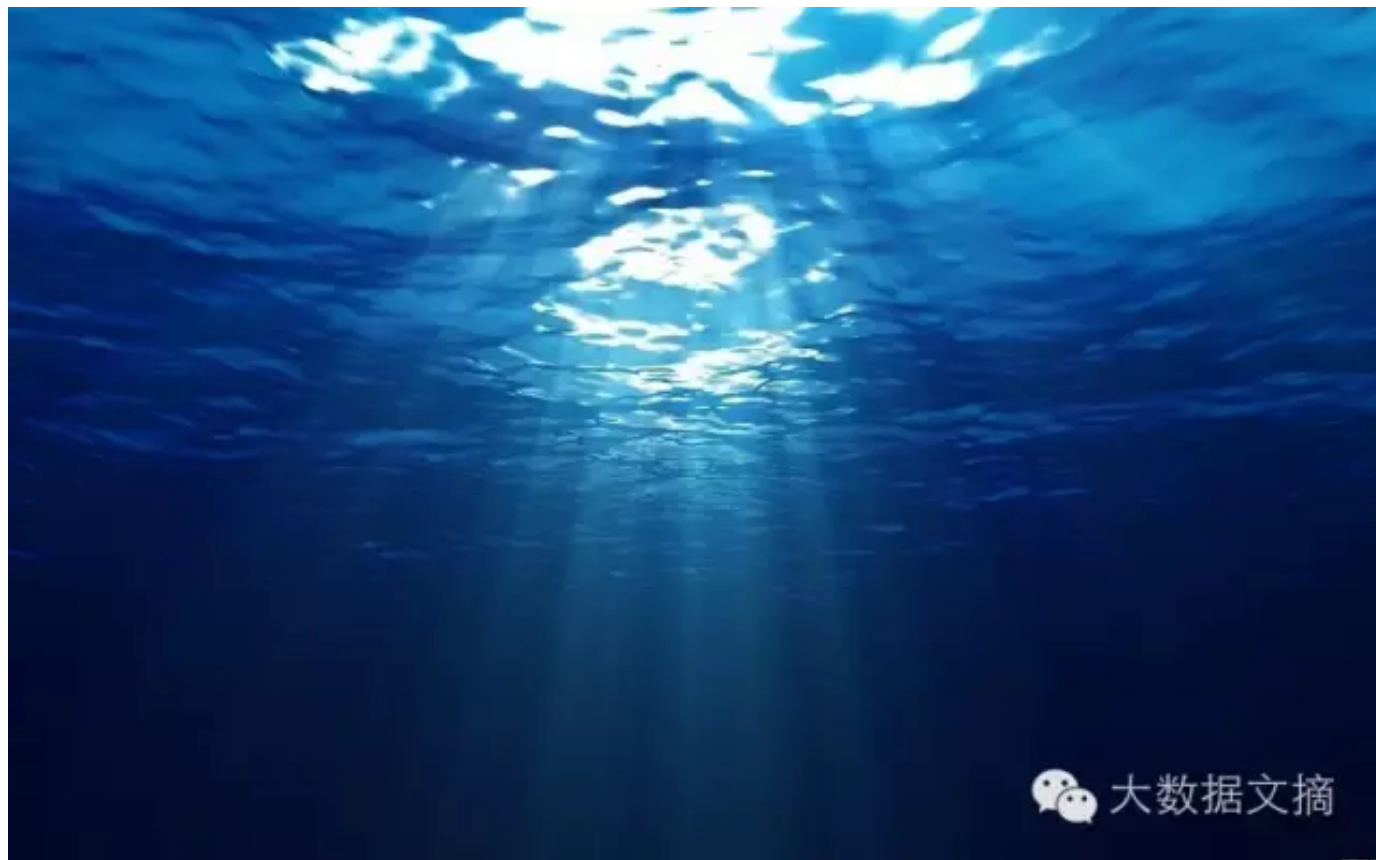


重磅启动！翻译斯坦福大学课程：深度学习与自然语言处理

2016-06-06 大数据文摘



人工智能	机器学习	AR/VR	可视化
医疗	金融	社会	数据开放
			非专栏

像追美剧一样追课程！

大数据文摘已获得斯坦福大学深度学习课程CS224d的翻译授权，重磅启动“斯坦福深度学习课程CS224d”的翻译工程，所有译文将会免费发布，计划每周发布1篇。期待你的加入，加入要求见文末，报名请点击文末“阅读原文”。

大数据文摘作品，转载需授权

作者|寒小阳 && 龙心尘

感谢@Fantzy同学的帮助

大数据文摘“机器学习”专栏介绍

本文为大数据文摘机器专栏推出的【数据科学/机器学习】学习分享项目启动篇，我们将以Stanford、Harvard等优秀高校的前沿数据科学课程为主线的学习与分享，将数据科学普及给更多国内的读者，也通过相互学习和交流加深对数据领域知识的理解认识。



课堂笔记：第一部分 春季2016

关键词：自然语言处理 (NLP) .词向量 (Word Vectors) .奇异值分解(Singular Value Decomposition). Skip-gram. 词组的持续爆 (CBOW) ,负采样样本 (Negative Sampling)

这是本课程的第一节，我们会先介绍自然语言处理 (NLP) 的概念和NLP现在所面对问题；然后开始讨论用数学向量代表自然语言词组的设想。最后我们会讨论现行的词向量构造方法。



1. 自然语言处理简介

在最开始咱们先说说什么是NLP。NLP的目的是设计出算法，让计算机“懂得”人类的自然语言，从而为人类执行任务。这些任务分为几个难度等级，例如

容易的任务：

- 语法检查
- 关键词搜索
- 查找同义词

中等难度的任务：

- 从网站，文件或其他来源中提取信息
- 比较有挑战的任务：
- 机器翻译（例如：中译英）
- 语意分析（提问者说的意思是什么）
- 指代分析（例如，“他”或“它”在一个特定文件中指的是什么）
- 回答问题（例如.回答“Jeopardy Questions”一种涉及人类社会各个方面的综艺问答）

在处理所有NLP任务的时候，我们首先需要解决非常重要的一个问题(可能是最重要的)：用什么方式将词组输入到模型中去。简单的NLP问题可能并不需要将词组作为独立个体对待 (atomic symbols) ，但现在的问题绝大多数需要这样一个预处理，来体现词组之间关联/相似性和区别。所以我们引入词向量的概念，如果把词编码成词向量，我们很容易从向量的角度去衡量不同的词之间的关联与差异（常用的距离测度法，包括Jaccard, Cosine, Euclidean等等，注：距离测度法，即用一个可观测度量的量来描述一个不能直接观测度量的量）。



2.词向量

我们拿英文举例：

英语中大约有1300万个词组（token，自定义字符串，译作词组），不过他们全部是独立的吗？并不是哦，比如有一些词组，“Feline猫科动物”和“Cat猫”，“Hotel宾馆”和“Motel汽车旅馆”，其实有一定的关联或者相似性在。因此，我们希望用词向量编码词组，使它代表在词组的N维空间中的一个点（而点与点之间有距离的远近等关系，可以体现深层一点的信息）。每一个词向量的维度都可能会表征一些意义（物理含义），这些意义我们用“声明speech”来定义。例如，语义维度可以用来表明时态（过去与现在与未来），计数（单数与复数），和性别（男性与女性）。

说起来，词向量的编码方式其实挺有讲究的。咱们从最简单的看起，最简单的编码方式叫做one-hot vector：假设我们的词库总共有n个词，那我们开一个1*n的高维向量，而每个词都会在某个索引index下取到1，其余位置全部都取值为0.词向量在这种类型的编码中如下图所示：

$$w^{aardcark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^a = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \dots$$

$$w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

大数据文摘

这种词向量编码方式简单粗暴，我们将每一个词作为一个完全独立的个体来表达。遗憾的是，这种方式下，我们的词向量没办法给我们任何形式的词组相似性权衡。例如：

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

大数据文摘

（注：这里W-1是W的逆矩阵，它们有关系：W-1*W=1，注意到hotel和motel是近义词）

究其根本你会发现，是你开了一个极高维度的空间，然后每个词语都会占据一个维度，因此没有办法在空间中关联起来。因此我们可能可以把词向量的维度降低一些，在这样一个子空间中，可能原本没有关联的词就关联起来了。



3.基于SVD的方法

这是一种构造词嵌入（即词向量）的方法，我们首先会遍历所有的文本数据集，然后统计词出现的次数，接着用一个矩阵 X 来表示所有的次数情况，紧接着对 X 进行奇异值分解得到一个USVT的分解。然后用 U 的行（rows）作为所有词表中词的词向量。对于矩阵 X ，我们有几种选择，咱们一起来比较一下。

3.1 词-文档矩阵

最初的想法是，我们猜测相互关联的词组同时出现在相同的文件中的概率很高。例如，“银行”、“债券”、“股票”、“钱”等都可能出现在一起。但是，“银行”、“章鱼”、“香蕉”和“曲棍球”可能不会一直一起出现。基于这个想法，我们建立一个词组文档矩阵 X ，具体是这么做的：遍历海量的文件，每次词组 i 出现在文件 j 中时，将 X_{ij} 的值加1。不过大家可想而知，这会是个很大的矩阵 $R|V| \times M$ ，而且矩阵大小还和文档个数 M 有关系。所以咱们最好想办法处理和优化一下。

3.2 基于窗口的共现矩阵 X

我们还是用一样的逻辑，不过换一种统计方式，把矩阵 X 记录的词频变成一个相关性矩阵。我们先规定一个固定大小的窗口，然后统计每个词出现在窗口中次数，这个计数是针对整个语料集做的。可能说得有点含糊，咱们一起来看个例子，假定我们有如下的3个句子，同时我们的窗口大小设定为1（把原始的句子分拆成一个一个的词）：

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

由此产生的计数矩阵如下：

$$X = \begin{matrix} & \begin{matrix} I & like & enjoy & deep & learning & NLP & flying & . \end{matrix} \\ \begin{matrix} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{matrix} & \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

然后我们对 X 做奇异值分解，观察观察奇异值（矩阵的对角元素），并根据我们期待保留的百分比来进行阶段（只保留前 k 个维度）：

$$\frac{\sum_{i=1}^k \sigma_i}{\sum_{i=1}^{|V|} \sigma_i}$$

大数据文摘

然后我们把子矩阵 $U_{1:|V|,1:k}$ 视作我们的词嵌入矩阵。也就是说，对于词表中的每一个词，我们都用一个 k 维的向量来表达。

对 X 采用奇异值分解

$$|V| \begin{bmatrix} |V| \\ X \end{bmatrix} = |V| \begin{bmatrix} | \\ | \\ \vdots \end{bmatrix} \begin{bmatrix} |V| \\ u_1 & u_2 & \dots \end{bmatrix} |V| \begin{bmatrix} |V| \\ \sigma_1 & 0 & \dots \\ 0 & \sigma_2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} |V| \begin{bmatrix} |V| \\ - & v_1 & - \\ - & v_2 & - \\ \vdots & \vdots & \vdots \end{bmatrix}$$

通过选择前 K 个奇异向量来进行降维：

$$|V| \begin{bmatrix} |V| \\ \hat{X} \end{bmatrix} = |V| \begin{bmatrix} | \\ | \\ \vdots \end{bmatrix} \begin{bmatrix} k \\ u_1 & u_2 & \dots \end{bmatrix} k \begin{bmatrix} k \\ \sigma_1 & 0 & \dots \\ 0 & \sigma_2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} k \begin{bmatrix} |V| \\ - & v_1 & - \\ - & v_2 & - \\ \vdots & \vdots & \vdots \end{bmatrix}$$

这两种方法都能产生词向量，它们能够充分地编码语义和句法的信息，但同时也带来了其他的问题：

- 矩阵的维度会经常变化（新的词语经常会增加，语料库的大小也会随时变化）。
- 矩阵是非常稀疏的，因为大多数词并不同时出现。
- 矩阵的维度通常非常高（ $\approx 10^6 \times 10^6$ ）
- 训练需要 $O(n^2)$ 的复杂度（比如 SVD）
- 需要专门对矩阵 X 进行特殊处理，以应对词组频率的极度不平衡的状况

当然，有一些办法可以缓解一下上述提到的问题：

- 忽视诸如 “he”、“the”、“has” 等功能词。
- 应用 “倾斜窗口”（ramp window），即：根据文件中词组之间的距离给它们的共现次数增加相应的权重。

- 使用皮尔森的相关性 (Pearson correlation) , 将0记为负数, 而不是它原来的数值。

不过缓解终归只是缓解, 咱们需要更合理地解决这些问题, 这也就是我们马上要提到的基于迭代的方法。



4.基于迭代的方法

现在我们退后一步, 来尝试一种新的方法。在这里我们并不计算和存储全局信息, 因为这会包含太多大型数据集和数十亿句子。我们尝试创建一个模型, 它能够一步步迭代地进行学习, 并最终得出每个单词基于其上下文的条件概率。

词语的上下文:

一个词语的上下文是它周围C个词以内的词。如果C=2, 句子"The quick brown fox jumped over the lazy dog"中单词"fox"的上下文为 {"quick", "brown", "jumped", "over"}.

我们想建立一个概率模型, 它包含已知和未知参数。每增加一个训练样本, 它就能从模型的输入、输出和期望输出 (标签) , 多学到一点点未知参数的信息。

在每次迭代过程中, 这个模型都能够评估其误差, 并按照一定的更新规则, 惩罚那些导致误差的参数。这种想法可以追溯到1986年 (Learning representations by back-propagating errors. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams (1988)), 我们称之为误差“反向传播”法。

4.1 语言模型 (1-gram, 2-gram等等)

首先, 我们需要建立一个能给“分词序列”分配概率的模型。我们从一个例子开始:

"The cat jumped over the puddle." (猫 跳 过 水坑)

一个好的语言模型会给这句话以很高的概率, 因为这是一个在语法和语义上完全有效的句子。同样地, 这句" stock boil fish is toy" (股票 煮 鱼 是 玩具) 就应该有一个非常低的概率, 因为它没有任何意义的。在数学上, 我们可以令任意给定的n个有序的分词序列的概率为:

$$P(w_1, w_2, w_3 \dots w_n)$$

大数据文摘

我们可以采用一元语言模型。它假定词语的出现是完全独立的, 于是可以将整个概率拆开相乘:

$$P(w_1, w_2, w_3 \dots w_n) = \prod_{i=1}^N P(w_i)$$

大数据文摘

看到这里，肯定很多同学就要喷了，这不对，词和词之间没有关联吗？确实，我们知道一句话中每一个词语都跟它前面的词语有很强的依赖关系，忽略这一点的话，一些完全无意义的句子，可能会有很高的概率。咱们稍微改一改，让一个词语的概率依赖于它前面一个词语。我们将这种模型称作bigram（2-gram，二元语言模型），表示为：

$$P(w_1, w_2, w_3 \dots w_n) = \prod_{i=2}^N P(w_i | w_{i-1})$$

 大数据文摘

看起来还是有点简单？恩，也对，我们只考虑一个词语依赖于其相邻的一个词语的关系，而不是考虑其依赖整个句子的情况。别着急，接下来将会看到，这种方法能让我们有非常显著的进步。考虑到前面“词-词”矩阵的情况，我们至少可以算出两个词语共同出现的概率。但是，旧话重提，这仍然要求储存和计算一个非常的大数据集里面的全部信息。

现在我们理解了“分词序列”的概率（其实就是N-gram语言模型啦），让我们观察一些能够学习到这些概率的例子。

4.2 连续词袋模型（CBOM）

有种模型是以{“The”，“cat”，“over”，“the”，“puddle”}为上下文，能够预测或产生它们中心的词语“jumped”，叫做连续词袋模型。

上面是最粗粒度的描述，咱们来深入一点点，看点细节。

首先，我们要建立模型的一些已知参数。它们就是将句子表示为一些one-hot向量，作为模型的输入，咱们记为 $x(c)$ 吧。模型的输出记为 $y(c)$ 吧。因为连续词袋模型只有一个输出，所以其实我们只需记录它为 y 。在我们上面举的例子中， y 就是我们已经知道的（有标签的）中心词（如本例中的“jumped”）。

好了，已知参数有了，现在我们一起定义模型中的未知参数。我们建立两矩阵， $V \in \mathbb{R}^{n \times |V|}$ 和 $U \in \mathbb{R}^{|V| \times n}$ 。其中的 n 是可以任意指定的，它用来定义我们“嵌入空间”（embedding space）的维度。 V 是输入词矩阵。当词语 w_i （译注： w_i 是只有第 i 维是1其他维是0的one-hot向量）作为模型的一个输入的时候， V 的第 i 列就是它的 n 维“嵌入向量”（embedded vector）。我们将 V 的这一列表示为 v_i 。类似的， U 是输出矩阵。当 w_j 作为模型输出的时候， U 的第 j 行就是它的 n 维“嵌入向量”。我们将 U 的这一行表示为 u_j 。要注意我们实际上对于每个词语 w_i 学习了两个向量。（作为输入词的向量 v_i ，和作为输出词的向量 u_j ）。

连续词袋模型（CBOM）中的各个记号：

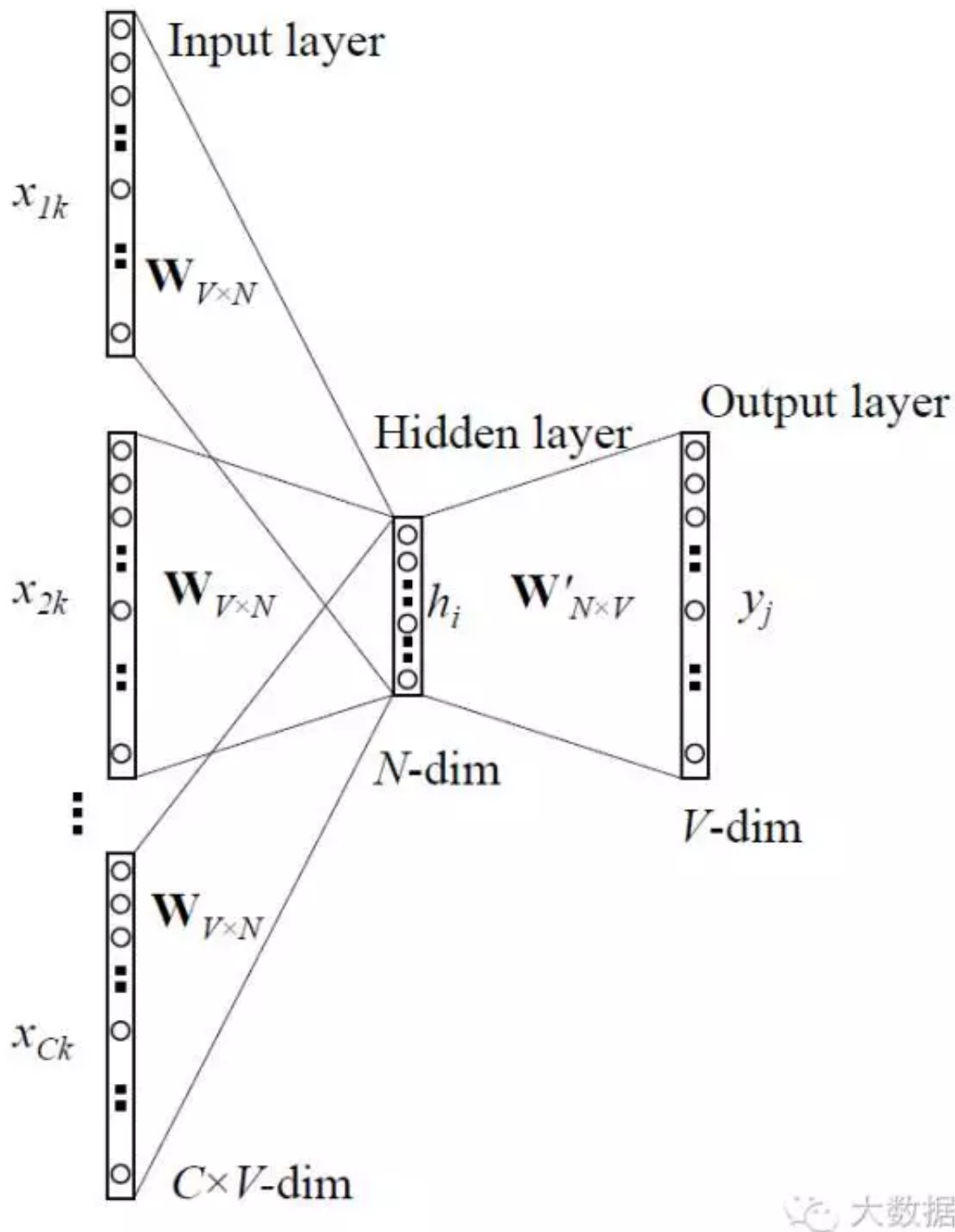
- w_i : 单词表 V 中的第 i 个单词
- $v \in \mathbb{R}^{n \times |V|}$: 输入词矩阵
- v_i : V 的第 i 列，单词 w_i 的输入向量
- $u \in \mathbb{R}^{|V| \times n}$: 输出词矩阵

- u_i : U 的第 i 行，单词 w_i 的输出向量

那这个模型是如何运作的呢？我们把整个过程拆分成以下几步：

1. 对于 m 个词长度的输入上下文，我们产生它们的one-hot向量
($x(c-m), \dots, x(c-1), x(c+1), \dots, x(c+m)$) 。
2. 我们得到上下文的嵌入词向量 ($v_{c-m+1} = Vx(c-m+1), \dots, v_{c+m} = Vx(c+m)$)
3. 将这些向量取平均 $\bar{v} = \frac{v_{c-m} + v_{c-m+1} + \dots + v_{c+m}}{2m}$
4. 产生一个得分向量 $z = U\bar{v}$
5. 将得分向量转换成概率分布形式 $\hat{y} = \text{softmax}(z)$
6. 我们希望我们产生的概率分布，与真实概率分布 y 相匹配。而 y 刚好也就是我们期望的真实词语的one-hot向量。

用一幅图来表示就是下面这个样子：



大数据文摘

通过上面说的种种步骤，我们知道有了矩阵U、V整个过程是如何运作的，那我们怎样找到U和V呢？——我们需要有一个目标函数。通常来说，当我们试图从已知概率学习一个新的概率时，最常见的是从信息论的角度寻找方法来评估两个概率分布的差距。其中广受好评又广泛应用的一个评估差异/损失的函数是交叉熵：

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

 大数据文摘


结合我们当下的例子， y 只是一个one-hot向量，于是上面的损失函数就可以简化为：

$$H(\hat{y}, y) = -y_i \log(\hat{y}_i)$$

 大数据文摘

我们用 c 表示 y 这个one-hot向量取值为1的那个维度的下标。所以在我们预测为准确值的情况下 $y^c = 1$ 。于是损失为 $-1 \log(1) = 0$ 。所以对于一个理想的预测值，因为预测得到的概率分布和真实概率分布完全一样，因此损失为0。现在让我们看一个相反的情况，也就是我们的预测结果非常不理想，此时 $y^c = 0.01$ 。计算得到的损失为 $-1 \log(0.01) \approx 4.605$ ，损失非常大，原本这才是标准结果，可是你给了一个非常低的概率，因此会拿到一个非常大的loss。可见交叉熵为我们提供了一个很好的衡量两个概率分布的差异的方法。于是我们最终的优化函数为：

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

 大数据文摘

我们用梯度下降法去更新每一个相关的词向量 u_c 和 v_j 。

4.3 Skip-Gram 模型

很上面提到的模型对应的另一种思路，是以中心的词语“jumped”为输入，能够预测或产生它周围的词语“The”，“cat”，“over”，“the”，“puddle”等。这里我们叫“jumped”为上下文。我们把它叫做Skip-Gram模型。

这个模型的建立与连续词袋模型（CBOW）非常相似，但本质上是交换了输入和输出的位置。我们令输入的one-hot向量（中心词）为 x （因为它只有一个），输出向量为 $y(j)$ 。 U 和 V 的定义与连续词袋模型一样。

Skip-Gram模型中的各个记号：

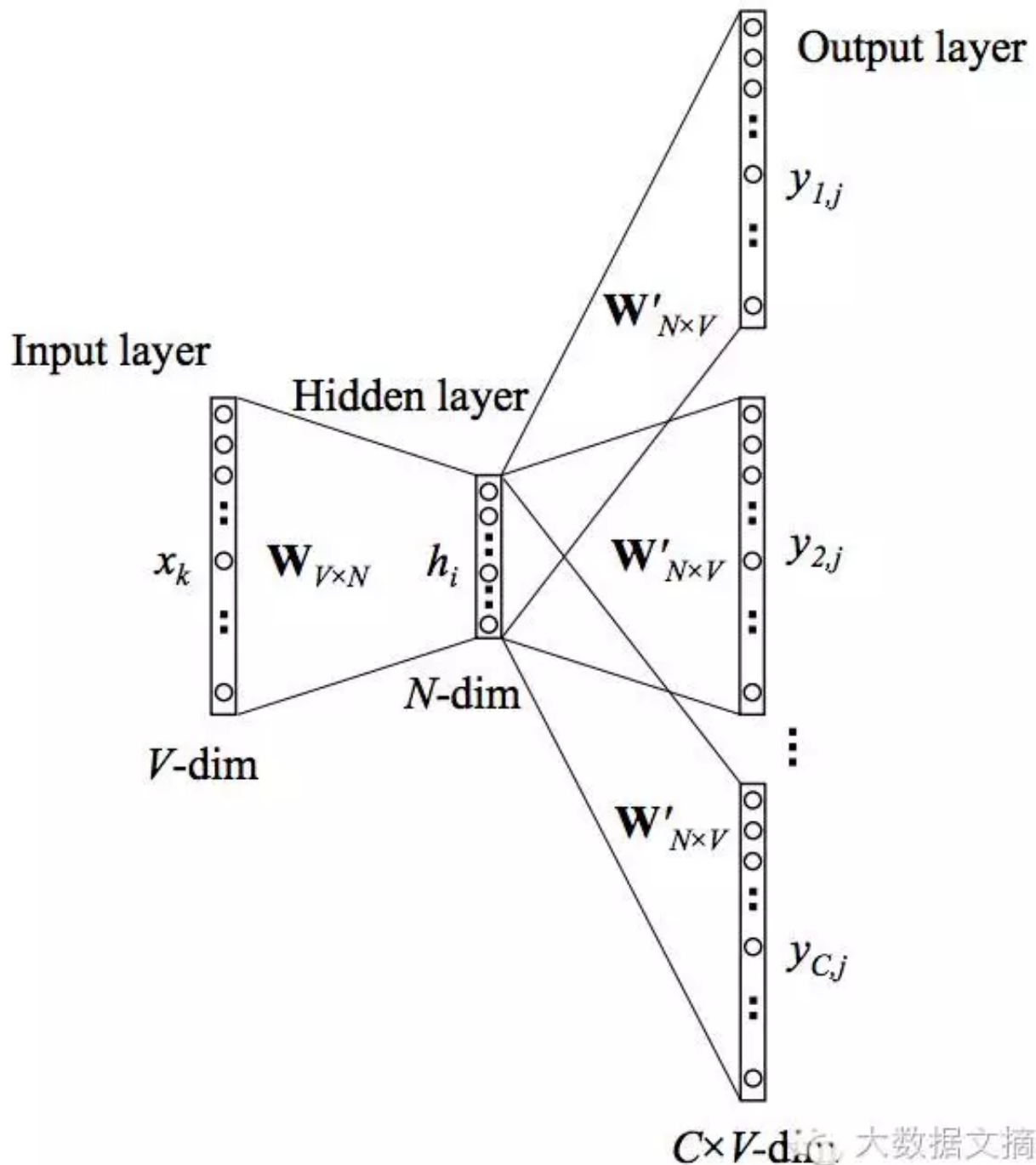
- w_i : 单词表 V 中的第 i 个单词
- $v \in \mathbb{R}^{n \times |V|}$: 输入词矩阵
- v_i : V 的第 i 列，单词 w_i 的输入向量

- $u \in \mathbb{R}^{|V| \times n}$ ：输出词矩阵
- u_i ：U的第i行，单词 w_i 的输出向量

对应到上面部分，我们可以把Skip-Gram 模型的运作方式拆分成以下几步：

1. 生成one-hot输入向量 x 。
2. 得到上下文的嵌入词向量 $vc = Vx$ 。
3. 因为这里不需要取平均值的操作，所以直接是 $v^{\wedge} = vc$ 。
4. 通过 $u = Uvc$ 产生 $2m$ 个得分向量 $u_{c-m}, \dots, u_{c-1}, u_{c+1}, \dots, u_{c+m}$ 。
5. 将得分向量转换成概率分布形式 $y = \text{softmax}(u)$ 。
6. 我们希望我们产生的概率分布与真实概率分布 $y_{c-m}, \dots, y_{c-1}, y_{c+1}, \dots, y_{c+m}$ 相匹配，也就是我们真实输出结果的one-hot向量。

用一幅图来表示这个过程如下：



像连续词袋模型一样，我们需要为模型设定一个目标/损失函数。不过不同的地方是我们这里需要引入朴素贝叶斯假设来将联合概率拆分成独立概率相乘。如果你之前不了解它，可以先跳过。这是一个非常强的条件独立性假设。也就是说只要给出了中心词，所有的输出词是完全独立的。

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

我们可以用随机梯度下降法去更新未知参数的梯度。

4.4 负面抽样 (Negative Sampling)

我们再次观察一下目标函数，注意到对整个单词表 $|V|$ 求和的计算量是非常巨大的，任何一个对目标函数的更新和求值操作都会有 $O(|V|)$ 的时间复杂度。我们需要一个思路去简化一下，我们想办法去求它的近似。

对于每一步训练，我们不去循环整个单词表，而只是抽象一些负面例子就够了！我们可以从一个噪声分布 $P_n(w)$ 中抽样，其概率分布与单词表中的频率相匹配。为了将描述问题的公式与负面抽样相结合，我们只需要更新我们的：

- 目标函数
- 梯度
- 更新规则

Mikolov ET AL.在他的《Distributed Representations of Words and Phrases and their Compositionality》中提出了负面抽样。虽然负面抽样是基于Skip-Gram 模型，它实际上是对一个不同的目标函数进行最优化。考虑一个“词-上下文”对 (w, c) ，令 $P(D = 1 | w, c)$ 为 (w, c) 来自于语料库的概率。相应的， $P(D = 0 | w, c)$ 则是不来自于语料库的概率。我们首先对 $P(D = 1 | w, c)$ 用sigmoid函数建模：


$$p(D = 1 | w, c, \theta) = \frac{1}{1 + e^{(-u_c^T v_w)}}$$

现在我们需要建立一个新的目标函数。如果 (w, c) 真是来自于语料库，目标函数能够最大化 $P(D = 1 | w, c)$ 。反之亦然。我们对这两个概率采用一个简单的最大似然法。（这里令 θ 为模型的参数，在我们的例子中，就是对应的 U 和 V 。）

$$\begin{aligned}
\theta &= \operatorname{argmax}_{\theta} \prod_{(w,c) \in D} P(D=1|w,c,\theta) \prod_{(w,c) \in \tilde{D}} P(D=0|w,c,\theta) \\
&= \operatorname{argmax}_{\theta} \prod_{(w,c) \in D} P(D=1|w,c,\theta) \prod_{(w,c) \in \tilde{D}} (1 - P(D=1|w,c,\theta)) \\
&= \operatorname{argmax}_{\theta} \sum_{(w,c) \in D} \log P(D=1|w,c,\theta) + \sum_{(w,c) \in \tilde{D}} \log(1 - P(D=1|w,c,\theta)) \\
&= \operatorname{argmax}_{\theta} \sum_{(w,c) \in D} \log \frac{1}{1 + \exp(-u_w^T v_c)} + \sum_{(w,c) \in \tilde{D}} \log \left(1 - \frac{1}{1 + \exp(-u_w^T v_c)}\right) \\
&= \operatorname{argmax}_{\theta} \sum_{(w,c) \in D} \log \frac{1}{1 + \exp(-u_w^T v_c)} + \sum_{(w,c) \in \tilde{D}} \log \left(\frac{1}{1 + \exp(u_w^T v_c)}\right)
\end{aligned}$$

注意这里的 \tilde{D} 表示“错误的”或者“负面的”语料库，像句子“stock boil fish is toy”就是从这样的语料库来的。不自然的句子应该有比较低的发生概率，我们可以从词库中随机采样来产生这样的“负面的”语料库。我们的新目标函数就变成了：

$$\log \sigma(u(c - m + j)^T \cdot v_c) + \sum_{k=1}^K \log \sigma(-\tilde{u}_k^T \cdot v_c)$$

 大数据文摘

在这里 $\{\tilde{u}_k | k=1, \dots, K\}$ 是从 $P_n(w)$ 中抽样取到的。需要多说一句的是，虽然关于怎么样最好地近似有许多讨论和研究，但是工作效果最好的似乎是指数为3/4的一元语言模型。至于为什么是3/4，下面有几个例子来帮助大家感性地理理解一下：

$$\begin{aligned}
is &: 0.9^{3/4} = 0.92 \\
constitution &: 0.09^{3/4} = 0.16 \\
bombastic &: 0.01^{3/4} = 0.032
\end{aligned}$$

 大数据文摘

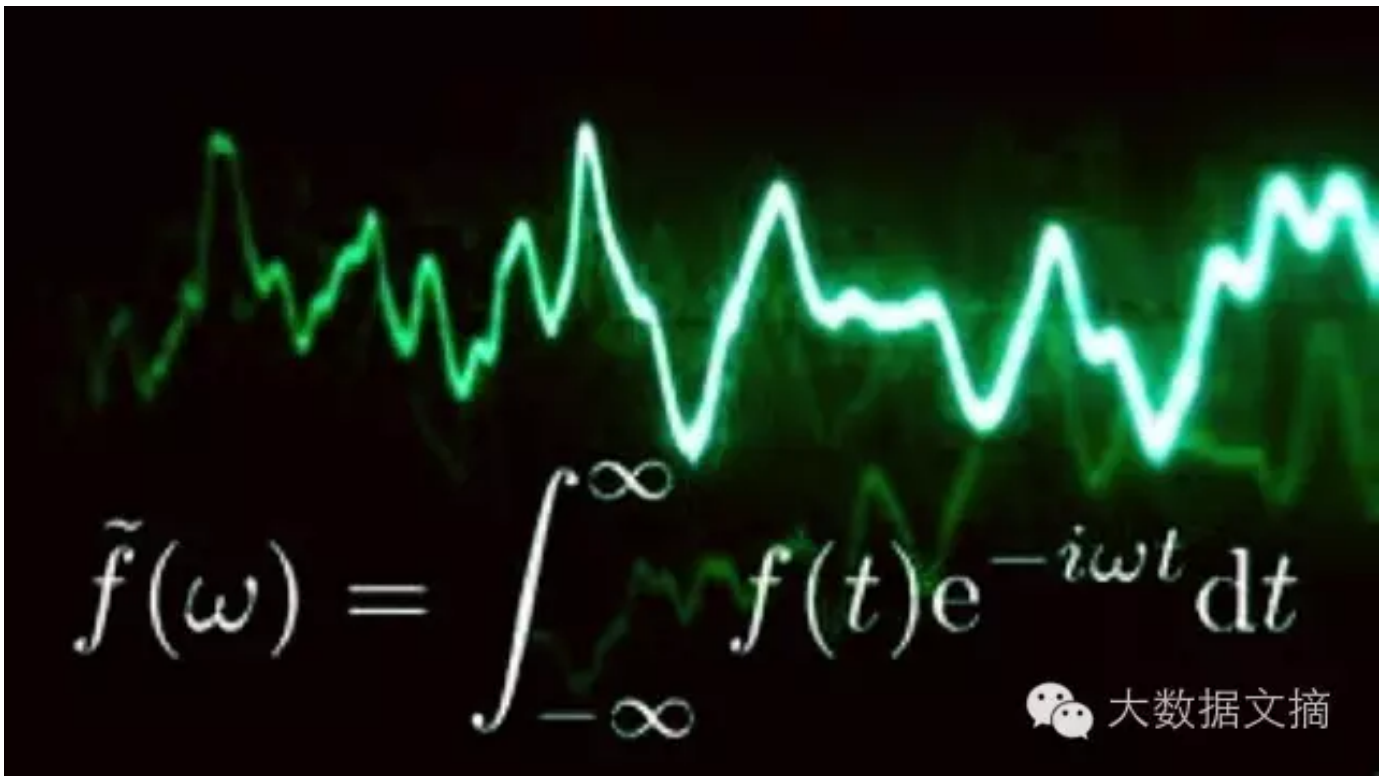
你看，经过3/4这样一个指数处理，“Bombastic”（少见）被采样的概率是之前的3倍，而“is”这个词（多见）被采样的概率只是稍微增长了一点点。

往期精彩文章推荐，点击图片可阅读

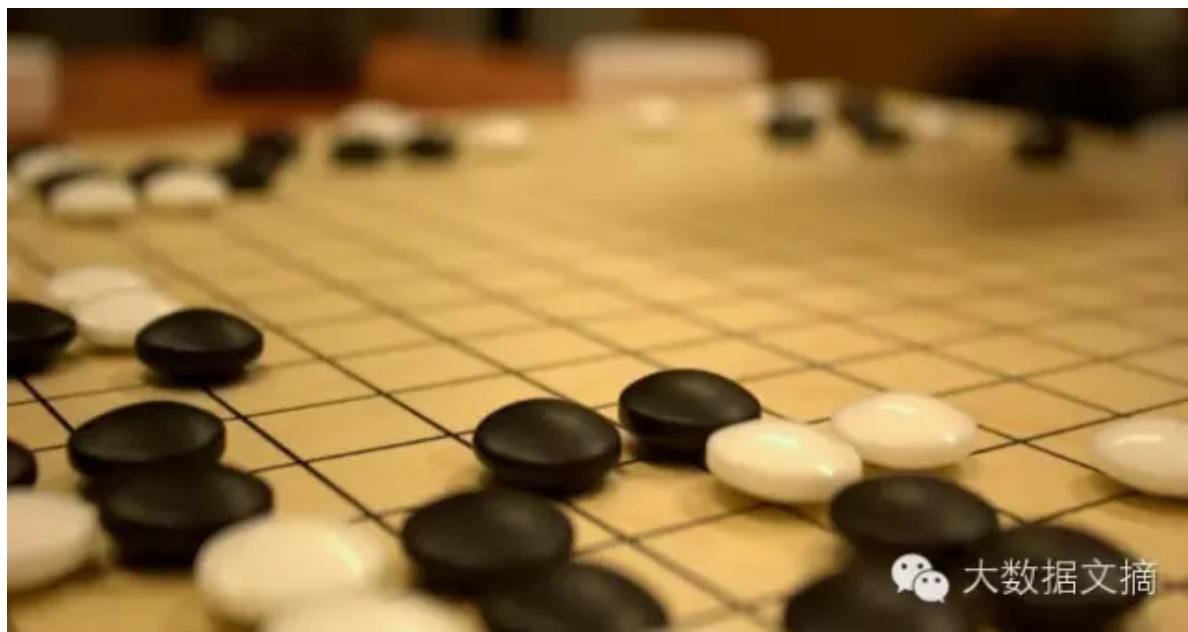
机器学习算法一览（附python和R代码）



神作：深入浅出傅里叶变换



读《Nature》论文，看AlphaGo养成机器



机器学习专栏介绍

《机器学习》是由大数据文摘和机器学习志愿者共同推出的聚焦于机器学习领域的专栏，内容将涵盖机器学习的基本理念、算法、国内外动态、深入见解等，每个月将会至少更新一次。

栏目主编：寒小阳与龙心尘，多年机器学习/数据挖掘/深度学习项目经验，专注海量数据上机器学习算法的应用与优化。坚持通过技术博客等方式交流分享机器学习心得体会。



寒小阳

高级算法工程师，北邮研究生毕业后一直从事互联网机器学习/数据挖掘/算法相关工作。有文本检索与挖掘、CTR预估、推荐系统、深度学习图像识别/分类/检索应用相关项目经验。希望能够通过大数据文摘平台，认识更多热爱机器学习的小伙伴，一起交流和分享机器学习相关的知识和应用经验，联系方式：hanxiaoyang.ml@gmail.com



龙心尘

高级算法工程师，北航毕业，多年互联网机器学习相关工作经验。有NLP、机器学习在安全领域应用，电商用户画像与搜索排序相关项目经验。希望能够通过大数据文摘平台，认识更多热爱机器学习的小伙伴，一起交流和分享机器学习相关的知识和应用经验，联系方式：
johnnygong.ml@gmail.com

招募要求：**1.背景需求（满足其一即可）：**

专业是机器学习方向、数学统计方向或者ee方向

从事机器学习、数据挖掘相关工作

参加过相关数据挖掘竞赛或有相关项目经历者优先

2.英文水平

翻译能力表达能力尚佳

3.工作投入

每周3个小时时长，持续3个月

如果你也是对是相关专业的大拿，如果你也对数据科学/机器学习感兴趣，且乐于分享，请点击文末 [“阅读原文”](#) 报名加入我们。



普及数据思维 传播数据文化

We talk data We deliver knowledge

关于转载授权大数据文摘作品，
请在大数据文摘后台留言“机构名称+文章标题+转载”，
申请过授权的不必再次申请，只要按约定转载即可。

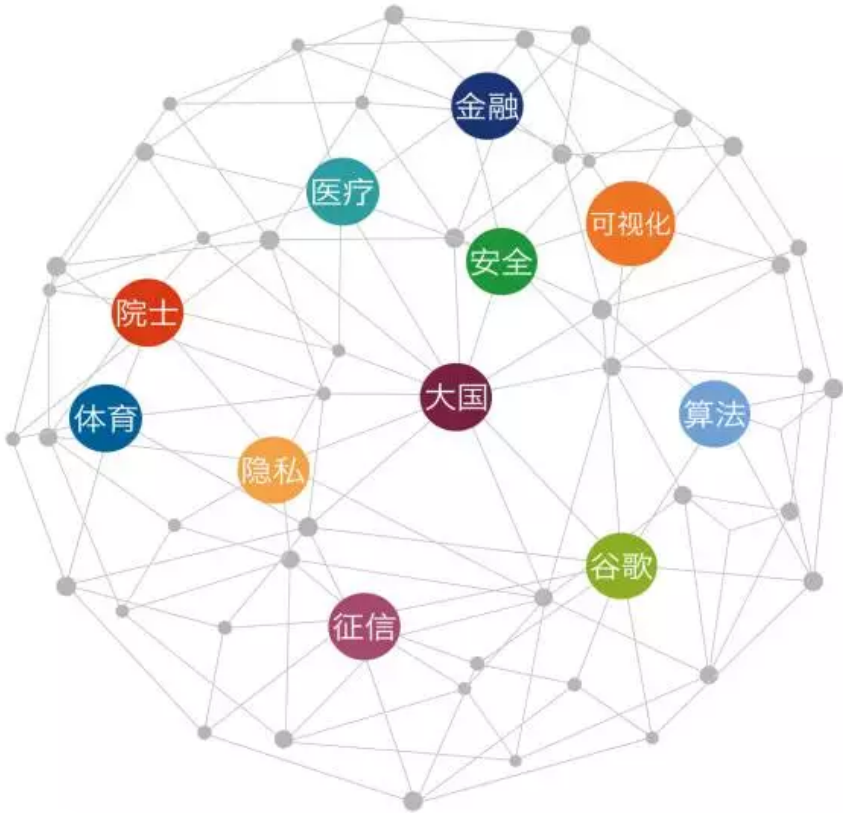


长按二维码 · 关注大数据文摘

动图

2015年2-3月干货文件打包下载，请点击大数据文摘底部菜单：[下载等.2-3月下载](#)





动图



动图

回复【志愿者】了解如何加入大数据文摘

动图

阅读原文