Word2Vec

1.

什么是Word2Vec和Embeddings?

Word2Vec是从大量文本语料中以无监督的方式学习语义知识的一种模型，它被大量地用在自然语言处理（NLP）中。那么它是如何帮助我们做自然语言处理呢？Word2Vec其实就是通过学习文本来用词向量的方式表征词的语义信息，即通过一个嵌入空间使得语义上相似的单词在该空间内距离很近。Embedding其实就是一个映射，将单词从原先所属的空间映射到新的多维空间中，也就是把原先词所在空间嵌入到一个新的空间中去。

我们从直观角度上来理解一下，cat这个单词和kitten属于语义上很相近的词，而dog和kitten则不是那么相近，iphone这个单词和kitten的语义就差的更远了。通过对词汇表中单词进行这种数值表示方式的学习（也就是将单词转换为词向量），能够让我们基于这样的数值进行向量化的操作从而得到一些有趣的结论。比如说，如果我们对词向量kitten、cat以及dog执行这样的操作：kitten - cat + dog，那么最终得到的嵌入向量（embedded vector）将与puppy这个词向量十分相近。

首先，考虑如何表示一个word

英语大概有约1300万单词。词向量就是将单词映射到词空间的一个点。

（注意，词向量在英文里有两个可以互相替换使用的说法： word embeddings和word vectors）

为什么使用词向量？一个最直观的原因是，我们可以找到一个N维（N小于1300万）的空间，足够编码我们所有的单词。

每一个维度可以编码一些含义，例如语义空间可以编码时态、单复数和性别。

one-hot向量（one-hot vector）：

one-hot向量就是利用一个 $R^{|V|\times 1}$ 向量来表示单词。

$|V|$ 是词汇表中单词的数量。

一个单词在英文词汇表中的索引位置是多少，那么相对应的那一行元素就是1，其他元素都是0。

$$w^{Aardvark} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{vmatrix} \qquad w^{zebra} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{vmatrix}$$

产生的问题是，任意两个单词的内积为0，无法区分他们在之间具有的不同的相似性

$$(w^{hotel})^T w^{motel} = 0$$

$$v^c = \begin{vmatrix} 0.2 \\ 0.4 \\ 1.3 \\ \vdots \\ 0.9 \end{vmatrix}$$

那么如何得到这样的一个向量呢?

矩阵奇异值分解

DeepLearning

两种方法：SkipGram/CBow

以下面这个句子为例：｛“The”，“cat”，‘ over”，“the’，“puddle”｝。

通过这些上下文单词预测中间缺少的单词“jump”，这种模型我们就成为连续袋模型。

1. 将输入句子中的单词生成one-hot向量（本文第二部分介绍过什么是one-hot向量）：

$$\left(x^{c-m},\ldots,x^{c-1},x^{c+1},\ldots,x^{c+m}\in R^{|V|}\right)$$

2. 通过将one-hot向量和输入矩阵 V 相乘，得到输入单词的词向量：

$$\nu_{c-m}=Vx^{c-m},\nu_{c-m+1}=Vx^{c-m+1},\ldots,\nu_{c+m}=Vx^{c+m}$$

3. 对上述词向量求平均得到：$\bar{\nu}=\frac{\nu_{c-m}+\nu_{c-m+1}+\cdots+\nu_{c+m}}{2m}\in R^n$

4. 上面的平均词向量与输出矩阵进行点乘运算，得到得分向量（score vector）

MathJax maximum macro substitution count exceeded; is there a recursive macro c

，我们知道，两个向量约相似，点乘得到的分数越高，因此将会使相似的词互相靠近，从而得到较高的分数。

5. 将分数转换成概率 $\bar{y}=softmax(z)\in R^{|V|}$ 。' softmax' 就是对$ \bar {y} $做如下运算 $\frac{e^{\bar{y}_i}}{\sum_{k=1}^{|V|}e^{\bar{y}_k}}$，使其每一个元素在[0,1]范围内，且和为1。

我们希望 $\bar{y}$ 与真实的 y 匹配，也就是真实的 y 的one-hot向量。

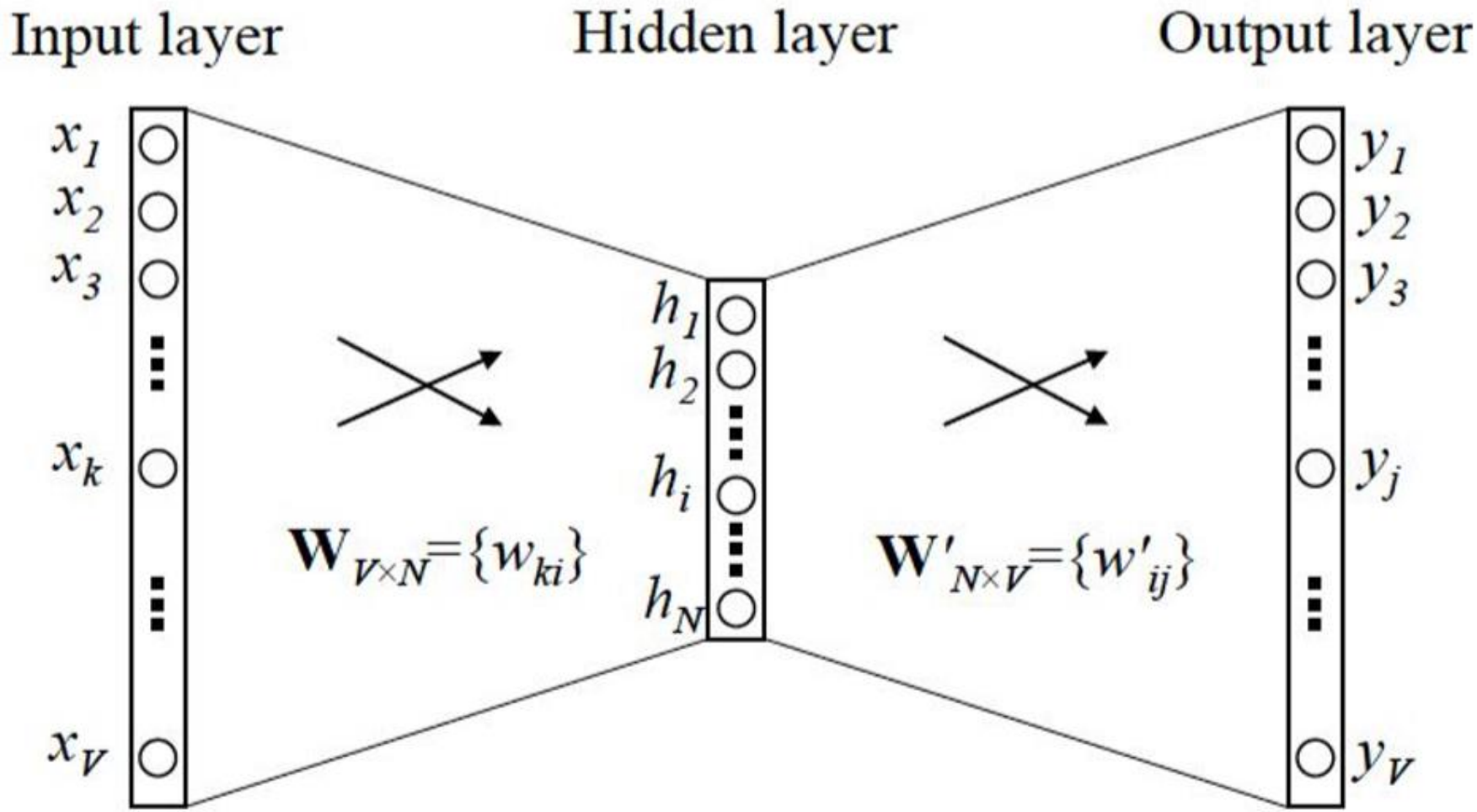
首先，为了计算预测值的准确性，我们先定义一个测量函数，这里，我们选择一个非常流行的测量函数：交叉熵（cross entropy）

$$H(\bar{y}, y) = -\sum_{j=1}^{|V|} y_j log \bar{y}_j$$

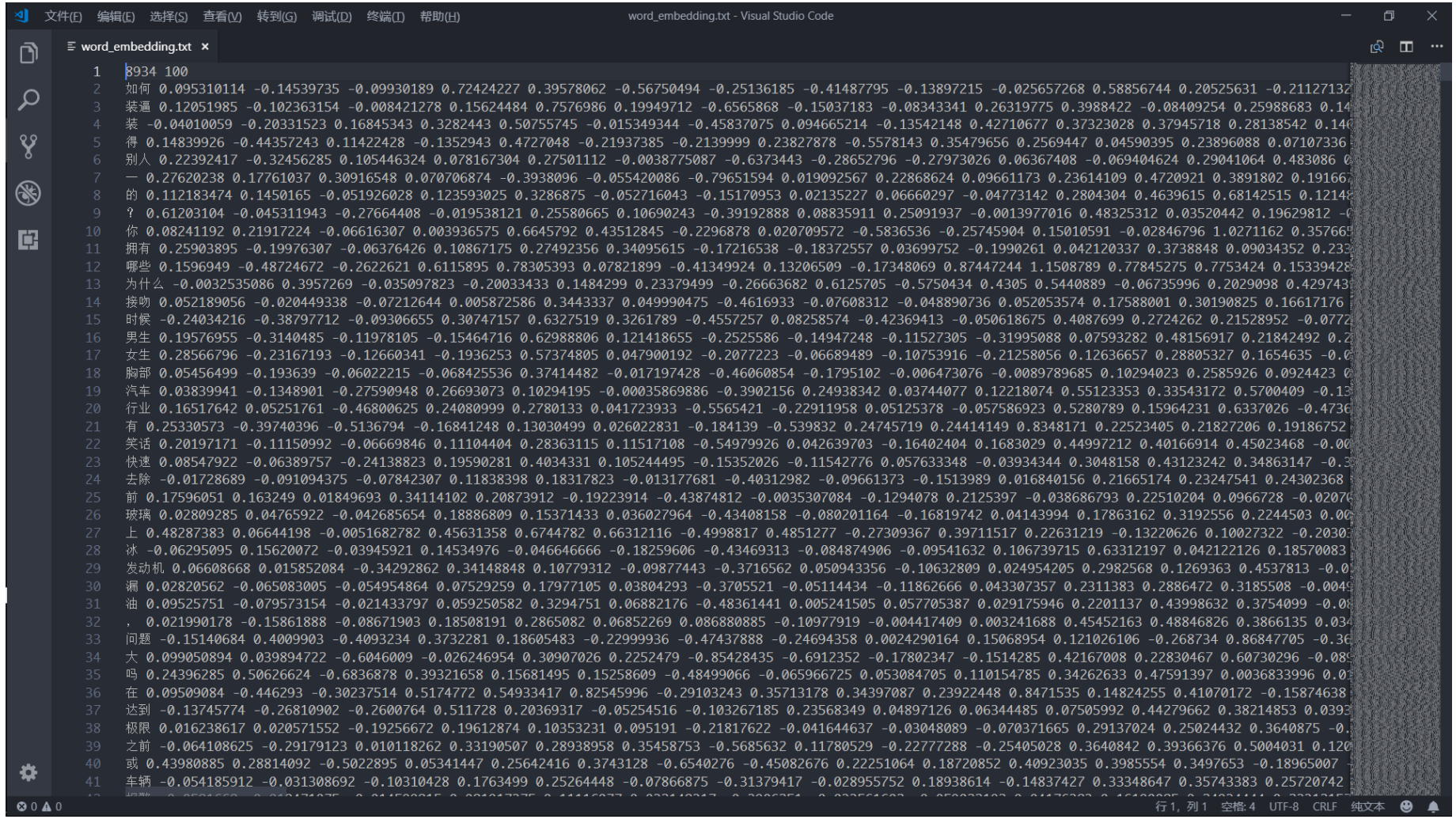
Skip-Gram模型与连续袋模型正好相反。

连续袋模型是根据周围的词来预测中心的词，而Skip-Gram模型是根据中心的词预测周围的词。

Skip-gram Model



Demo的地址
https://github.com/Adoni/word2vec_pytorch/blob/master/model.py



负采样
https://www.cnblogs.com/pinard/p/7249903.html

Google 官方的word2Vec项目

https://github.com/tmikolov/word2vec