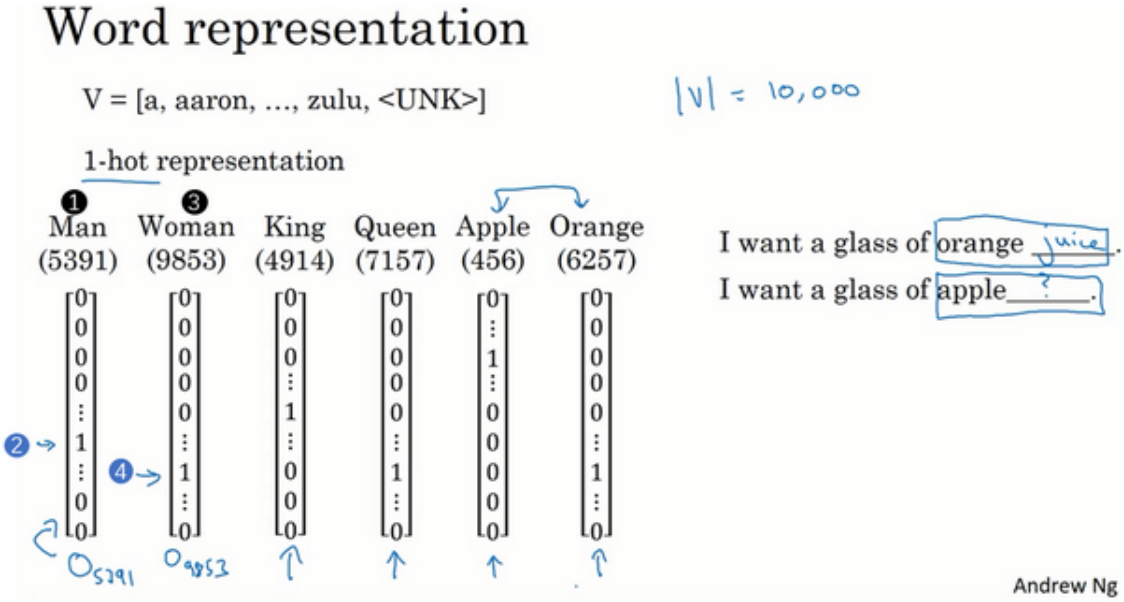
2.1 词汇表征（Word Representation）

上周我们学习了 **RNN**、**GRU** 单元和 **LSTM** 单元。本周你会看到我们如何把这些知识用到 **NLP** 上，用于自然语言处理，深度学习已经给这一领域带来了革命性的变革。其中一个很关键的概念就是词嵌入（word embeddings），这是语言表示的一种方式，可以让算法自动的理解一些类似的词，比如男人对女人，比如国王对王后，还有其他很多的例子。通过词嵌入的概念你就可以构建 **NLP** 应用了，即使你的模型标记的训练集相对较小。这周的最后我们会消除词嵌入的偏差，就是去除不想要的特性，或者学习算法有时会学到的其他类型的偏差。

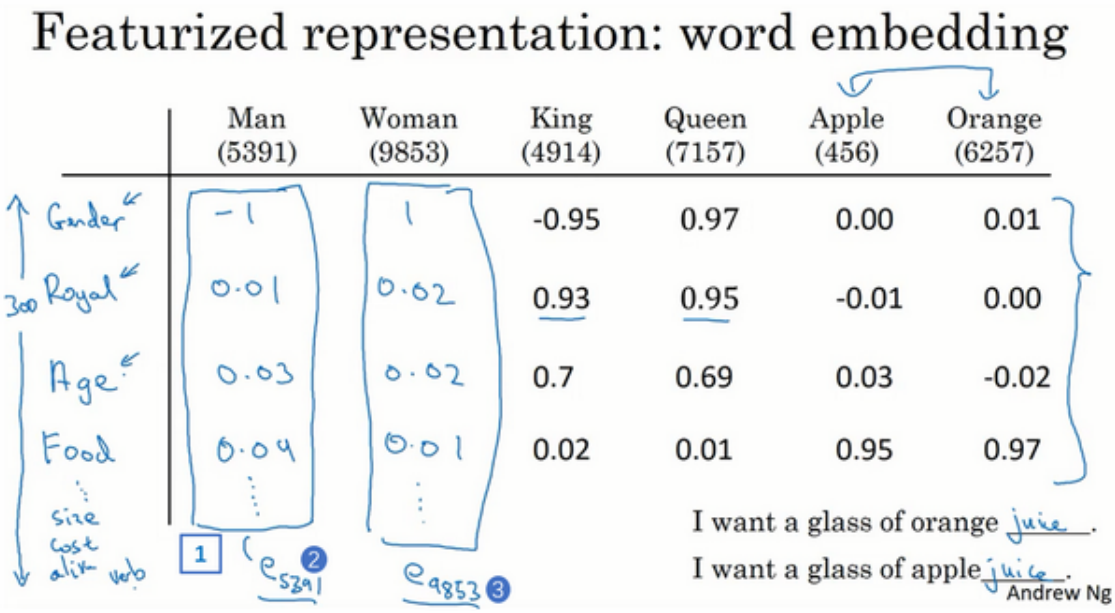


现在我们先开始讨论词汇表示，目前为止我们一直都是用词汇表来表示词，上周提到的词汇表，可能是 10000 个单词，我们一直用 **one-hot** 向量来表示词。比如如果 **man**（上图编号 1 所示）在词典里是第 5391 个，那么就可以表示成一个向量，只在第 5391 处为 1（上图编号 2 所示），我们用O_{5391}代表这个量，这里的O代表 **one-hot**。接下来，如果 **woman** 是编号 9853（上图编号 3 所示），那么就可以用O_{9853}来表示，这个向量只在 9853 处为 1（上图编号 4 所示），其他为 0，其他的词 **king**、**queen**、**apple**、**orange** 都可以这样表示出来这种表示方法的一大缺点就是它把每个词孤立起来，这样使得算法对相关词的泛化能力不强。

举个例子，假如你已经学习到了一个语言模型，当你看到**“I want a glass of orange ___”**，那么下一个词会是什么？很可能是 **juice**。即使你的学习算法已经学到了**“I want a glass of orange juice”**这样一个很可能的句子，但如果看到**“I want a glass of apple ___”**，因为算法不知道 **apple** 和 **orange** 的关系很接近，就像 **man** 和 **woman**，**king** 和 **queen** 一样。所以算法很

难从已经知道的 **orange juice** 是一个常见的东西，而明白 **apple juice** 也是很常见的东西或者说常见的句子。这是因为任何两个 **one-hot** 向量的内积都是 0，如果你取两个向量，比如 **king** 和 **queen**，然后计算它们的内积，结果就是 0。如果用 **apple** 和 **orange** 来计算它们的内积，结果也是 0。很难区分它们之间的差别，因为这些向量内积都是一样的，所以无法知道 **apple** 和 **orange** 要比 **king** 和 **orange**，或者 **queen** 和 **orange** 相似地多。

换一种表示方式会更好，如果我们不用 **one-hot** 表示，而是用特征化的表示来表示每个词，**man**，**woman**，**king**，**queen**，**apple**，**orange** 或者词典里的任何一个单词，我们学习这些词的特征或者数值。



举个例子，对于这些词，比如我们想知道这些词与 **Gender**（**性别**）的关系。假定男性的性别为-1，女性的性别为+1，那么 **man** 的性别值可能就是-1，而 **woman** 就是 1。最终根据经验 **king** 就是-0.95，**queen** 是+0.97，**apple** 和 **orange** 没有性别可言。

另一个特征可以是这些词有多 **Royal**（**高贵**），所以这些词，**man**，**woman** 和高贵没太关系，所以它们的特征值接近 0。而 **king** 和 **queen** 很高贵，**apple** 和 **orange** 跟高贵也没太大关系。

那么 **Age**（**年龄**）呢？**man** 和 **woman** 一般没有年龄的意思，也许 **man** 和 **woman** 隐含着成年人的意思，但也可能是介于 **young** 和 **old** 之间，所以它们（**man** 和 **woman**）的值也接近 0。而通常 **king** 和 **queen** 都是成年人，**apple** 和 **orange** 跟年龄更没什么关系了。

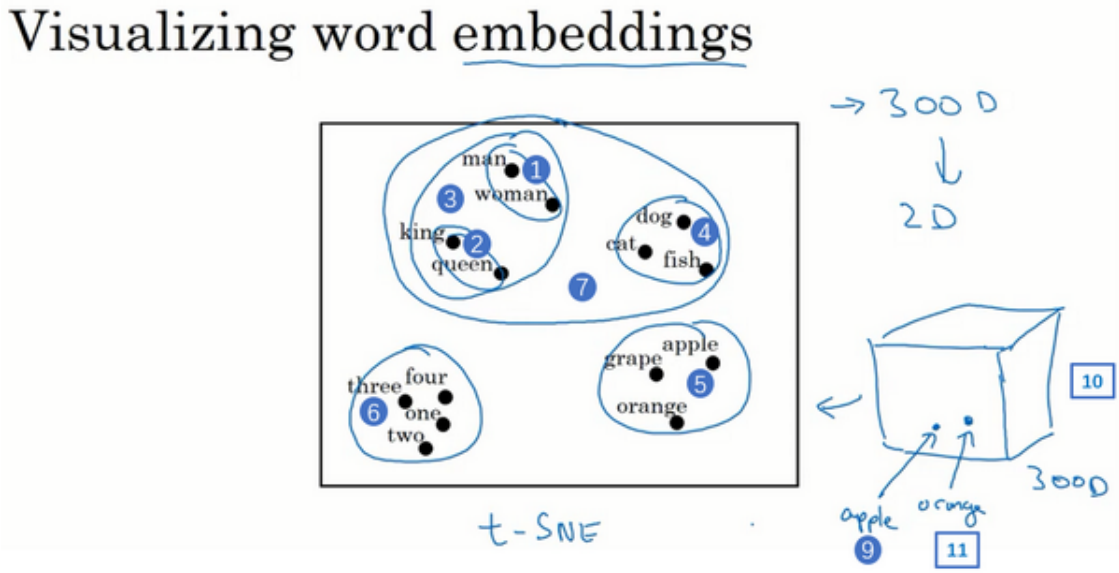
还有一个特征，这个词是否是 **Food**（**食物**），**man** 不是食物，**woman** 不是食物，**king** 和 **queen** 也不是，但 **apple** 和 **orange** 是食物。

当然还可以有很多的其他特征，从 **Size**（**尺寸大小**），**Cost**（**花费多少**），这个东西是不

是 **alive**（**活的**），是不是一个 **Action**（**动作**），或者是不是 **Noun**（**名词**）或者是不是 **Verb**（**动词**），还是其他的等等。

所以你可以想很多的特征，为了说明，我们假设有 300 个不同的特征，这样的话你就有了这一列数字（上图编号 1 所示），这里我只写了 4 个，实际上是 300 个数字，这样就组成了一个 300 维的向量来表示 **man** 这个词。接下来，我想用e_{5391}这个符号来表示，就像这样（上图编号 2 所示）。同样这个 300 维的向量，我用e_{9853}代表这个 300 维的向量用来表示 **woman** 这个词（上图编号 3 所示），这些其他的例子也一样。现在，如果用这种表示方法来表示 **apple** 和 **orange** 这些词，那么 **apple** 和 **orange** 的这种表示肯定会非常相似，可能有些特征不太一样，因为 **orange** 的颜色口味，**apple** 的颜色口味，或者其他的一些特征会不太一样，但总的来说 **apple** 和 **orange** 的大部分特征实际上都一样，或者说都有相似的值。这样对于已经知道 **orange juice** 的算法很大几率上也会明白 **apple juice** 这个东西，这样对于不同的单词算法会泛化的更好。

后面的几个视频，我们会找到一个学习词嵌入的方式，这里只是希望你能理解这种高维特征的表示能够比 **one-hot** 更好的表示不同的单词。而我们最终学习的特征不会像这里一样这么好理解，没有像第一个特征是性别，第二个特征是高贵，第三个特征是年龄等等这些，新的特征表示的东西肯定会更难搞清楚。尽管如此，接下来要学的特征表示方法却能使算法高效地发现 **apple** 和 **orange** 会比 **king** 和 **orange**，**queen** 和 **orange** 更加相似。



如果我们能够学习到一个 300 维的特征向量，或者说 300 维的词嵌入，通常我们可以做一件事，把这 300 维的数据嵌入到一个二维空间里，这样就可以可视化了。常用的可视化算法是 **t-SNE 算法**，来自于 **Laurens van der Maaten** 和 **Geoff Hinton** 的论文。如果观察这种

词嵌入的表示方法，你会发现 **man** 和 **woman** 这些词聚集在一块（上图编号 1 所示），**king** 和 **queen** 聚集在一块（上图编号 2 所示），这些都是人，也都聚集在一起（上图编号 3 所示）。动物都聚集在一起（上图编号 4 所示），水果也都聚集在一起（上图编号 5 所示），像 1、2、3、4 这些数字也聚集在一起（上图编号 6 所示）。如果把这些生物看成一个整体，他们也聚集在一起（上图编号 7 所示）。

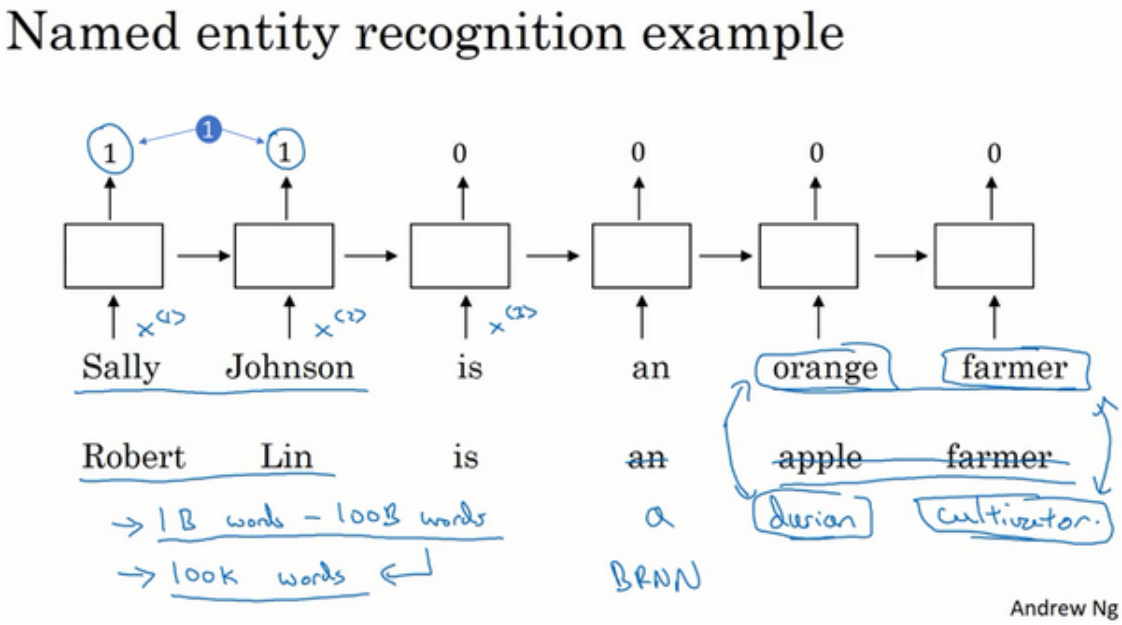
在网上你可能会看到像这样的图用来可视化，300 维或者更高维度的嵌入。希望你能有个整体的概念，这种词嵌入算法对于相近的概念，学到的特征也比较类似，在对这些概念可视化的时候，这些概念就比较相似，最终把它们映射为相似的特征向量。这种表示方式用的是在 300 维空间里的特征表示，这叫做嵌入（**embeddings**）。之所以叫嵌入的原因是，你可以想象一个 300 维的空间，我画不出来 300 维的空间，这里用个 3 维的代替（上图编号 8 所示）。现在取每一个单词比如 **orange**，它对应一个 3 维的特征向量，所以这个词就被嵌在这个 300 维空间里的一个点上了（上图编号 9 所示），**apple** 这个词就被嵌在这个 300 维空间的另一个点上了（上图编号 10 所示）。为了可视化，**t-SNE 算法**把这个空间映射到低维空间，你可以画出一个 2 维图像然后观察，这就是这个术语嵌入的来源。

词嵌入已经是 **NLP** 领域最重要的概念之一了，在自然语言处理领域。本节视频中你已经知道为什么要学习或者使用词嵌入了，下节视频我们会深入讲解如何用这些算法构建 **NLP** 算法。

2.2 使用词嵌入（Using Word Embeddings）

上一个视频中，你已经了解不同单词的特征化表示了。这节你会看到我们如何把这种表示方法应用到 **NLP** 应用中。

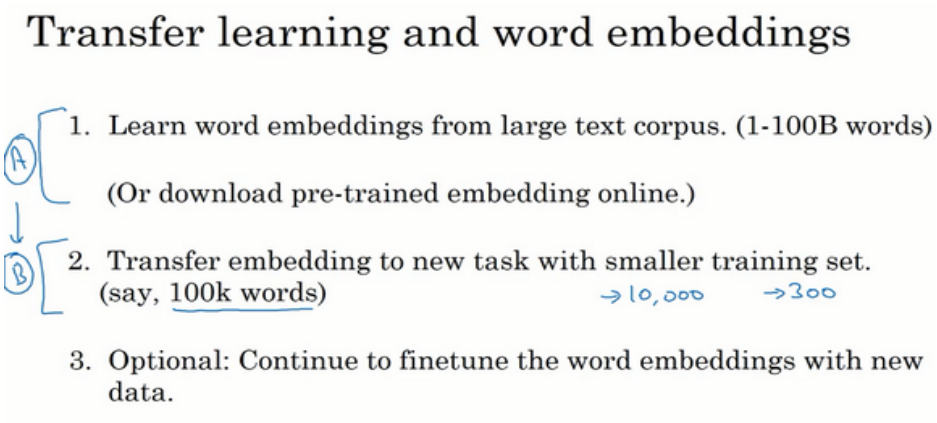
我们从一个例子开始，我们继续用命名实体识别的例子，如果你要找出人名，假如有一个句子：**“Sally Johnson is an orange farmer.”**（**Sally Johnson 是一个种橙子的农民**），你会发现 **Sally Johnson** 就是一个人名，所以这里的输出为 1。之所以能确定 **Sally Johnson** 是一个人名而不是一个公司名，是因为你知道种橙子的农民一定是一个人，前面我们已经讨论过用 **one-hot** 来表示这些单词，$x^{<1>}$ ，$x^{<2>}$等等。



但是如果你用特征化表示方法，嵌入的向量，也就是我们在上个视频中讨论的。那么用词嵌入作为输入训练好的模型，如果你看到一个新的输入：**“Robert Lin is an apple farmer.”**（**Robert Lin 是一个种苹果的农民**），因为知道 **orange** 和 **apple** 很相近，那么你的算法很容易就知道 **Robert Lin** 也是一个人，也是一个人的名字。一个有意思的情况是，要是测试集里这句话不是**“Robert Lin is an apple farmer.”**，而是不太常见的词怎么办？要是你看到：**“Robert Lin is a durian cultivator.”**（**Robert Lin 是一个榴莲培育家**）怎么办？**榴莲**（**durian**）是一种比较稀罕的水果，这种水果在新加坡和其他一些国家流行。如果对于一个命名实体识别任务，你只有一个很小的标记的训练集，你的训练集里甚至可能没有 **durian**（**榴莲**）或者 **cultivator**（**培育家**）这两个词。但是如果你有一个已经学好的词嵌入，它会告诉你 **durian**（**榴莲**）是水果，就像 **orange**（**橙子**）一样，并且 **cultivator**（**培育家**），做培育工作的人其实跟 **farmer**（**农民**）差不多，那么你就有可能从你的训练集里的**“an orange farmer”**（**种橙子的农民**）归

纳出**“a durian cultivator”**（**榴莲培育家**）也是一个人。

词嵌入能够达到这种效果，其中一个原因就是学习词嵌入的算法会考察非常大的文本集，也许是从网上找到的，这样你可以考察很大的数据集可以是 1 亿个单词，甚至达到 100 亿也都是合理的，大量的无标签的文本的训练集。通过考察大量的无标签文本，很多都是可以免费下载的，你可以发现 **orange**（**橙子**）和 **durian**（**榴莲**）相近，**farmer**（**农民**）和 **cultivator**（**培育家**）相近。因此学习这种嵌入表达，把它们都聚集在一块，通过读取大量的互联网文本发现了 **orange**（**橙子**）和 **durian**（**榴莲**）都是水果。接下来你可以把这个词嵌入应用到你的命名实体识别任务当中，尽管你只有一个很小的训练集，也许训练集里有 100,000 个单词，甚至更小，这就使得你可以使用迁移学习，把你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，能够分辨 **orange**（**橙子**）、**apple**（**苹果**）和 **durian**（**榴莲**）都是水果的知识，然后把这些知识迁移到一个任务中，比如你只有少量标记的训练数据集的命名实体识别任务中。当然了，这里为了简化我只画了单向的 **RNN**，事实上如果你想用在命名实体识别任务上，你应该用一个双向的 **RNN**，而不是这样一个简单的。



总结一下，这是如何用词嵌入做迁移学习的步骤。

第一步，先从大量的文本集中学习词嵌入。一个非常大的文本集，或者可以下载网上预训练好的词嵌入模型，网上你可以找到不少，词嵌入模型并且都有许可。

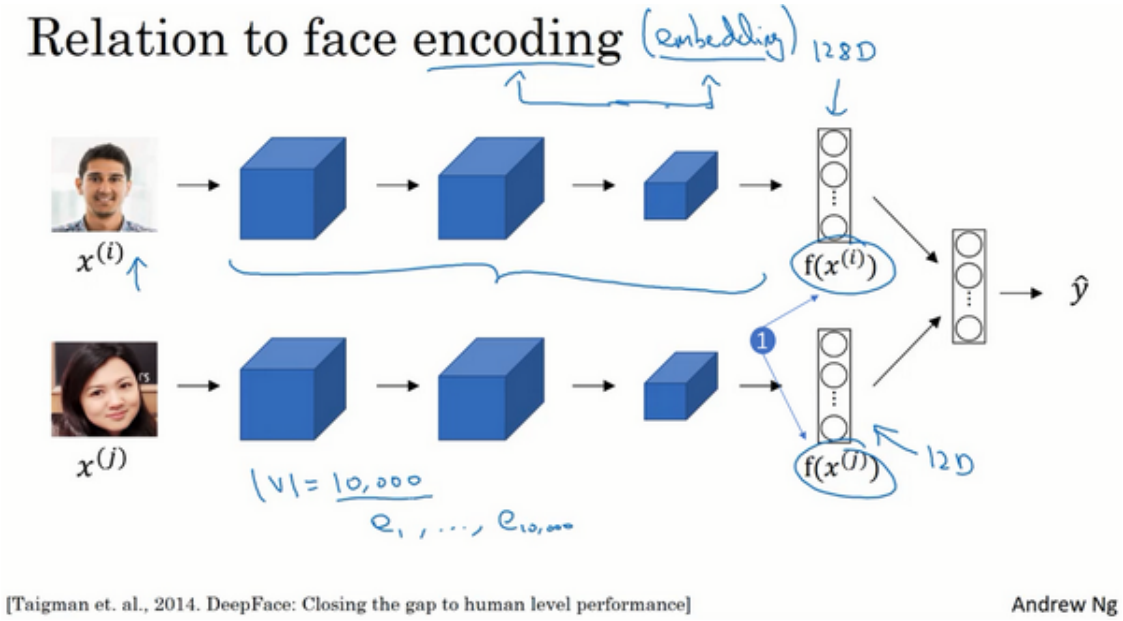
第二步，你可以用这些词嵌入模型把它迁移到你的新的只有少量标注训练集的任务中，比如说用这个 300 维的词嵌入来表示你的单词。这样做的一个好处就是你可以用更低维度的特征向量代替原来的 10000 维的 one-hot 向量，现在你可以用一个 300 维更加紧凑的向量。尽管 one-hot 向量很快计算，而学到的用于词嵌入的 300 维的向量会更加紧凑。

第三步，当你在你新的任务上训练模型时，在你的命名实体识别任务上，只有少量的标记数据集上，你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，只有这个

第二步中有很大的数据集你才会这样做，如果你标记的数据集不是很大，通常我不会在微调词嵌入上费力气。

当你的任务的训练集相对较小时，词嵌入的作用最明显，所以它广泛用于 **NLP** 领域。我只提到一些，不要太担心这些术语（下问列举的一些 **NLP** 任务），它已经用在命名实体识别，用在文本摘要，用在文本解析、指代消解，这些都是非常标准的 **NLP** 任务。

词嵌入在语言模型、机器翻译领域用的少一些，尤其是你做语言模型或者机器翻译任务时，这些任务你有大量的数据。在其他的迁移学习情形中也一样，如果你从某一任务 **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。所以对于很多 **NLP** 任务这些都是对的，而对于一些语言模型和机器翻译则不然。



[Taigman et. al., 2014. DeepFace: Closing the gap to human level performance] Andrew Ng

最后，词嵌入和人脸编码之间有奇妙的关系，你已经在前面的课程学到了关于人脸编码的知识了，如果你上了卷积神经网络的课程的话。你应该还记得对于人脸识别，我们训练了一个 **Siamese** 网络结构，这个网络会学习不同人脸的一个 128 维表示，然后通过比较编码结果来判断两个图片是否是同一个人脸，这个词嵌入的意思和这个差不多。在人脸识别领域大家喜欢用编码这个词来指代这些向量$f(x^{(i)})$，$f(x^{(j)})$（上图编号 1 所示），人脸识别领域和这里的词嵌入有一个不同就是，在人脸识别中我们训练一个网络，任给一个人脸照片，甚至是没有见过的照片，神经网络都会计算出相应的一个编码结果。上完后面几节课，你会更明白，我们学习词嵌入则是有一个固定的词汇表，比如 10000 个单词，我们学习向量e_1到e_{10000}，学习一个固定的编码，每一个词汇表的单词的固定嵌入，这就是人脸识别与我们接下来几节视频要讨论的算法之间的一个不同之处。这里的术语编码（**encoding**）和嵌入（**embedding**）可以互换，所以刚才讲的差别不是因为术语不一样，这个差别就是，人脸识别中的算法未来

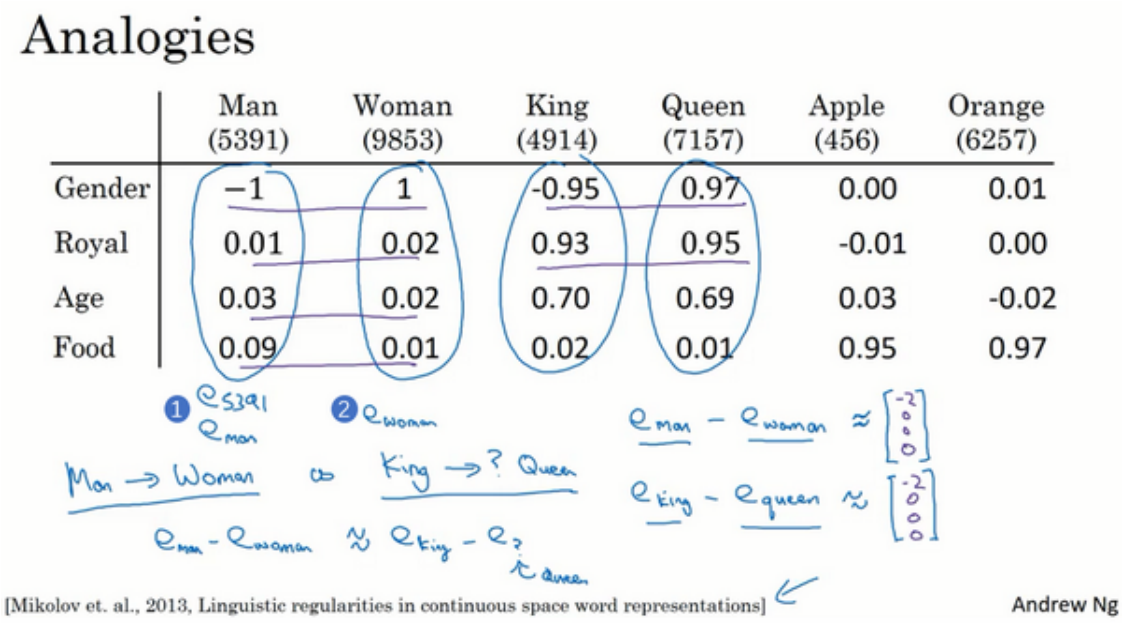
可能涉及到海量的人脸照片，而自然语言处理有一个固定的词汇表，而像一些没有出现过的单词我们就记为未知单词。

这节视频里，你看到如何用词嵌入来实现这种类型的迁移学习，并且通过替换原来的**one-hot** 表示，而是用之前的嵌入的向量，你的算法会泛化的更好，你也可以从较少的标记数据中进行学习。接下来我会给你展示一些词嵌入的特性，这之后再讨论学习这些词嵌入的算法。下个视频我们会看到词嵌入在做类比推理中发挥的作用。

2.3 词嵌入的特性（Properties of Word Embeddings）

到现在，你应该明白了词嵌入是如何帮助你构建自然语言处理应用的。词嵌入还有一个迷人的特性就是它还能帮助实现类比推理，尽管类比推理可能不是自然语言处理应用中最重要的，不过它能帮助人们理解词嵌入做了什么，以及词嵌入能够做什么，让我们来一探究竟。

这是一系列你希望词嵌入可以捕捉的单词的特征表示，假如我提出一个问题，**man** 如果对应 **woman**，那么 **king** 应该对应什么？你们应该都能猜到 **king** 应该对应 **queen**。能否有一种算法来自动推导出这种关系，下面就是实现的方法。



我们用一个四维向量来表示 **man**，我们用e_{5391}来表示，不过在这节视频中我们先把它（上图编号 1 所示）称为e_{man}，而旁边这个（上图编号 2 所示）表示 **woman** 的嵌入向量，称它为e_{woman}，对 **king** 和 **queen** 也是用一样的表示方法。在该例中，假设你用的是四维的嵌入向量，而不是比较典型的 50 到 1000 维的向量。这些向量有一个有趣的特性，就是假如你有向量e_{man}和e_{woman}，将它们进行减法运算，即

$$e_{\text{man}} - e_{\text{woman}} = \begin{bmatrix} -1 \\ 0.01 \\ 0.03 \\ 0.09 \end{bmatrix} - \begin{bmatrix} 1 \\ 0.02 \\ 0.02 \\ 0.01 \end{bmatrix} = \begin{bmatrix} -2 \\ -0.01 \\ 0.01 \\ 0.08 \end{bmatrix} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

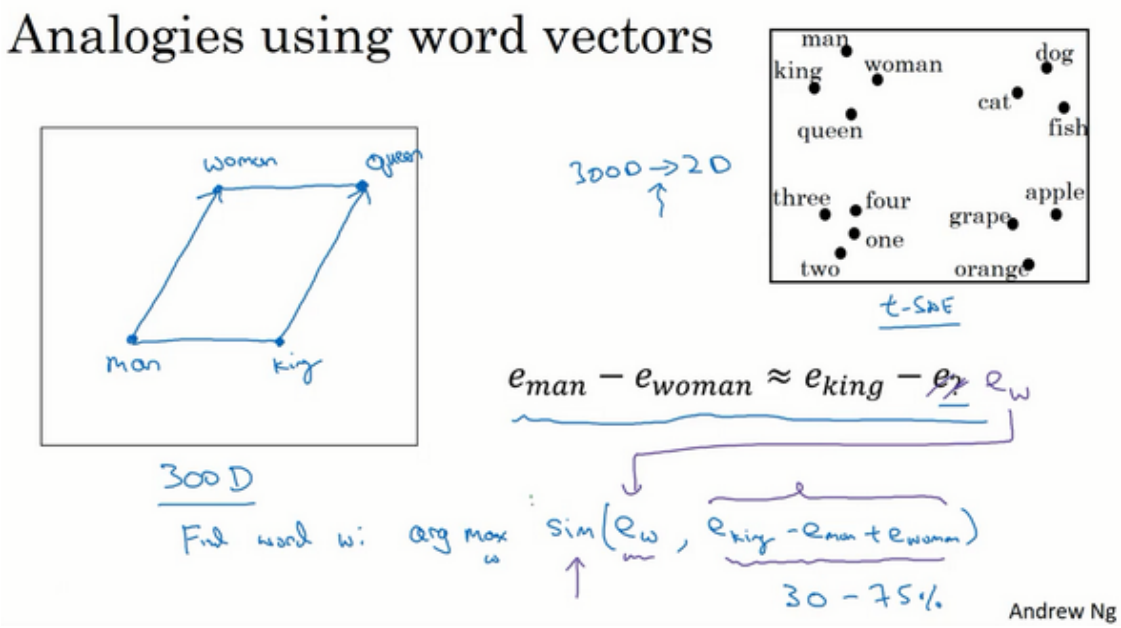
类似的，假如你用e_{king}减去e_{queen}，最后也会得到一样的结果，即

$$e_{\text{king}} - e_{\text{queen}} = \begin{bmatrix} -0.95 \\ 0.93 \\ 0.70 \\ 0.02 \end{bmatrix} - \begin{bmatrix} 0.97 \\ 0.95 \\ 0.69 \\ 0.01 \end{bmatrix} = \begin{bmatrix} -1.92 \\ -0.02 \\ 0.01 \\ 0.01 \end{bmatrix} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

这个结果表示，**man** 和 **woman** 主要的差异是 **gender**（**性别**）上的差异，而 **king** 和 **queen**

之间的主要差异，根据向量的表示，也是 **gender**（**性别**）上的差异，这就是为什么$e_{\text{man}} - e_{\text{woman}}$和$e_{\text{king}} - e_{\text{queen}}$结果是相同的。所以得出这种类比推理的结论的方法就是，当算法被问及 **man** 对 **woman** 相当于 **king** 对什么时，算法所做的就是计算$e_{\text{man}} - e_{\text{woman}}$，然后找出一个向量也就是找出一个词，使得$e_{\text{man}} - e_{\text{woman}} \approx e_{\text{king}} - e_{?}$，也就是说，当这个新词是 **queen** 时，式子的左边会近似地等于右边。这种思想首先是被 **Tomas Mikolov** 和 **Wen-tau Yih** 还有 **Geoffrey Zweig** 提出的，这是词嵌入领域影响力最为惊人和显著的成果之一，这种思想帮助了研究者们对词嵌入领域建立了更深刻的理解。

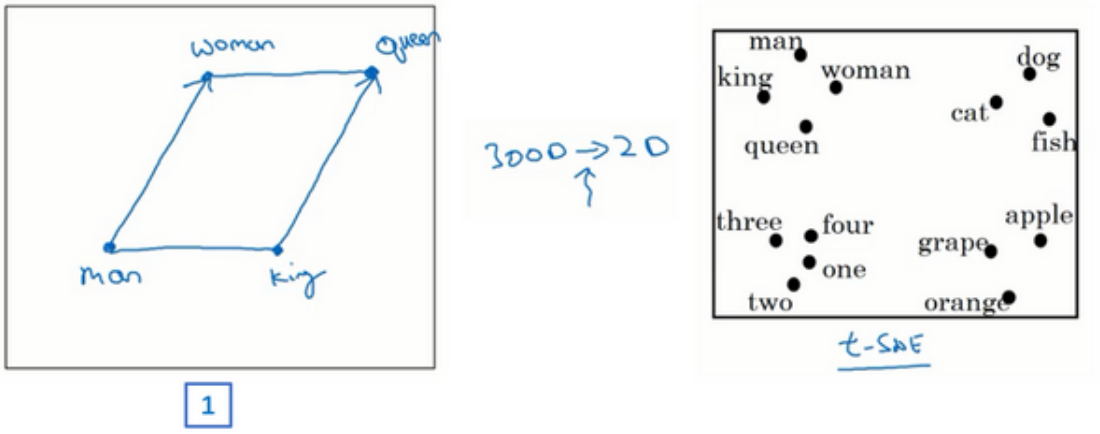
（**Mikolov T, Yih W T, Zweig G. Linguistic regularities in continuous space word representations[J]. In HLT-NAACL, 2013.**）



让我们来正式地探讨一下应该如何把这种思想写成算法。在图中，词嵌入向量在一个可能有 300 维的空间里，于是单词 **man** 代表的就是空间中的一个点，另一个单词 **woman** 代表空间另一个点，单词 **king** 也代表一个点，还有单词 **queen** 也在另一点上（上图编号 1 方框内所示的点）。事实上，我们在上个幻灯片所展示的就是向量 **man** 和 **woman** 的差值非常接近于向量 **king** 和 **queen** 之间的差值，我所画的这个箭头（上图编号 2 所示）代表的就是向量在 **gender**（**性别**）这一维的差，不过不要忘了这些点是在 300 维的空间里。为了得出这样的类比推理，计算当 **man** 对于 **woman**，那么 **king** 对于什么，你能做的就是找到单词 **w** 来使得，$e_{\text{man}} - e_{\text{woman}} \approx e_{\text{king}} - e_w$这个等式成立，你需要的就是找到单词 **w** 来最大化e_w与$e_{\text{king}} - e_{\text{man}} + e_{\text{woman}}$的相似度，即

$$Find\ word\ w: argmax\ Sim(e_w, e_{\text{king}} - e_{\text{man}} + e_{\text{woman}})$$

所以我做的就是我把这个e_w全部放到等式的一边，于是等式的另一边就会是$e_{\text{king}} - e_{\text{man}} + e_{\text{woman}}$。我们有一些用于测算$e_w$和$e_{\text{king}} - e_{\text{man}} + e_{\text{woman}}$之间的相似度的函数，然后通过方程找到一个使得相似度最大的单词，如果结果理想的话会得到单词 **queen**。值得注意的是这种方法真的有效，如果你学习一些词嵌入，通过算法来找到使得相似度最大化的单词 **w**，你确实可以得到完全正确的答案。不过这取决于过程中的细节，如果你查看一些研究论文就不难发现，通过这种方法来做类比推理准确率大概只有 30%~75%，只要算法猜中了单词，就把该次计算视为正确，从而计算出准确率，在该例子中，算法选出了单词 **queen**。



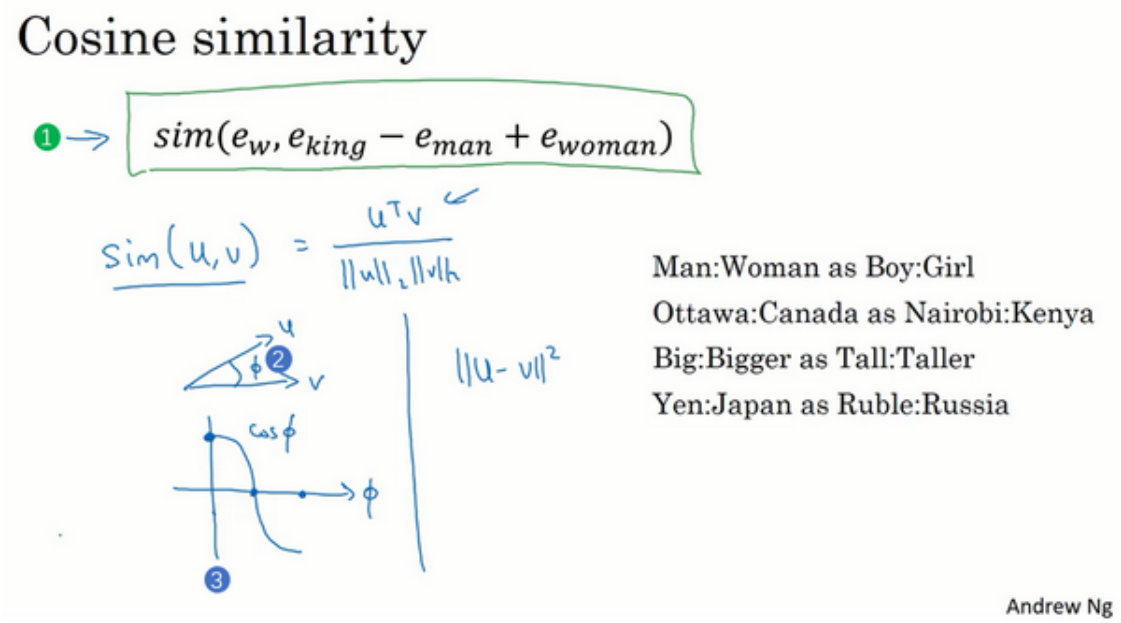
1

在继续下一步之前，我想再说明一下左边的这幅图（上图编号 1 所示），在之前我们谈到过用 **t-SNE 算法来将单词可视化**。**t-SNE 算法**所做的就是把这些 300 维的数据用一种非线性的方式映射到 2 维平面上，可以得知 **t-SNE** 中这种映射很复杂而且很非线性。在进行 **t-SNE** 映射之后，你不能总是期望使等式成立的关系，会像左边那样成一个平行四边形，尽管在这个例子最初的 300 维的空间内你可以依赖这种平行四边形的关系来找到使等式成立的一对类比，通过 **t-SNE 算法**映射出的图像可能是正确的。但在大多数情况下，由于 **t-SNE** 的非线性映射，你就没法再指望这种平行四边形了，很多这种平行四边形的类比关系在 **t-SNE** 映射中都会失去原貌。

现在，再继续之前，我想再快速地列举一个最常用的相似度函数，这个最常用的相似度函数叫做**余弦相似度**。这是我们上个幻灯片所得到的等式（下图编号 1 所示），在余弦相似度中，假如在向量u和v之间定义相似度:$\text{sim}(u,v) = \frac{u^T v}{||u||_2 ||v||_2}$

现在我们先不看分母，分子其实就是u和v的内积。如果 u 和 v 非常相似，那么它们的内积将会很大，把整个式子叫做余弦相似度，其实就是因为该式是u和v的夹角的余弦值，所以这个角（下图编号 2 所示）就是 Φ 角，这个公式实际就是计算两向量夹角 Φ 角的余弦。你应该还记得在微积分中，Φ 角的余弦图像是这样的（下图编号 3 所示），所以夹角为 0 度

时，余弦相似度就是 1，当夹角是 90 度角时余弦相似度就是 0，当它们是 180 度时，图像完全跑到了相反的方向，这时相似度等于-1，这就是为什么余弦相似度对于这种类比工作能起到非常好的效果。 距离用平方距离或者欧氏距离来表示:$||u-v||^2$



参考资料：余弦相似度 为了测量两个词的相似程度，我们需要一种方法来测量两个词的两个嵌入向量之间的相似程度。给定两个向量u和v，余弦相似度定义如下：

$$CosineSimilarity(u,v) = \frac{u.v}{||u||_2||v||_2} = cos(\theta)$$

其中 $u.v$ 是两个向量的点积（或内积），$||u||_2$是向量u的范数（或长度），并且 θ 是向量u和v之间的角度。这种相似性取决于角度在向量u和v之间。如果向量u和v非常相似，它们的余弦相似性将接近 1；如果它们不相似，则余弦相似性将取较小的值。

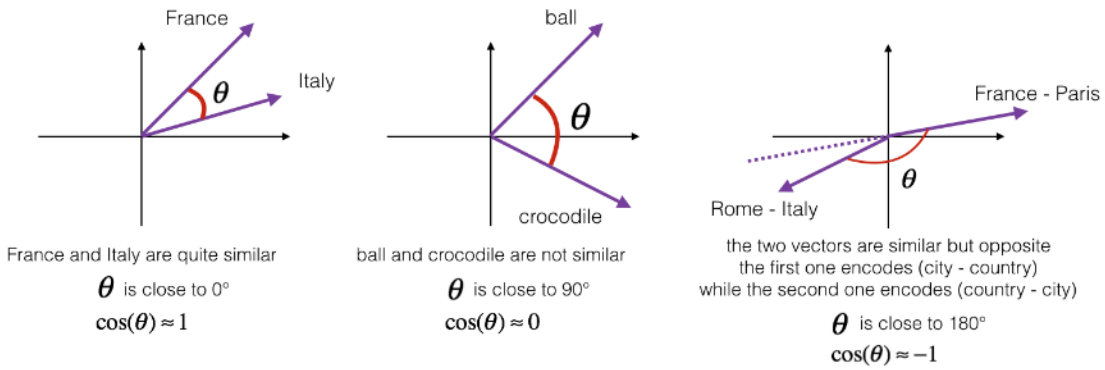


图 1：两个向量之间角度的余弦是衡量它们有多相似的指标，角度越小，两个向量越相似。

从学术上来说，比起测量相似度，这个函数更容易测量的是相异度，所以我们需要对其取负，这个函数才能正常工作，不过我还是觉得余弦相似度用得更多一点，这两者的主要区

别是它们对u和v之间的距离标准化的方式不同。

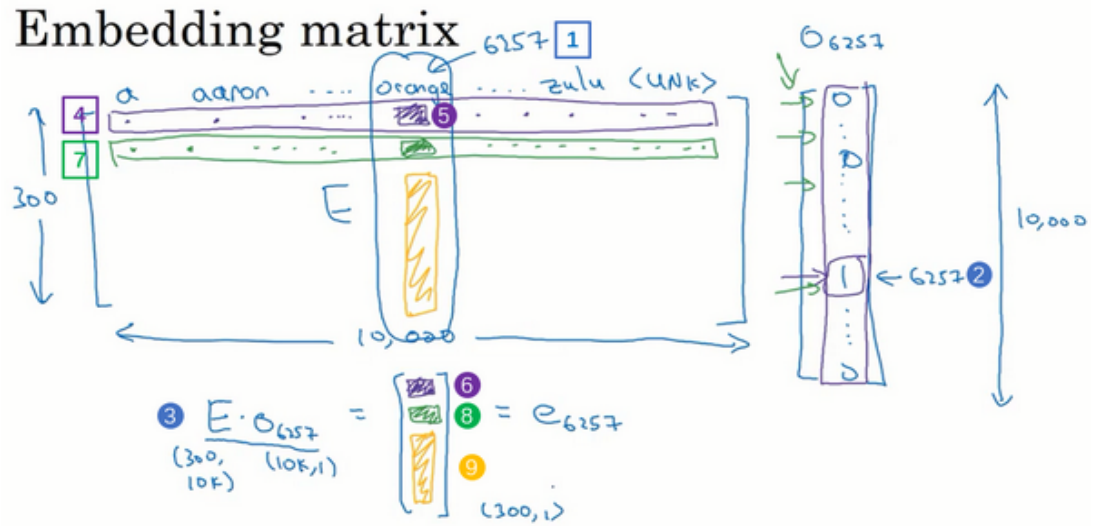
词嵌入的一个显著成果就是，可学习的类比关系的一般性。举个例子，它能学会 **man** 对于 **woman** 相当于 **boy** 对于 **girl**，因为 **man** 和 **woman** 之间和 **king** 和 **queen** 之间，还有 **boy** 和 **girl** 之间的向量差在 **gender**（**性别**）这一维都是一样的。它还能学习 **Canada**（**加拿大**）的首都是 **Ottawa**（**渥太华**），而渥太华对于加拿大相当于 **Nairobi**（**内罗毕**）对于 **Kenya**（**肯尼亚**），这些都是国家中首都城市名字。它还能学习 **big** 对于 **bigger** 相当于 **tall** 对于 **taller**，还能学习 **Yen**（**円**）对于 **Janpan**（**日本**），**円**是日本的货币单位，相当于 **Ruble**（**卢比**）对于 **Russia**（**俄罗斯**）。这些东西都能够学习，只要你在大型的文本语料库上实现一个词嵌入学习算法，只要从足够大的语料库中进行学习，它就能自主地发现这些模式。

在本节视频中，你见到了词嵌入是如何被用于类比推理的，可能你不会自己动手构建一个类比推理系统作为一项应用，不过希望在这些可学习的类特征的表示方式能够给你一些直观的感受。你还看知道了余弦相似度可以作为一种衡量两个词嵌入向量间相似度的办法，我们谈了许多有关这些嵌入的特性，以及如何使用它们。下节视频中，我们来讨论如何真正的学习这些词嵌入。

2.4 嵌入矩阵（Embedding Matrix）

接下来我们要将学习词嵌入这一问题具体化，当你应用算法来学习词嵌入时，实际上是学习一个嵌入矩阵，我们来看一下这是什么意思。

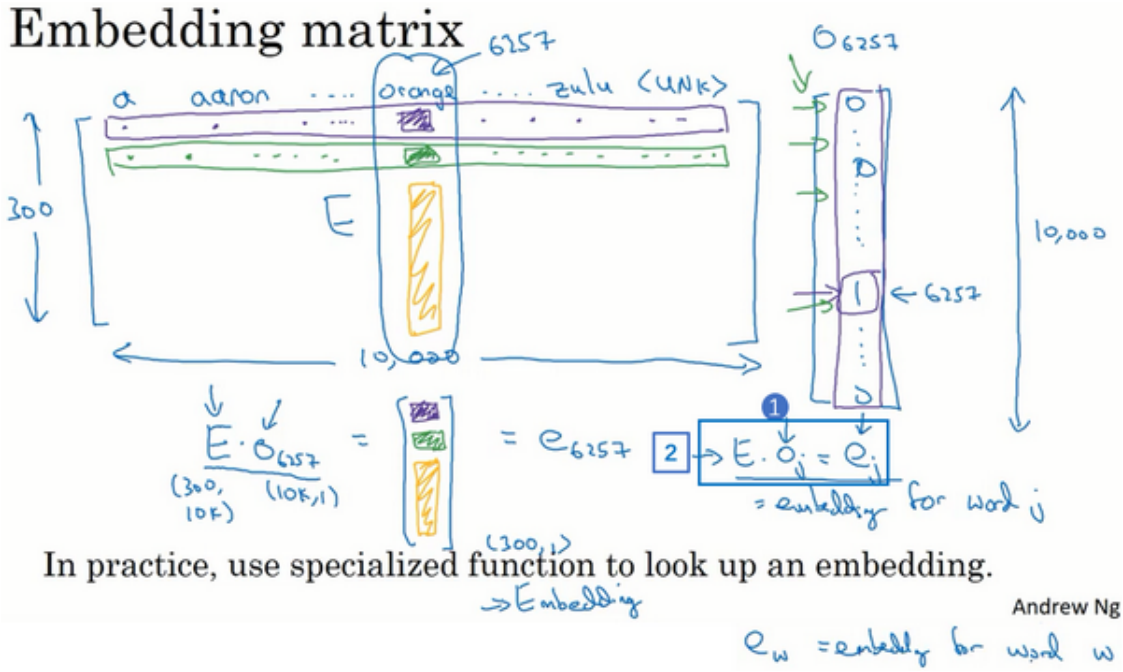
和之前一样，假设我们的词汇表含有 10,000 个单词，词汇表里有 **a**，**aaron**，**orange**，**zulu**，可能还有一个未知词标记**<UNK>**。我们要做的就是学习一个嵌入矩阵E，它将是一个 300×10,000 的矩阵，如果你的词汇表里有 10,000 个，或者加上未知词就是 10,001 维。这个矩阵的各列代表的是词汇表中 10,000 个不同的单词所代表的不同向量。假设 **orange** 的单词编号是 6257（下图编号 1 所示），代表词汇表中第 6257 个单词，我们用符号O_{6527} 来表示这个 **one-hot** 向量，这个向量除了第 6527 个位置上是 1（下图编号 2 所示），其余各处都为 0，显然它是一个 10,000 维的列向量，它只在一个位置上有 1，它不像图上画的那么短，它的高度应该和左边的嵌入矩阵的宽度相等。



假设这个嵌入矩阵叫做矩阵E，注意如果用E去乘以右边的 one-hot 向量（上图编号 3 所示），也就是O_{6527}，那么就会得到一个 300 维的向量，E是 300×10,000 的，O_{6527}是 10,000×1 的，所以它们的积是 300×1 的，即 300 维的向量。要计算这个向量的第一个元素，你需要做的是把E的第一行（上图编号 4 所示）和O_{6527}的整列相乘，不过O_{6527}的所有元素都是 0，只有 6257 位置上是 1，最后你得到的这个向量的第一个元素（上图编号 5 所示）就是 **orange** 这一列下的数字（上图编号 6 所示）。然后我们要计算这个向量的第二个元素，就是把E的第二行（上图编号 7 所示）和这个O_{6527}相乘，和之前一样，然后得到第二个元素（上图编号 8 所示），以此类推，直到你得到这个向量剩下的所有元素（上图编号 9 所示）。

这就是为什么把矩阵E和这个 **one-hot** 向量相乘，最后得到的其实就是这个 300 维的列，

就是单词 **orange** 下的这一列，它等于e_{6257}，这个符号是我们用来表示这个 300×1 的嵌入向量的符号，它表示的单词是 **orange**。



更广泛来说，假如说有某个单词 w，那么e_w就代表单词 w 的嵌入向量。同样，EO_j，O_j就是只有第j个位置是 1 的 one-hot 向量，得到的结果就是e_j，它表示的是字典中单词 j 的嵌入向量。

在这一小节中，要记住的一件事就是我们的目标是学习一个嵌入矩阵E。在下节视频中你将会随机地初始化矩阵E，然后使用梯度下降法来学习这个 300×10,000 的矩阵中的各个参数，E乘以这个 **one-hot** 向量（上图编号 1 所示）会得到嵌入向量。再多说一点，当我们写这个等式（上图编号 2 所示）的时候，写出这些符号是很方便的，代表用矩阵E乘以 **one-hot** 向量O_j。但当你动手实现时，用大量的矩阵和向量相乘来计算它，效率是很低下的，因为 **one-hot** 向量是一个维度非常高的向量，并且几乎所有元素都是 0，所以矩阵向量相乘效率太低，因为我们要乘以一大堆的 0。所以在实践中你会使用一个专门的函数来单独查找矩阵E的某列，而不是用通常的矩阵乘法来做，但是在画示意图时（上图所示，即矩阵E乘以 **one-hot** 向量示意图），这样写比较方便。但是例如在 **Keras** 中就有一个嵌入层，然后我们用这个嵌入层更有效地从嵌入矩阵中提取出你需要的列，而不是对矩阵进行很慢很复杂的乘法运算。