使用词袋表示法(TF-IDF)，如果文集包含 10,000 个单词，那么表示每个文本需要用一个 10,000 维的向量，如果文本只有 1,000 个单词，那么这个向量有 9,000 个位置为 0，高维的稀疏的向量严重影响处理的速度。基于 SVD 的潜在语义分析，相对于词袋表示法，一定程度解决了维度灾难问题。

7.2.3 文本表示总结与延伸

在第一章“数据科学概论”以及本章的前文，我们已经对文本的几种表示法进行了介绍，但是不成体系。在这里，我们对文本的表示做一个简单总结，并且做进一步的延伸讨论。

1. 文本的表示和单词向量化(Word Vectorization)

在自然语言处理中，我们要让计算机能够理解自然语言、甚至能够生成自然语言。第一个挑战是，人们是通过单词和语句来进行沟通的，而计算机则只能处理数字。于是，首先需要把文本转换成计算机能够处理的数字。

把一个个的单词转换为向量的形式，这就是单词的表示形式(Representation)。我们希望这个表示形式，能够捕抓到单词的意思(Meanings)、单词之间的语义关系(Semantic Relationships)、以及单词所出现的上下文等必要的信息。

2. 独热编码(One-Hot Encoding)

对单词进行编码，最为直接的方式是独热编码。其基本原理很简单，在第一章已经简单介绍过。

首先查看整个词汇表有多少个单词，比如有 1,500 个。在此基础上，建立一个有序表，序号从 0 到 1,499，每个位置代表一个单词。

对于某个单词，我们对其进行向量表示的时候，把它表示为一个 1,500 维的向量，向量的第 i 个分量(对应这个单词)置为 1，其余分量都置为 0。

比如，假设整个字典有三个单词，分别是 Monkey、Eat 和 Banana。那么表示 Monkey 的向量为<1, 0, 0>，表示 Eat 的向量为<0, 1, 0>，最后表示 Banana 的向量为<0,0,1>。

在此基础上，可以对文档(Document)进行编码。如果文档出现两个单词，把两个单词的向量相加即可得到文档的编码。比如，一个文档出现 Monkey 和 Banana，整个文档的向量化表示为<1,0,1>，即<1,0,1>=<1,0,0>+<1,0,1>。

3. 词频(Term Frequency)

我们看到“Monkey eat banana”和“Monkey eat banana banana”两句话，是有区别的。第二句话强调了两次 Banana。以独热方式编码，两个文档都表示为<1,1,1>，没看出区别来。

为此，引入基于词频的向量化方法，在文档表示的向量的各个分量上，保存的是各个单词的频率(词频)，而不是 0/1 而已。比如上述两个文档分别表示为<1,1,1>，<1,1,2>。注意，词频包括绝对词频和相对词频，绝对词频即词项计数(Counter)，相对词频需要把绝对词频除以文档的长度。这里指的是绝对词频。

3.1 文档的向量表示和单词的向量表示的关系

假设有个文集 C，有 D 个文档$\{d_1,d_2,\ldots,d_D\}$，里面包含 N 个不同的单词(Unique Tokens)，那么这 N 个单词构成一个字典。

整个文集可以表示为 D*N 的矩阵 M，M 的第 i 行第 j 列对应的元素，表示第 i 个文档中，第 j 个单词出现的频率。

比如，有一个文集包含两个文档。D1: He is a lazy boy. She is also lazy。D2: Neeraj is a lazy person。那么字典表(已经去掉停用词)为['He', 'She', 'lazy', 'boy', 'Neeraj', 'person']。这里，D=2，N=6。

矩阵 M 的具体取值，如表 2 所示。

表 7-2 文档的向量表示和单词的向量表示

	He	She	lazy	boy	Neeraj	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

从行的方向看，我们看到了 D1 和 D2 的向量化表示。而从列的方向看，我们可以把每列看作单词的向量化表示，比如 lazy 的向量化表示为$<2,1>^T$。

可以看到，基于词频的向量化，对文档进行向量化的同时，也对单词进行了向量化。

4. TF-IDF

基于绝对词频的向量化表示，存在内在的缺陷。(1) 首先，第一个问题是文档长度和词频的问题。比如，D1 很短，只有 10 个单词，出现 5 次 Lazy，而 D2 很长，有 1,000 个单词，出现 10 个 Lazy。从 Lazy 的“浓度”来看，D1 好像更浓一点，但是基于词频的向量化表示无法告诉我们这些。

(2) 此外，还有一个问题，在整个文集中，有些单词几乎每个文档都出现，区分度不明显，比如"a"、"the"。而有些单词，仅仅在整个文集的某几个文档出现，极具区分度。比如，整个文集只有两个文档谈论梅西，从而包含"Messi"这个单词，而文集中的其它文档都没有"Messi"这个单词。

解决上述问题的办法，是在 D*N 的矩阵中，每个单元格保存的不是词频，而是 TF-IDF 权重。其中 TF 解决第一个问题，IDF 解决第二个问题。

TF = (Number of times term t appears in a document)/(Number of terms in the document)。

TF 是一个相对词频。比如，D1 的 Lazy 的浓度为 5/10，而 D2 的 Lazy 的浓度为 10/1000=1/100，D1 里面的 Lazy 的浓度更高，很合理。

IDF = log(D/Dt)，其中 D 是整个文档集的文档数量，Dt 表示包含单词 t 的文档的数量。比如，文档数量为 200，所有文档都出现"the"，那么 IDF("the")=log(200/200)=log(1)，只有 2 个文档出现"Messi"，那么 IDF("Messi")=log(200/2)=log(100)，可以看到 log(100)>log(1)，IDF 补偿了"Messi"的重要性。在此基础上，TF-IDF=TF * IDF。

表 7-3 文档的 TF-IDF 向量化表示

	He	She	lazy	boy	Neeraj	person
D1	tf-idf(he,d1)	tf-idf(she,d1)	tf-idf(lazy,d1)	tf-idf(boy,d1)	tf-idf(Neeraj,d1)	tf-idf(person,d1)
D2	tf-idf(he,d2)	tf-idf(she,d2)	tf-idf(lazy,d2)	tf-idf(boy,d2)	tf-idf(Neeraj,d2)	tf-idf(person,d2)

我们观察表 3 所表达的矩阵，它和表 2 的区别，只是每个单元格从词频(绝对词频 Counter)，变成 TF-IDF 权重。同样，行向量就是文档的向量化，列向量就是单词的向量化。

5. 按照固定大小的上下文窗口创建的单词共现矩阵

我们看两个句子，“Apple is a fruit”和“Mango is a fruit”，我们看到 Apple 和 Mango 是相似的单词，它们都是水果，它们都和单词 Fruit 在一块。也就是相似的单词经常一起出现，或者出现在类似的上下文。共现 (Co-occurrence)，指的是在一个给定的文集中，单词 w1 和单词 w2 在一定大小的上下文窗口中，共同出现的次数。

上下文窗口 (Context Window)，指的是以某个单词为基准的左边和右边的若干单词构成的窗口。比如 Fox 左右两侧灰色单元格里面的单词，构成单词 Fox 的宽度为 2 的上下文窗口。

Quick	Brown	Fox	Jump	Over	The	Lazy	Dog

对于 Over 这个单词来讲，它的宽度为 2 的上下文窗口，则如下所示。

Quick	Brown	Fox	Jump	Over	The	Lazy	Dog

假设，现在我们有一个文集，包含三个文档，具体如下：

Corpus = He is not lazy. He is intelligent. He is smart.

我们就可以计算单词之间的共现矩阵(Co-Occurrence Matrix)[91]。

	He	is	not	lazy	intelligent	smart
He	0	4	2	1	2	1
is	4	0	1	2	2	1

[91] https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.

not	2	1	0	1	0	0
lazy	1	2	1	0	0	0
intelligent	2	2	0	0	0	0
smart	1	1	0	0	0	0

对于这个矩阵中的深色单元格里面的 4 和 0，我们来分析这个频率是怎么计算出来的。

灰色单元格里面的 4 表示‘He’ 和‘is’ 在大小为 2 的上下文窗口中共现的次数。从以下表格中我们可以看到，它们‘He’ 和‘is’共现的次数是 4。

句子	He is not lazy	He is intelligent	He is smart
上下文	He 的上下文 He is not is 和 not 的上下文 He is not lazy lazy 的上下文 is not lazy	三个单词的上下文都是 He is intelligent	三个单词的上下文都是 He is smart

而单词 ‘lazy’和‘intelligent’ 没有在这样大小的上下文窗口中共现过，所以计数为 0。

这个共现矩阵并不是可以直接使用的单词的向量表示(Word Vector Representation)。需要使用一定的方法进行分解，分解为为一系列因子(Factors)，包括 PCA 和 SVD[92]等。这些因子的组合，称为单词的向量化表示。

比如，我们对 V*V 的共现矩阵进行 PCA 分解，我们得到 V 个主成分，我们可以从 V 个主成分中，选择 K 个主成分，构成新的矩阵为 V*K。这时候，每个单词就可以表示为 K 维向量，而不是 V 维向量。K 的大小一般选择几百为宜。

上述共现矩阵方法的主要优势有：(1) 它保留了单词之间的语义关系(Semantic Relationship)，比如 man 和 woman 比起 man 和 apple 更加接近。(2) 加上 PCA/SVD 处理，可以产生比其它方法更加准确的单词的向量表示。(3) 经过计算后，可以多次使用。

共现矩阵的主要劣势是，进行计算的时候，需要大量的内存来存储共现矩阵。

6. CBOW 和 Skip-Gram

一般来讲，我们通过一个单词出现的上下文，能够猜测出它所表达的意思。或者说，一个单词和另外一个单词的意思是相似的，如果它们出现在类似的上下文中，于是可以互相替换。这就是所谓的“分布式假说”(Distributional Hypothesis)。它的核心思想是上下文相似的词，其语义也相似。

这个假说给了我们构造词向量的思路。One-Hot Encoding、绝对词频向量化、以及 TF-IDF 向量化都是高维稀疏的表示法，处理起来不是很方便。我们希望构造单词的某种稠密的向量化表示，只要表示类似含义的单词，它们的词向量表示也相似即可。

这种类型的向量化表示，也称为词嵌入(Word Embedding)[93]。词汇表中的词语被映射到由实数构成的稠密向量里。词嵌入，是一系列语言模型和特征学习技术的统称。

注意，传统的词袋模型，也是用向量表示词项，但是这样的向量是高维和稀疏的。一般来讲，在 NLP 语境中，词向量表示通过各种技术得到的词项的低维的稠密的向量表示。词向量的思想，来源于 Hinton 提出的词的分布式表示(Distributed Representation)，之后 Bengio 把这个概念引入到语言模型中，提出了神经网络语言模型(Neural Network Language Model)。

近年来，利用神经网络来预测(计算)单词的向量化表示，成为研究的热点，取得了意想不到的效果。Google 提出的词向量技术 Word2vec，广为人知，它是为了利用神经网络从大量无标注的文本中提取有用信息而设计的。

它的基本的过程是，用单词和单词的上下文来训练一个神经网络结构，然后用网络结构训练后形成的连接权重来表示各个单词。主要的策略包括 CBOW 和 Skip-Gram 两种。

6.1 Continuous Bag of Words (CBOW)

---

[92] 网址 https://math.stackexchange.com/questions/3869/what-is-the-intuitive-relationship-between-svd-and-pca 给出了 PCA 和 SVD 的区别和联系; 关于 PCA 算法的具体细节，请参考“数据的深度分析(下)”一章.

[93] https://monkeylearn.com/blog/word-embeddings-transform-text-numbers/.

假设我们有个文集，包含一个文档“Hey, this is sample corpus using only one context word.”。在此基础上，我们使用 One-Hot Encoding 对各个单词进行编码，如表 4 所示。

表 7-4 One-Hot Encoding

单词	Hey	this	is	sample	corpus	using	only	one	context	word
Hey	1	0	0	0	0	0	0	0	0	0
this	0	1	0	0	0	0	0	0	0	0
is	0	0	1	0	0	0	0	0	0	0
sample	0	0	0	1	0	0	0	0	0	0
corpus	0	0	0	0	1	0	0	0	0	0
using	0	0	0	0	0	1	0	0	0	0
only	0	0	0	0	0	0	1	0	0	0
one	0	0	0	0	0	0	0	1	0	0
context	0	0	0	0	0	0	0	0	1	0
word	0	0	0	0	0	0	0	0	0	1

在这张表中，每行表示各个单词的独热向量化表示，即独热编码。

CBOW 的基本思路是，用一个大小为 N 的滑动窗口，扫描整个文集的各个文档，在某个单词的两边，各寻找 N 个单词。这 2*N 个单词构成输入，而这个单词作为输出，构造一个训练样本。用一系列的样本训练神经网络。如图 11(a)所示。

比如，我们用“corpus”单词两边的上下文(One-Hot Encoded Vectors)，来预测“corpus”。

输入如下，

Is	0	0	1	0	0	0	0	0	0	0
sample	0	0	0	1	0	0	0	0	0	0
Using	0	0	0	0	0	1	0	0	0	0
Only	0	0	0	0	0	0	1	0	0	0

输出如下，

Corpus	0	0	0	0	1	0	0	0	0	0

这个神经网络模型有一个输入层，如果 N 为 2，那么它接受 4 个单词的 One-Hot Encoded 向量，每个向量都是 1*V 的大小(V 是字典表的大小)(在图 11(a)所示的 CBOW 神经网络中，输入是一系列的 V*1 向量)。

CBOW 的计算过程如下：

(1) 输入数据是 4 个 V*1 的向量，首先求均值$\bar{x}$。

W 的大小是 V*N_{dim}，N_{dim}是一个为了创建稠密的向量表示而设定的一个维度，V 可能是 10,000，而 N_{dim} 可能是 300，N_{dim}<<V。

$h = \frac{1}{C} W^T \sum_{c=1}^{C} x^{(c)} = W^T \bar{x}$ (注：N_{dim} *V * V *1)

(2) 在 h 基础上计算 u

$u = W'^T h = \frac{1}{C} \sum_{c=1}^{C} W'^T W^T x^{(c)} = W'^T W^T \bar{x}$ (注：V*N_{dim}*N_{dim}*1)

W'是一个 N_{dim} *V 的权重矩阵。读者可以注意到，$W'^T W^T \bar{x}$向量计算的维度关系为(V*N_{dim}) *(N_{dim} *V) *(V*1)，所以输出是一个 V*1 的向量。

(3) 最后进行 SoftMax 规范化

$$y = Softmax(u) = Softmax(W'^T W^T \bar{x})$$

SoftMax 函数，也称归一化指数函数，是逻辑函数的一种推广。它能将一个含任意实数的 K 维的向量 z 的“压缩”到另一个 K 维实向量 σ(z) 中，使得每一个元素(分量)的范围都在(0,1)之间，并且所有元素(分量)的和为 1。具体的公式如下：

$$f(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{n} e^{z_i}}$$

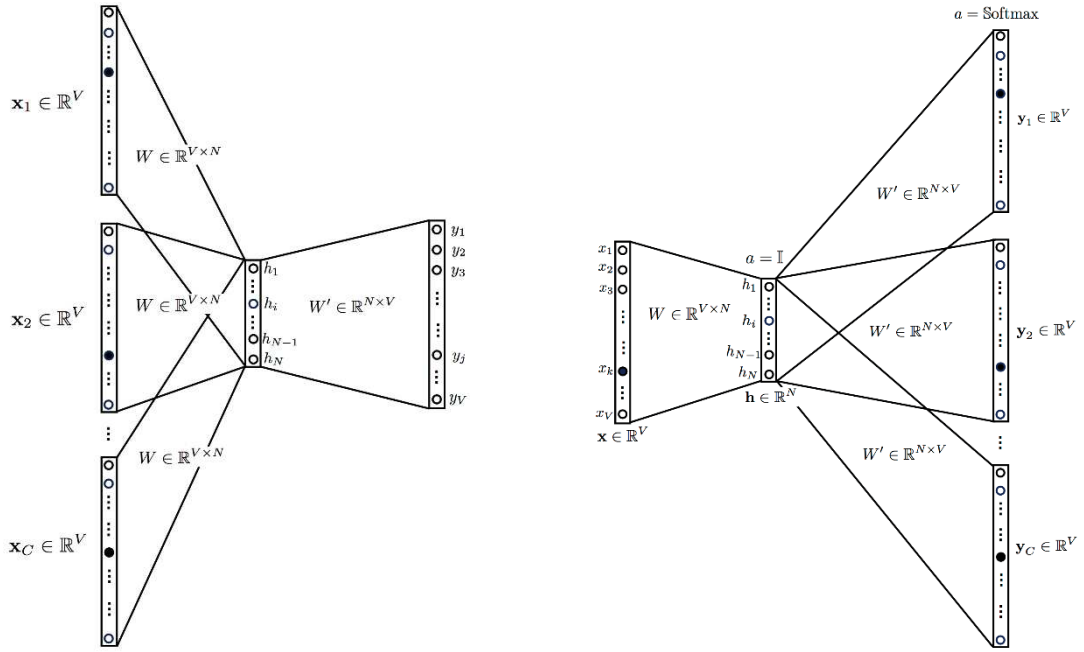
SoftMax 输出，即 y 的第 i 个位置的分量，表示这个输出为第 i 个单词的概率。

经过上述计算，得到神经网络的输出，和预期的输出(即“corpus”的 One-Hot Encoded Vector)比较，经过误差的反向传播来修改 W 和W'。

Corpus	0	0	0	0	1	0	0	0	0	0

注意，W 是一个 V*N_{Dim} 的矩阵，这个矩阵的第 i 行，是一个 N_{dim} 向量，可以用来作为 i 个 Word 的词向量，具体如下[94]，

$$[0\ 0\ \ldots\ 1\ 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ \ldots & \ldots & \ldots \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10\ 12\ 19]$$



(a) CBOW (b) Skip Gram

图 7-11 CBOW 的网络结构与 Skip Gram 网络结构[95]

6.2 Skip-Gram

CBOW 是利用上下文预测一个单词，而 Skip-Gram 则是用某个单词预测它的上下文。比如，我们用“corpus”来预测“corpus”单词两边的上下文(One-Hot Encoded Vectors)。

输入如下，

Corpus	0	0	0	0	1	0	0	0	0	0

输入如下，

Is	0	0	1	0	0	0	0	0	0	0
sample	0	0	0	1	0	0	0	0	0	0
Using	0	0	0	0	0	1	0	0	0	0
Only	0	0	0	0	0	0	1	0	0	0

用一个样本，预测四个样本，好像不太可能。实际上可以的，最后的结果往往比 CBOW 还要好。具体的网络结构如图 11(b)所示。

Skip-Gram 的计算过程为：

(1) 输入数据是 1 个 V*1 的向量，W 是一个 V*N_{dim} 的权重矩阵，计算 h。

$h = W^T x$ (注：N_{dim}*V*V*1)

[94] https://www.leiphone.com/news/201706/PamWKpfRFEI42McI.html.
[95] http://www.claudiobellei.com/2018/01/06/backprop-word2vec/.

(2) $W'^T W^T x$，计算 u_c。W'是一个 N_{dim} *V 的权重矩阵。

$u_c = W'^T h = W'^T W^T x, c = 1, \ldots, C$ (注：V*N_{dim}*N_{dim}*1)

$Y_c = Softmax(u_c) = Softmax(W'^T W^T x), c = 1, \ldots, C$

输出向量是一样的(包括 u_c)，即 $y_1=y_2\cdots=y_C$。。

利用 4 个 V*1 的输出和预期的输出进行比较，构造损失函数。对误差进行反向传播，修改W'和 W。

具体来讲，模型希望输出的 One-Hot Encoded Vector，为所有 c 个上下文词语对应 One-Hot Encoded Vector 的位置都是 1，而其它位置则都是 0，用这个作为损失函数的依据，进行反向传播和训练。输出对应 corpus 的上下文，One-Hot Encoded Vectors 如下。

Is	0	0	1	0	0	0	0	0	0	0
sample	0	0	0	1	0	0	0	0	0	0
Using	0	0	0	0	0	1	0	0	0	0
Only	0	0	0	0	0	0	1	0	0	0

那么四个向量的求和结果如下，即用这个向量，评价模型的损失。

Sum	0	0	1	1	0	1	1	0	0	0

最后，表示每个单词的 Vector 是从W'产生的。具体办法，请参考上文的 CBOW 模型的做法。

7. Word Embedding 的一些有趣的结果[96]

Word Embedding 能够发现语言里的一些相似性和语义关系。比如文献[97][98][99][100]找到了特定名词的性别关系、名词的单数和复数关系等，如图 12 所示。

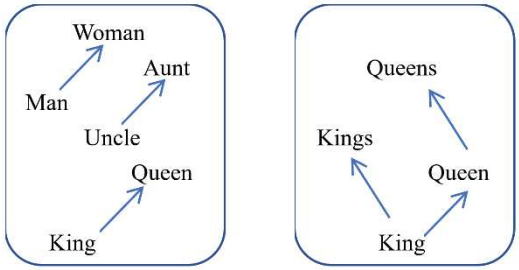


图 7-12 Word Embedding—名词的性别、名次的单数和复数

他们还发现了国家和首都的对应关系，如图 13 所示。

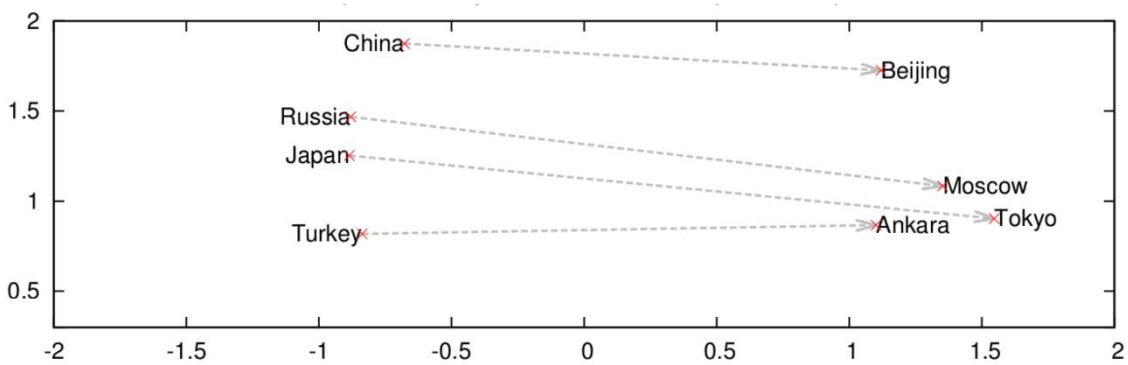


图 7-13 Word Embedding—国家和首都关系(局部)[92]

[96] https://www.springboard.com/blog/introduction-word-embeddings/.

[97] Tomas Mikolov, Wen-tau Yih, Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. Proceedings of NAACL-HLT 2013, pages 746–751.

[98] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781, 2013.

[99] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. https://arxiv.org/abs/1310.4546, 2013.

[100] Tomas Mikolov, Quoc V. Le, Ilya Sutskever. Exploiting Similarities among Languages for Machine Translation. https://arxiv.org/abs/1309.4168, 2013.

由于每个单词的意义，蕴含在这个单词和其它单词的关系中。模型学习到的词向量，体现了某种语言里的某种结构(单词间的关系)，和另外一种语言里的结构具有相似性。预示着 Word Embedding 可以用在机器翻译中。如图 14 所示。

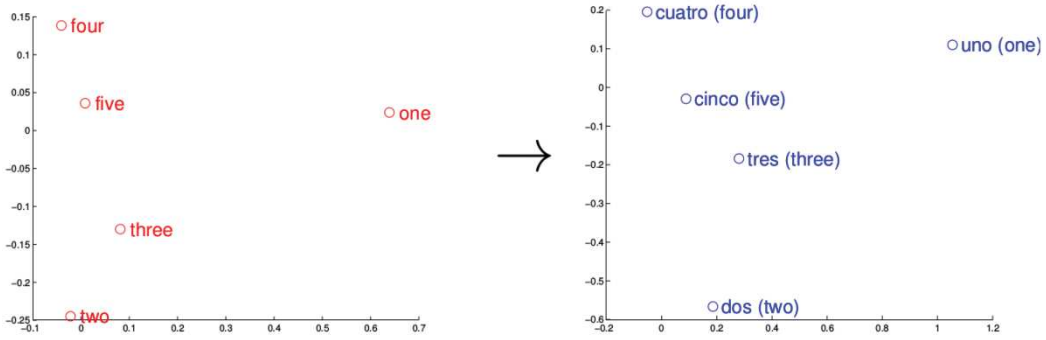


图 7-14 Word Embedding—语言间的相似性(局部)[92]

第一个 Word Embedding 技术即 Word2vec 很成功，激发了一系列 Word Embedding 技术的研发，包括 WordRank、Stanford 的 GloVe、以及 Facebook 的 FastText 等。

这些技术[101]一方面致力于改善 Word2vec，一方面从不同粒度考虑对文本的向量化，包括字符(Character)、单词成分(Sub-word)、单词(Word)、短语(Phrase)、句子(Sentence)、文档(Document)等。使得我们不仅可以在单词层面，也可以在句子层面、和文档层面考虑它们的相似度，如图 15 所示。

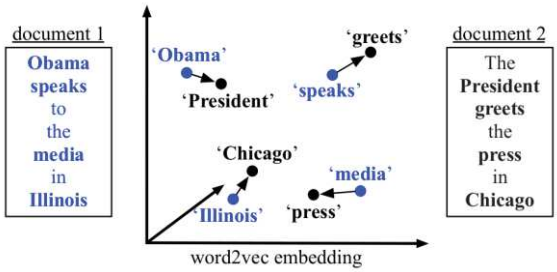


图 7-15 文档相似度

一般来说词语是表达语义的基本单元。有相当一部分研究者将文章或者句子作为文本处理的基本单元，于是产生了 doc2vec 和 str2vec 技术。注意，对文本进行向量化，大部分的研究都是通过词向量化实现的。

词嵌入(Word Embedding)技术把文本里面蕴含的知识，想办法向量化，把文本转化为一种有意义的形式，方便进行后续处理。

8. Word Embedding 的应用

词向量可以作为深度学习模型的输入，执行一系列下游任务。词向量也可以作为传统机器学习模型比如支持向量机(SVM)、朴素贝叶斯分类器(NB)等的输入，完成分类等不同的机器学习任务。

本质上，我们可以把词向量看作是单词的浓缩的表示。这种浓缩的表示，捕抓了语言的上下文中表达的单词的意义和单词的关系，有利于进行迁移学习(Transfer Learning)。迁移学习是一种把一个领域学习到的知识，应用到一个新领域、解决其问题的技术。

7.2.4 文本分类(Text Classification)

文本分类，是把文档集合中的每个文档，划分到一个预先定义的一个主题类别(Predefined Subject Category)。文本分类是文本分析和挖掘的一项重要工作。把电子邮箱收到的邮件，适当进行分类，分为正常邮件和垃圾(Spam)邮件，就是文本分类的一个应用实例。此外，确定文档的作者(Authorship

[101] Matt Kusner, Yu Sun, Nicholas Kolkin, Kilian Weinberger. From word embeddings to document distances. ICML 2015, pp. 957-966.