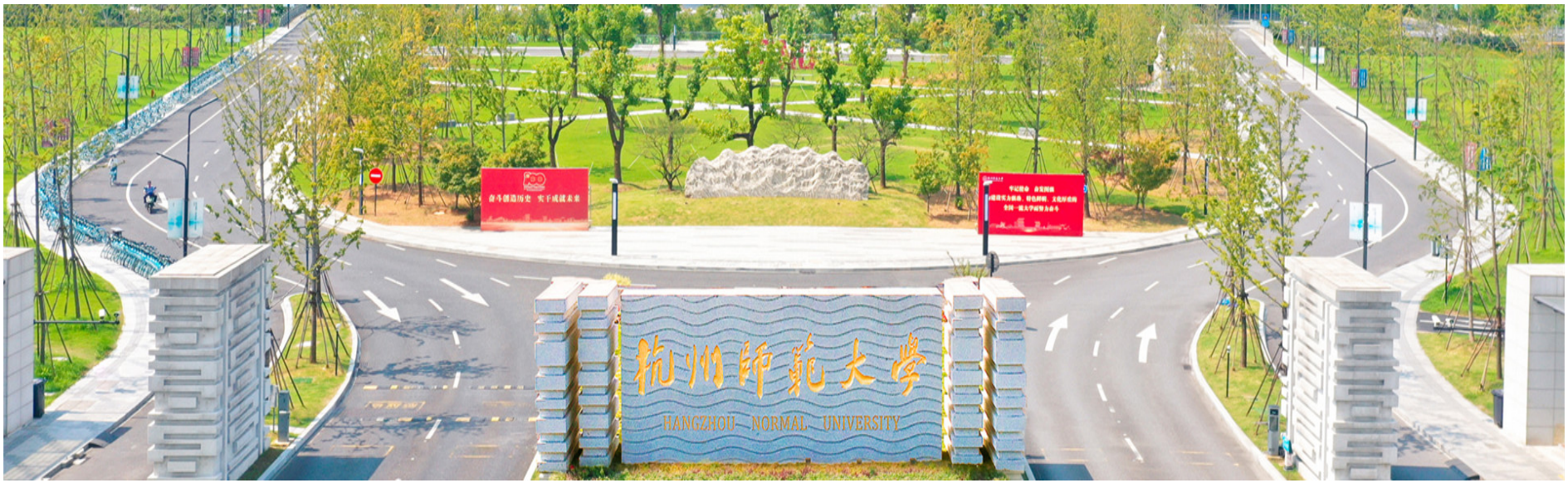
第六讲 - 向量语义与嵌入

张建章

阿里巴巴商学院
杭州师范大学

2025-02-01



目录

分布假设：词语的意义并非孤立存在，而是与它在文本中出现的环境密切相关。早在1950年代，Joos、Harris、Firth 等语言学家便观察到：**相近语义的词往往会在类似的上下文中共同出现**。这一观察形成了“**分布假设**”，即“词语相似性可以通过其分布特点来度量”。

向量表示：心理学家 Osgood 等人在 1957 年提出，将词语的情感语义信息用多维空间中的一个点来表示，提出了三个关键维度：愉悦度（valence）、唤起度（arousal）和支配度（dominance），每个词可以用一个三维向量来进行描述，从而刻画其在情感上的位置。

稀疏表示到密集嵌入的转变：① 早期，基于共现计数的稀疏（高维）向量表示方法，如 tf-idf 和 PPMI 等；② 现在，基于自监督学习的密集型（低维）词向量模型。稀疏表示虽然直观，但维数庞大且难以捕捉深层语义结构；而密集嵌入（如 word2vec、GloVe）则能在较低维空间中更有效地反映词语间的语义距离和相似性。

目录

1. 同义关系（Synonymy）

当两个词在某个语义层面具有相似或近似的意义时，便可称其为同义词。同义的形式化定义为：若两个词可以在任意句子中相互替换而不改变句子的真值条件，则它们为同义词。**由于形式上的差别必然伴随某种语义差异，在实际中很难找到绝对意义完全相同的词，因此同义关系通常指近似同义。**

2. 词语相似性（Word Similarity）

词语并不总是严格的同义，而更多是具有相似性。以“cat”与“dog”为例，它们在具体语义上并非替换性同义，但在人们的直观感知中仍被视为语义上相近。这种相似性为句子或文本的整体语义比较提供了依据。

3. 词语关联性（Word Relatedness）

除了相似性外，词语之间还存在其他类型的关联性。例如“coffee”与“cup”虽然在属性上没有太多重合，却在实际使用中密切相关，因为它们常共同出现在饮用咖啡的情境中，**属于同一个语义场**。

语义场（semantic field）是指覆盖特定语义范畴、且彼此之间存在结构化关联的一组词汇，如外科医生、手术刀、护士等均属于医院相关词汇。语义场为发现文本中的主题结构和关联关系提供了理论支撑。

目录

文档与词汇的向量空间模型

将文档和词汇映射到高维向量空间中的核心思想是：利用共现矩阵构建向量表示，以捕捉文本中蕴含的统计信息与语义联系。

1. 利用文档构造向量（Term-Document Matrix）

在文档向量表示中，每一行对应词汇表中的一个词，每一列对应文档集合中的一个文档。矩阵中每个单元格记录了某一词在某一文档中出现的次数。

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

每个文档可以看作一个向量，其维度等于词汇表大小。在此模型中，文档间的相似性通过比较对应向量（如在“battle”与“fool”这两个维度上的分布）得以体现。如果两个文档在大多数词汇上具有类似的计数分布，那么它们在向量空间中距离较近，表示语义上具有相似性。

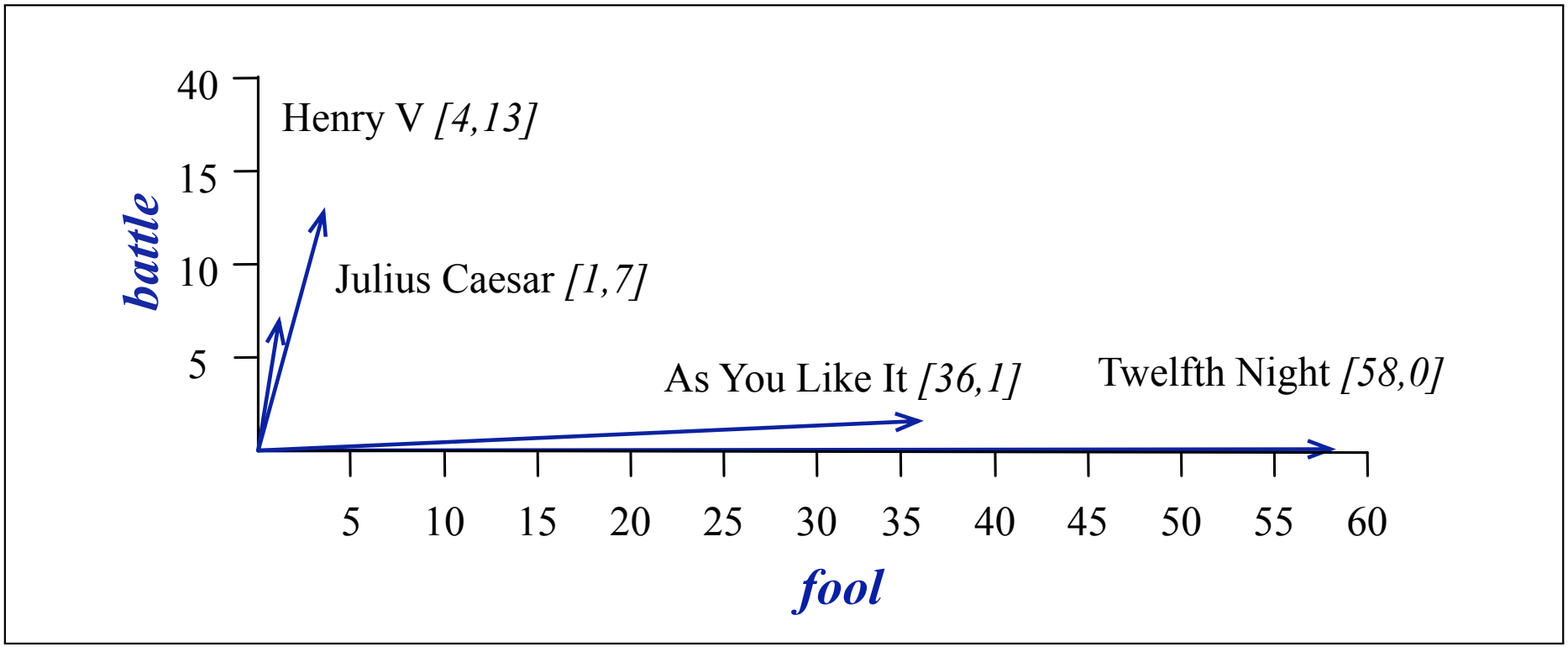


Figure 6.4 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

以文档为维度构造词向量

1. 词向量的构造

利用上述的词—文档矩阵，每个词可以以一行向量表示，其维度仍然是文档数。即一个词的向量反映了该词在不同文档中出现的频率分布。

	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

Figure 6.5 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each word is represented as a row vector of length four.

基于分布假设，相似词往往出现在相似的文档中，因此其向量在文档维度下会表现出相近的数值分布。通过比较这些行向量，可以捕获词汇之间的语义相似性。这种方法直接利用文档统计信息来刻画每个词的语义特征，从而使得相同或相近意义的词（尽管形式不同）在向量空间中趋于聚集，便于在后续任务中进行相似性度量和信息检索。

以词为维度构造词向量——词—词共现矩阵

1. 词—词（Term-Term 或 Word-Word）矩阵

除了使用文档作为向量维度 (粗粒度)，还可以利用词**在上下文窗口内的共现情况**来构造向量 (细粒度)。此时，每一行仍表示目标词，但每一列对应的不是整个文档，而是语境中的另一个词。统计目标词与各上下文词在固定窗口（例如左右各4个词）中的共现次数，形成的便是词—词共现矩阵。由于词汇量通常较大（例如常见词可能达到一万个甚至五万个），因此这种矩阵往往是高维且稀疏的。

	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	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

例如，“cherry”与“strawberry”的共现向量在与“pie”或“sugar”等词上的计数较高，从而体现了二者在饮食或甜点语境下的密切联系。

例如，“digital”与“information”在相关上下文中的共现频次表明了两者的语义相关性。

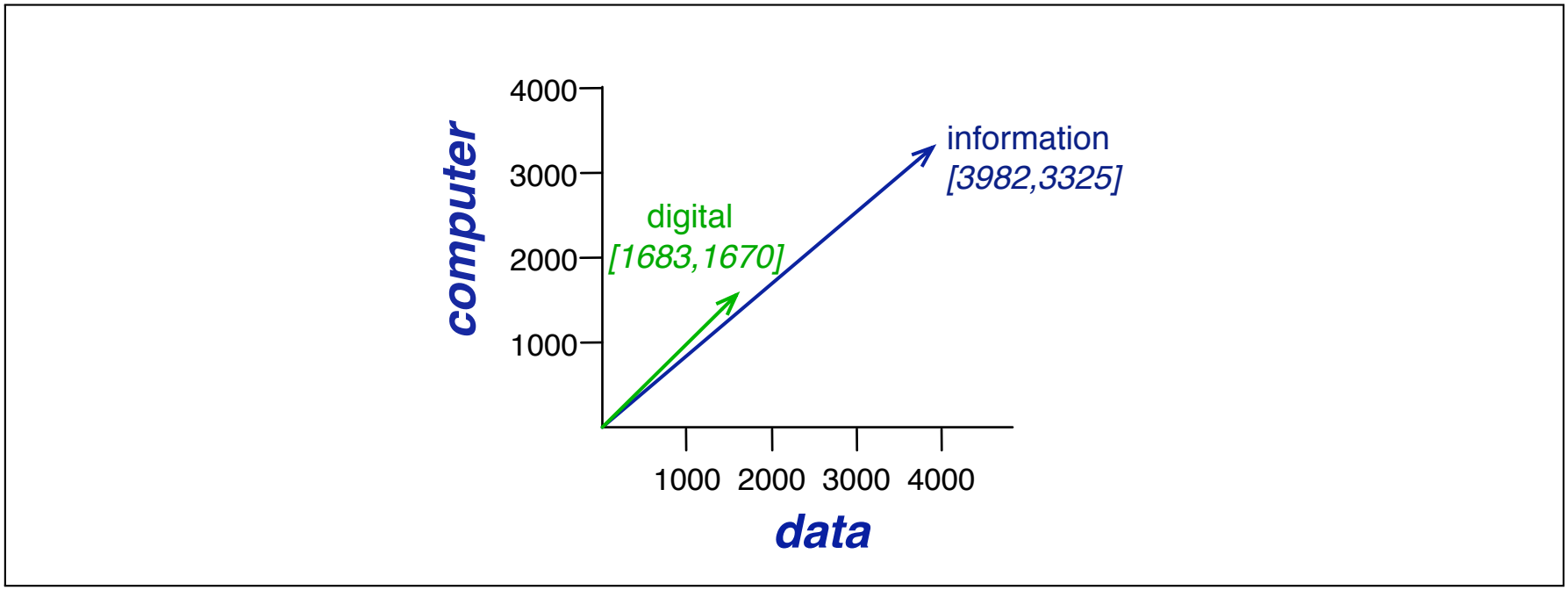


Figure 6.7 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

细粒度上下文的优势：利用词—词共现矩阵，可以更直接地反映词汇在局部语境中的相似性。与文档级别的统计不同，局部共现更能揭示词与词之间微妙的语义差异，对于捕捉同义、相似以及联想关系均具有较高的敏感度。

目录

余弦相似度的基本思想

在向量语义模型中，最常用的比较两个向量之间的相似程度的衡量指标是“余弦相似度”，即，向量之间夹角的余弦值。

1. 余弦相似度的定义

余弦相似度基于向量的点积（内积）运算。对于两个同维度的向量 $\mathbf{v}$ 与 $\mathbf{w}$，它们的点积定义为：

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i$$

为避免直接使用点积时因向量长度不同而导致的偏差，将点积结果归一化，定义余弦相似度为：

$$\cos(\theta) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}|\ |\mathbf{w}|}$$

其中，向量的模长 $|\mathbf{v}|$ 定义为：

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

点积反映共同特征：当两个向量在相同维度上均取较高数值时，点积将较大；而若两个向量在不同维度上呈现较高值，则点积相对较低。这说明点积可反映它们在语义空间中是否在相同方向上存在较强重合。

归一化消除长度差异：长向量（频次较高）在原始点积上会天然具有较高值，为了不受这一因素影响，采用模长归一化，使得余弦相似度只反映向量之间的夹角大小。**对归一化后的单位向量，其点积直接等于余弦值：**

$$\cos(\theta) = \frac{\mathbf{v}}{|\mathbf{v}|} \cdot \frac{\mathbf{w}}{|\mathbf{w}|}$$

2. 余弦相似度的优点

- **尺度无关性：**由于余弦相似度对向量的模长进行了标准化，使得无论词的出现频次如何，只要其分布的“形状”相似（即各维度上占比类似），其余弦值就会较高。

- **取值范围明确：**当所有向量分量均为非负数时（如基于词频构造的共现向量），余弦值的范围从 0 到 1，数值越大说明相似度越高，这在具体应用中直观易懂。

- **计算简便、效率较高：**余弦相似度依赖基本的线性代数运算（点积和模长计算），实现简单且计算速度快，适用于大规模文本数据处理。

示例说明与应用场景

构造一个包含“cherry”、“digital”及“information”的共现计数矩阵，计算词语之间的余弦相似度：

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

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

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

上例结果直观地表明，“information”与“digital”在统计上（从共现模式看）具有高度相似性，而与“cherry”则相去甚远，从而体现出余弦相似度在捕捉词义相似关系上的有效性。

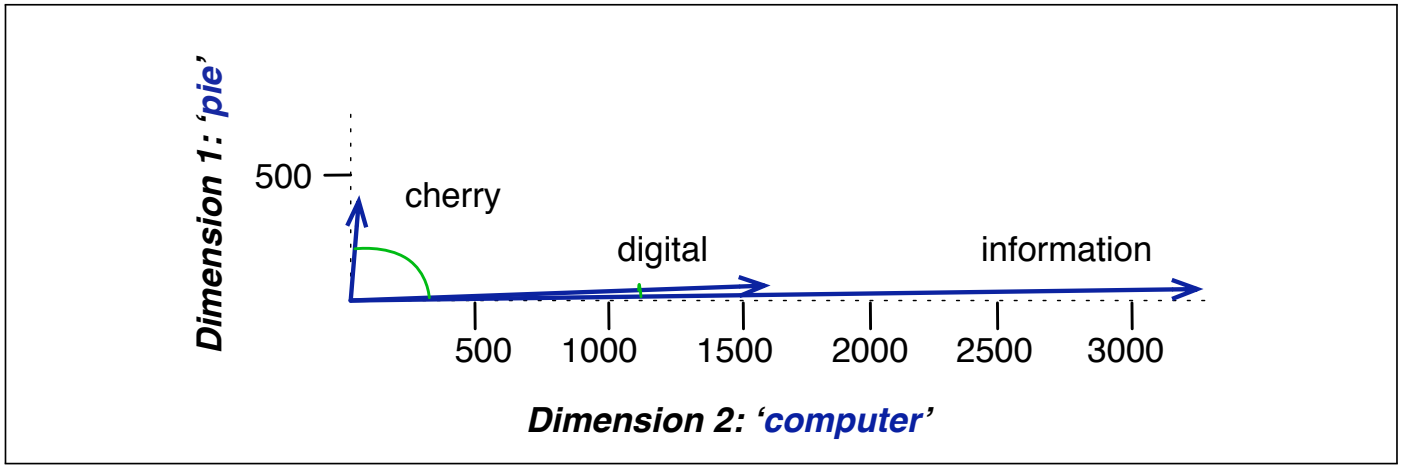


Figure 6.8 A (rough) graphical demonstration of cosine similarity, showing vectors for three words (*cherry*, *digital*, and *information*) in the two dimensional space defined by counts of the words *computer* and *pie* nearby. The figure doesn't show the cosine, but it highlights the angles; note that the angle between *digital* and *information* is smaller than the angle between *cherry* and *information*. When two vectors are more similar, the cosine is larger but the angle is smaller; the cosine has its maximum (1) when the angle between two vectors is smallest (0°); the cosine of all other angles is less than 1.

词相似性计算：度量两个词的向量表示的相似性，进而在同义词发现、词义消歧、信息检索等任务中发挥重要作用。

文档相似度计算：通过计算文档向量间的余弦相似度，可以评估两个文档在主题、内容上是否接近，辅助分类、聚类等任务的开展。

目录

问题背景及动机

频次统计的局限性：在构造词—文档或词—词共现矩阵时，通常直接使用原始频次来描述词语的共现情况。然而，原始频次分布往往呈现高度偏斜的特性：**某些常见词（如“the”、“good”）在所有文档中均出现较频繁，这使得这些词在向量表示中占据较大的权重，导致其缺乏区分性**。为了更好地捕捉对文档语义具有鉴别作用的单词，同时降低那些高频且信息量不高的单词的影响，就需要**对原始频次进行“加权”处理**。

权衡信息量与常见性的矛盾：一个单词在某文档中出现得越频繁，通常表明该单词对文档的主题具有较大贡献；但另一方面，如果一个单词在整个文档集合中均频繁出现，则它对区分各文档的重要性较低。因此，**如何在“词频高代表重要”与“全局高频代表普遍性”之间取得平衡，是词频-逆文档频 (TF-IDF)设计的初衷**。

TF-IDF 的两个核心组成部分

TF-IDF 模型将单词在文档中的重要性拆解为两个相乘的部分：

1. 词频（Term Frequency, tf）

词频 $tf_{t,d}$ 表示单词 t 在文档 d 中出现的次数。最简单的定义就是直接使用原始计数：

$$tf_{t,d} = \mathsf{count}(t, d)$$

对频次的对数平滑处理：为避免词频极高的单词对模型产生过度影响，同时体现“出现 100 次并不意味着比出现 10 次重要 10 倍”这一直觉，通常**采用对数变换对频次进行“平滑”：**

$$tf_{t,d} = \begin{cases} 1 + \log_{10}(\mathsf{count}(t, d)) & \text{if } \mathsf{count}(t, d) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

例如，当单词在文档中出现 1 次、10 次、100 次时，对应的 tf 分别为 1、2、3，从而**弱化频次增加带来的线性影响**。

逆文档频率（Inverse Document Frequency, idf）

逆文档频率 idf_t 用于**降低在整个文档集合中普遍出现的单词的权重**：

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

其中 N 为文档总数，df_t 为包含单词 t 的文档数。当 $df_t = N$ 时，idf_t 取最低值（0），以反映这些普遍性高的单词在区分文档时信息量较低。

文档频率 df_t 表示单词 t 在多少个文档中出现；与之不同，集合频率指单词在所有文档中出现的总次数。即便两个单词在全集中出现的总次数相同，但如果一个单词只在极少数文档中出现，而另一个单词则几乎遍布所有文档，则前者能够更好地区分文档主题，应赋予更高权重。例如，莎士比亚全集中“Romeo”与“action”两词出现次数相同，但“Romeo”只在一部戏剧中出现，因此其 idf 值要高于“action”。

TF-IDF 权重的计算及示例

将词频与逆文档频率相乘，得到单词 t 在文档 d 中的 TF-IDF 权重：

$$w_{t,d} = tf_{t,d} \times idf_t$$

该公式同时考虑了**单词在局部文档中的重要性与其在全局集合中的区分性**，从而对各个单词赋予合适的权重。

计算实例：在37部莎士比亚戏剧中统计词语的逆文档频，如下：

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

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

	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

Figure 6.9 A portion of the tf-idf weighted term-document matrix for four words in Shakespeare plays, showing a selection of 4 plays, using counts from Fig. 6.2. For example the 0.085 value for *wit* in *As You Like It* is the product of $\text{tf} = 1 + \log_{10}(20) = 2.301$ and $\text{idf} = .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

对于“good”这一单词，虽然在每个戏剧中均出现频繁，但由于它在所有戏剧中的文档频率很高，其 *idf* 值趋近于 0，因此最终的 TF-IDF 权重也很低，反映出其对区分不同戏剧主题信息量较小。同样，出现频率较低且局限于少数文档的单词（例如“Romeo”）会获得较高的 *idf* 值，从而使其 TF-IDF 权重较高，突显其在反映特定主题上的作用。

目录

1. 基本理念

点互信息 (Pointwise Mutual Information, PMI)用于判断两个词之间的“关联”程度，**通过比较实际共现概率与独立假设下共现概率的比值，来定量衡量词与词之间的协同程度。**

2. 数学定义

对于目标词 w 与上下文词 c，PMI 定义为：

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

其中，$P(w, c)$ 表示在语料中 w 和 c同时出现的概率（使用最大似然估计，从共现计数归一化得到）；$P(w)$ 与 $P(c)$ 分别是 w 与 c 各自的出现概率。

概率比值的含义：分子 $P(w, c)$ 反映了观察到的共现情况，分母 $P(w)P(c)$ 则是假设二者互相独立时的共现概率。取对数后可以得到一个数值，该值越大说明二者**实际共现频率**越超过**独立假设下共现的概率**，因而关联性越强。

3. 正点互信息（PPMI）

PMI的取值范围从负无穷到正无穷，正值表示实际共现频率超过独立假设下的预期随机共现的概率，负值表示实际共现频率低于独立预期，但在实际语料中，负PMI值往往不具备可靠性且难以解释。为此，采用PPMI（Positive PMI），将所有负值置零。

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

计算过程：首先构建词—词共现矩阵 F（行对应目标词，列对应语境词），利用统计数据计算：

- 联合概率：$p_{ij} = \dfrac{f_{ij}}{\sum_{i,j} f_{ij}}$；
- 边缘概率：$p_{i*} = \sum_j \dfrac{f_{ij}}{\sum_{i,j} f_{ij}}$ 与 $p_{*j} = \sum_i \dfrac{f_{ij}}{\sum_{i,j} f_{ij}}$；
- 计算每个单元格的 PMI，并将小于零的值置为 0。

计算示例

	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

Figure 6.10 Co-occurrence counts for four words in 5 contexts in the Wikipedia corpus, together with the marginals, pretending for the purpose of this calculation that no other words/contexts matter.

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

	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	

Figure 6.11 Replacing the counts in Fig. 6.6 with joint probabilities, showing the marginals in the right column and the bottom row.

	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

Figure 6.12 The PPMI matrix showing the association between words and context words, computed from the counts in Fig. 6.11. Note that most of the 0 PPMI values are ones that had a negative PMI; for example PMI(*cherry,computer*) = -6.7, meaning that *cherry* and *computer* co-occur on Wikipedia less often than we would expect by chance, and with PPMI we replace negative values by zero.

大多数0 PPMI值对应的原始PMI实为负数。例如，PMI(樱桃, 计算机) = -6.7，这意味着在维基百科中“樱桃”和“计算机”共同出现的频率比独立假设下预期随机共同出现的概率还要低。

改进与平滑策略

PMI 指标倾向于对低频上下文词赋予非常高的值，因为低频事件下，即使共现次数较小，其比值 $\frac{P(w,c)}{P(w)P(c)}$ 也可能被放大，为了缓解对低频词的偏差，可以**使用幂次调整上下文概率或者进行Laplace 平滑：**

1. 幂次平滑（α参数）

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

通常取 $\alpha = 0.75$ 能取得较好的平衡效果。这样，对低频上下文词，其概率经过幂次平滑后增大，从而降低计算出的 PMI 值，避免极端偏高。

2. Laplace 平滑

另外，也可以通过在共现计数中加入一个小常数 k（例如 0.1 至 3 的范围内），以达到平滑效果，从而防止零计数和降低低频词对结果的不合理影响。

目录

向量语义模型中，每个目标词都可由一个向量表示，其维度对应于整个文档集合（即词—文档矩阵）或局部上下文词（即词—词共现矩阵）。在这两种表示中，每一维的值均由词在特定文档中或在指定上下文中的出现次数得到，两种权重函数分别针对不同情形发挥作用：

TF-IDF：在词—文档矩阵中，通过对局部词频进行对数平滑，再与逆文档频率的乘积，降低了在所有文档中高频出现的词的权重，突出那些能区分文档特征的低频词。

PPMI：在词—词共现矩阵中，通过计算目标词与上下文词联合出现概率与独立出现概率比值的对数，并将负值置零，使得实际共现显著高于预期的词对获得较高权重，从而更好地揭示词汇间的语义关联。

语义比较：向量相似性计算

无论是 TF-IDF 还是 PPMI 加权后得到的向量，向量语义模型均采用余弦相似度作为两向量间相似性比较的主要指标。通过计算两个向量夹角的余弦值，能够衡量它们在语义空间中方向上的接近程度，余弦值越接近 1，说明两个词或文档在语义上越相似；反之，数值较低则表明二者差异较大。

文档向量的构造方法：将文档中所有词的向量进行求和后取平均，得到该文档的中心向量（**质心**）：

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

这一操作实际是在多维空间中寻找一“平均点”，该点能**最小化与文档中各词向量之间的平方距离总和**。

具体应用场景

文档相似性计算：信息检索、文档聚类及抄袭检测等任务。

词汇相似性计算：词汇意义拓展、语义演变追踪、词义消歧及文本自动释义。

其他应用：新闻推荐、数字人文研究（如比较不同文本版本间的内容相似性）等多种场景。

核心思路均为利用向量空间中各单元的加权分布，客观量化文本中信息的相似性，为后续分类、聚类和检索提供依据。

目录

模型背景与动机

传统的文本表示方法（如基于词—文档或词—词共现矩阵的**稀疏向量表示**）虽然能够通过加权策略（TF-IDF、PPMI 等）捕捉词语之间的语义信息，但**其向量维度通常高达数万且稀疏**，既计算资源消耗较大，又可能**无法充分捕捉词汇之间的细微相似性**。

Word2vec 提出了一种通过神经网络方法训练得到的**低维、密集向量（即词嵌入）的表示方法**，其优点在于降低了参数规模、提高了计算效率，并在各种自然语言处理任务上表现出优越性。

基本思想与模型直观

1. 嵌入向量的定义

Word2vec 的核心在于将词映射到一个低维连续向量空间中，这些向量通常维度在 50 到 1000 之间，与传统表示中基于词频的高维稀疏向量相比，具有更紧凑和更连续的特性。其**每个维度虽没有明确的语义解释，但整体能够捕捉词的语义和句法信息。**

2. 自监督学习范式

模型利用大量无标注文本中的上下文信息构造训练数据，无需人工标注。具体来说，通过**利用“目标词与上下文词共现”这一自然现象，将该共现关系作为隐式监督信号，从而直接从数据中“学习”词的分布式表示。**

模型构成：Skip-gram with Negative Sampling（SGNS）

1. 正、负样本构造

Word2vec 中最常用的实现方式为 Skip-gram 模型，其基本思想是：

① 将句子中**中心词**与其在一定窗口（例如 ±2 个词）内的**上下文词**视为正样本。

② 针对每一个正样本，随机从词汇中采样若干“噪声 (上下文)词”，构建负样本，要求这些噪声词不等于目标词。

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

This example has a target word w (apricot), and 4 context words in the $L = \pm 2$ window, resulting in 4 positive training instances (on the left below):

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

2. 基于嵌入相似度的概率计算

模型的核心假设是：**若一个候选上下文词的嵌入向量与目标词的嵌入向量相似，则该词更有可能在目标词的上下文中出现**。直观地，向量之间的相似性可以由它们的点积（内积）来衡量，即

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

但由于点积值的范围是 $-\infty$ 到 $+\infty$，无法直接解释为概率。

3. 点积到概率的转换：使用 Sigmoid 函数

为了将点积转化为概率，模型使用 Logistic 回归中的 Sigmoid 函数：

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

正例概率表达：将目标词 w 与候选上下文 c 的点积经过 Sigmoid 转换后，得到

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

这一概率值介于 0 和 1 之间，高值表示 c 很可能为真实上下文词。

对应地，**负例的概率定义为：**

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

4. 整体分类器模型与多上下文处理

当目标词 w 拥有多个上下文词（记为 $c_1, c_2, \ldots, c_L$）时，模型假设这些上下文词相互独立，则整个上下文窗口为正样本的联合概率为：

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

其对数形式为

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \sigma(c_i \cdot w)$$

整体上，Skip-gram 模型训练一个二元分类器，其输入为目标词与候选上下文词对，通过计算二者嵌入向量的点积并经过 Sigmoid 转换，得到该候选词是否为真实上下文的概率。

模型参数：每个词的两种嵌入，① 作为目标词时的向量表示，② 作为上下文词或噪音词时的向量表示。

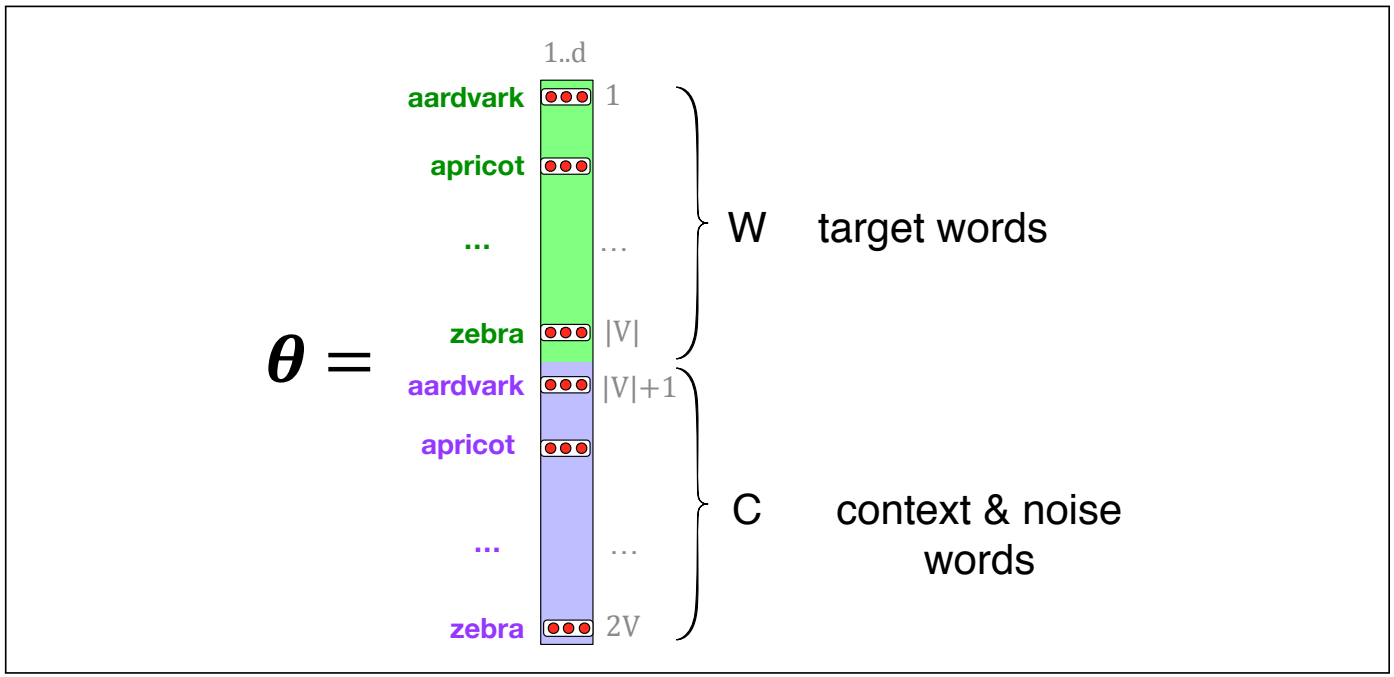


Figure 6.13 The embeddings learned by the skipgram model. The algorithm stores two embeddings for each word, the target embedding (sometimes called the input embedding) and the context embedding (sometimes called the output embedding). The parameter θ that the algorithm learns is thus a matrix of $2|V|$ vectors, each of dimension d, formed by concatenating two matrices, the target embeddings **W** and the context+noise embeddings **C**.

参数(词嵌入向量) 学习

1. 负样本构造

为了让模型不仅学会靠近真实上下文词，还能区分噪声词，每个正样本对应生成 k 个负样本对。负样本采用随机抽取方式，从词汇表中抽出不与目标词 w 相同的噪声词 c_{neg}。这一步骤中，噪声词的采样概率并非简单的频率，而是**按照加权的 unigram 分布进行采样**，其概率为

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

一般选择 $\alpha = 0.75$，这**有助于提高低频词被采样的概率，从而减小频率极差带来的影响**。

2. 训练目标与损失函数

为了使得模型能够区分“真实”上下文词和噪声词，构造如下目标函数。**对于一个正样本 (w, c_{pos}) 及其 k 个负样本** (w, c_{neg_i}) $(i = 1, 2, \ldots, k)$，定义整体损失函数为：

$$L = -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right]$$

其中，

- w 表示目标词的当前嵌入向量；
- c_{pos} 表示正样本中上下文词的向量；
- c_{neg_i} 表示第 i 个负采样噪声词的向量；
- $\sigma(x) = \frac{1}{1+\exp(-x)}$ 为 sigmoid 函数，其作用是将词向量间的点积转换为 (0,1) 区间内的概率值。

这样设计的损失函数反映了两个目标：

- **最大化**正样本对 w 与 c_{pos} 的点积，从而使真实上下文词对的相似度更高；

- **最小化**目标词与负样本噪声词之间的相似度（即最大化 $\log \sigma(-c_{neg_i} \cdot w)$），促使非上下文词对不具备语义相关性；

3. 模型参数 (嵌入表示)

模型为每个词同时学习了两种嵌入：一种是作为“目标词”（输入嵌入）的向量 w，另一种是作为“上下文词”（输出嵌入）的向量 c。这两个向量通常分别存储在目标矩阵 W 和上下文矩阵 C 中。这种双重表示的设计有助于模型在训练过程中更好地捕捉词与词之间的相关性。

最终实际使用时，每个词语的嵌入向量表示，既可以将两种嵌入相加（即 $w + c$），也可以只使用 W。

上下文窗口大小 L 的选取会对训练效果产生影响，通常需要根据具体任务进行在开发集上调参。

为最小化上述损失函数，采用随机梯度下降（SGD）方法进行优化，具体步骤包括计算损失函数对于各个嵌入向量参数的偏导数，然后根据这些梯度对参数进行更新。

① 对正样本上下文词 c_{pos} 的梯度：

$$\frac{\partial L}{\partial c_{pos}} = \left[\sigma(c_{pos} \cdot w) - 1\right] w$$

② 对每个负样本上下文词 c_{neg} 的梯度：

$$\frac{\partial L}{\partial c_{neg}} = \sigma(c_{neg} \cdot w) w$$

③ 对目标词 w 的梯度：

$$\frac{\partial L}{\partial w} = \left[\sigma(c_{pos} \cdot w) - 1\right] c_{pos} + \sum_{i=1}^{k} \sigma(c_{neg_i} \cdot w) c_{neg_i}$$

基于这些梯度，令学习率为 η，可以得到参数更新公式：

① **正样本上下文词更新：**

$$c_{pos} \leftarrow c_{pos} - \eta \left[\sigma(c_{pos} \cdot w) - 1\right] w$$

② **负样本上下文词更新：**

$$c_{neg} \leftarrow c_{neg} - \eta \, \sigma(c_{neg} \cdot w) w$$

③ **目标词更新：**

$$w \leftarrow w - \eta \left(\left[\sigma(c_{pos} \cdot w) - 1\right] c_{pos} + \sum_{i=1}^{k} \sigma(c_{neg_i} \cdot w) c_{neg_i} \right)$$

算法首先以随机初始化的目标矩阵 W 和上下文矩阵 C 作为起点，然后**遍历训练语料**，采用梯度下降法对 W 和 C 进行调整。

其他类型的静态词向量

除word2vec之外，还有其他类型的静态词嵌入模型，这些模型虽然均生成**固定、不随上下文变化的词向量**，但在处理词汇覆盖和词形稀疏性等问题上存在不同的改进策略。

1. fastText 模型 （1）fastText 是 word2vec 的一种扩展，其主要改进在于解决未知词（在测试语料中出现但在训练中未见到的词）和词形稀疏性问题。

（2）具体做法是利用子词（subword）建模，即不仅将词视作一个整体，同时分解为一系列字符 n-gram，并在每个词向量表示中加入这些 n-gram 的信息。

（3）例如，对于单词“where”，模型会在其左右加上特殊边界符号形成“<where>”，再从中提取长度为 3 的 n-gram，如“<wh”，“whe”，“her”，“ere”，“re>”。

（4）在训练过程中，会为每个 n-gram 学习一个嵌入；最终，一个词的向量由该词自身与其所有 n-gram 嵌入的加和构成，从而不仅可以表示训练中见过的词，还可以对未知词进行建模。

2. GloVe 模型

（1）GloVe（Global Vectors）是另一种被广泛使用的静态词嵌入模型。

（2）其主要思想在于利用全局共现统计信息，即基于词与词之间共现概率的比值来捕获词的语义关系。

（3）GloVe 模型既结合了类似 PPMI 这类基于计数的方法的直观优势，又能像 word2vec 那样获得低维、连续的稠密向量表示。（4）数学上可以证明 word2vec 隐式地在优化某个与 PPMI 矩阵相关的目标函数，从而揭示了这两类方法之间的内在联系。

未完待续