$p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w) = [\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{d_j^w}.$$

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

注 4.2 在 §3.3 中，最后得到的条件概率为

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

具体见 (3.6) 式，由于这里有个归一化操作，因此显然成立

$$\sum_{w\in\mathcal{D}} p(w|Context(w)) = 1.$$

然而，对于由 (4.3) 定义的概率，是否也能满足上式呢？这个问题留给读者思考.

将 (4.3) 代入对数似然函数 (4.1)，便得

$$\begin{aligned}\mathcal{L} &= \sum_{w\in\mathcal{C}} \log \prod_{j=2}^{l^w} \{[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{1-d_j^w} \cdot [1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{d_j^w}\} \\ &= \sum_{w\in\mathcal{C}} \sum_{j=2}^{l^w} \{(1-d_j^w)\cdot \log[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] + d_j^w \cdot \log[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\}, \end{aligned} \tag{4.4}$$

为下面梯度推导方便起见，将上式中双重求和符号下花括号里的内容简记为 $\mathcal{L}(w,j)$，即

$$\mathcal{L}(w,j) = (1-d_j^w)\cdot \log[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] + d_j^w \cdot \log[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]. \tag{4.5}$$

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

至此，已经推导出对数似然函数 (4.4)，这就是 CBOW 模型的目标函数，接下来讨论它的优化，即如何将这个函数最大化. word2vec 里面采用的是**随机梯度上升法***. 而梯度类算法的关键是给出相应的梯度计算公式，因此接下来重点讨论梯度的计算.

随机梯度上升法的做法是：每取一个样本 $(Context(w), w)$，就对目标函数中的所有（相关）参数做一次刷新. 观察目标函数 $\mathcal{L}$ 易知，该函数中的参数包括向量 $\mathbf{x}_w, \theta_{j-1}^w$，$w \in \mathcal{C}$，$j = 2, \cdots, l^w$. 为此，先给出函数 $\mathcal{L}(w, j)$ 关于这些向量的梯度.

首先考虑 $\mathcal{L}(w, j)$ 关于 θ_{j-1}^w 的梯度计算.

$$
\begin{aligned}
\frac{\partial \mathcal{L}(w,j)}{\partial \theta_{j-1}^w} &= \frac{\partial}{\partial \theta_{j-1}^w}\left\{(1-d_j^w)\cdot \log[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] + d_j^w \cdot \log[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\right\} \\
&= (1-d_j^w)[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\mathbf{x}_w - d_j^w \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\mathbf{x}_w \quad \text{(利用 (2.1) 式)} \\
&= \left\{(1-d_j^w)[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] - d_j^w \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\right\}\mathbf{x}_w \quad \text{(合并)} \\
&= \left[1 - d_j^w - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\right]\mathbf{x}_w.
\end{aligned}
$$

于是，θ_{j-1}^w 的更新公式可写为

$$\theta_{j-1}^w := \theta_{j-1}^w + \eta\left[1 - d_j^w - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\right]\mathbf{x}_w,$$

其中 η 表示**学习率**，下同.

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

接下来考虑 $\mathcal{L}(w,j)$ 关于 $\mathbf{x}_w$ 的梯度. 观察 (4.5) 可发现, $\mathcal{L}(w,j)$ 中关于变量 $\mathbf{x}_w$ 和 θ_{j-1}^w 是**对称**的 (即两者可交换位置), 因此, 相应的梯度 $\frac{\partial \mathcal{L}(w,j)}{\partial \mathbf{x}_w}$ 也只需在 $\frac{\partial \mathcal{L}(w,j)}{\partial \theta_{j-1}^w}$ 的基础上对这两个向量交换位置就可以了, 即

$$\frac{\partial \mathcal{L}(w,j)}{\partial \mathbf{x}_w} = \left[1 - d_j^w - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\right] \theta_{j-1}^w.$$

到这里, 细心的读者可能已经看出问题来了: 我们的最终目的是要求词典 $\mathcal{D}$ 中每个词的词向量, 而这里的 $\mathbf{x}_w$ 表示的是 $Context(w)$ 中各词词向量的累加. 那么, 如何利用 $\frac{\partial \mathcal{L}(w,j)}{\partial \mathbf{x}_w}$ 来对 $\mathbf{v}(\widetilde{w}), \widetilde{w} \in Context(w)$ 进行更新呢? word2vec 中的做法很简单, 直接取

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

---

*求最小值用 (随机) 梯度下降法, 求最大值用 (随机) 梯度上升法, 这种基于梯度的方法通常统称为 (随机) 梯度下降法. 这里强调“上升”是想提醒大家这是在求最大值.

即把 $\sum_{j=2}^{l^w} \frac{\partial \mathcal{L}(w,j)}{\partial \mathbf{x}_w}$ 贡献到 $Context(w)$ 中每一个词的词向量上. 这个应该很好理解, 既然 $\mathbf{x}_w$ 本身就是 $Context(w)$ 中各词词向量的累加, 求完梯度后当然也应该将其贡献到每个分量上去.

注 4.3 当然, 读者这里需要考虑的是: 采用**平均**贡献会不会更合理？即使用公式

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

其中 $|Context(w)|$ 表示 $Context(w)$ 中词的个数.

NLP基础_词向量_Word2Vec_CBOW

- CBOW扩展：

下面以样本 $(Context(w), w)$ 为例, 给出 CBOW 模型中采用随机梯度上升法更新各参数的伪代码.

1. $\mathbf{e} = 0$.
2. $\mathbf{x}_w = \sum_{u \in Context(w)} \mathbf{v}(u)$.
3. **FOR** $j = 2 : l^w$ **DO**
   {
   3.1 $q = \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)$
   3.2 $g = \eta(1 - d_j^w - q)$
   3.3 $\mathbf{e} := \mathbf{e} + g\theta_{j-1}^w$
   3.4 $\theta_{j-1}^w := \theta_{j-1}^w + g\mathbf{x}_w$
   }
4. **FOR** $u \in Context(w)$ **DO**
   {
   $\mathbf{v}(u) := \mathbf{v}(u) + \mathbf{e}$
   }

注意, 步 3.3 和步 3.4 不能交换次序, 即 θ_{j-1}^w 应等贡献到 $\mathbf{e}$ 后再做更新.

注 4.4 结合上面的伪代码, 简单给出其与 *word2vec* 源码中的对应关系如下: *syn0* 对应 $\mathbf{v}(\cdot)$, *syn1* 对应 θ_{j-1}^w, *neu1* 对应 $\mathbf{x}_w$, *neu1e* 对应 $\mathbf{e}$.

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram扩展：

1. **输入层**：只含当前样本的中心词 w 的词向量 $\mathbf{v}(w) \in \mathbb{R}^m$.

2. **投影层**：这是个恒等投影，把 $\mathbf{v}(w)$ 投影到 $\mathbf{v}(w)$. 因此，**这个投影层其实是多余的**，这里之所以保留投影层主要是方便和 CBOW 模型的网络结构做对比.

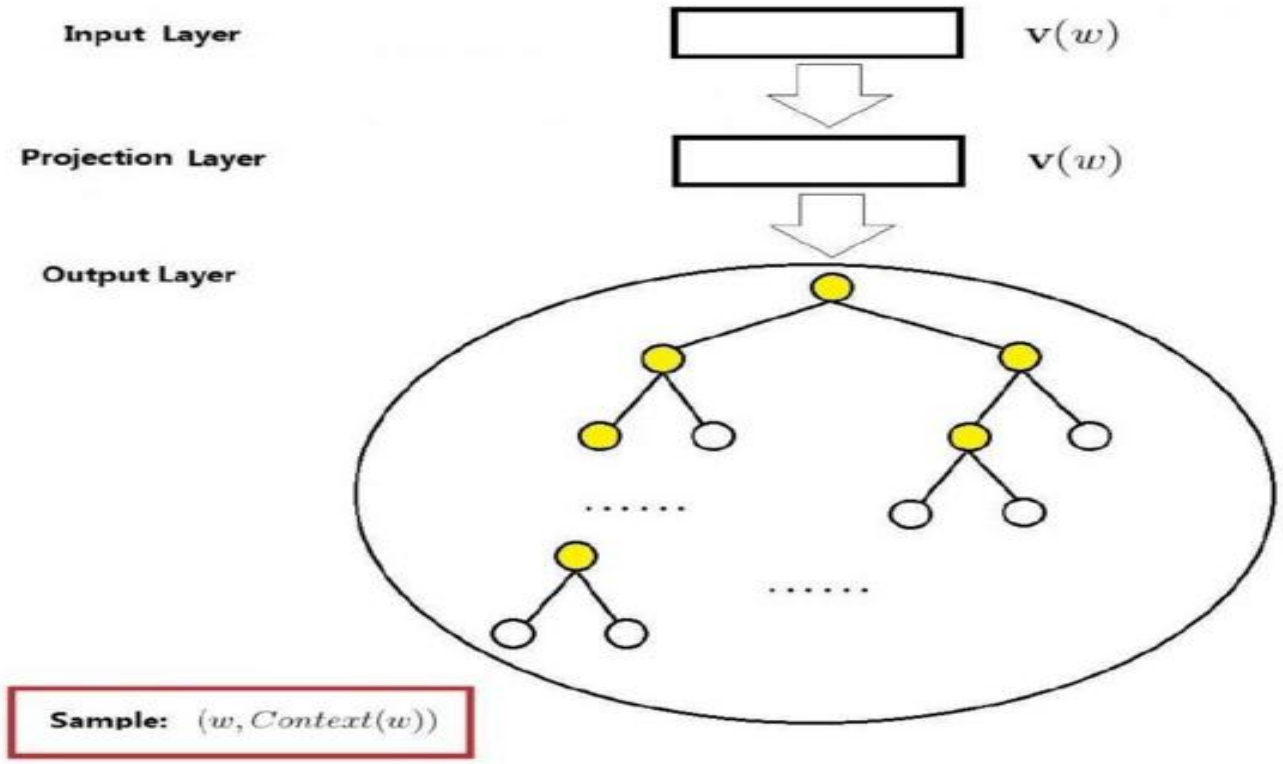


图 12 Skip-gram 模型的网络结构示意图

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram扩展：

3. **输出层**: 和 CBOW 模型一样, 输出层也是一棵 Huffman 树.

§4.2.2 梯度计算

对于 Skip-gram 模型, 已知的是当前词 w, 需要对其上下文 $Context(w)$ 中的词进行预测, 因此目标函数应该形如 (4.2), 且关键是条件概率函数 $p(Context(w)|w)$ 的构造, Skip-gram 模型中将其定义为

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

上式中的 $p(u|w)$ 可按照上小节介绍的 Hierarchical Softmax 思想, 类似于 (4.3) 地写为

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

其中

$$p(d_j^u|\mathbf{v}(w), \theta_{j-1}^u) = [\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{d_j^u}. \tag{4.6}$$

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram扩展：

将 (4.6) 依次代回, 可得对数似然函数 (4.2) 的具体表达式

$$\begin{aligned}\mathcal{L} &= \sum_{w\in\mathcal{C}} \log \prod_{u\in Context(w)} \prod_{j=2}^{l^u} \{[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1-\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{d_j^u}\} \\ &= \sum_{w\in\mathcal{C}} \sum_{u\in Context(w)} \sum_{j=2}^{l^u} \{(1-d_j^u)\cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1-\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]\}.\end{aligned} \tag{4.7}$$

同样, 为下面梯度推导方便起见, 将三重求和符号下花括号里的内容简记为 $\mathcal{L}(w,u,j)$, 即

$$\mathcal{L}(w,u,j) = (1-d_j^u)\cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1-\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)].$$

至此, 已经推导出对数似然函数的表达式 (4.7), 这就是 Skip-gram 模型的目标函数. 接下来同样利用**随机梯度上升法**对其进行优化, 关键是要给出两类梯度.

首先考虑 $\mathcal{L}(w,u,j)$ 关于 θ_{j-1}^u 的梯度计算 (与 CBOW 模型对应部分的推导完全类似).

$$\begin{aligned}\frac{\partial \mathcal{L}(w,u,j)}{\partial \theta_{j-1}^u} &= \frac{\partial}{\partial \theta_{j-1}^u}\{(1-d_j^u)\cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1-\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]\} \\ &= (1-d_j^u)[1-\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]\mathbf{v}(w) - d_j^u \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)\mathbf{v}(w) \quad \text{(利用 (2.1) 式)} \\ &= \{(1-d_j^u)[1-\sigma(\mathbf{v}(w)^\top \mathbf{v}_{j-1}^u)] - d_j^u \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)\}\mathbf{v}(w) \quad \text{(合并)} \\ &= [1-d_j^u - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]\mathbf{v}(w).\end{aligned}$$

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram扩展：

于是，θ_{j-1}^u 的更新公式可写为

$$\theta_{j-1}^u := \theta_{j-1}^u + \eta \left[1 - d_j^u - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)\right] \mathbf{v}(w).$$

接下来考虑 $\mathcal{L}(w, u, j)$ 关于 $\mathbf{v}(w)$ 的梯度. 同样利用 $\mathcal{L}(w, u, j)$ 中 $\mathbf{v}(w)$ 和 θ_{j-1}^w 的**对称性**, 有

$$\frac{\partial \mathcal{L}(w, u, j)}{\partial \mathbf{v}(w)} = \left[1 - d_j^u - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)\right] \theta_{j-1}^u.$$

于是，$\mathbf{v}(w)$ 的更新公式可写为

$$\mathbf{v}(w) := \mathbf{v}(w) + \eta \sum_{u \in Context(w)} \sum_{j=2}^{l^u} \frac{\partial \mathcal{L}(w, u, j)}{\partial \mathbf{v}(w)}.$$

下面以样本 $(w, Context(w))$ 为例, 给出 Skip-gram 模型中采用随机梯度上升法更新各参数的伪代码.

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram扩展：

$\mathbf{e} = \mathbf{0}$

FOR $u \in Context(w)$ **DO**

{

  FOR $j = 2 : l^u$ **DO**

  {

    1. $q = \sigma(\mathbf{v}(w)^\top \theta^u_{j-1})$
    2. $g = \eta(1 - d^u_j - q)$
    3. $\mathbf{e} := \mathbf{e} + g\theta^u_{j-1}$
    4. $\theta^u_{j-1} := \theta^u_{j-1} + g\mathbf{v}(w)$

  }

}

$\mathbf{v}(w) := \mathbf{v}(w) + \mathbf{e}$

但是, word2vec 源码中, 并不是等 $Context(w)$ 中的所有词都处理完后才刷新 $\mathbf{v}(w)$, 而是, 每处理完 $Context(w)$ 中的一个词 u, 就及时刷新一次 $\mathbf{v}(w)$, 具体为

NLP基础_词向量_Word2Vec_Skip-gram

- Skip-gram扩展：

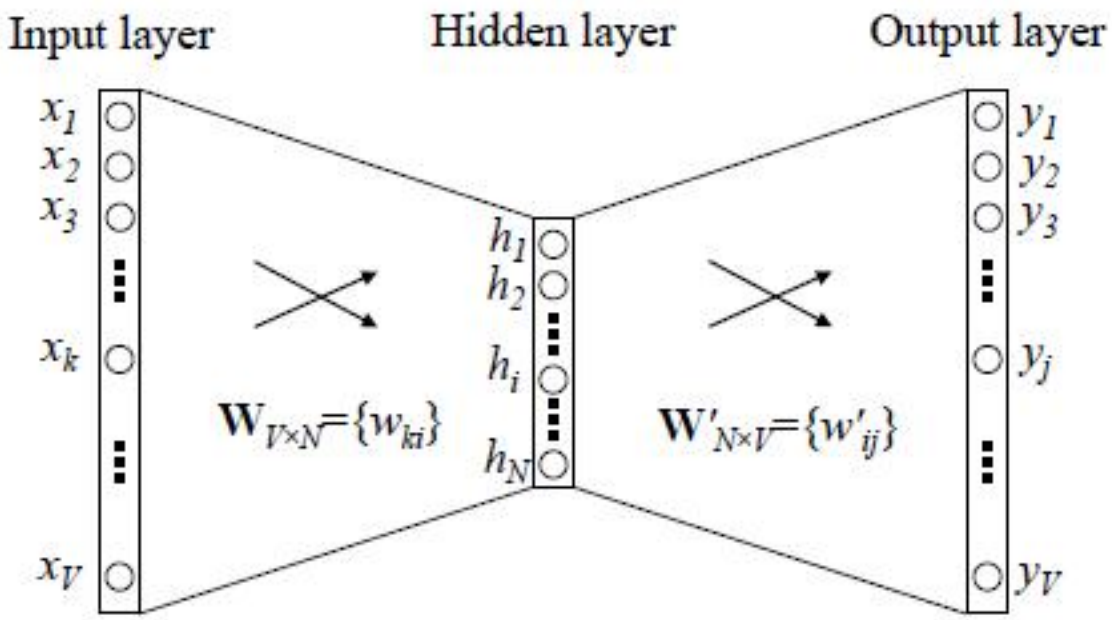
```
FOR u ∈ Context(w) DO
{
    e = 0
    FOR j = 2 : l^u DO
    {
        1. q = σ(v(w)^⊤ θ^u_{j-1})
        2. g = η(1 − d^u_j − q)
        3. e := e + gθ^u_{j-1}
        4. θ^u_{j-1} := θ^u_{j-1} + gv(w)
    }
    v(w) := v(w) + e
}
```

同样, 需要注意的是, 循环体内的步 3 和步 4 不能交换次序, 即 θ^u_{j-1} 要等贡献到 $\mathbf{e}$ 后才更新.

注 4.5 结合上面的伪代码, 简单给出其与 *word2vec* 源码中的对应关系如下: *syn0* 对应 $\mathbf{v}(\cdot)$, *syn1* 对应 θ^u_{j-1}, *neu1e* 对应 $\mathbf{e}$.

NLP基础_词向量_Word2Vec_Hierarchical Softmax

- 输入层到隐藏层之间的映射，没有采用神经网络的线性转换加激活函数的方式，而且采用简单的均值的方式来实现；
- 从隐藏层到输出的softmax层之间的计算量采用霍夫曼树进行了改进。
- 参考：https://www.cnblogs.com/pinard/p/7243513.html

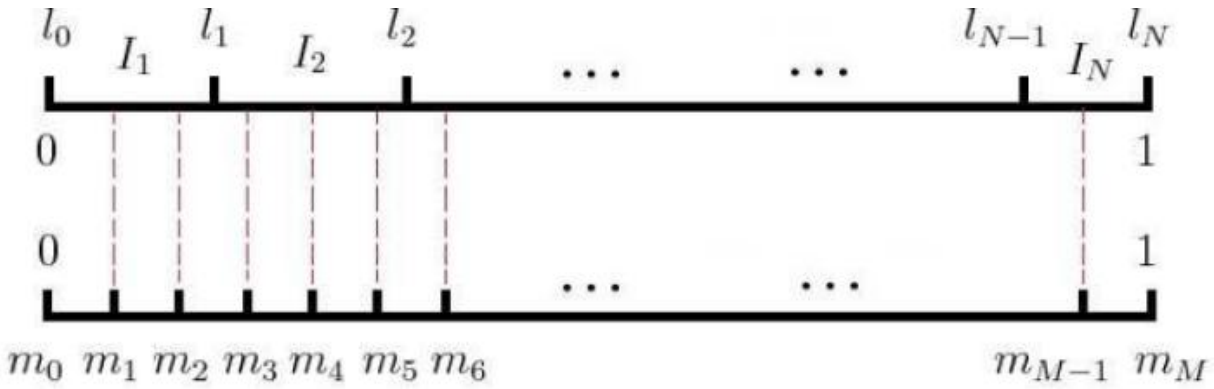


NLP基础_词向量_Word2Vec_Negative Sampling

$$
\begin{aligned}
\text{minimize } J &= -\log P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c | \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}
$$

NLP基础_词向量_Word2Vec_Negative Sampling

- Hierarchical Softmax有一个非常大的缺点，就是**针对很生僻的词，在霍夫曼树中，这个词所在的位置非常低，也就是路径非常长，计算量就比较大**；
- Negative Sampling(负采样<训练过程中的>)将实际类别当做正例，从所有的非实际类别中随机的抽取K个类别作为负例(不是等概率)，然后进行线性转换，最后进行损失函数以及反向传播更新参数。
- 参考：https://www.cnblogs.com/pinard/p/7249903.html



THANKS!