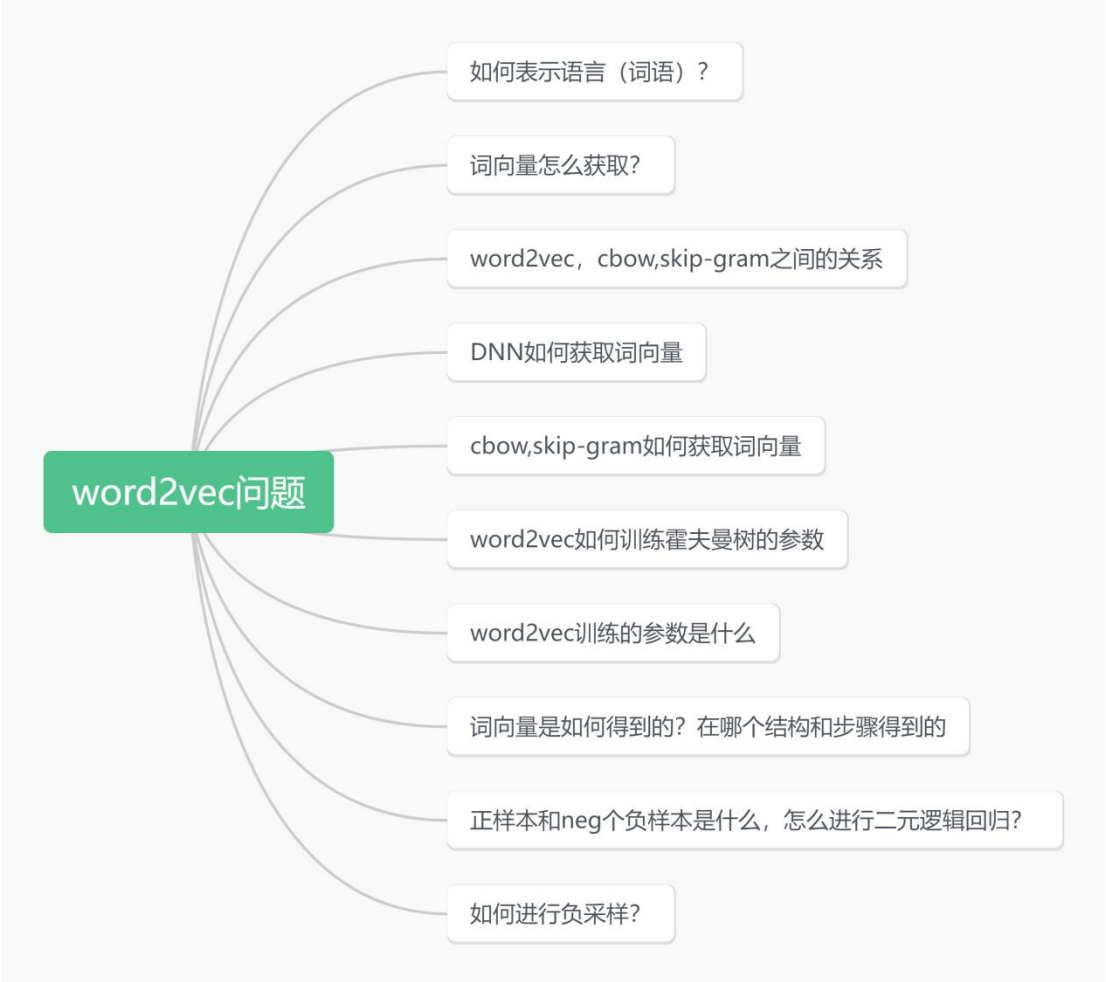


语言模型与词向量表达方式

1 语言模型

语言模型（language model）就是建模按照语言某种顺序排列的概率。把一段自然语言文本看作一段离散的时间序列。假设一段长度为 T 的文本中的词依次为 w1,w2,...,wT，那么在离散的时间序列中 wt，可看作在时间步 t（time step）的输出或标签。

给定一个长度为 T 的词的序列，语言模型将计算该序列的概率：

$$P(w1,w2,...,wT)$$

在机器翻译中，如果对英文“you go first”逐词翻译成中文的话，可能得到“你走先”“你先走”等排列方式的文本序列。如果语言模型判断出“你先走”的概率大于其他排列方式的文本序列的概率，我们就可以把“you go first”翻译成“你先走”。

语言模型可用于提升语音识别和机器翻译的性能。例如，在机器翻译中，如果对英文“you go first”逐词翻译成中文的话，可能得到“你走先”“你先走”等排列方式的文本序列。如果语言模型判断出“你先走”的概率大于其他排列方式的

文本序列的概率，我们就可以把“you go first”翻译成“你先走”。在语音识别中，给定一段“厨房里食油用完了”的语音，有可能会输出“厨房里食油用完了”和“厨房里石油用完了”这两个读音完全一样的文本序列。如果语言模型判断出前者的概率大于后者的概率，我们就可以根据相同读音的语音输出“厨房里食油用完了”的文本序列。

2 词向量表达方式

前面多次提到，学习神经网络模型的核心就是抓住数据流，即数据的形式和维度，然后思考模型是怎么挖掘变换数据中的特征。所以深度学习获取数据集的一大关键就是如何将现实中的物理信息转化成计算机可识别的数字信息，之前学习的 CV 模块，输入一般是图片，而计算机“看”到的图片是三通道的 RGB 数字（ channel, height, weight），后续网络处理也无非就是将这些数字进行各种变换挖掘特征。那么对于文字和语言该如何处理呢？该怎么将人类抽象出来的文字符号表示为计算机可识别的数字呢？这就是自然语音处理(NLP)的第一步，将词语转变成数字。

在深度学习中，将人类抽象出来的自然语言的基本单元——词语转变成计算机可识别处理的数字向量，便是词向量处理，也叫词嵌入（Word Embedding）。词向量，就和表示图像的多维张量一样，可以表征词语的特征，并从中挖掘变换学习特征用于完成不同的任务。

那么该如何将词语表示成向量呢？根据词向量我们可以判断表示的是特定的词，一般有两类表示方法，一类是基于词本身的特征（比如最简单的独热编码中即使用词在词汇表中的位置来表征），另一类是基于和其它词的关系来表示。近年来，词向量已逐渐成为自然语言处理的基础知识。注意区分词向量的维度和词汇表的长度。接下来详细介绍几种常见的词向量表示方法。

2.1 独热编码 One-Hot Encoding

一种最简单的词向量方式是独热编码 one-hot representation，首先定义一个含有 n 个词语的词汇表（想象成不同的字典），然后每个单词定义一个向量 1*n（该向量长度即为词汇表的长度），单词在词汇表的位置 a 对应的向量位置为 1，其余位置的全为 0。

举个简单的例子，假如词汇表中有 5 个词，第 3 个词表示“你好”这个词，那么该词对应的 one-hot 编码即为 00100（第 3 个位置为 1，其余为 0）。

One-hot Representation 形式很简洁，使用起来也很简单，只需要给每个单词分配一个编号，然后在对应的向量位置记为 1 就好了。但也存在 2 个很大的问题：

（1）维度灾难：词汇表一般都非常大，假如词汇表有 10k 个词，那么一个词向量的长度就需要达到 10k，而其中却仅有一个位置是 1，其余全是 0，内存占用过高且表达效率很低。

（2）无法体现出**词与词之间的关系**：比如“爱”和“喜欢”这两个词，它们的意思是相近的，但基于 one-hot 编码后的结果取决于它们在词汇表中的位置，不能很好地刻画词与词之间的相似性。因为任何两个 one-hot 向量之间的内积都是 0，很难区分他们之间的差别。

2.2. 基于统计方法

2.2.1. 共现 co-occurrence 矩阵

这种方法就是利用周围的词的关系来表示某个词，比如小明最好的朋友是小红，那就用“小明最好的朋友”来表示“小红”。通过从大量的语料文本中构建一个共现矩阵来定义词向量，具体来说，通过统计一个事先指定大小的窗口内的词语共现次数，将其作为当前词语的词向量。

例如，有语料如下：

I like deep learning.

I like NLP.

I enjoy flying.

设窗口大小为 1，三句话中，观察 i 前后各 1 个单词，发现“like”出现了 2 次，enjoy 出现了 1 次，所以 i 就可以表示为 02100000。以此类推，其余则其共现矩阵如下：

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

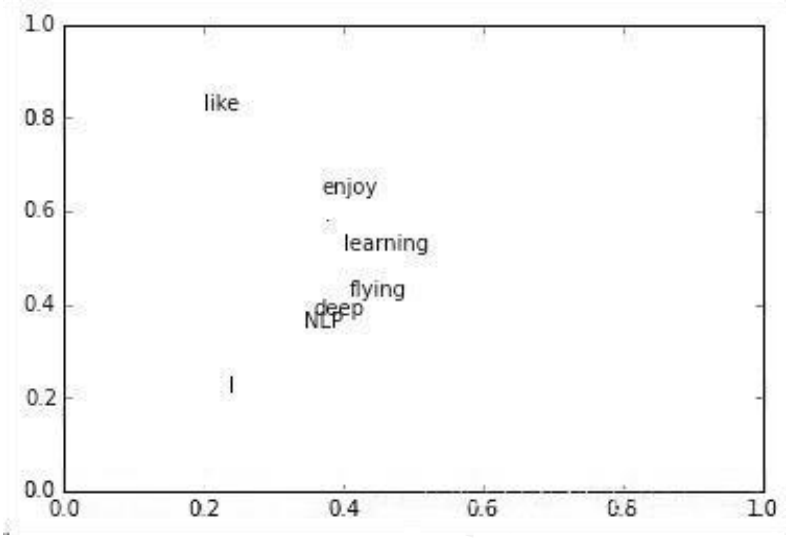
矩阵定义的词向量在一定程度上缓解了 one-hot 向量相似度为 0 的问题，但没有解决数据稀疏性和维度灾难的问题。

2.2.2. SVD（奇异值分解）

基于 co-occurrence 矩阵得到的离散词向量仍然存在着高维和稀疏性的问题，一个自然而然的解决思路是对原始词向量进行降维，从而得到一个稠密的连续词向量。 对共现矩阵进行 SVD 分解，得到正交矩阵 U，对 U 进行归一化得到矩阵如下：

I	0.24	0.21	0.10	0.38	-0.18	-0.18	-0.42	-0.06
like	0.20	0.82	-0.17	0.31	0.18	-0.23	0.13	0.14
enjoy	0.37	0.64	0.16	0.00	-0.58	0.64	0.00	-0.31
deep	0.36	0.38	0.35	-0.07	0.45	0.08	0.55	-0.47
learning	0.40	0.52	-0.50	-0.43	0.35	0.16	-0.47	-0.40
NLP	0.35	0.35	-0.22	-0.19	0.13	0.49	0.21	0.66
flying	0.41	0.42	-0.40	-0.38	-0.51	-0.43	0.42	-0.12

SVD 得到了 word 的**稠密（dense）矩阵**，该矩阵具有很多良好的性质：**语义相近的词在向量空间相近**，甚至**可以一定程度反映 word 间的线性关系**。



2.3 基于语言模型(language model)：词向量/词嵌入 Word Embedding

Distributed representation 可以解决 One hot representation 的问题，它最早是 Hinton 于 1986 年提出的。它的思路是通过训练，将每个词都映射到一个较短的词向量上来，能够体现词与词之间的关系，所有的这些词向量就构成了向量空间，进而可以用普通的统计学的方法来研究词与词之间的关系。这就是 word embedding，即指的是将词转化成一种分布式表示，又称词向量。分布式表示将词表示成一个定长的连续的稠密向量，这个较短的词向量维度是多大呢？这个一般需要我们在训练时自己来指定。

那么应该如何设计这种方法呢？最方便的途径是设计一个**可学习的权重矩阵 W（这个矩阵是致密的）**，将 one-hot 向量与这个矩阵进行点乘，即得到新的词向量表示结果，使得**意思相近的词有相近的表示结果**。下面章节会详细介绍如何生成词向量。

2.3.1 降维

假设词汇表共有 5 个词语，“爱” 和 “喜欢” 这两个词经过 one-hot 后分别表示为 10000 （1*5）和 00001，权重矩阵设计如下（5*3）：

[w00, w01, w02
  w10, w11, w12
  w20, w21, w22
  w30, w31, w32
  w40, w41, w42]

那么两个词点乘后的结果分别是 [w00, w01, w02] 和 [w40, w41, w42]，在网络学习过程中（这两个词后面通常都是接主语，如“你”，“他”等，或者在翻译场景，它们被翻译的目标意思也相近，它们要学习的目标一致或相近），权重矩阵的参数会不断进行更新，从而使得 [w00, w01, w02] 和 [w40, w41, w42] 的值越来越接近。

其实，可以将这种方式看作是一个 **lookup table**：对于每个 word，进行 word embedding 就相当于一个 lookup 操作，在表中查出一个对应结果。

需要注意的是，要获取的词向量是这里要学习的参数矩阵，另一方面，对于以上这个例子，我们还把向量的维度从 5 维压缩到了 3 维 [w00, w01, w02] 和 [w40, w41, w42]。因此，word embedding 还可以起到**降维**的效果。

A* B=C

在上述公式中，经过训练，原来的独热编码 10 个元素的 A 矩阵 2*5，变成参数矩阵 B 中对应行的 2*3，直观上大小缩小了近一半。

更进一步地，假设一个 100Wx10W 的原始矩阵，训练得到 10Wx20 的参数矩阵，降低 10w/20=5000 倍。

总结：在某种程度上，Embedding 层实现了降维的作用，核心就是把词向量的长度从词汇表的长度降到了人为设置的一个常数。

再延伸一下，Embedding 本质是用一个低维稠密的向量表示一个对象，这里的对象可以是一个词（Word2vec），也可以是一个物品（Item2vec），亦或是网络关系中的节点（Graph Embedding）。

在 Pytorch 框架下，可以使用 *torch.nn.Embedding* 来实现 word embedding：

```
class Embeddings(nn.Module):
    def __init__(self, d_model, vocab):
        super(Embeddings, self).__init__()
```

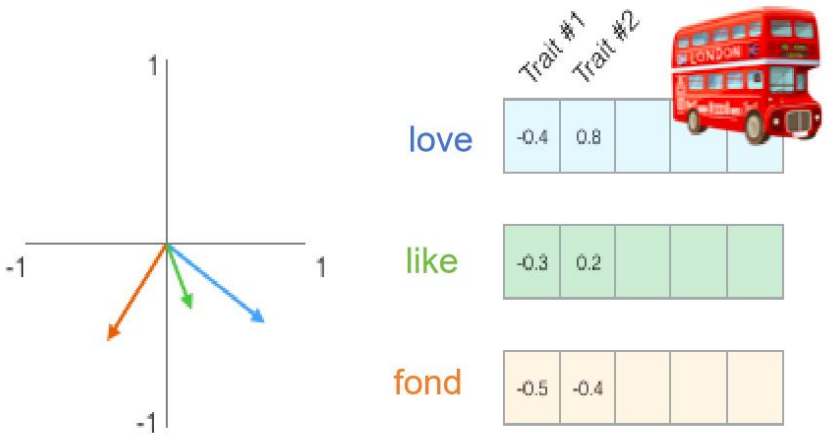
```
        self.ebd = nn.Embedding(voca_size, d_model)
        self.d_model = d_model

    def forward(self, x):
        return self.ebd t(x) * math.sqrt(self.d_model)
```

其中，*voca*_size 代表词汇表中的单词量，one-hot 编码后词向量的长度就是这个值；*d_model* 代表权重矩阵的列数，通常为 512，就是要将词向量的维度从 *voca*_size 编码到 *d_model*。

2.3.2 词与词之间关系

前面讲到 one-hot 编码词向量不能表示词与词之间的关系，为什么呢？一般来说，数学中使用两个向量之间的距离或者余弦来表示相似性。而 one-hot 编码因为每个词只在词汇表位置处为 1，每个词向量的内积都是 0，那么任意两个向量的“距离”或者说“相似性”都为 0.

以下图为例，可以看到 love 和 like 的词向量余弦相似度是比较近的。



简单总结一下，分布式克服了 one-hot 表示的缺点，有以下 2 个有点:

(1) 可以表征词与词之间的相似关系：利用词之间的余弦来表征“距离”或“相似度”，这对很多自然语言处理的任务非常有帮助。

(2) 维度更低，包含信息更多： 相比 one-hot 的向量长度为词汇表长度，分布式词向量维度为人为设置的常数，维度更低；而且相比 one-hot 仅含有词在词汇表的位置这一信息，词向量能够包含更多信息，并且每一维都有特定的含义。在采用 one-hot 特征时，可以对特征向量进行删减，词向量则不能。

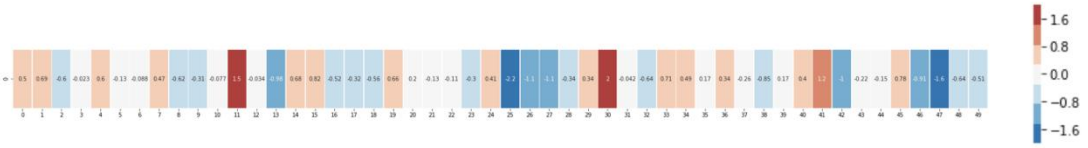
3 词向量可视化

这一小节主要是从可视化的角度来展示一下词向量是如何表征词与词之间的关系的。

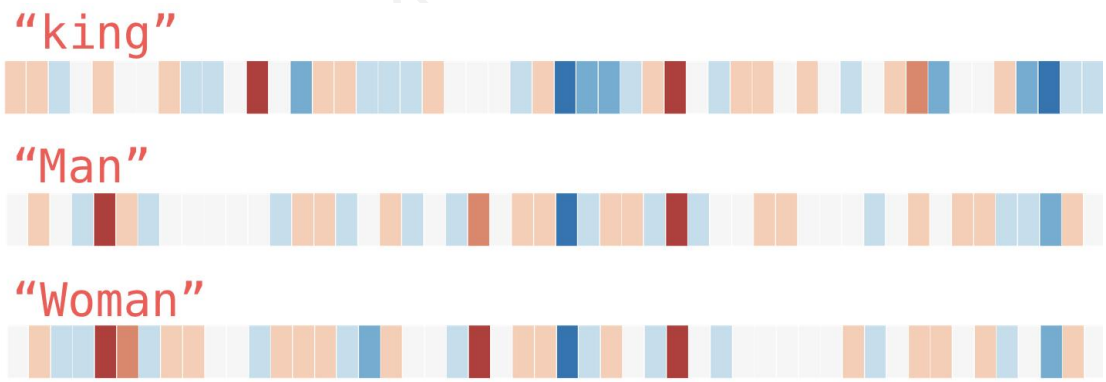
以单词 King 的词向量（通过维基百科语料训练的 Glove 向量）为例，如下：

[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 ,
-0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 ,
0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344
, -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

根据值对单元格进行颜色编码（接近 2 则为红色，接近 0 则为白色，接近-2 则为蓝色）：

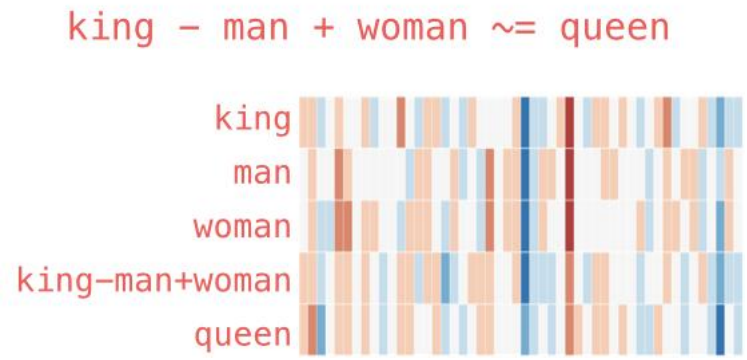


忽略数字仅考虑颜色，将“king”与其它单词进行比较：



可以看到，man 和 woman 颜色相似的维度会比它们任一和 King 相似的维度要多，也就是说这两个词在更多的特征维度是相似的，“距离”更近。

再来看一个经典例子，



可以看到，“king-man+woman”的词向量和“queen”的词向量非常相似。

4 词嵌入用做迁移学习

如果对于一个命名实体识别任务，只有一个很小的标记的训练集，训练集里可能没有某些词，但是如果有一个已经学好的词嵌入，就可以用迁移学习，把从互联网上免费获得的大量的无标签文本中学习到的知识迁移到一个命名实体识别任务中。

如果从某一任务 A 迁移到某个任务 B，只有 A 中有大量数据，而 B 中数据少时，迁移的过程才有用。

用词嵌入做迁移学习的步骤：

（1）先从大量的文本集中学习词嵌入，或者可以下载网上预训练好的词嵌入模型。

（2）把这些词嵌入模型迁移到新的只有少量标注训练集的任务中。

（3）考虑是否微调，用新的数据调整词嵌入。当在新的任务上训练模型时，如命名实体识别任务上，只有少量的标记数据集上，可以自己选择要不要继续微调，用新的数据调整词嵌入。实际中，只有第二步中有很大的数据集时才会这样做，如果你标记的数据集不是很大，通常不建议在微调词嵌入上费力气。

5. 生成词向量的方式

前面讲了很多词向量的优势和特色，只要得到词汇表中的词对应的向量，就可以像图像处理那样得到了原始的输入，就可以进一步挖掘变换数据里面的特征，完成各种各样的任务。一个比较实用的场景是找同义词，得到词向量后，假如想找出与“爱”相似的词，建立好词向量后，对计算机来说，只要拿这个词的词向量跟其他词的词向量一一计算欧式距离或者 cos 距离，得到距离小于某个值的那些词，就是它的同义词。

那么该如何训练得到合适的词向量呢？

训练方法较多，word2vec 是其中一种。还要注意的是每个词在不同的语料

库和不同的训练方法下，得到的词向量可能是不一样的。由于是用向量表示，而且用较好的训练算法得到的词向量一般是有空间上的意义的。

语言模型生成词向量是通过训练神经网络语言模型 NNLM（neural network language model），词向量作为语言模型的附带产出。NNLM 背后的基本思想是对出现在上下文环境里的词进行预测，这种对上下文环境的预测本质上也是一种对共现统计特征的学习。 较著名的采用 neural network language model 生成词向量的方法有：Skip-gram、CBOW、LBL、NNLM、C&W、GloVe 等。接下来，以 word2vec 为例，讲解基于神经网络语言模型的词向量生成。

下节将详细介绍。

参考资料

[1] https://zh.d2l.ai/index.html
https://blog.csdn.net/qq_27586341/article/details/90146342