22 嵌入向量, 词嵌入, 子词嵌入, 全局向量的词嵌入

VCG

概要

- 嵌入向量（Embeddings）
- 词嵌入（Word2vec）
  - Skip-Gram
  - CBOW
- 子词嵌入（fastText）
- 全局向量的词嵌入（GloVe）

根本问题：如何让机器理解词义？

- 词义理解的核心挑战：计算机原生处理的是数值而非抽象的语言符号。我们需要将词语（Token）转化为机器可以计算的数学对象——向量
  - 这种向量表示必须能够编码词语的语义信息。
  - 语义相似的词，其向量在空间中也应该彼此接近
    - 国王 vs. 王后 -> 距离近
    - 国王 vs. 香蕉 -> 距离远
  - 这就是“嵌入（Embedding）”的本质目标

词嵌入（Word2vec）

早期的词嵌入 - 独热编码 (One-Hot Encoding)

- 独热编码 (One-Hot Encoding)
  - 将每个词映射为一个高维稀疏向量
    - 向量长度 = 词汇表大小 (N)
    - 只有对应词的索引位置为1，其余全为0。
      - 国王 -> [1, 0, 0, ..., 0]
      - 王后 -> [0, 1, 0, ..., 0]
      - 香蕉 -> [0, 0, 1, ..., 0]

- 但是，这种表示方法存在致命缺陷…

独热编码的困境

- 语义鸿沟 (Semantic Gap)
  - 任意两个不同词的独热向量都是正交的
  - dot(国王, 王后) = 0
  - dot(国王, 香蕉) = 0
  - 模型无法从向量本身判断“国王”和“王后”比“国王”和“香蕉”更相似。它只编码了身份，未编码语义
- 维度灾难与稀疏性 (Curse of Dimensionality & Sparsity)
  - 当词汇表很大时（如几十万），向量维度极高
  - 这给计算和存储带来巨大挑战，也使得模型难以泛化
- 核心目标：需要一种密集(Dense)、低维(Low-dimensional)的向量表示，它必须能编码词语间的语义关系

词嵌入理论基石：分布式假设

- Distributional Hypothesis，语言学基石
  - "A word is characterized by the company it keeps." (J.R. Firth, 1957)
  - 一个词的意义，由其频繁出现的上下文所决定
    - 恒星 常与 行星、星系、发光 共同出现；香蕉 常与 水果、黄色、食物 共同出现
  - 合理推断
    - 如果两个词的上下文相似，那么它们的语义也应该相似
- 语义相近的词，它们的词向量在向量空间中的距离也相近
  - 这是所有现代词嵌入方法的理论原点
- 分布式表示 (Distributed Representation):
  - 词嵌入通常是低维的，而且是稠密的
  - 一个词的意义不是由某一个维度决定的，而是“分布”在所有维度上。每个维度都可能代表一个抽象的、我们无法直接解释的语义特征
  - 通过计算向量间的余弦相似度等来衡量词语的语义相似性

词嵌入核心策略：通过“代理任务”学习

- 如何设计一个任务，让模型在完成任务的过程中，将词语的上下文信息“压缩”进一个低维稠密向量里？
- 解决方案：代理任务 (Proxy Task)
  - 不直接去“学习向量”，而是让模型去完成一个看似简单的预测任务。词向量是完成这个任务时产生的副产品
- 如，大规模的“完形填空”游戏
  - “天空是蓝色的，草地是 ___ 的。”
  - 为了填对“绿色”，模型必须理解“草地”和“绿色”之间的语义关联

- Word2vec通过让机器进行亿万次这样的“猜词游戏”来学习词向量

Skip-Gram

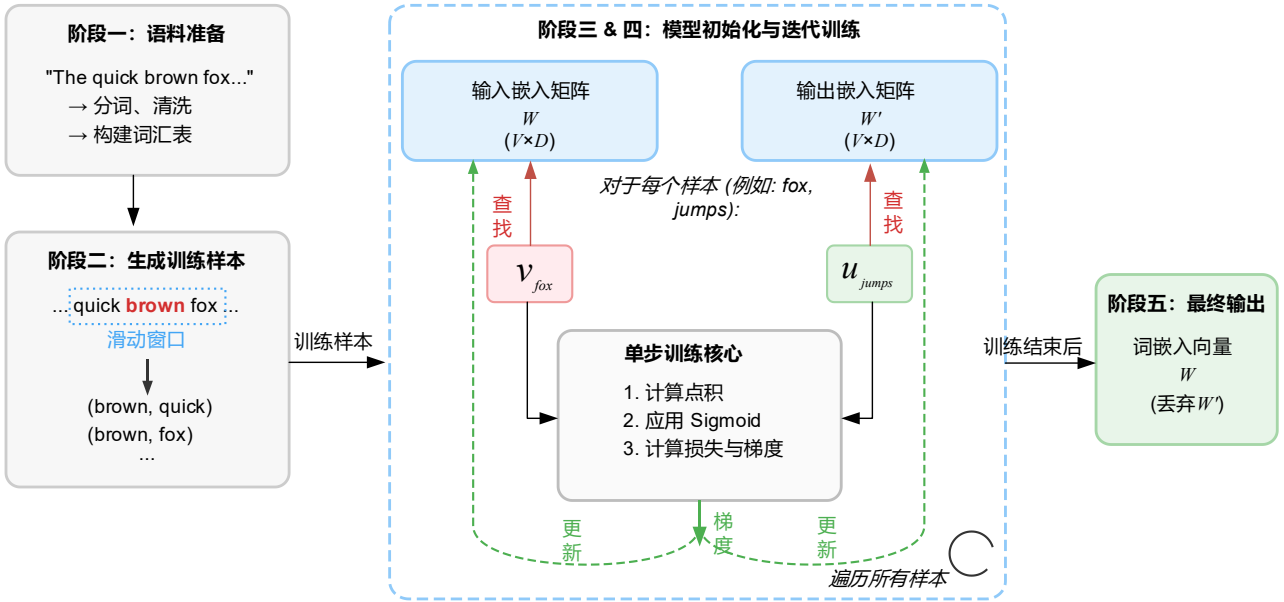
方法一：Word2vec

- 由Google于2013年提出，开创了预测式词嵌入的范式。它将“分布式假设”转化为一个具体的预测任务

- 两种经典架构：
  - Skip-Gram (跳元模型)
    - 任务： 根据中心词，预测其上下文词
    - 直觉： "绿色" -> [..., "草地", "是", "的", ...]
  - Continuous Bag-of-Words (CBOW, 连续词袋模型)
    - 任务： 根据上下文词，预测中心词。
    - 直觉： [..., "草地", "是", "的", ...] -> "绿色"

Word2vec 深入解析 (1): Skip-Gram 工作机制

- 给定中心词 w_c，最大化其上下文窗口内所有真实上下文词的联合概率
  - 将一个复杂的语言建模问题简化为一个高效的、可扩展的词语对预测任务
- 为词汇表每个词 w 学习两个向量表达v_w (中心词)u_w (上下文词)，调整v 和 u，使得整个语料库的似然函数最大化



Skip-Gram 模型：宏观工作流程

Skip-Gram 宏观工作流程

- 整个过程可以分解为五个主要阶段：从原始文本到最终可用的词向量
- 以一个具体的例子贯穿始终，假设我们的输入句子是 "The quick brown fox jumps over the lazy dog“
- 设定的窗口大小为5（即中心词左右各2个词）

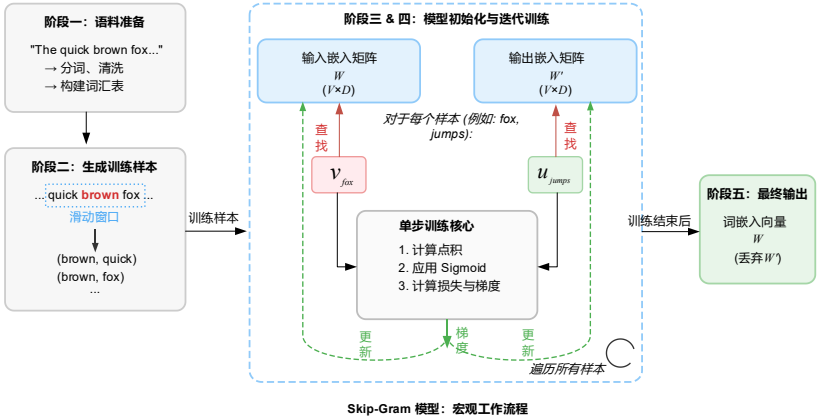
阶段一：语料准备与词汇表构建

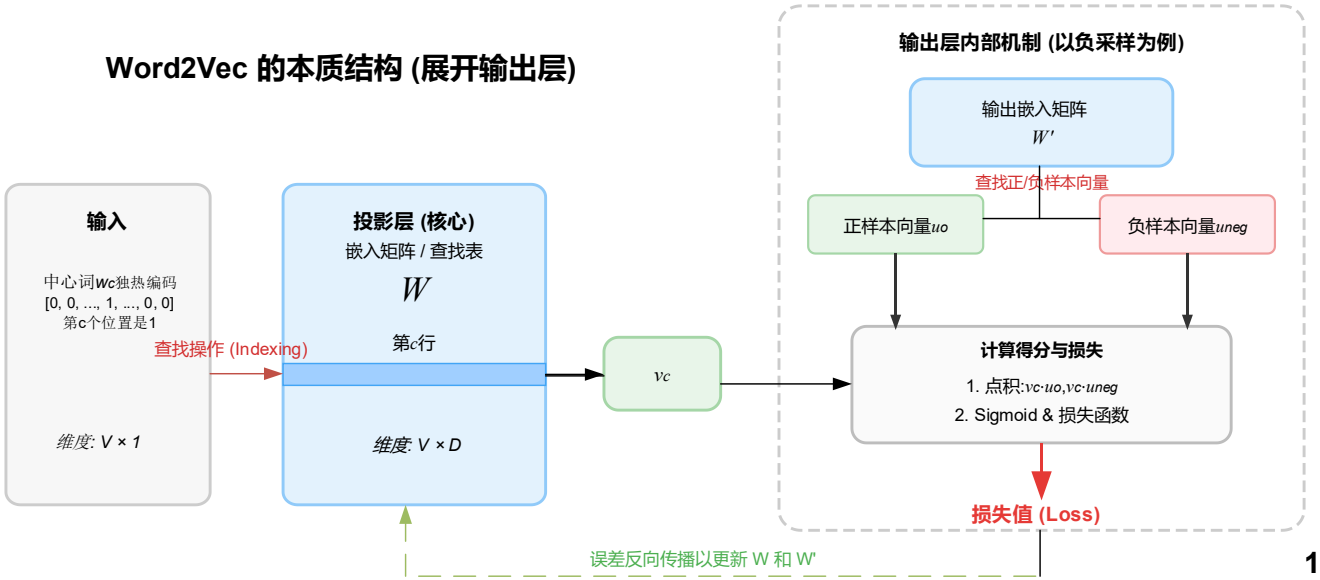
- 将非结构化的原始文本转换为机器可以处理的结构化数据
- 动作
  - 分词 (Tokenization)： 将文本分割成一个个独立的词语（tokens）。例如，"The quick brown fox..." -> ['The', 'quick', 'brown', 'fox', ...]
  - 文本清洗 (Text Cleaning)： 将所有词语转为小写，去除标点符号等。['The', 'quick', ...] -> ['the', 'quick', ...]
  - 构建词汇表 (Build Vocabulary)： 统计所有不重复的词语，并根据词频进行筛选（如，丢弃出现次数低于某个阈值的低频词）。为词汇表中的每个唯一词语分配一个唯一的整数ID
- 输出
  - 一个词汇表（Vocabulary），即一个从词语到整数ID的映射。
    - 如：{'the': 0, 'quick': 1, 'brown': 2, 'fox': 3, ...}
  - 一个将整个语料库表示为ID序列的列表。例如：[0, 1, 2, 3, 4, 5, 0, 6, 7]

阶段二：生成训练样本

- 根据分布假说，从语料库中创建大量的 (中心词, 上下文词) 训练对
- 动作
  - 滑动窗口 (Sliding Window)： 在ID序列上移动一个固定大小的窗口
  - 提取词对： 在每个窗口位置，将窗口中心的词作为“中心词”，将窗口内所有其他词作为“上下文词”，并生成相应的词对
- 输出： 一个巨大的训练样本集
  - 示例： 对于序列 ... quick brown fox jumps over ... (ID: ... 1, 2, 3, 4, 5 ...)
    - 当窗口中心是 brown (ID: 2) 时，上下文是 the, quick, fox, jumps。生成的样本对为：(brown, the), (brown, quick), (brown, fox), (brown, jumps)
    - 当窗口中心是 fox (ID: 3) 时，上下文是 quick, brown, jumps, over。生成的样本对为：(fox, quick), (fox, brown), (fox, jumps), (fox, over)
- 此过程遍历整个语料库，产生数百万甚至数十亿的训练样本

Word2Vec


阶段一：语料准备
"The quick brown fox..."
→ 分词、清洗
→ 构建词汇表
阶段二：生成训练样本
...quick brown fox...
滑动窗口
(brown, quick)
(brown, fox)
...
训练样本
阶段三 & 四：模型初始化与迭代训练
输入嵌入矩阵
W
(V×D)
输出嵌入矩阵
W'
(V×D)
对于每个样本 (例如: fox, jumps):
查找
查找
v fox
u jumps
单步训练核心
1. 计算点积
2. 应用 Sigmoid
3. 计算损失与梯度
更新
梯度
更新
遍历所有样本
训练结束后
阶段五：最终输出
词嵌入向量
W
(丢弃W')
Skip-Gram 模型：宏观工作流程


Word2Vec 的本质结构 (展开输出层)
输入
中心词wc独热编码
[0, 0, ..., 1, ..., 0, 0]
第c个位置是1
维度: V × 1
查找操作 (Indexing)
投影层 (核心)
嵌入矩阵 / 查找表
W
第c行
维度: V × D
vc
输出层内部机制 (以负采样为例)
输出嵌入矩阵
W'
查找正/负样本向量
正样本向量uo
负样本向量uneg
计算得分与损失
1. 点积:vc·uo,vc·uneg
2. Sigmoid & 损失函数
损失值 (Loss)
误差反向传播以更新 W 和 W'

阶段三：模型初始化

- 准备好神经网络的参数，即我们最终要学习的词向量
- 动作
  - 创建两个权重矩阵
    - 输入嵌入矩阵 W (维度 $V \times D$)：每一行代表一个词作为“中心词”时的初始向量
    - 输出嵌入矩阵 W' (维度 $V \times D$)：每一行代表一个词作为“上下文词”时的初始向量
  - 使用小的随机数对这两个矩阵进行初始化
- 输出
  - 两个随机初始化的嵌入矩阵。这些矩阵在训练开始时没有任何语义信息

阶段四：迭代训练

- 遍历所有训练样本对，调整两嵌入矩阵向量，使得频繁共现词对在向量空间中更接近
- 单步动作（处理一个正样本对，如 (fox, jumps)）
  - a. 获取正样本向量： 从矩阵 W 中查找中心词 fox 的输入向量 v_{fox}；从矩阵 W' 中查找上下文词 jumps 的输出向量 u_{jumps}
  - b. 生成负样本： 从词汇表中随机抽取 k 个词作为负样本（例如，apple, car）
  - c. 获取负样本向量： 从矩阵 W' 中查找负样本词 apple 和 car 的输出向量 u_{apple} 和 u_{car}
  - d. 前向传播计算得分：
    - 计算正样本得分：$score_{pos} = v_{fox} \cdot u_{jumps}$；计算负样本得分：$score_{neg1} = v_{fox} \cdot u_{apple}, score_{neg2} = v_{fox} \cdot u_{car}$
  - e. 计算损失： 将所有得分通过Sigmoid函数σ，并根据负采样损失函数计算总损失
    - 目标让 $\sigma(score_{pos})$ 趋近于1，让所有 $\sigma(score_{neg})$ 趋近于0
  - f. 反向传播与更新： 计算损失函数关于所涉及向量（$v_{fox}, u_{jumps}, u_{apple}, u_{car}$）的梯度，并使用梯度下降法对这些向量进行微小的更新。每一步只更新这 $k+2$ 个词对应的向量

优化目标 - 损失函数的定义

- 正样本的优化目标
  - 最大化 $\sigma(v_c^T u_o)$ 。在对数空间里，最大化$\log \sigma(v_c^T u_o)$
- 负样本的优化目标
  - 最大化 " 不是上下文 " 的概率 。在对数空间里，最大化 $\log\left(1-\sigma\left(v_c^T u_{neg}\right)\right)=\log \sigma\left(-v_c^T u_{neg}\right)$
- 组合成完整的目标函数
  - 对于一个正样本和 k 个负样本，最大化所有这些事件的联合对数似然

$$\text{Objective } = \log \sigma(v_c^T u_o) + \sum_{i=1}^{k} \log \sigma\left(-v_c^T u_{\text{neg},i}\right)$$

- 转化为损失函数
  - 损失函数（Loss Function）

$$L = -\left(\log \sigma(v_c^T u_o) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)}[\log \sigma(-v_c^T u_i)]\right)$$

  - 其中，v_c 是中心词的向量，u_o 是真实上下文词（正样本）的向量，u_i 是一个负样本词的向量，$\mathbb{E}_{w_i \sim P_n(w)}[\ldots]$ 表示对从噪声分布 $P_n(w)$ 中采样的 k 个负样本求期望（实现时，求和取平均）

阶段五：提取并使用词向量

- 获取训练完成的、蕴含语义信息的最终词向量
- 动作：
  - 训练结束后，丢弃输出嵌入矩阵 W'
  - 保留输入嵌入矩阵 W 作为最终的词向量查找表
- 产出
  - 一个维度为 $V \times D$ 的矩阵，其中第 i 行就是词汇表中第 i 个词的最终词嵌入。这个向量可以用于各种下游NLP任务，如计算词语相似度、文本分类、情感分析等

GloVe: Global Vectors for Word Representation

学习词向量的两大流派

- 矩阵分解法 (Matrix Factorization)
  - LSA (Latent Semantic Analysis)
  - 构建词-文档/词-词共现矩阵，通过SVD等方法降维
  - 能利用全局统计信息
  - 在词语类比任务上表现不佳，难以捕捉精细的线性语义
- 局部上下文预测法 (Local Context Prediction)
  - Word2Vec (Skip-Gram, CBOW)
  - 基于滑动窗口，通过局部上下文进行预测
  - 在词语类比任务上表现优异，能学习到向量的线性结构
  - 对全局统计信息的利用是间接和低效的
- 能否设计一个模型，既能直接利用全局统计信息，又能学习到有意义的线性子结构？

核心洞察：意义蕴含在共现概率的“比率”中

- 思考 ice 和 steam 的关系。仅看单个共现概率意义不大，但它们的比率揭示了深刻的语义关系

探针词 (k)	P(k | ice)	P(k | steam)	比率: P(k | ice) / P(k | steam)	结论
solid	高	低	>> 1	与 ice 强相关 与 steam 弱相关
gas	低	高	<< 1	与 ice 弱相关 与 steam 强相关
water	高	高	≈ 1	与两者都相关
fashion	低	低	≈ 1	与两者都无关

- 结论： 概率的比率，而非概率本身，才是区分词语、编码意义的关键。这个比率就像一个“语义指纹”
- GloVe的目标： 学习词向量，使其向量运算能够直接模拟这个概率比率

从洞察到模型 (1):建立数学框架

- 第一步：将“比率”与向量联系起来
  - 我们的目标是找到一个函数 F，它能将词向量映射到共现概率的比率上：

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

    - w_i, w_j: 中心词 i, j 的向量；$\tilde{w}_k$: 上下文词 k 的向量；$P_{ik} = X_{ik}/X_i$: 词 k 出现在词 i 上下文中的概率
- 为了捕捉线性结构，施加约束：
  - 输入为向量差：我们关心的是词 i 和 j 的关系，所以输入应为 $w_i - w_j$。

$$F(w_i - w_j, \tilde{w}_k)$$

  - 使用点积：将向量空间映射到标量空间最直接的方式是点积

$$F((w_i - w_j)^T \tilde{w}_k)$$

  - 利用同态性质：左边是向量的差，右边是概率的商。为了匹配运算，我们希望 F 是一个同态映射。具体来说，我们希望 F 能将向量空间的加法（或减法）映射到实数空间的乘法（或除法）
  - 指数函数 $F = \exp$ 是一个完美的选择

$$\exp((w_i - w_j)^T \tilde{w}_k) = \frac{\exp(w_i^T \tilde{w}_k)}{\exp(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$$

从洞察到模型 (2):推导最终形式

- 第二步：简化得到最终模型
  - 从 $\exp(w_i^T \tilde{w}_k) = P_{ik} = X_{ik}/X_i$ 出发，两边取对数

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$

    - 这个形式还不够好，因为 $\log(X_i)$ 这一项只与 i 有关，与 k 无关，它破坏了模型的对称性
  - 关键步骤：引入偏置项 (Bias)
    - 注意到 $\log(X_i)$ 可以被吸收到一个只与 i 相关的偏置项 b_i 中。同时，为了保持 $w \leftrightarrow \tilde{w}$ 的对称性，我们再为 $\tilde{w}_k$ 增加一个偏置项 $\tilde{b}_k$
    - 最终，我们得到了GloVe的核心模型方程

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

  - 解读：
    - 这是一个极其简洁的对数-双线性模型
    - 它不再预测概率，而是直接拟合共现次数的对数
    - 它将全局统计量 $\log(X_{ik})$ 与由模型参数构成的线性结构直接联系起来

GloVe的目标函数

- 目标函数：加权最小二乘
  - 我们的目标是让模型预测值 $w_i^T \tilde{w}_j + b_i + \tilde{b}_j$ 尽可能接近真实值 $\log(X_{ij})$。最自然的方法是使用最小二乘法。
  - 但简单的最小二乘会给所有词对同等的权重，这有两个问题：
    - 停用词问题：像 (the, is) 这样的共现次数极高，会在损失中占据主导地位。
    - 零共现问题：绝大多数词对的共现次数为0，$\log(0)$ 无定义。
- 解决方案：加权最小二乘 (Weighted Least Squares)
  - $J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}) \right)^2$
    - $f(X_{ij})$ 是一个权重函数。
    - 求和只在 $X_{ij} > 0$ 的词对上进行，自然地解决了 $\log(0)$ 的问题。

关键的权重函数 $f(x)$

- 权重函数 $f(x)$ 的设计哲学
- 目标：
  - 对于罕见的共现，权重不能为0，它们也包含信息。
  - 对于频繁的共现（停用词），权重不能过高，需要设置上限。
  - 函数应平滑、非递减。
- GloVe的权重函数设计：

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

  - $x_{\max}$ 是一个阈值（论文中推荐100）
  - α 是一个超参数（论文中推荐0.75）

GloVe 完整算法流程

- 构建共现矩阵 X
  - 确定窗口大小
  - 遍历整个语料库，统计所有词对 (i, j) 的共现次数 X_{ij}。可以采用距离加权
- 初始化参数
  - 随机初始化中心词向量矩阵 W $(V \times D)$ 和上下文词向量矩阵 $\widetilde{W}$ $(V \times D)$
  - 随机初始化偏置项 b 和 $\tilde{b}$ (均为 $V \times 1$ 向量)
- 迭代训练
  - 对于多轮迭代 (Epochs):
    - 打乱共现矩阵中的所有非零项
    - 对于每一个 (i, j) 对，使用随机梯度下降法（如AdaGrad）来最小化损失项 $f(X_{ij})(\ldots)^2$
- 得到最终词向量
  - 训练结束后，最终的词向量是中心词向量和上下文词向量的和：

$$W_{final} = W + \widetilde{W}$$

Word2vec vs. GloVe

- 哪个更好？
  - 两者在下游任务上表现都非常出色，各有千秋。
  - GloVe的训练更快，对中等大小的语料库非常有效。
  - Word2Vec的在线学习特性使其对超大规模、流式数据的处理更具灵活性

特性	GloVe	Word2Vec (Skip-Gram)
模型类型	计数模型 (Count-based)	预测模型 (Predictive)
核心思想	显式地拟合全局共现次数的对数	隐式地学习统计规律，通过局部上下文预测
数据利用	一次性构建全局共现矩阵	在线学习，一次一个上下文窗口
训练速度	训练通常更快，因为收敛所需迭代次数少	对于巨大语料库，无需构建大矩阵，可在线训练
理论联系	后续研究表明，Word2Vec也等价于分解一个隐式的共现矩阵	

结论与历史地位

- 结论与要点回顾
  - GloVe成功地将矩阵分解法（全局统计）和预测法（线性结构）的优点结合在了一起
  - 其理论基石是“意义蕴含在共现概率的比率中”
  - 通过一系列优雅的推导，将复杂的比率问题转化为一个简单的对数-双线性拟合问题
  - 有效的加权方案：通过 $f(x)$ 函数，巧妙地平衡了高频词和低频词的影响
- 历史地位
  - GloVe与Word2Vec共同定义了静态词嵌入技术的黄金时代
  - 它们是后续更复杂的上下文相关模型（如ELMo, BERT）出现之前，NLP领域最重要、最广泛使用的基础技术之一
  - 理解GloVe对于理解表示学习的演进至关重要

fastText

子词模型：fastText

- Word2vec/GloVe 的共同缺陷：将词视为原子单位
- 这个假设带来了两个严重问题
  - 未登录词 (Out-of-Vocabulary, OOV)
    - 模型无法为训练语料中未出现过的词生成向量
  - 忽略形态学 (Morphology)
    - help, helpful, helpless, unhelpful 被视为四个完全独立的词。
    - 模型无法利用它们共享的词根 help，导致学习效率低下。
    - 对于形态丰富的语言（德语、芬兰语等）尤其严重。
- fastText 的核心目标：通过利用词的内部结构（子词信息）来解决这些问题

fastText 的核心思想

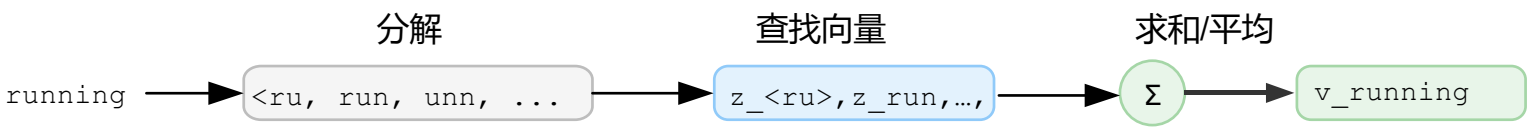
- 一个词的向量是其所有子词（character n-grams）向量的和
  - fastText 不再将词视为原子，而是将其表示为字符 n-gram (character n-grams) 的集合
- 示例：词 where (假设 n=3 到 6)
  - 添加边界符号: <where>
  - 提取 n-grams:
    - 3-grams: <wh, whe, her, ere, re>
    - 4-grams: <whe, wher, here, ere>
    - 5-grams: <wher, where, here>
    - 6-grams: <where, where>
  - 包含原词: 最后，加入特殊子词 <where> 本身。
  - 词 where 的最终表示 = { <wh, whe, her, ..., <where> }

fastText模型架构：对 Skip-Gram 的精巧扩展

- 在 Skip-Gram 框架内重新定义输入
  - FastText 的整体架构与 Skip-Gram 完全相同（预测上下文），核心创新在于如何构建中心词的输入向量。

Word2Vec（Skip－Gram）	FastText
1．输入中心词 c	1．输入中心词 c
2．在词嵌入矩阵W 中查找	2．将 c 分解为子词集合 G_c
3．得到向量 $v_c = W[c]$	3．在子词嵌入矩阵 z 中查找每个子词 $g \in G_c$ 的向量 z_g
	4．将所有子词向量求和 / 平均，得到最终向量 $v_c = \sum_{g \in G_c} z_g$

FastText 输入向量构建流程



FastText 的两大优势

- **优势 1: 优雅地解决 OOV 问题**
  - 对于一个未登录词，如 aquaphobia：
    - Word2Vec: 返回 <UNK> 向量，信息为零
    - FastText:
      - 将其分解为子词：{ <aq, aqu, qua, ..., bia>, <aquaphobia> }
      - 即使从未见过 <aquaphobia> 这个整体，但很可能在训练集中见过 aqua (与水相关) 和 phobia (与恐惧相关) 的子词
      - 通过组合这些已知子词的向量，FastText 可以**合成**一个非常有意义的、近似的向量来表示 aquaphobia
- **优势 2: 高效捕捉形态学信息**
  - helpful 和 unhelpful 共享子词 help 和 ful
  - 在训练过程中，对这两个词的更新都会贡献于 help 和 ful 的向量学习
  - 这使得模型能够自动学习到前缀 un- 具有“否定”的语义功能
  - 对于罕见词（如 braggadocio），即使它本身出现次数很少，但其子词（如 brag）可能很常见，从而帮助其学习到一个更高质量的向量

结论

- 要点回顾
  - 核心创新: 通过引入子词信息 (character n-grams)，打破了词的原子性假设
  - 架构: 本质上是 Word2Vec (Skip-Gram/CBOW) 的一个增强版，而非全新架构
  - 主要优势:
    - 能为任何词（包括 OOV）生成向量
    - 能高效学习和利用形态学信息
  - 代价: 子词词典巨大，模型体积和内存占用远大于 Word2Vec
- 在词嵌入技术谱系中
  - 是对 Word2Vec 的一个直接、向后兼容的功能扩展
  - 与 BPE/WordPiece 的对比
    - FastText: 使用基于规则的、穷举式的子词切分
    - BPE/WordPiece: 使用基于数据驱动的、更智能的子词切分，是现代 Transformer 模型（BERT, GPT）的标配
- 完美解决了词的内部结构问题，是从“原子词”到“上下文词”演进中的关键一步

总结

- 嵌入向量（Embeddings）
- 词嵌入（Word2vec）
  - Skip-Gram
  - CBOW
- 子词嵌入（fastText）
- 全局向量的词嵌入（GloVe）