6 向量语义与嵌入

> 荃者所以在鱼,得鱼而忘荃 Nets are for fish; Once you get the fish, you can forget the net.
>
> 言者所以在意,得意而忘言 Words are for meaning; Once you get the meaning, you can forget the words.
>
> 庄子(Zhuangzi), Chapter 26

洛杉矶因其高速公路上的沥青而闻名。然而，在这座城市的中心，还有一片不同的沥青——拉布雷亚沥青坑（La Brea Tar Pits）。这片沥青保存了数百万件化石骨骼，它们来自更新世时期的最后一个冰河时代。其中一件化石是剑齿虎（Smilodon），其长长的犬齿使其立刻能被识别出来。大约五百万年前，另一种完全不同的剑齿虎——袋剑虎（Thylacosmilus）生活在阿根廷和南美洲的其他地区。袋剑虎是有袋类动物，而剑齿虎是胎盘类哺乳动物，但袋剑虎也拥有相同的长上犬齿，并且像剑齿虎一样，袋剑虎的下颌骨上也有一个保护性骨板。这两种哺乳动物的相似性是许多平行或趋同进化的例子之一，在这些例子中，特定的环境或情境导致了不同物种演化出非常相似的结构（Gould, 1980）。

在一种与生物无关的“有机体”——词语——的相似性中，上下文（context，语境）的作用同样重要。出现在相似语境中的词语往往具有相似的含义。这种词语分布相似性与意义相似性之间的联系被称为**分布假设**（**distributional hypothesis**）。这一假设最早由20世纪50年代的语言学家提出，如Joos（1950）、Harris（1954）和Firth（1957），他们发现，像*oculist*和*eye-doctor*这样的同义词往往出现在相同的环境中（例如，靠近像“*eye* 眼睛”或“*examined* 检查”这样的词），并且两个词之间的意义差异“大致对应于它们在环境中的差异”（Harris, 1954, p. 157）。

在本章中，我们将介绍**向量语义**（**vector semantics**），它通过直接从文本的分布中学习单词的意义表示（称为**嵌入**，**embeddings**）来实现这种语言学假设。这些表示用于每一个涉及到语义的自然语言处理应用中，而我们在此介绍的**静态嵌入**（**static embeddings**）是更强大的**动态**或**上下文化嵌入**（如**BERT**）基础（这些嵌入将在第11章中讨论）。

这些词的表示也是本书中**表示学习（representation learning）**方法的第一个例子，即自动学习输入文本的有用表示，而不是通过特征工程手动创建。表示学习是自然语言处理研究的一个重要方向（Bengio et al., 2013）。

6.1 词汇语义学

让我们首先介绍一些词义的基本原则。我们应该如何表示一个词的意义呢？在第3章的n-gram模型和经典的自然语言处理应用中，我们对词语的表示是一个字母串，或者作为词汇表中的索引。这种表示方式与哲学中的某种传统并没有太大区别，或许你在入门逻辑课中见过这种方式，即通过将词用小写的大写字母拼写出来来表示词的意义；例如用DOG来表示“狗”的意义，用CAT来表示“猫”的意义，或者用撇号（DOG'）来表示。

通过大写字母来表示词的意义是一种相当不浅显的模型。你可能见过一个源于语义学家Barbara Partee的笑话版本（Carlson, 1977）：

Q：What’s the meaning of life?

A：LIFE'

显然，我们能做得比这更好！毕竟，我们希望一个词义模型能够为我们完成各种任务。它应该告诉我们有些词意义相似（猫与狗相似），有些词是反义词（冷与热相对），一些词具有正面含义（快乐），而另一些词具有负面含义（悲伤）。它应该表示出像买、卖、付这样词语的意义，提供了同一购买事件的不同视角。（If I buy something from you, you’ve probably sold it to me, and I likely paid you.）更普遍地说，一个词义模型应该能够让我们得出推论，解决与意义相关的任务，如问 答或对话。

在本节中，我们将总结一些这样的标准，借鉴词义的语言学研究成果，称为**词汇语义学**（**lexical semantics**）；我们将在附录G和第21章中回到并扩展这一列表。

词形和词义

让我们从查看一个词（我们选择mouse）在字典中的定义开始（简化自在线词典WordNet）：

mouse (N)

a. 任意一种小型啮齿动物...

b. 一种手动设备，用于控制光标...

在这里，形式“mouse”是**词元**（**lemma**），也称为引用形式。形式“mouse”也是词形“mice”的词元；字典不会为像“mice”这样的词形形式提供单独的定义。同样，“sing”是“sing, sang, sung”的词元。在许多语言中，动词的不定式形式被用作词元，因此西班牙语“dormir”（睡觉）是“duermes”（你睡觉）的词元。具体形式如“sung”或“carpets”或“sing”或“duermes”被称为**词形**（**wordforms**）。

如上例所示，每个词元可以有多个意义；词形“mouse”可以指代啮齿动物或光标控制设备。我们称“mouse”意义的每一个方面为**词义**（**word sense**）。词形可以是**多义**（**polysemous**）的，这可能会使解释变得困难（输入“mouse info”到搜索引擎中的人是在寻找宠物还是工具？）。第11章和附录G将讨论多义性问题，并介绍**词义消歧任务**（**word sense disambiguation**），即确定在特定上下文中使用了哪个词义。

同义关系（Synonymy）

词义的一个重要组成部分是词义之间的关系。例如，当一个词的某个意义与另一个词的某个意义相同或几乎相同时，我们称这两个词义是**同义词**（**synonyms**）。同义词包括以下配对：

couch/sofa vomit/throw up filbert/hazelnut car/automobile

同义关系（词义之间的同义性）的一个更正式的定义是，如果两个词可以在任何句子中替换而不改变句子的真值条件（即该句子在什么情况下会为真），那么这两个词就是同义词。

虽然像“car/automobile”或“water/H2O”这样的词对的替换是保持真值的，但这些词的意义仍然不完全相同。实际上，几乎没有两个词的意义完全相同。语义学的基本原则之一，称为**对比原则**（**principle of contrast**）（Girard 1718, Bréal 1897, Clark 1987），表明语言形式的差异总是与意义的差异相关联。例如，“H2O”这个词在科学背景下使用，而在徒步旅行指南中使用它是不合适的

——使用“water”会更合适——而这种语域的差异是词义的一部分。因此，**实际上“同义词”这个词被用来描述一种近似或粗略的同义关系。**

词义相似性(Word Similarity)

虽然词语没有太多的同义词，但大多数词语确实有很多相似的词。猫不是狗的同义词，但猫和狗肯定是相似的词语。从同义性到相似性，转向讨论词汇之间的关系（如相似性）会更有用。处理词汇可以避免我们必须致力于某种特定的词义表示，这将简化我们的任务。

词义相似性的概念在更大的语义任务中非常有用。知道两个词汇的相似程度可以帮助计算两个短语或句子意义的相似程度，这是诸如问答、释义和总结等任务的重要组成部分。获得词汇相似性值的一种方法是让人类判断一个词与另一个词的相似程度。许多数据集源于这样的实验。例如，SimLex-999数据集（Hill等，2015）在0到10的尺度上给出相似性值，下面是一些例子，从近似同义词（vanish，disappear）到几乎没有任何共同点的词对（hole，agreement）。

vanish	disappear	9.8
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

词义相关性 (Word Relatedness)

两个词汇的含义可以通过除相似性之外的方式相关联。这类关系的一种叫做词义相关性（Budanitsky和Hirst, 2006），在心理学中传统上称为词语联想（word association）。例如，考虑“coffee”和“cup”这两个词的意义。咖啡与杯子并不相似；它们几乎没有共同特征（咖啡是一种植物或饮料，而杯子是一种有特定形状的制成品）。但咖啡和杯子显然是相关的；它们通过参与日常事件（用杯子喝咖啡的事件）相关联。类似地，“scalpel”和“surgeon”也不相似，但它们在事件上是相关的（外科医生通常使用手术刀）。

词义相关性的一种常见形式是当它们属于同一个**语义场**（**semantic field**）。语义场是一组覆盖特定语义领域并与彼此有结构关系的词汇。例如，词语可以通过属于医院的语义场（如surgeon, scalpel, nurse, anesthetic, hospital）、餐馆的语义场（如waiter, menu, plate, food, chef）或房屋的语义场（如door, roof, kitchen, family, bed）相关联。语义场也与主题模型相关，例如潜在狄利克雷分配（LDA），它通过对大量文本进行无监督学习，推导出文本中的一组相关词。语义场和主题模型是发现文档主题结构的非常有用的工具。

在附录G中，我们将介绍更多词义之间的关系，如上位词关系或IS-A关系，反义词关系（antonymy）和部分-整体关系（meronymy）。

语义框架与角色

与语义场紧密相关的是语义框架的概念。语义框架是一组表示特定类型事件中的视角或参与者的词语。例如，商业交易是一种事件，其中一个实体向另一个实体交易金钱以换取某些商品或服务，此后商品易手或服务可能被执行。这个事件可以通过使用“buy”（从买家的视角描述事

件）、“sell”（从卖家的视角描述）或“pay”（专注于金钱方面）等动词或“buyer”等名词在词汇上编码。框架具有语义角色（如买方、卖方、商品、金钱），句子中的词语可以承担这些角色。

了解“buy”和“sell”之间的这种关系可以使系统知道，像“Sam bought the book from Ling”这样的句子可以被释义为“Ling sold the book to Sam”，并且Sam在框架中承担买方的角色，Ling承担卖方的角色。识别这种改写对于问答非常重要，并且有助于在机器翻译中转换视角。

词语的内涵

最后，词汇具有感情意义或内涵。内涵一词在不同领域有不同含义，但在这里我们用它来指与作者或读者的情感、情绪、意见或评价相关的词义方面。例如，一些词汇有积极的内涵（wonderful），而另一些则有消极的内涵（dreary）。即使在其他方面意义相似的词汇，内涵也可能不同；考虑fake、knockoff、forgery与copy、replica、reproduction之间的内涵差异，或innocent（积极内涵）与naive（消极内涵）之间的差异。一些词汇表达正面评价（great，love），另一些则表达负面评价（terrible，hate）。正面或负面评价的语言称为情感，如我们在第4章中所见，词汇情感在情感分析、立场检测以及政治语言和消费者评论的NLP应用中起着重要作用。

早期关于感情意义的研究（Osgood等，1957）发现，词汇在感情意义上沿着三个重要维度变化：

1. 价效（valence）：刺激的愉悦度
2. 唤醒度（arousal）：刺激引发情感的强度
3. 控制度（dominance）：刺激施加的控制程度

因此，像happy或satisfied这样的词在价效上得分很高，而unhappy或annoyed得分较低。excited在唤醒度上得分很高，而calm得分较低。controlling在控制度上得分很高，而awed或influenced得分较低。因此，每个词汇都可以通过三个数字表示，分别对应其在三个维度上的评分：

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

Osgood等（1957）注意到，通过使用这三个数字来表示词汇的意义，模型实际上将每个词汇表示为三维空间中的一个点，一个向量，其三个维度分别对应该词在三条尺度上的评分。**这一革命性的观点认为词汇意义可以表示为空间中的一个点**（例如，heartbreak的部分意义可以表示为[2.45, 5.65, 3.58]）是我们将在下节中介绍的向量语义模型的首次体现。

6.2 向量语义学

向量语义学是自然语言处理中表示词义的标准方法，帮助我们建模上一节中提到的许多的有关词义方面。这个模型的根源可以追溯到20世纪50年代，当时两个重要的想法相互融合：Osgood在1957年提出的用三维空间中的一个点来表示词的含义的想法，以及像Joos（1950）、Harris（1954）和Firth（1957）这样的语言学家提出的通过词在语言使用中的分布，即它的邻近词或语法环境，来定义词义的主张。他们的观点是，出现在非常相似的分布中的两个词（即它们的邻近词相似）具有相似的含义。

例如，假设你不知道“ongchoi”这个词的含义（这是从粤语中借来的新词），但你在以下语境中看到了它：

(6.1) Ongchoi和蒜炒在一起非常美味。

(6.2) Ongchoi放在米饭上非常出色。

(6.3) …ongchoi叶子配咸味酱汁…

再假设你在其他语境中也见过很多这些上下文词：

(6.4) …菠菜和蒜炒在一起，配米饭…

(6.5) …瑞士甜菜茎和叶子非常美味…

(6.6) …羽衣甘蓝和其他咸味的绿叶菜…

Ongchoi与米饭、蒜、美味和咸等词一起出现，而像菠菜、瑞士甜菜、羽衣甘蓝等词也出现在类似的语境中，这可能表明ongchoi是一种类似于这些其他绿叶菜的绿叶蔬菜。我们可以通过计算机模拟来实现这一点，方法是仅仅统计与ongchoi相关的上下文词。

向量语义学的核心思想是通过邻近词的分布（在后续内容中会看到具体方法）将一个词表示为多维语义空间中的一个点。用于表示词的向量称为**嵌入（embeddings）**，尽管这个术语有时更严格地仅用于表示稠密向量，如word2vec（见6.8节），而不是稀疏的tf-idf或PPMI向量（见6.3-6.6节）。嵌入一词来源于其数学意义，即从一个空间或结构映射到另一个空间或结构，尽管其含义已经发生了变化；参见本章末尾的讨论。

图6.1展示了用于情感分析的嵌入的可视化，将选定的词从60维空间投影到二维空间中。不同区域分别表示正面词汇、负面词汇和中性功能词。

向量语义学中词汇相似性的细粒度模型为NLP应用提供了巨大的威力。第4章或第5章中的情感分类器等NLP应用依赖于训练集和测试集中出现相同的词汇。但是通过将词汇表示为嵌入，分类器只要看到一些意思相似的词就可以分配情感。正如我们将看到的，向量语义模型可以通过无监督学习从文本中自动获得。

在本章中，我们将介绍两个最常用的模型。在tf-idf模型中，一个重要的基线，词义由邻近词的计数的简单函数定义。我们将看到这种方法产生了非常长的向量，这些向量是稀疏的，即大多数值为零（因为大多数词几乎从未出现在其他词的上下文中）。我们还将介绍word2vec模型系列，用于构建具有有用语义属性的短而稠密的向量。我们还将介绍余弦相似度，这是使用嵌入计算两个词、两个句子或两个文档之间语义相似性的标准方法，是问答、摘要或自动作文评分等实际应用中的重要工具。

6.3 词汇与向量

> “在三维空间中，向量最重要的属性是{位置，位置，位置}”
>
> —— Randall Munroe, https://xkcd.com/2358/

向量或分布式语义模型通常基于共现矩阵，这是表示单词共现频率的一种方式。我们将讨论两种常见的矩阵：词-文档（term-document）矩阵和词-词（term-term）矩阵。

6.3.1 向量与文档

在**词-文档矩阵**（**term-document matrix**）中，每一行代表词汇表中的一个单词，每一列代表文档集合中的一个文档。图6.2显示了一个词-文档矩阵的小样本，其中展示了莎士比亚四部戏剧中四个词的出现频次。这个矩阵中的每个单元格表示特定单词（由行定义）在特定文档（由列定义）中出现的次数。例如，“fool”在《第十二夜》中出现了58次。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

图 6.2 四部莎士比亚戏剧中四个单词的术语文档矩阵。每个单元格包含（行）单词在（列）文档中出现的次数。

图6.2中的词-文档矩阵最早定义为信息检索中**向量空间模型**（**vector space model**）的一部分（Salton，1971）。在该模型中，文档被表示为一个计数向量，即图6.3中的一列。

回顾一些基础线性代数，向量本质上只是一个数字列表或数组。因此，《皆大欢喜》被表示为列表[1,114,36,20]（图6.3中的第一列向量），而《朱利叶斯·凯撒》则被表示为列表[7,62,1,2]（第三列向量）。向量空间是向量的集合，其特征在于它的维度。三维向量空间中的向量在每个维度上都有一个元素。我们可以宽泛地将四维空间中的向量称为四维向量，每个维度上有一个元素。在图6.3的例子中，我们选择使文档向量的维度为4，这样它们能放在页面上；在实际的词-文档矩阵中，文档向量的维度会是|V|（词汇表大小）。

向量空间中的数字顺序表明了文档变化的不同维度。这些向量的第一个维度对应单词“battle”出现的次数，我们可以比较每个维度，注意到例如《皆大欢喜》和《第十二夜》的向量在第一个维度上的值相似（分别为1和0）。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

图 6.3 四部莎士比亚戏剧中四个单词的术语文档矩阵。红色框显示每个文档都表示为长度为 4 的列向量。

我们可以将文档的向量看作|V|维空间中的一个点；因此，图6.3中的文档是四维空间中的点。由于四维空间很难可视化，图6.4展示了一个二维可视化；我们任意选择了单词“battle”和“fool”对应的维度。

词-文档矩阵最初被定义为用于文档信息检索任务中找到相似文档的一种方法。相似的文档往往包含相似的单词，如果两个文档有相似的单词，它们的列向量也趋于相似。《皆大欢喜》[1,114,36,20]和《第十二夜》[0,80,58,15]的向量看起来彼此更相似（更多关于愚人和智慧的内容，而不是战斗）相比于《朱利叶斯·凯撒》[7,62,1,2]或《亨利五世》[13,89,4,3]。这在原始数字中很明显；在第一个维度（战斗）上，喜剧的数值较低，而其他剧的数值较高，我们可以通过图6.4直观地看到这一点；我们很快将更加正式地量化这一直觉。

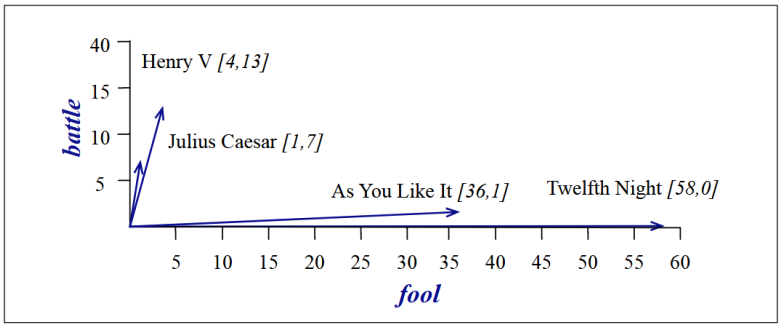

图 6.4 四个莎士比亚戏剧文档的文档向量的空间可视化，仅显示两个维度，对应于单词“战斗”和“傻瓜”。喜剧傻子维度颜值高，战斗维度颜值低

当然，真实的词-文档矩阵不会只有4行和4列，更不用说只有2列了。通常，词-文档矩阵有|V|行（词汇表中每个词的一个行）和D列（集合中每个文档的一个列）；正如我们将看到的，词汇表的大小通常是数以万计的，而文档数量可能非常庞大（想象一下所有网页的集合）。

信息检索（Information retrieval，IR）是从某个集合的D个文档中找到与查询q最匹配的文档d的任务。因此，在IR中，我们也将查询表示为一个向量，长度同样为|V|，并且我们需要一种方法来比较两个向量，以找到它们的相似度。（进行IR还需要高效的方式来存储和操作这些向量，利用这些向量大多为稀疏的事实，即大部分为零）。

在本章稍后部分，我们将介绍向量比较过程中涉及的一些组件：tf-idf词权重和余弦相似度度量。

6.3.2 将单词表示为向量：文档维度

我们已经看到，文档可以作为向量在向量空间中表示。但向量语义也可以用于表示单词的含义。我们通过将每个单词与一个词向量相关联来做到这一点——这里是行向量而不是列向量，因此具有不同的维度，如图6.5所示。词“fool”的四维向量[36,58,1,4]对应莎士比亚的四部戏剧。在相同的四个维度上，其他三个词的向量是：wit [20,15,2,3]；battle [1,0,7,13]；以及 good [114,80,62,89]。

对于文档，我们已经看到相似的文档具有相似的向量，因为相似的文档往往有相似的词。同样的原则也适用于单词：相似的单词具有相似的向量，因为它们往往出现在相似的文档中。因此，词-文档矩阵允许我们通过单词倾向出现的文档来表示其含义。

6.3.3 将单词表示为向量：词汇维度

另一种将单词表示为文档计数向量的替代方法是使用词-词矩阵（term-term matrix），这也被称为词-词矩阵（word-word matrix）或词-上下文（term-context matrix）矩阵，其中列取决于单词而不是文档。因此，这个矩阵的维度是|V|×|V|，每个单元格记录了某个训练语料库中行（目标）单词和列（上下文）单词在某种上下文中共同出现的次数。上下文可以是文档，在这种情况下，该单元格表示两个单词出现在同一文档中的次数。然而，最常见的是使用较小的上下文，通常是单词周围的一个窗口，例如向左4个单词和向右4个单词，在这种情况下，单元格的值表示在某个训练语料库中，列单词在行单词的±4个单词窗口中出现的次数。以下是一些单词在其窗口中的四个示例：

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

如果我们对每个单词（例如“strawberry”）的每次出现进行统计，并统计其周围的上下文单词，我们就得到了一个词-词共现矩阵。图6.6显示了从维基百科语料库（Davies, 2015）计算出的这四个单词的词-词共现矩阵的简化的子集。

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

图 6.6 维基百科语料库中四个单词的共现向量，显示六个维度（出于教学目的精心挑选）。数字向量用红色标出。请注意，真正的向量将具有更多的维度，因此会稀疏得多。

在图6.6中可以注意到，单词“cherry”和“strawberry”彼此之间更加相似（“pie”和“sugar”通常出现在它们的窗口中），而与其他单词（例如“digital”）的相似性较低；反之，单词“digital”和“information”比与“strawberry”更相似。图6.7展示了一个空间可视化。

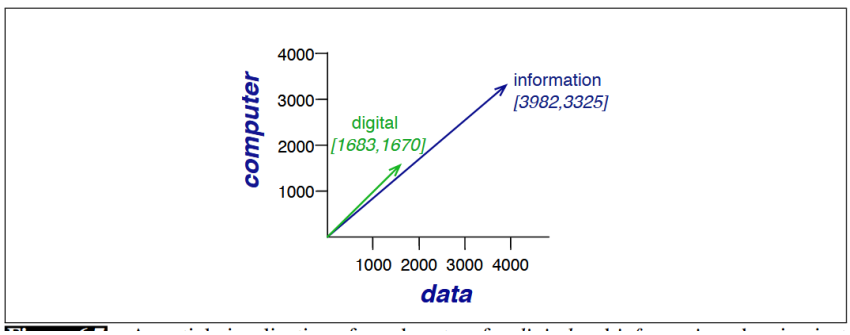

图 6.7 digitalk和information的词向量的空间可视化，仅显示两个维度，对应于词data和computer。

请注意，向量的维度|V|通常是词汇表的大小，通常在10,000到50,000个单词之间（使用训练语料库中最常用的单词；保留大约最常用的50,000个单词之后的单词通常没有太大帮助）。由于这些数字大多数是零，因此这些是稀疏向量表示；有高效的算法可以用于存储和计算稀疏矩阵。

现在我们已经有了一些直观的理解，接下来我们将深入探讨计算单词相似度的细节。之后，我们会讨论单元格加权的方法。

6.4 使用余弦来衡量相似

为了衡量两个目标词 v 和 w 之间的相似性，我们需要一种度量标准，该标准接受两个向量（具有相同的维度，可能都以单词为维度，长度为 |V|，或都以文档为维度，长度为 |D|），并给出它们的相似性度量。目前最常用的相似性度量是向量之间角度的余弦值。

余弦——就像大多数用于自然语言处理的向量相似性度量一样——基于线性代数中的点积运算，也称为内积：

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

点积作为相似性度量的原因在于，当两个向量在相同维度上具有较大的值时，点积的值通常会较高。相反，若两个向量在不同的维度上为零（正交向量），则它们的点积为0，表示它们的强烈不相似性。

然而，作为相似性度量，原始的点积存在一个问题：它偏向于较长的向量。向量长度定义如下：

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

如果向量较长，在每个维度上都有较高的值，那么点积就会更高。因为频率较高的词汇往往与更多的单词共现，并且在每个单词上具有较高的共现值，因此它们的向量会更长。因此，频繁词汇的原始点积值通常较高。但这是一个问题；我们希望相似性度量能够告诉我们两个词的相似程度，而不受它们频率的影响。

为了解决这个问题，我们通过将点积除以两个向量的长度来规范化点积。这一规范化的点积正好与两个向量之间的角度余弦值相同，这基于向量 a 和 b 的点积定义：

$$\begin{aligned} \mathbf{a}\cdot\mathbf{b} &= |\mathbf{a}||\mathbf{b}|\cos\theta \\ \frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} &= \cos\theta \end{aligned}$$

因此，向量 v 和 w 之间的余弦相似性度量可以如下计算：

$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

在某些应用中，我们会预先对每个向量进行规范化，方法是将它除以其长度，创建长度为1的单位向量。因此，我们可以通过将向量 a 除以 |a| 来计算一个单位向量。对于单位向量，点积与余弦相同。

余弦值的范围从1（向量指向相同方向）到0（正交向量），再到-1（向量指向相反方向）。但由于原始频率值是非负的，这些向量的余弦值范围为0到1。

让我们看看余弦如何计算出“cherry”和“digital”哪个词与“information”的含义更接近，使用下表中的原始计数值。

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry, information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .018$$

$$\cos(\text{digital, information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

模型决定“information”与“digital”远比与“cherry”更接近，这个结果看起来是合理的。图6.8展示了一个可视化。

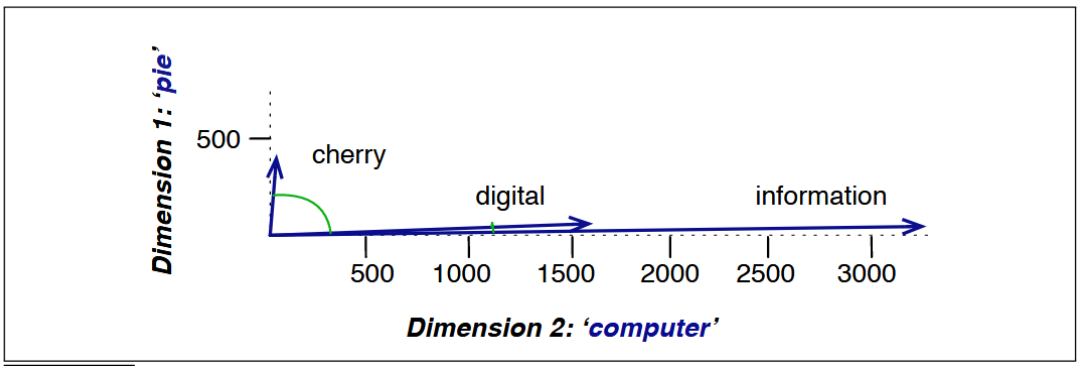

图 6.8 余弦相似度的（粗略）图形演示，显示由附近单词“computer”和“pie”的计数定义的二维空间中三个单词（cherry、digital 和 information）的向量。该图没有显示余弦，但突出显示了角度；请注意，数字和信息之间的角度小于樱桃和信息之间的角度。当两个向量越相似时，余弦越大，但夹角越小；当两个向量之间的角度最小（0°）时，余弦达到最大值（1）；所有其他角度的余弦都小于 1。

6.5 TF-IDF：向量中的词项加权

上述共现矩阵每个单元格表示频率，可能是单词与文档的频率（图6.5），或者是单词与其他单词的频率（图6.6）。然而，原始频率并不是衡量单词关联的最佳方式。原始频率非常偏斜，且区分性不强。如果我们想知道“cherry”和“strawberry”共享的上下文与“digital”和“information”不同的上下文，像“the”、“it”或“they”这种与各种词频繁共现但对任何特定单词都没有信息量的词无法提供良好的区分性。我们在图6.3中看到的莎士比亚语料库也是如此；单词“good”在不同戏剧之间的区分性不强，因为它是一个常见词，在每个戏剧中的频率都大致相同且较高。

这有点像悖论。频繁出现在附近的单词（例如“pie”出现在“cherry”附近）比只出现一次或两次的单词更重要。然而，过于频繁出现的单词——如“the”或“good”这样的常见词——却是不重要的。我们如何在这两个相互矛盾的约束之间取得平衡呢？

有两种常见的解决方案：在本节中，我们将介绍通常用于文档维度的**tf-idf加权**。在下一节中，我们将介绍**PPMI算法**（通常用于单词维度）。

tf-idf加权（此处“-”是连字符，而非减号）是两个词项的乘积，每个词项捕捉一种直观感受：第一个是词频（**term frequency**，Luhn，1957）：即单词t在文档d中的出现频率。我们可以直接使用原始计数作为词频：

$$\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

然而，更常见的是，我们会对原始频率进行一定程度的压缩，通常使用频率的对数（log10）。其直观理解是，一个单词在文档中出现100次并不意味着它比只出现1次的单词与文档的关联性高出100倍。我们还需要对出现次数为0的情况做特殊处理，因为0的对数是未定义的：

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if} \quad \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

如果我们使用对数加权，在文档中出现0次的词项的tf=0，出现1次的词项的tf=1+log10(1)=1，出现10次的词项的tf=1+log10(10)=2，出现100次的词项的tf=1+log10(100)=3，出现1000次的词项的tf=4，依此类推。

tf-idf的第二个因子用于给仅在少数文档中出现的词赋予更高的权重。只出现在少数文档中的词项有助于区分这些文档，而那些在整个集合中频繁出现的词项并没有太大帮助。词项t的文档频率 df_t 是指它出现在多少个文档中。文档频率不同于词项的集合频率，后者是指该词在整个集合中出现的总次数。以莎士比亚的37部戏剧为例，单词“Romeo”和“action”在整个剧集中出现的总次数相同（都出现了113次），但它们的文档频率差别很大，因为“Romeo”只在一部戏剧中出现。如果我们的目标是找到与“Romeo”的浪漫纠葛有关的文档，那么“Romeo”一词应当被赋予很高的权重，而不是“action”：

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

我们通过**逆文档频率**（**inverse document frequency**, idf）或idf词权重（Sparck Jones，1972）来强调像“Romeo”这样具有区分性的词项。idf的定义为**N/** df_t 的比值，其中N是集合中的文档总数，而 df_t 是词项t出现的文档数。一个词项出现的文档越少，其权重越高。对于出现在所有文档中的词项，最低的权重为1。通常，什么算作一个文档是显而易见的：在莎士比亚的作品中，我们会使用一部戏剧；处理百科文章集合（如维基百科）时，文档是一个维基页面；处理报纸文章时，文档是一篇文章。偶尔，语料库中可能没有合适的文档划分，你可能需要自己将语料库划分为文档，以便计算idf。

由于许多集合中的文档数量很大，这一度量通常也通过对数函数进行压缩。因此，逆文档频率（idf）的最终定义如下所示：

$$\mathrm{idf}_t = \log_{10} \left(\frac{N}{\mathrm{df}_t} \right)$$

以下是一些在莎士比亚语料库中的单词的idf值（以及其基于的文档频率df值），这些单词从极具信息量的只出现在一部戏剧中的单词如“Romeo”，到出现在少数几部戏剧中的词

如“salad”或“Falstaff”，再到非常常见的词如“fool”，甚至是像“good”或“sweet”这样出现在所有37部戏剧中而完全没有区分性的信息词。

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

因此，对于文档 d 中的词 t，其权重 wt使用结合了词频tf和拟文档频率idf的 tf-idf 来衡量：

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

图6.9将tf-idf加权应用于图6.2中的莎士比亚词-文档矩阵，使用的是公式6.12中的tf公式。请注意，维度对应于单词“good”的tf-idf值现在全都变为0；由于该单词出现在每个文档中，tf-idf加权使其被忽略。同样，出现在37部戏剧中36部的单词“fool”权重也大大降低。

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.246	0	0.454	0.520
good	0	0	0	0
fool	0.030	0.033	0.0012	0.0019
wit	0.085	0.081	0.048	0.054

图 6.9 莎士比亚戏剧中四个单词的 tf-idf 加权术语文档矩阵的一部分，使用图 6.2 中的计数显示了 4 个戏剧的选择。例如，《皆大欢喜》中 wit 的 0.085 值是 tf = 1+log10(20) = 2.301 和 idf = .037 的乘积。请注意，idf 权重消除了无处不在的单词“good”的重要性，并大大降低了几乎无处不在的单词“fool”的影响。

tf-idf加权是一种用于信息检索中共现矩阵加权的方法，但它在自然语言处理的许多其他方面也发挥着作用。它也是一个很好的基准方法，值得首先尝试。我们将在第6.6节中探讨其他加权方法，如PPMI（正向点互信息）。

6.6 逐点互信息 (Pointwise Mutual Information，PMI)

另一种用于词-词矩阵的加权函数是 PPMI（正点互信息，positive pointwise mutual information），这是 tf-idf 的替代方法。当向量维度对应于单词而不是文档时，PPMI 就被使用。PPMI 的直觉来源于这样的观点：衡量两个单词之间关联的最佳方式是考察它们在语料库中的共同出现频率，是否远高于我们预期的随机出现频率。

点互信息 (PMI) 是 NLP 中最重要的概念之一，由 Fano 于 1961 年提出。它是一种度量两个事件 x 和 y 共同出现的频率与它们独立出现的预期频率之间的差异：

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

目标词 w 和上下文词 c 之间的点互信息（Church 和 Hanks 在 1989 年、1990 年提出）定义如下：

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

其中分子表示我们在观察中这两个词共同出现的次数（假设我们用最大似然估计法来计算概率）。分母表示我们在假设两个词独立出现的情况下，它们共同出现的预期频率；回想一下，两个独立事件同时发生的概率只是这两个事件各自发生概率的乘积。因此，这个比率为我们提供了一个估计值，表示这两个词共同出现的次数是否远高于我们预期的随机情况。

PMI 是一个有用的工具，特别是在我们需要找到强关联词的时候。PMI 值的范围从负无穷到正无穷，但负 PMI 值（意味着两个词比预期的更少共同出现）在语料库不够大的情况下往往不可靠。要区分两个个体概率都是 10^{-6} 的词是否比随机情况下更少共同出现，我们需要确保它们共同出现的概率显著小于 10^{-12}，这种精确度要求一个巨大的语料库。此外，用人类判断来评估这种“无关”的分数是否可能也不确定。因此，更常用的是正点互信息（PPMI），它将所有负 PMI 值替换为零：

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

更正式地说，假设我们有一个共现矩阵 F，矩阵有 W 行（单词）和 C 列（上下文），其中 f_ij 给出单词 w_i 与上下文 c_j 共同出现的次数。这可以转换为一个 PPMI 矩阵，其中 PPMI_ij 给出单词 w_i 和上下文 c_j 的 PPMI 值（我们也可以将其表示为 PPMI(w_i, c_j) 或 PPMI(w=i, c=j)），计算如下：

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0)$$

我们来看一些 PPMI 的计算例子。我们将使用图 6.10，它重复了图 6.6 并包含所有的计数边缘值，为了简化计算，假设这些是唯一重要的单词/上下文。

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

图 6.10 维基百科语料库中 5 个上下文中的四个单词以及边缘词的共现计数，为了计算的目的假装其他单词/上下文无关紧要。

例如，我们可以计算 PPMI(information, data)，假设图 6.6 包含了所有相关的单词上下文维度，具体步骤如下：

$$
\begin{aligned}
P(\text{w=information, c=data}) &= \frac{3982}{11716} = .3399 \\
P(\text{w=information}) &= \frac{7703}{11716} = .6575 \\
P(\text{c=data}) &= \frac{5673}{11716} = .4842 \\
\text{PPMI(information,data)} &= \log_2(.3399/(.6575 * .4842)) = .0944
\end{aligned}
$$

图 6.11 显示了根据图 6.10 中的计数计算出的联合概率，图 6.12 则显示了 PPMI 值。不出意料，cherry 和 strawberry 与 pie 和 sugar 都高度相关，而 data 与 information 关系较弱。

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

图 6.11 用联合概率替换图 6.6 中的计数，显示右列和底行的边缘。

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

图 6.12 PPMI 矩阵显示单词和上下文单词之间的关联，根据图 6.11 中的计数计算得出。请注意，大多数 0 PPMI 值都是具有负 PMI 的值；例如 PMI(cherry,computer) = -6.7，这意味着 cherry 和 Computer 在维基百科上同时出现的频率比我们预期的要少，并且使用 PPMI，我们将负值替换为零。

PMI 存在的一个问题是它对低频事件有偏差；非常罕见的单词往往具有很高的 PMI 值。减少这种偏差的一个方法是稍微改变 P(c) 的计算，使用一个不同的函数 Pα(c)，其中上下文词的概率被提升到 α 次方：

$$
\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)
$$

$$
P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}
$$

Levy 等人在 2015 年发现，α = 0.75 的设置可以改善嵌入在各种任务上的表现（这一做法借鉴了跳字模型中类似的加权方法，如公式 6.32 所述）。这是因为将计数提升至 α = 0.75 增加了赋予罕见上下文的概率，因此降低了它们的 PMI（当 c 很罕见时，Pα(c) > P(c)）。

另一种可能的解决方案是拉普拉斯平滑：在计算 PMI 之前，向每个计数添加一个小常数 k（通常取 0.1-3），从而收缩（折扣）所有非零值。k 值越大，非零计数折扣越多。

6.7 TF-IDF 或 PPMI 向量模型的应用

总而言之，我们迄今为止描述的向量语义模型将目标词表示为一个向量，其维度对应于大型文档集合中的文档（词-文档矩阵），或是某个相邻窗口内词的计数（词-词矩阵）。每个维度的值是计数，并通过 tf-idf（用于词-文档矩阵）或 PPMI（用于词-词矩阵）加权，这些向量是稀疏的（因为大多数值为零）。

该模型通过计算词 x 和词 y 的 tf-idf 或 PPMI 向量的余弦值来度量它们之间的相似性；余弦值越高，相似性越大。整个模型有时被称为 tf-idf 模型或 PPMI 模型，依据的是加权函数的不同。

tf-idf 的语义模型通常用于文档功能，例如判断两个文档是否相似。我们通过获取文档中所有词的向量，并计算这些向量的中心点(均值的多维版本)来表示一个文档。一组向量的中心点是与该组向量的平方距离之和最小的那个向量。给定 k 个词向量 w1, w2, ..., wk，文档的中心向量 d 如下所示：

$$d = \frac{w_1 + w_2 + ... + w_k}{k}$$

对于两篇文档，我们可以分别计算它们的文档向量 d1 和 d2，并通过计算 cos(d1, d2) 来估计这两篇文档的相似性。文档相似性对各种应用场景都有用，如信息检索、抄袭检测、新闻推荐系统，甚至在数字人文学科中用于比较不同版本的文本，判断哪些版本彼此相似。

无论是 PPMI 模型还是 tf-idf 模型，都可以用于计算词语相似性，例如用于查找词语的释义、跟踪词语意义的变化，或在不同语料库中自动发现词语的不同含义。例如，我们可以通过计算目标词 w 与 V − 1 个其他词的余弦值，排序并选择前 10 个，来找到与任何目标词 w 最相似的 10 个词。

6.8 Word2vec

在前面的章节中，我们已经了解到如何将一个词表示为稀疏的、长的向量，其维度对应于词汇表中的词或集合中的文档。现在我们介绍一种更强大的词表示方法：**嵌入**（**embeddings**），即短而密集的向量。与之前看到的向量不同，嵌入是短的，维度 d 通常在 50-1000 之间，而不是我们之前见到的更大词汇表大小 |V| 或文档数量 D。这些 d 维度并没有明确的解释。并且这些向量是密集的：向量中的条目不再是稀疏的、大多数为零的计数或计数的函数，而是可以**为负**的实数值。

事实证明，在每个自然语言处理（NLP）任务中，密集向量比稀疏向量表现得更好。虽然我们不能完全理解所有原因，但我们有一些直觉。将词表示为 300 维的密集向量要求分类器学习的权重远少于将词表示为 50,000 维的稀疏向量，这样更小的参数空间可能有助于泛化和避免过拟合。密集向量在捕捉同义词方面也可能表现得更好。例如，在稀疏向量表示中，像 car 和 automobile 这样的同义词的维度是独立且无关的；因此，稀疏向量可能无法捕捉到一个词与 car 为邻居和另一个词与 automobile 为邻居之间的相似性。

在本节中，我们介绍一种计算嵌入的方法：使用负采样的跳词模型（**skip-gram with negative sampling**），有时简称为 **SGNS**。跳词算法是一个名为 word2vec 的软件包中的两种算法之一，因此有时这个算法会被不严格地称为 word2vec（Mikolov et al. 2013a, Mikolov et al. 2013b）。word2vec 方法速度快，训练高效，且可以很方便地在线获取代码和预训练嵌入。word2vec 嵌入是**静态嵌入**，这意味着该方法为词汇表中的每个词学习一个固定的嵌入。在第11章中，我们将介绍学习**动态上下文嵌入**的方法，例如流行的 **BERT** 表示家族，其中每个词的向量在不同的上下文中是不同的。

word2vec 的直觉是，取代统计词 w 在 apricot 附近出现的频率，我们将训练一个二元分类器来进行一个预测任务：“词 w 是否可能出现在 apricot 附近？”实际上，我们并不关心这个预测任务本身；相反，我们将使用分类器学习的权重作为词的嵌入。

这一革命性的直觉在于，我们可以仅使用文本作为这种分类器的隐式监督训练数据；出现在目标词 apricot 附近的词 c 作为问题“词 c 是否可能出现在 apricot 附近？”的黄金“正确答案”。这种方法通常被称为自监督（self-supervision），它避免了任何人工标注的监督信号的需要。这个想法最早在神经语言建模任务中提出，当时 Bengio 等人（2003）和 Collobert 等人（2011）展示了神经语言模型（即学习通过之前的词预测下一个词的神经网络）可以仅使用文本中的下一个词作为其监督信号，并且可以在执行该预测任务的过程中学习每个词的嵌入表示。

我们将在下一章讨论如何使用神经网络，但 word2vec 是一种比神经网络语言模型更简单的模型，主要体现在两个方面。首先，word2vec 简化了任务（将其转化为二元分类，而不是词预测）。其次，word2vec 简化了架构（训练一个**逻辑回归分类器**，而不是需要更复杂训练算法的多层神经网络）。跳词模型的直觉如下：

1. 将目标词和相邻上下文词作为正样本。
2. 从词汇表中随机采样其他词作为负样本。
3. 使用逻辑回归训练一个分类器来区分这两种情况。
4. 使用学习到的权重作为嵌入。

6.8.1 分类器

我们先来思考一下分类任务，然后再讨论如何训练。假设有一个句子，其中目标词是 “apricot”，并且我们使用的是 ±2 个上下文词的窗口：

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w     c3       c4
```

我们的目标是训练一个分类器，给定一个由目标词 **w** 和候选上下文词 **c** 组成的元组 (w, c)（例如 (apricot, jam) 或 (apricot, aardvark)），分类器将返回 **c** 是否**是真实上下文词的概率**（对于 jam 为真，对于 aardvark 为假）：

$$P(+|w,c)$$

词 **c** 不是目标词 **w** 的真实上下文词的概率就是 1 减去公式 6.24 中的结果。

$$P(-|w,c) = 1 - P(+|w,c)$$

分类器如何计算这个概率 P？跳词模型的直觉是基于嵌入的相似性：如果某个词的嵌入向量与目标词的嵌入相似，那么该词很可能出现在目标词附近。为了计算这些密集嵌入之间的相似性，我们依赖于这样的直觉：两个向量如果点积较大，它们就相似（毕竟，余弦相似度本质上就是归一化的点积）：

$$Similarity(w,c) \approx \mathbf{c} \cdot \mathbf{w}$$

点积 **c**・**w** 并不是一个概率，它只是一个从 −∞ 到 ∞ 的数（因为 word2vec 嵌入中的元素可以是负数，点积也可能为负）。为了将点积转换为概率，我们将使用逻辑回归的核心——逻辑函数或 sigmoid 函数 σ (x)：

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

我们可以将词 **c** 是目标词 **w** 的真实上下文词的概率建模如下：

$$P(+|w,c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c} \cdot \mathbf{w})}$$

sigmoid 函数返回的是一个介于 0 和 1 之间的数值，但为了让它成为概率，我们还需要确保两种可能事件（“c 是上下文词” 和 “c 不是上下文词”）的总概率加和为 1。因此，我们估计词 **c** 不是目标词 **w** 的真实上下文词的概率如下：

$$\begin{aligned} P(-|w,c) &= 1 - P(+|w,c) \\ &= \sigma(-\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(\mathbf{c} \cdot \mathbf{w})} \end{aligned}$$

公式 6.28 给出了一个词的概率，但在窗口中有多个上下文词。跳词模型做了一个简化假设，即所有上下文词是独立的，因此我们可以将它们的概率相乘：

$$\begin{aligned} P(+|w,c_{1:L}) &= \prod_{i=1}^{L} \sigma(\mathbf{c_i} \cdot \mathbf{w}) \\ \log P(+|w,c_{1:L}) &= \sum_{i=1}^{L} \log \sigma(\mathbf{c_i} \cdot \mathbf{w}) \end{aligned}$$

总之，跳词模型训练了一个概率分类器，该分类器在给定一个目标词 **w** 和其上下文窗口 **L** 个词 **c1:L** 的情况下，根据这个上下文窗口与目标词的相似性来分配一个概率。这个概率基于将目标词嵌入与每个上下文词嵌入的点积应用到逻辑（sigmoid）函数上。为了计算这个概率，我们只需获得词汇表中每个目标词和上下文词的嵌入。

图 6.13 展示了我们需要的参数的直观表示。跳词模型实际上为每个词存储了两个嵌入，一个用于词作为目标词时，另一个用于词作为上下文词时。因此，我们需要学习的参数是两个矩阵 **W** 和 **C**，每个矩阵都包含词汇表 **V** 中每个词的嵌入。现在让我们转向学习这些嵌入（这也是训练这个分类器的最终目标）。

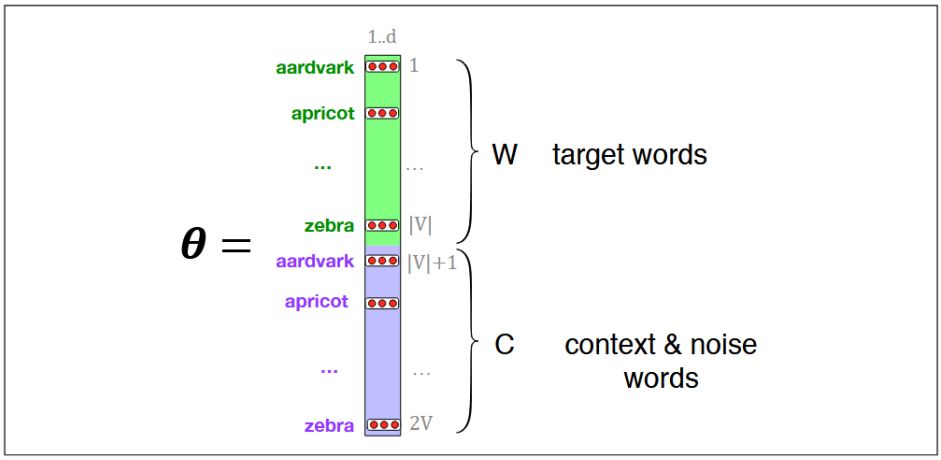

图 6.13 Skipgram 模型学习到的嵌入。该算法为每个单词存储两个嵌入，目标嵌入（有时称为输入嵌入）和上下文嵌入（有时称为输出嵌入）。因此，算法学习的参数 θ 是一个 2|V | 的矩阵。向量，每个维度为 d，通过连接两个矩阵（目标嵌入 W 和上下文+噪声嵌入 C）形成。

6.8.2 学习跳词嵌入

跳词（skip-gram）嵌入的学习算法以一个文本语料库和选定的词汇表大小 **N** 作为输入。它首先为这 **N** 个词汇表中的每个词分配一个随机的嵌入向量，然后通过迭代地调整每个词 **w** 的嵌入，使其更像文本中出现在其附近的词的嵌入，同时与不出现在其附近的词的嵌入区别开来。我们从下面的一个训练数据示例开始：

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w     c3       c4
```

这个例子中的目标词是 w （“apricot”），在 $L = \pm 2$ 的上下文窗口内有 4 个上下文词，产生了 4 个正训练实例（如下图左侧所示）：

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

为了训练一个二元分类器，我们还需要负例。事实上，使用负采样的跳词模型（SGNS）比正例使用更多的负例（它们之间的比例由参数 **k** 决定）。因此，对于每个 (w, cpos) 的正训练实例，我们会创建 **k** 个负样本，每个负样本由目标词 **w** 加上一个“噪声词” **cneg** 组成。噪声词是从词汇表中随机选取的，且不等于目标词 “w”。在上图右侧显示的例子中，当 **k = 2** 时，每个正例 (w, cpos) 会有 2 个负例。

噪声词的选择依据它们加权的单词频率 “pα(w)”，其中 α 是一个权重。如果我们按照未加权的频率 p(w) 进行采样，意味着以单词 the 的单词频率 p("the") 选择 the 作为噪声词，或以 p("aardvark") 选择 aardvark 作为噪声词，依此类推。但在实际操作中，通常将 α 设置为 0.75，即使用加权频率 p_(3/4)(w)：

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

设置 α = 0.75 可以提高性能，因为它稍微提高了稀有噪声词的概率：对于稀有词，Pα(w) 大于 P(w)。为说明这一直觉，我们可以用 α = 0.75 和两个事件 P(a) = 0.99 和 P(b) = 0.01 来计算概率。由下面计算结果可知，α = 0.75 增加了稀有事件 b 的概率，从 0.01 提高到 0.03：

$$\begin{aligned} P_\alpha(a) &= \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = 0.97 \\ P_\alpha(b) &= \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = 0.03 \end{aligned}$$

给定一组正负训练实例和初始嵌入集，学习算法的目标是调整这些嵌入，以实现以下目标：

- 最大化从正例中抽取的目标词和上下文词对 (w, cpos) 的相似性。
- 最小化从负例中抽取的目标词和噪声词对 (w, cneg) 的相似性。

如果我们考虑一个词/上下文对 (w, cpos) 及其 k 个噪声词 cneg1 ... cnegk，我们可以用以下损失函数 L （因此有一个负号）来表达这两个目标。这里，第一项表示我们希望分类器为真实上下文词 cpos 分配一个较高的邻近概率，第二项表示我们希望为每个噪声词 cneg 分配一个较高的非邻近概率，所有这些概率相乘，因为我们假设它们是独立的：

$$\begin{aligned} L &= -\log\left[P(+|w, c_{pos}) \prod_{i=1}^{k} P(-|w, c_{neg_i})\right] \\ &= -\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log P(-|w, c_{neg_i})\right] \\ &= -\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log\left(1 - P(+|w, c_{neg_i})\right)\right] \\ &= -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right] \end{aligned}$$

也就是说，我们希望最大化目标词与实际上下文词的点积，并最小化目标词与 **k** 个负采样非邻居词的点积。

我们使用随机梯度下降（SGD）来最小化这个损失函数。图 6.14 展示了学习过程中的一步直观图。

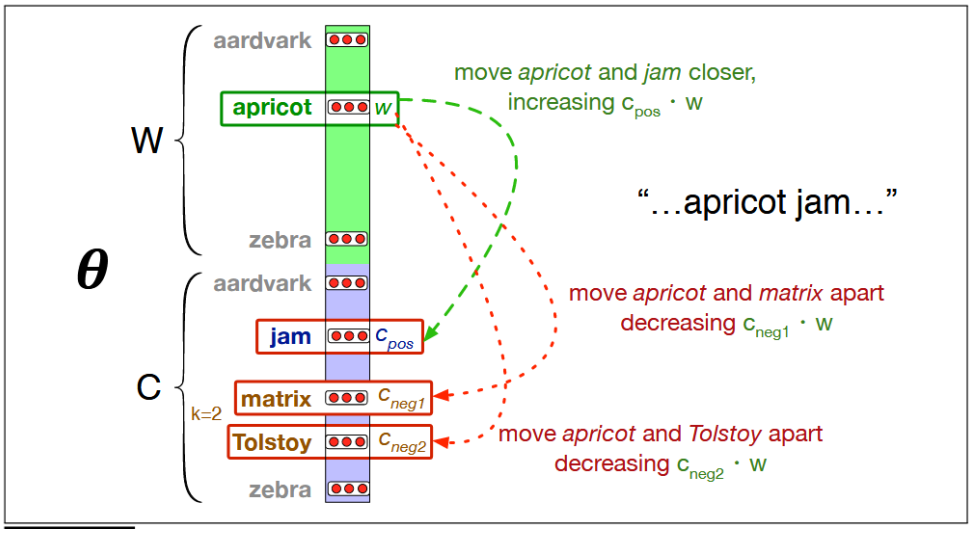

图 6.14 梯度下降一步的直觉。 Skip-gram 模型尝试改变嵌入，以便目标嵌入（此处为 apricot）更接近（具有较高的点积）附近单词（此处为 jam）的上下文嵌入，并且远离（具有较低的点积）上下文嵌入附近不会出现的干扰词（这里是托尔斯泰和矩阵）。

为了得到梯度，我们需要对公式 6.34 进行不同嵌入的求导。结果如下（证明作为本章末尾的练习）：

$$\begin{aligned}\frac{\partial L}{\partial c_{pos}} &= [\sigma(\mathbf{c}_{pos}\cdot\mathbf{w})-1]\mathbf{w}\\ \frac{\partial L}{\partial c_{neg}} &= [\sigma(\mathbf{c}_{neg}\cdot\mathbf{w})]\mathbf{w}\\ \frac{\partial L}{\partial w} &= [\sigma(\mathbf{c}_{pos}\cdot\mathbf{w})-1]\mathbf{c}_{pos}+\sum_{i=1}^{k}[\sigma(\mathbf{c}_{neg_i}\cdot\mathbf{w})]\mathbf{c}_{neg_i}\end{aligned}$$

从时间步 t 到 t+1 的随机梯度下降的更新方程如下所示：

$$\begin{aligned}\mathbf{c}_{pos}^{t+1} &= \mathbf{c}_{pos}^{t}-\eta[\sigma(\mathbf{c}_{pos}^{t}\cdot\mathbf{w}^{t})-1]\mathbf{w}^{t}\\ \mathbf{c}_{neg}^{t+1} &= \mathbf{c}_{neg}^{t}-\eta[\sigma(\mathbf{c}_{neg}^{t}\cdot\mathbf{w}^{t})]\mathbf{w}^{t}\\ \mathbf{w}^{t+1} &= \mathbf{w}^{t}-\eta\left[[\sigma(\mathbf{c}_{pos}^{t}\cdot\mathbf{w}^{t})-1]\mathbf{c}_{pos}^{t}+\sum_{i=1}^{k}[\sigma(\mathbf{c}_{neg_i}^{t}\cdot\mathbf{w}^{t})]\mathbf{c}_{neg_i}^{t}\right]\end{aligned}$$

就像在逻辑回归中一样，学习算法从随机初始化的 **W** 和 **C** 矩阵开始，然后通过遍历训练语料库，使用梯度下降调整 **W** 和 **C**，以通过 (公式 6.38)-(公式 6.40) 的更新来最小化公式 6.34 中的损失。

回顾一下，跳词模型为每个词 **i** 学习两个独立的嵌入：目标嵌入 **wi** 和上下文嵌入 **ci**，它们分别存储在两个矩阵中，即目标矩阵 **W** 和上下文矩阵 **C**。通常的做法是将它们相加，用向量 **wi + ci** 表示词 **i**。或者，我们也可以丢弃 **C** 矩阵，仅用向量 **wi** 表示每个词 **i**。

与简单的基于计数的方法（如 tf-idf）一样，上下文窗口大小 **L** 会影响跳词嵌入的性能，实验通常会在开发集上调整参数 L。

6.8.3 其他类型的静态嵌入

静态嵌入有许多不同的类型。FastText（Bojanowski 等，2017）是 Word2Vec 的一种扩展，解决了我们迄今为止提到的 Word2Vec 的一个问题：它对未知词汇——出现在测试语料库中但未在训练语料库中见过的词汇——处理得不够好。一个相关的问题是词汇稀疏性，特别是在形态丰富的语言中，每个名词和动词的众多形式可能很少出现。FastText 通过使用子词模型解决这些问题，将每个词表示为其本身加上组成的 n-gram 的集合，并为每个词添加特殊的边界符号“<”和“>”。例如，当 n=3 时，单词 "where" 会表示为 "<where>" 以及字符 n-gram 的序列：

$$\texttt{<wh, whe, her, ere, re>}$$

然后为每个组成的 n-gram 学习一个 skip-gram 嵌入，单词 "where" 由其组成的所有 n-gram 嵌入的和表示。未知的词汇可以仅通过其组成 n-gram 的和来表示。一个 FastText 开源库，包括157种语言的预训练嵌入，可在 https://fasttext.cc 获取。

另一个非常广泛使用的静态嵌入模型是 GloVe（Pennington 等，2014），其全称为 Global Vectors（全局向量），因为该模型基于捕捉全局语料库统计信息。GloVe 基于词-词共现矩阵的概率比率，结合了基于计数的模型（如 PPMI）的直觉，同时还捕捉了像 Word2Vec 这样的方法所使用的线性结构。

事实证明，像 Word2Vec 这样的密集嵌入与像 PPMI 这样的稀疏嵌入有一个优雅的数学关系，其中 Word2Vec 可以被看作是隐式优化 PPMI 矩阵的某个函数（Levy 和 Goldberg，2014c）。

6.9 可视化嵌入

> “只要维度在两个左右，我在多个维度中都能看得很清楚。”
>
> ——已故经济学家马丁·舒比克

可视化嵌入是帮助理解、应用和改进这些词语意义模型的重要目标。但是，如何可视化一个（例如）100维的向量呢？

最简单的方式是列出嵌入在空间中的单词 w 的最相似单词，方法是将词汇表中所有单词的向量按照与 w 的向量的余弦相似度排序。例如，使用 GloVe 算法计算的青蛙（frog）嵌入的最相似的 7 个单词是：frogs、toad、litoria、leptodactylidae、rana、lizard 和 eleutherodactylus（Pennington 等, 2014）。

另一种可视化方法是使用聚类算法，展示嵌入空间中哪些单词与其他单词相似的层次结构表示。下图使用名词的一些嵌入向量的层次聚类作为可视化方法（Rohde 等, 2006）。

然而，最常见的可视化方法可能是将一个词的 100 维向量投影到 2 维中。图 6.1 和图 6.16 显示了这样的可视化，使用了一种名为 t-SNE 的投影方法（van der Maaten 和 Hinton, 2008）。

6.10 嵌入的语义属性

本节我们将简要总结一些已经被研究的嵌入语义属性。

不同类型的相似性或关联：一个与稀疏 tf-idf 向量和密集的 word2vec 向量相关的向量语义模型参数是用于收集计数的上下文窗口的大小。通常，目标词的每一侧上下文窗口在 1 到 10 个单词之间（总上下

文为 2-20 个单词）。

窗口大小的选择取决于表示的目标。较短的上下文窗口往往会导致表示更加语法化，因为信息来自于紧邻的单词。当向量是通过较短的上下文窗口计算时，与目标词 w 最相似的单词往往是具有相同词性、语义相似的单词。当向量是通过较长的上下文窗口计算时，目标词 w 的最高余弦相似度的单词往往是主题相关但不相似的单词。

例如，Levy 和 Goldberg（2014a）表明，使用 ±2 窗口的跳字模型（skip-gram），与单词“霍格沃茨”（Hogwarts，来自哈利波特系列）最相似的单词是其他虚构学校的名字：Sunnydale（来自《吸血鬼猎人巴菲》）或 Evernight（来自一部吸血鬼系列）。而使用 ±5 的窗口时，与霍格沃茨最相似的单词是与哈利波特系列主题相关的其他单词：邓布利多、马尔福和混血王子。

通常，还需要区分两种类型的单词相似性或关联（Schütze 和 Pedersen, 1993）。如果两个单词通常相互靠近，则称它们具有**一级共现**（**first-order co-occurrence**，有时称为**组合关联**）。因此，write（写作）是 book（书）或 poem（诗）的一级关联。两个单词具有**二级共现**（**second-order co-occurrence**有时称为**范式关联**）则是指它们有相似的邻居。因此，write 是 said（说）或 remarked（评论）等词的二级关联。

类比/关系相似性：嵌入的另一种语义属性是它们捕捉关系含义的能力。在早期的认知向量空间模型中，Rumelhart 和 Abrahamson（1973）提出了**平行四边形模型**（**parallelogram model**）来解决类似于“a 之于 b 就像 a* 之于什么？”的简单类比问题。在这些问题中，系统会给出一个问题，例如“apple:tree::grape:?”，即“苹果之于树，葡萄之于什么？”，并且需要填写单词“vine”（葡萄藤）。在平行四边形模型中，如图 6.15 所示，从单词“apple”到单词“tree”的向量加上“grape”的向量，返回该点最近的单词。

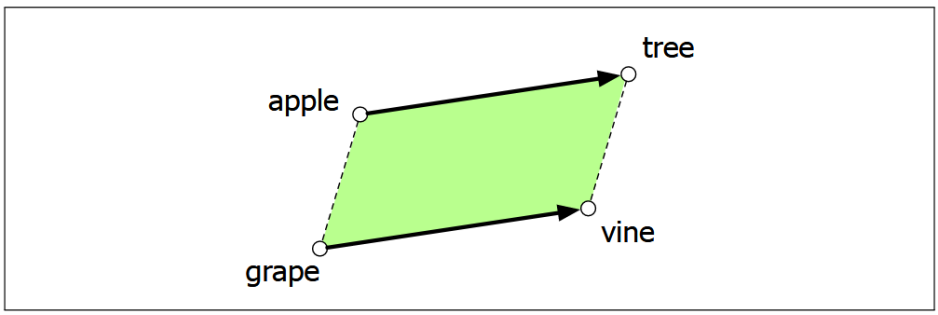

图 6.15 类比问题的平行四边形模型（Rumelhart 和 Abrahamson，1973）：向量vine 的位置可以通过从向量tree 中减去 向量apple 并奖赏向量grape来找到。

在早期关于稀疏嵌入的研究中，学者们表明，稀疏向量模型可以解决这样的类比问题（Turney 和 Littman, 2005），但由于它在 word2vec 或 GloVe 向量中的成功，平行四边形方法受到了更多的现代关注（Mikolov 等, 2013c, Levy 和 Goldberg, 2014b, Pennington 等, 2014）。例如，向量king − 向量man + 向量woman 的结果是一个接近 queen 的向量。同样，向量Paris − 向量France + 向量Italy 结果接近 # » Rome 的向量。因此，嵌入模型似乎提取了诸如 MALE-FEMALE（男性-女性）或 CAPITAL-CITY-OF（首都-城市）的关系表示，甚至比较级/最高级，如 GloVe 的图 6.16 所示。

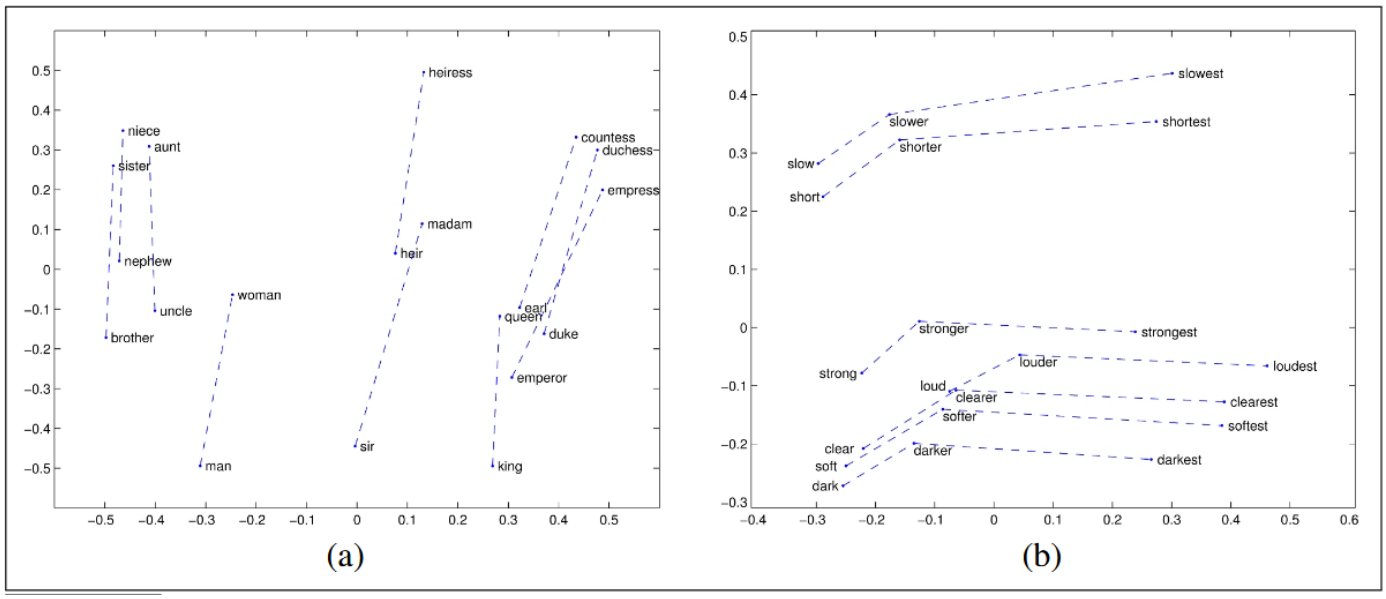

图 6.16 GloVe 向量空间的关系属性，通过将向量投影到二维上来显示。 (a) 向量king − 向量man + 向量woman 的结果是一个接近 queen 的向量。 (b) 偏移似乎捕捉了比较和最高级的形态（Pennington 等人，2014）。

对于 a : b :: a⍰ : b⍰ 问题，即算法给定向量 a、b 和 a⍰，并必须找到 b⍰，平行四边形方法如下，

$$\hat{\mathbf{b}}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ \text{distance}(\mathbf{x}, \mathbf{b} - \mathbf{a} + \mathbf{a}^*)$$

使用一些距离函数，如欧几里得距离。

有一些需要注意的地方。例如，平行四边形算法在 word2vec 或 GloVe 嵌入空间中返回的最近值通常实际上并不是 b⍰，而是输入的 3 个单词之一或它们的形态变体（即 cherry:red::potato:x 返回 potato 或 potatoes 而不是 brown），因此这些必须明确排除。此外，虽然嵌入空间在涉及常见单词、小距离和某些关系（如将国家与其首都相关联或动词/名词与其屈折形式相关联的任务）中表现良好，但嵌入的平行四边形方法在其他关系中效果不佳（Linzen 2016, Gladkova 等, 2016, Schluter 2018, Ethayarajh 等, 2019a），实际上 Peterson 等（2020）认为平行四边形方法总体上过于简单，无法模拟人类在形成此类类比时的认知过程。

6.10.1 嵌入与历史语义

嵌入也可以作为研究词义随时间变化的有用工具，方法是计算多个嵌入空间，每个空间基于特定时间段内编写的文本。例如，图 6.17 显示了过去两个世纪中英语单词词义变化的可视化结果，这些结果通过从历史语料库（如 Google n-grams (Lin 等, 2012) 和美国历史语料库 (Davies, 2012)）为每个十年构建单独的嵌入空间计算得出。

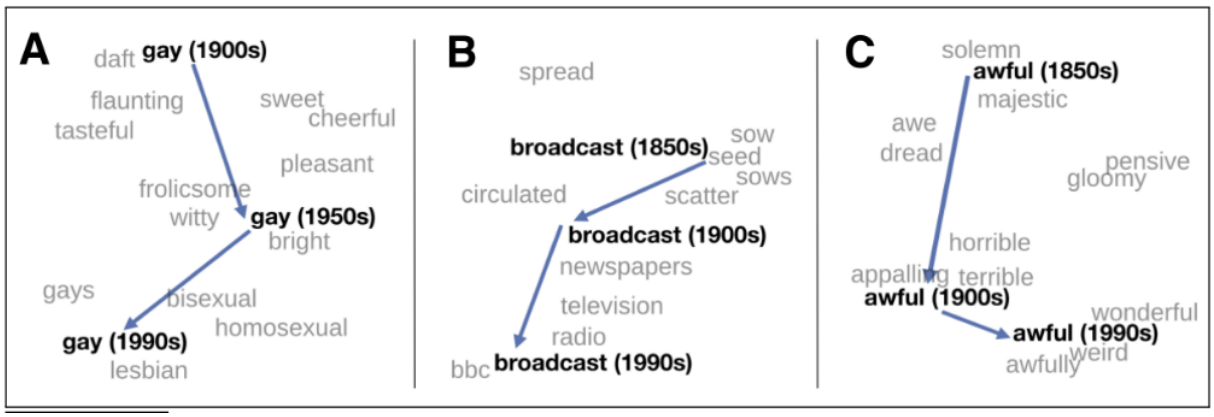

图 6.17 使用 word2vec 向量对 3 个英语单词的语义变化进行 t-SNE 可视化。每个单词的现代意义以及灰色上下文单词都是根据最近（现代）时间点嵌入空间计算的。早期点是根据早期历史嵌入空间计算的。可视化展示了 gay 一词从与“快乐”或“嬉戏”相关的含义到指代同性恋的变化，现代“broadcast”含义的广播从最初的播种含义的发展，awful 从意思“充满敬畏”转变为意思“可怕或令人震惊”

6.11 偏见与嵌入

除了能够从文本中学习词义外，嵌入模型也会重现文本中隐含的偏见和刻板印象。正如前一节所示，嵌入模型可以粗略地模拟关系相似性：‘queen’ 是 ‘king’ - ‘man’ + ‘woman’ 的最近词，这意味着类比 man:woman::king:queen。但这些相同的嵌入类比也展示了性别刻板印象。例如，Bolukbasi 等人（2016）发现，在基于新闻文本训练的 word2vec 嵌入中，‘computer programmer’ - ‘man’ + ‘woman’ 最接近的职业是 ‘homemaker’（家庭主妇），而嵌入同样暗示了 ‘father’ 对应 ‘doctor’，就像 ‘mother’ 对应 ‘nurse’。这种现象可能会导致 Crawford（2017）和 Blodgett 等人（2020）所称的分配性伤害，即当系统不公平地向不同群体分配资源（如工作或信贷）时产生的伤害。例如，使用嵌入作为寻找潜在程序员或医生的搜索算法可能错误地降低带有女性名字的文档的权重。

事实证明，嵌入不仅仅反映其输入的统计数据，还会放大偏见；在嵌入空间中，性别词汇比输入文本中的统计数据更加具有性别偏见（Zhao 等人，2017；Ethayarajh 等人，2019b；Jia 等人，2020），而且嵌入的偏见比实际的劳动力就业统计更加夸张（Garg 等人，2018）。

嵌入还编码了人类推理中的隐含关联。隐含关联测试（Greenwald 等人，1998）通过测量人们标记词语时在不同类别中的反应时间差异，来衡量人们在概念（如 ‘花’ 或 ‘昆虫’）与属性（如 ‘愉快’ 和 ‘不愉快’）之间的关联。使用这种方法的研究表明，美国人往往将非洲裔美国人的名字与不愉快的词语联系在一起（相比欧洲裔美国人的名字），男性名字与数学联系更多，女性名字与艺术联系更多，老年人的名字则与不愉快的词语联系在一起（Greenwald 等，1998；Nosek 等，2002a；Nosek 等，2002b）。Caliskan 等人（2017）使用 GloVe 向量和余弦相似度重现了这些隐含关联的发现，而不是依赖于人类反应时间。例如，‘Leroy’ 和 ‘Shaniqua’ 等非洲裔美国人的名字与不愉快词语的 GloVe 余弦相似度更高，而 ‘Brad’、‘Greg’ 和 ‘Courtney’ 等欧洲裔美国人的名字则与愉快词语的余弦相似度更高。这些嵌入中的问题是 ‘表现性伤害’（Crawford，2017；Blodgett 等，2020）的一个例子，即系统贬低或忽视某些社会群体时造成的伤害。任何利用词语情感的嵌入感知算法都有可能加剧对非洲裔美国人的偏见。

最近的研究集中于尝试消除这些偏见的方法，例如通过开发嵌入空间的转换，去除性别刻板印象但保留定义性别的特征（Bolukbasi 等，2016；Zhao 等，2017），或改变训练过程（Zhao 等，2018）。然而，尽管这些去偏见方法可能减少嵌入中的偏见，但并不能完全消除它（Gonen 和 Goldberg，2019），这仍然是一个未解决的问题。

历史嵌入也被用于衡量过去的偏见。Garg 等人（2018）使用历史文本中的嵌入来衡量职业嵌入与不同种族或性别名字嵌入之间的关联（例如女性名字与男性名字相对于职业词语如‘图书管理员’或‘木匠’的相对余弦相似度）在整个 20 世纪的变化。他们发现这些余弦值与历史上女性或种族群体在这些职业中的比例存在相关性。历史嵌入还重现了旧时对种族刻板印象的调查结果；1933 年实验参与者将‘勤奋’或‘迷信’等形容词与中国民族联系在一起的倾向，与通过 1930 年代文本训练的嵌入中，中国姓氏与这些形容词的余弦值呈相关性。他们还记录了历史上的性别偏见，例如与能力相关的形容词（‘聪明’、‘智慧’、‘有思想’、‘足智多谋’）与男性词汇的余弦值比女性词汇更高，并显示这种偏见自 1960 年以来逐渐减少。我们将在后续章节中回到自然语言处理中的偏见问题。

6.12 向量模型的评估

向量模型最重要的评估指标是任务的外部评估，即在自然语言处理（NLP）任务中使用向量，并观察其在性能上是否优于其他模型。

尽管如此，进行内部评估也是有用的。最常见的内部评估指标是测试向量在相似性任务上的表现，通过计算算法的词语相似性分数与人类赋予的词语相似性评分之间的相关性。WordSim-353（Finkelstein 等，2002）是一个常用的评分集，针对 353 对名词进行 0 到 10 的评分；例如，（plane, car）平均得分为 5.77。SimLex-999（Hill 等，2015）是一个更复杂的数据集，它量化了相似性（例如 cup, mug），而不是关联性（例如 cup, coffee），并包括具体和抽象的形容词、名词和动词对。TOEFL 数据集是由 80 个问题组成的集合，每个问题由一个目标词和 4 个额外的词选项组成；任务是选择正确的同义词，例如：Levied 最接近的意思是：imposed, believed, requested, correlated（Landauer 和 Dumais, 1997）。所有这些数据集中的词语都没有提供上下文。

稍微更真实一些的是包含上下文的内在相似性任务。斯坦福上下文词相似性（SCWS）数据集（Huang 等，2012）和词语上下文（WiC）数据集（Pilehvar 和 Camacho-Collados，2019）提供了更丰富的评估场景。SCWS 提供了 2003 对词语在句子上下文中的人类评分，而 WiC 提供了目标词在两个句子上下文中的不同用法；详见附录 G。语义文本相似性任务（Agirre 等，2012；Agirre 等，2015）评估的是句子级相似性算法的性能，任务包括一组句子对，每对句子都有人工标注的相似性分数。

另一个用于评估的任务是类比任务（在第 24 页讨论过），系统需要解决类似 a 之于 b 就像 a* 之于 b* 的问题，给定 a、b 和 a*，并找到 b*（Turney 和 Littman，2005）。为此任务创建了多个元组集合（Mikolov 等，2013a；Mikolov 等，2013c；Gladkova 等，2016），涵盖了形态变化（如 city:cities::child:children）、词典关系（如 leg:table::spout:teapot）和百科关系（如 Beijing:China::Dublin:Ireland），有些数据集取自 SemEval-2012 Task 2 的 79 种不同关系的数据集（Jurgens 等，2012）。

所有嵌入算法都存在固有的变异性。例如，由于初始化的随机性和随机负采样，像 word2vec 这样的算法即使在同一个数据集上也可能产生不同的结果，并且文集中的个别文档可能对生成的嵌入有

很大影响（Tian 等，2016；Hellrich 和 Hahn，2016；Antoniak 和 Mimno，2018）。因此，当使用嵌入来研究特定语料库中的词语关联时，最好的做法是使用引导采样对文档进行多次训练并对结果进行平均（Antoniak 和 Mimno，2018）。

6.13 总结

- 在向量语义中，词语被建模为一个向量——一个位于高维空间中的点，也称为**嵌入**。本章我们聚焦于**静态嵌入**，即每个词语映射到一个固定的嵌入。
- 向量语义模型分为两类：**稀疏模型**和**密集模型**。在稀疏模型中，每个维度对应于词汇表 V 中的一个词，单元格是**共现计数**的函数。**词-文档**矩阵对词汇表中的每个词（术语）都有一行，对每个文档都有一列。**词-上下文**或**词-词**矩阵对词汇表中的每个（目标）词都有一行，对词汇表中的每个上下文术语都有一列。两种常见的稀疏加权方法是：**tf-idf 加权**（按词频和逆文档频率加权每个单元格）和 **PPMI**（逐点正互信息），后者最常用于词-上下文矩阵。
- 密集向量模型的维度通常在 50 到 1000 之间。像 **skip-gram** 这样的 **word2vec** 算法是计算密集嵌入的流行方法。skip-gram 训练一个逻辑回归分类器来计算两个词‘可能在文本中同时出现’的概率。这个概率通过两个词的嵌入之间的点积计算得出。
- Skip-gram 使用随机梯度下降法训练分类器，通过学习使附近出现的词语嵌入具有高点积，而噪声词语嵌入具有低点积。
- 其他重要的嵌入算法包括 **GloVe**，这是一种基于词共现概率比率的方法。
- 无论是使用稀疏向量还是密集向量，词语和文档的相似性都是通过向量之间**点积**的某种函数来计算的。两个向量的余弦——标准化的点积——是最流行的此类度量标准。