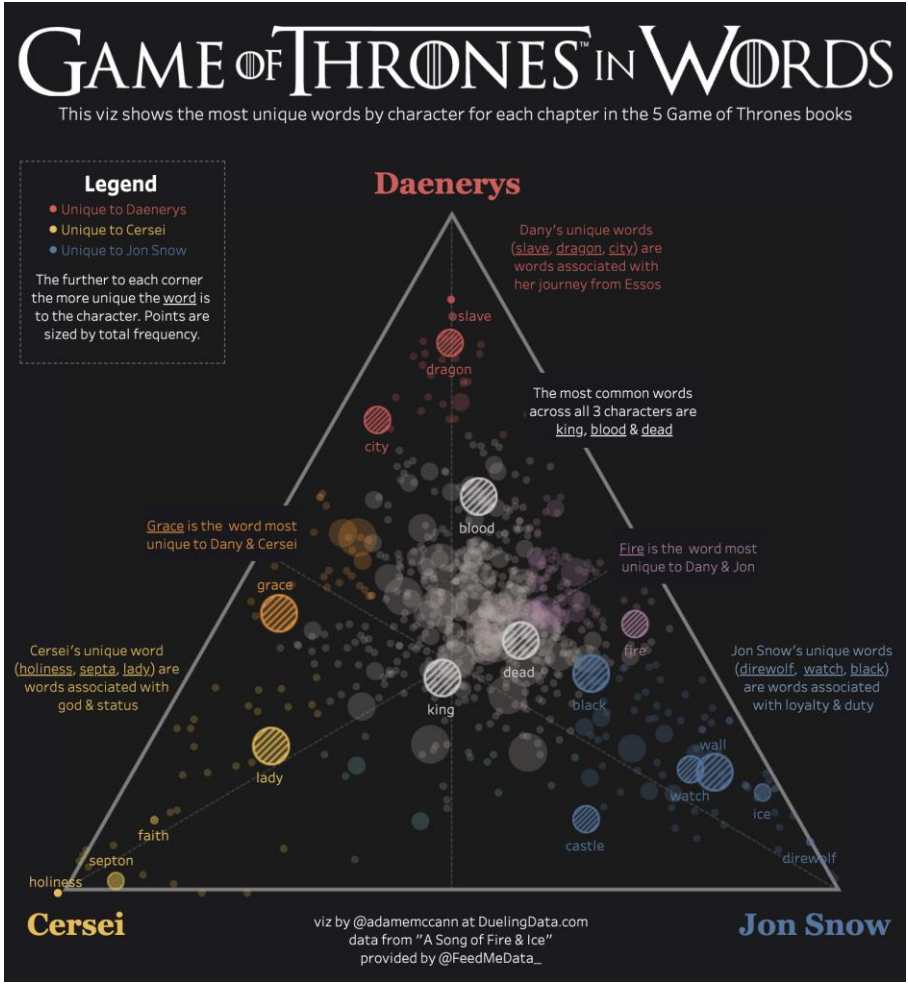
22 嵌入向量, 词嵌入, 子词嵌入, 全局向量的词嵌入

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）

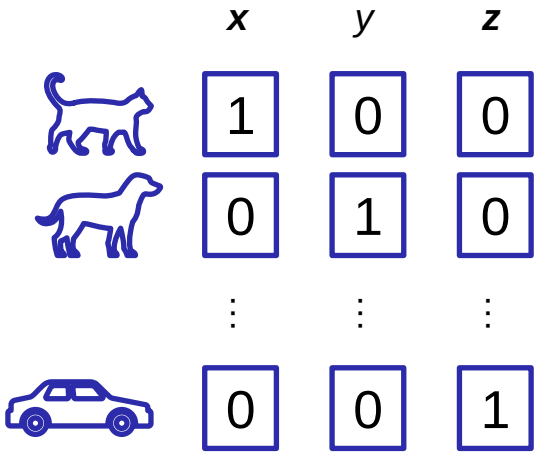
词嵌入（Word2vec）



动机

- 单热向量法将目标对象/单词映射到固定长度向量
- 这些向量仅包含身份信息，而不包含语义含义，如：

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{z}, \mathbf{y} \rangle = 0$$

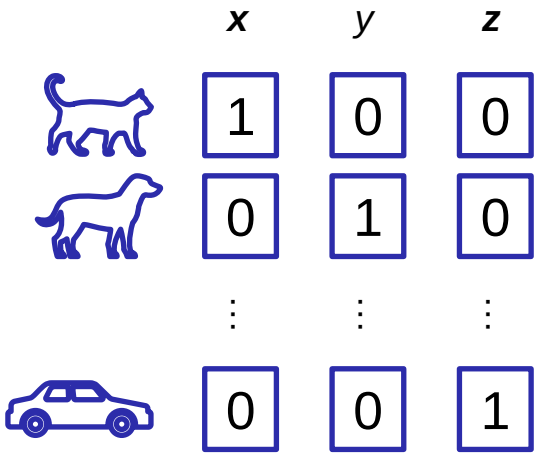


词嵌入（word2vec）

- 学习每个单词的嵌入向量
  - google模型：80-300维
- 需要定义相似度， 衡量相似性，如

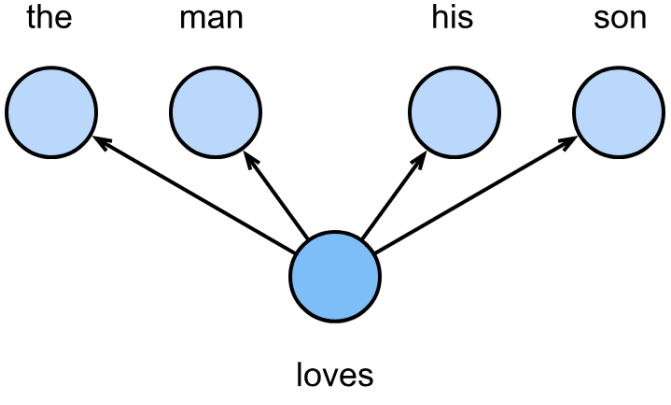
$$\langle \mathbf{x}, \mathbf{y} \rangle > \langle \mathbf{z}, \mathbf{y} \rangle$$

- 建立概率模型
- 最大化似然函数优化



Skip-Gram 模型

- 一个单词可用于生成它周围的单词
- 给定中心词，每个上下文词是独立生成的



$$\mathbb{P}(\text{"the"}, \text{"man"}, \text{"his"}, \text{"son"} \mid \text{"loves"})$$

$$= \mathbb{P}(\text{"the"} \mid \text{"loves"}) \cdot \mathbb{P}(\text{"man"} \mid \text{"loves"}) \cdot \mathbb{P}(\text{"his"} \mid \text{"loves"}) \cdot \mathbb{P}(\text{"son"} \mid \text{"loves"})$$

似然函数

全部概率求和非常昂贵

	词	嵌入
中心词	w_c	$\mathbf{v}_c \in \mathbb{R}^d$
上下文	w_o	$\mathbf{u}_o \in \mathbb{R}^d$

$$\mathbb{P}(w_o \mid w_c) = \frac{exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

$\mathcal{V}$: 所有上下文

- 给定长度为T的序列，上下文窗口长度为 m，似然函数：

$$\prod_{t=1}^{T} \prod_{-m \le j \le m, j \ne 0} \mathbb{P}(w^{(t+j)} \mid w^{(t)})$$

Skip-Gram 模型训练

- 所有上下文词的概率:

$$\prod_{t=1}^{T} \prod_{-m \le j \le m, j \ne 0} P\left(w^{(t+j)} \mid w^{(t)}\right),$$

- 相当于最小化以下损失函数

$$-\sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log P\left(w^{(t+j)} \mid w^{(t)}\right)$$

- 涉及中心词 w_c 和上下文词 w_o 的对数条件概率为

$$\log P(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log\left(\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^\top \mathbf{v}_c\right)\right)$$

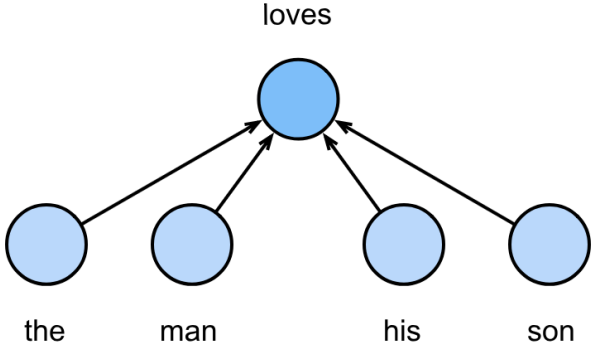
- 相对于中心词向量 $\mathbf{v}_c$ 的梯度为

$$\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} = \mathbf{u}_o - \sum_{j \in \mathcal{V}} P\left(w_j \mid w_c\right)\mathbf{u}_j$$

- 时间复杂度和$\mathcal{V}$的大小有关

CBOW 模型

- CBOW - Continuous Bag Of Words
- 基于上下文词生成中心词



$$\mathbb{P}(\textit{"loves"} \mid \textit{"the"}, \textit{"man"}, \textit{"his"}, \textit{"son"})$$

似然函数

- 计算概率

$$\mathbb{P}(w_c \mid w_{o_1}, \ldots, w_{o_{2m}}) = \frac{\exp\left(\frac{1}{2m}\mathbf{u}_c^\top(\mathbf{v}_{o_1} + \cdots + \mathbf{v}_{o_{2m}})\right)}{\sum_{i \in \mathcal{V}} \exp\left(\frac{1}{2m}\mathbf{u}_i^\top(\mathbf{v}_{o_1} + \cdots + \mathbf{v}_{o_{2m}})\right)}$$

- 可能性

$$\prod_{t=1}^{T} \mathbb{P}(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

近似训练

- 跳元模型的主要思想是使用softmax运算来计算基于给定的中心词 w_c 生成上下文字 w_o 的条件概率
- 跳元模型的梯度计算和 (14.1.15)中的连续词袋模型的梯度计算都包含求和。在一个词典$\mathcal{V}$上（通常有几十万或数百万个单词）求和的梯度的计算成本是巨大的

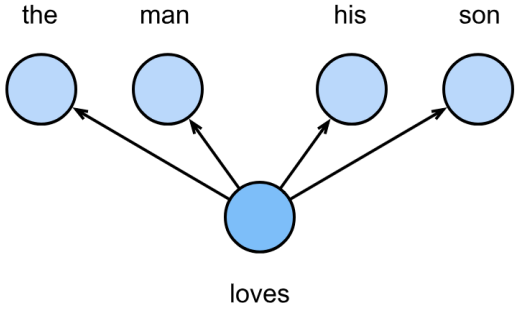
$$\mathbb{P}(w_o \mid w_c) = \frac{exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

- 以跳元模型为例来描述这两种近似训练方法

负采样

- 给定中心词 w_c 的上下文窗口，任意上下文词 w_o 来自该上下文窗口的被认为是由下式建模概率的事件【将中心词和上下文词同时出现在相同窗口作为一个“事件”】

$$P(D = 1 \mid w_c, w_o) = \sigma(\mathbf{u}_o^\top \mathbf{v}_c)$$



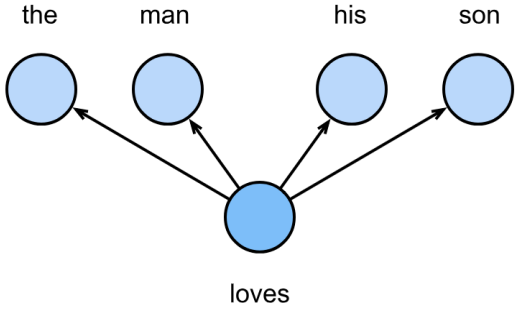
  - 其中σ为sigmoid激活函数
- 等价于转换了模型
  - 第一个输入不变
  - 原模型的预测标签输出作为第二个输入

负采样

- 新输出是原模型的softmax输出：即评价是否合理的数值

- 给定长度为 T 的文本序列, 以 $w^{(t)}$ 表示时间步 t 的词，并使上下文窗口为 m ，考虑最大化联合概率：

$$\prod_{t=1}^{T}\prod_{-m\leq j\leq m, j\neq 0} P\left(D=1 \mid w^{(t)}, w^{(}\right.$$



负采样

- 联合概率$\prod_{t=1}^{T}\prod_{-m\le j\le m,j\ne 0}P\left(D=1\mid w^{(t)},w^{(t+j)}\right)$只考虑那些正样本的事件
- 为了使目标函数更有意义，负采样添加从预定义分布中采样的负样本

负采样

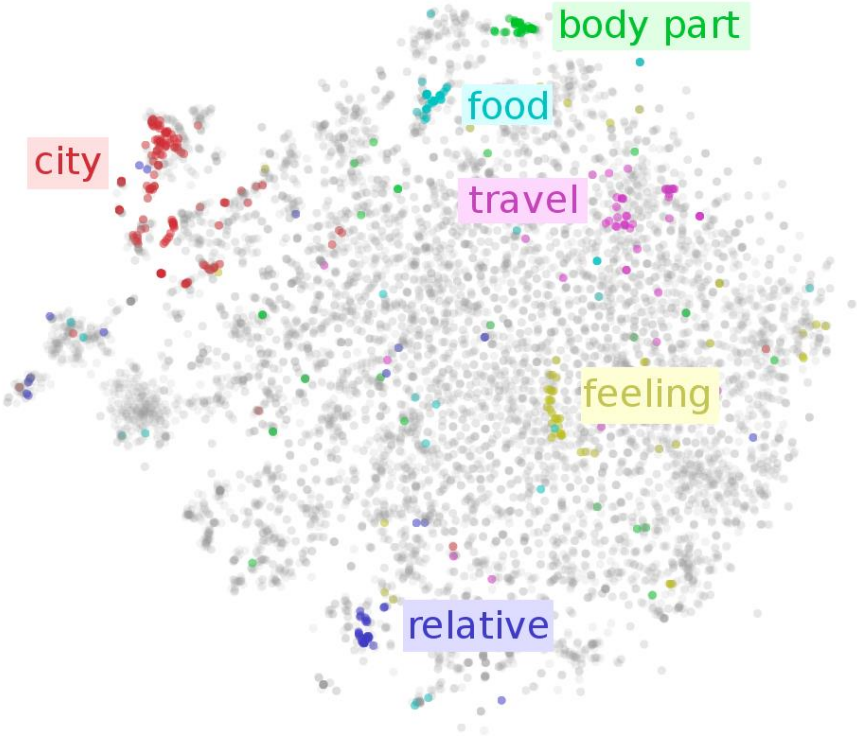
- 用 S 表示上下文词 w_o 来自中心词 w_c 的上下文窗口的事件。对于这个涉及 w_o 的事件，从预定义分布 $P(w)$ 中采样K个不是来自这个上下文窗口噪声词。用 N_k 表示噪声词 $w_k(k=1,\ldots,K)$ 不是来自 w_c 的上下文窗口的事件。假设正例和负例 $S, N_1, \ldots, N_K$ 的这些事件是相互独立的。通过事件 $S, N_1, \ldots, N_K$ 近似条件概率:

$$P\left(w^{(t+j)} \mid w^{(t)}\right) = P\left(D=1 \mid w^{(t)}, w^{(t+j)}\right) \prod_{k=1, w_k \sim P(w)}^{K} P\left(D=0 \mid w^{(t)}, w_k\right)$$

- 每个训练步的梯度计算成本与词表大小无关，而是线性依赖于**K**
  - 当将超参数K设置为较小的值时，在负采样的每个训练步处的梯度的计算成本较小

代码 ...

更多嵌入模型



子词嵌入（fastText）

- 英语单词通常有内部结构和形成方法
  - dog，dogs，dogcatcher
- 每个中心词可以表示为一组子词
  - where -> <where>
  - n-gram (n = 3)：“<wh”，“whe”，“her”，“ere”，“re>”
- 适用于冗长但不常见的单词
  - 例如：pneumonoultramicroscopicsilicovolcanoconiosis（火山肺矽病）

子词嵌入（fastText）

- 对于单词 w，$\mathcal{G}_w$ 是长度为 3-6 的词的并集
- 中心向量

$$\mathbf{u}_w = \sum_{g \in \mathcal{G}_w} \mathbf{u}_g$$

- 其余模型与 skip-gram 相同

全局向量的词嵌入（GloVe）

- 表示: $q_{ij} = \frac{\exp(\mathbf{u}_j^\top \mathbf{v}_i)}{\sum_{k \in \mathcal{V}} \exp(\mathbf{u}_k^\top \mathbf{v}_i)}$

- 重写 skip-gram 的负对数似然函数

$$-\log \left[\prod_{t=1}^{T} \prod_{-m \le j \le m, j \ne 0} \mathbb{P}(w^{(t+j)} \mid w^{(t)}) \right]$$

- 相当于 $-\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} x_{ij} \log q_{ij}$ （足量样本 x_{ij}）

全局向量的词嵌入（GloVe）

- 其它表示：

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} x_{ij}\log q_{ij} = \sum_{i\in\mathcal{V}} x_i \sum_{j\in\mathcal{V}} p_{ij}\log q_{ij}$$

交叉熵

$$x_i = \sum_j x_{ij}$$

$$p_{ij} = \frac{x_{ij}}{x_i}$$

全局向量的词嵌入（GloVe）

- 用（易于计算的）对数平方损失（log square loss）替换交叉熵损失（cross entropy loss）

$$\sum_{j\in\mathcal{V}} p_{ij}\log q_{ij} \rightarrow \sum_{j\in\mathcal{V}} (\log p_{ij} - \log q'_{ij})^2$$

- 为中心词和上下文词添加偏见词 $q'_{ij} = \exp(\mathbf{u}_j^\top \mathbf{v}_i)$
- 用 [0,1] 中的单调递增函数替换权重 x_i

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} h(x_{ij})\left(\mathbf{u}_j^\top \mathbf{v}_i + b_i + c_j - \log x_{ij}\right)^2$$

代码...

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）