



机器学习班第**15**课

# 词嵌入原理

李韶华

第一期 2016年8-10月



# 提纲

- 自然语言处理的基本概念
- 直观上认识和理解词嵌入
- Bengio的NNLM; Mikolov的word2vec
- 矩阵分解做词嵌入: GloVe算法
- 影响性能的因素; 怎么调参
- “多词义”词嵌入
- 应用: CNN做句子分类



# NLP常见任务

- 自然语言处理: Natural Language Processing (NLP)
- 词性标注 heat (v.) water (n.) in (p.) a (det.) pot (n.)
- 分词 大水沟/很/难/过
- 文本分类/聚类
- 自动摘要
- 阅读理解, 指代消解 小明放学了, 妈妈去接他
- 机器翻译 小心地滑 → Slide carefully
- 问答系统: 问题和答案自动匹配
- 人机对话 (Siri, 小冰.....)
- .....



# NLP基本方法

- 传统: 基于规则
- 现代: 基于统计机器学习
  - 给定一个很大的训练语料(许多文章的集合)
  - HMM, CRF, SVM, LDA, CNN, LSTM...
  - 规则隐含在模型参数里, 自动从语料学到
  - 越来越像机器学习在文本处理方面的应用
- 未来: 统计机器学习+知识库+规则
  - 知识图谱

# NLP vs. 图像处理

- 文本输入: 一串词
  - 特点: 完全离散, 词与词表面上看不到相似性, 看不到语义
  - 任务: 理解、挖掘深层语义.
  - 需利用词的长程相关性
- 图像输入: 一个像素矩阵
  - 特点: 像素在空间上基本连续, 颜色上渐变, 天然的相似性定义, “所见即所得”
  - 任务: 直观、浅层理解

一只/猫/趴/在/键盘/上,  
键盘/在/卧室/里。  
问: 猫在哪个房间?



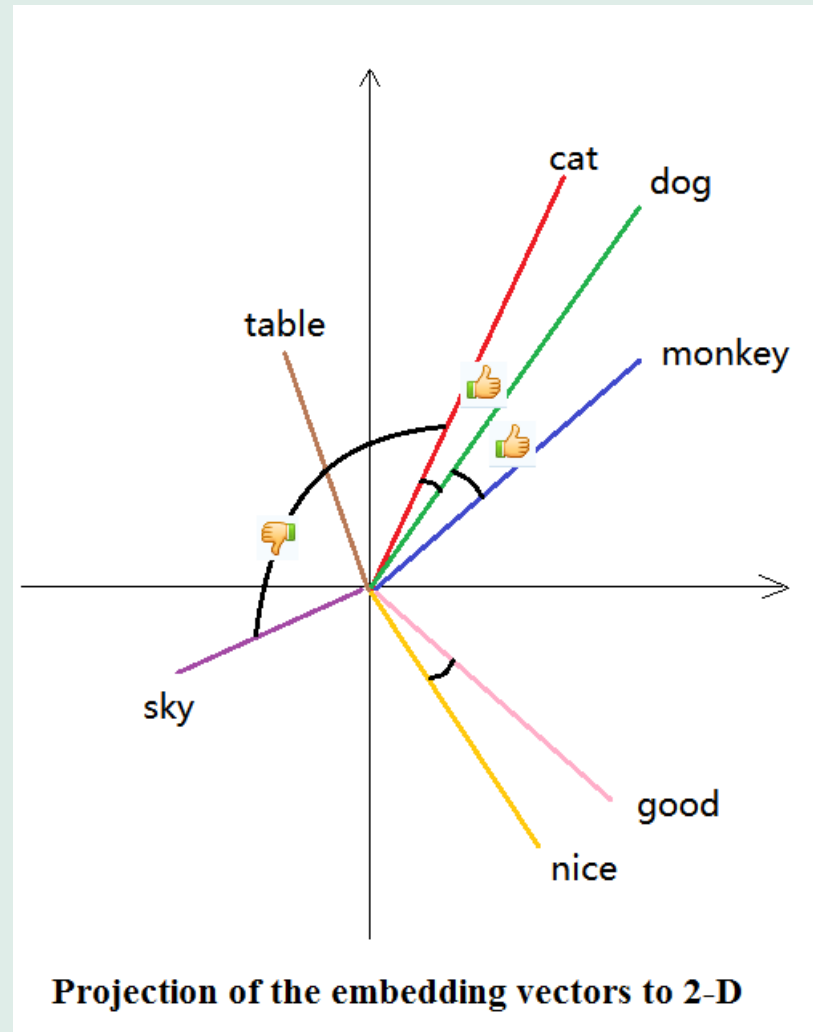
问: 键盘上什么动物?

# NLP系统的输入—One-hot 编码

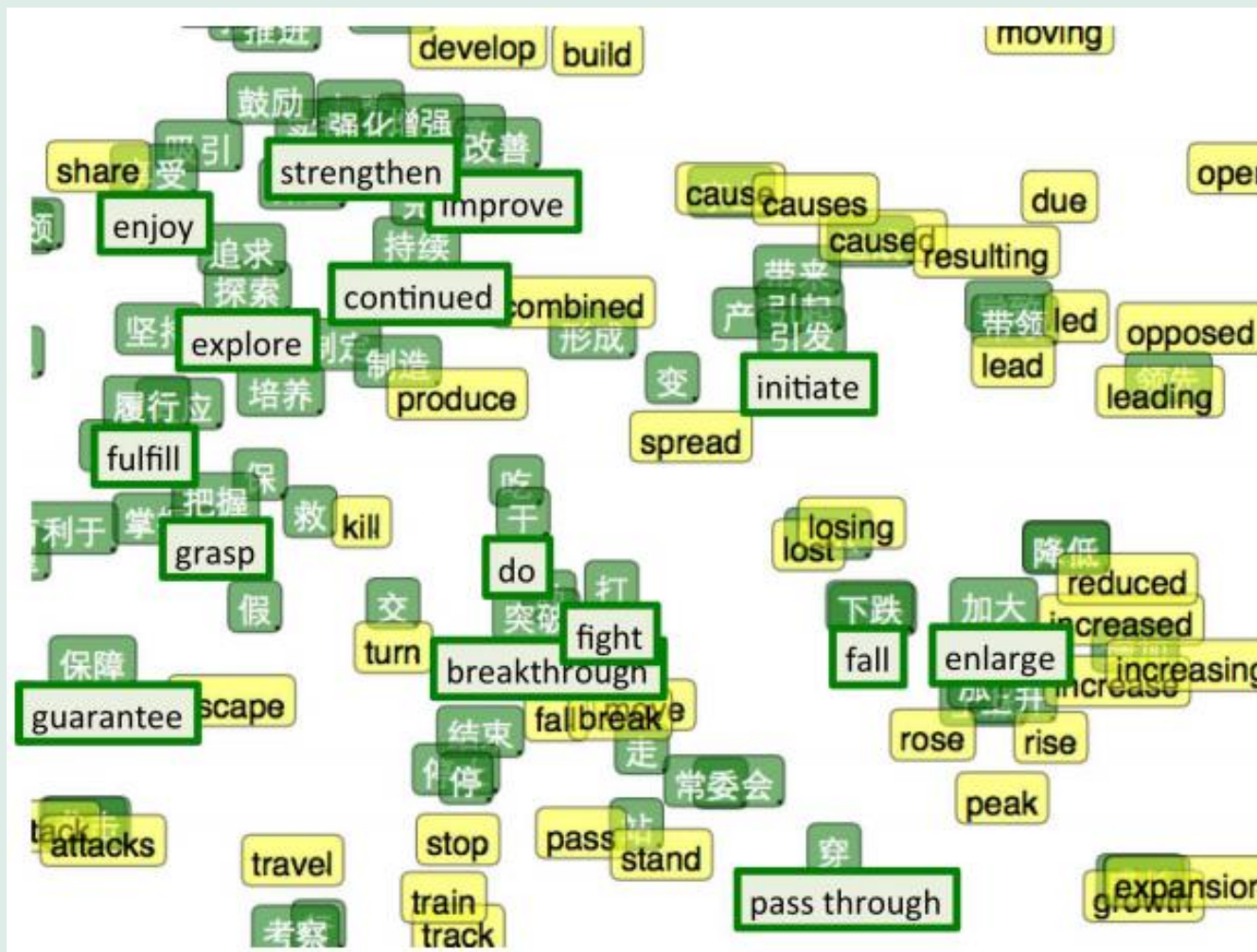
- 把文本表示成适合机器处理的形式
  - 简单做法: 词用one-hot编码表示
  - 每个词用一个01向量中的一维表示
  - “凉快”: (1, 0, 0..., 0), “凉爽”: (0, 1, 0, ...0)
  - 权力/的/游戏: (1, 0, 0, 1, 1, 0, 0, ...)
  - 冰/与/火/之/歌: (0, 1, 1, 0, 0, 1, 1, ...)
- one-hot编码的问题
  - 高维(几万-几十万), 高度稀疏
  - 看不出不同词之间的语义相似性
  - 难以做模糊匹配: “天气凉爽” vs. “天气凉快”

# 词嵌入初印象

- 词映射到嵌入空间中的低维连续向量
  - Cat:  $(-0.065, -0.035, 0.019, -0.026, 0.085, \dots)$
  - Dog:  $(-0.019, -0.076, 0.044, 0.021, 0.095, \dots)$
  - Table:  $(0.027, 0.013, 0.006, -0.023, 0.014, \dots)$
- 相似词映射到相似方向
  - 向量方向编码了语义
  - Cosine相似度衡量相似性



# 多种语言可映射到同一嵌入空间





# 词嵌入做类比题

- $v(\text{“国王”}) - v(\text{“王后”}) \approx v(\text{“男”}) - v(\text{“女”})$
- $v(\text{“英国”}) + v(\text{“首都”}) \approx v(\text{“伦敦”})$
- 词嵌入编码了语义空间中的线性关系
  - 解释: 向量的不同维度编码了不同的语义
  - 语义的组合  $\longleftrightarrow$  向量的组合
- 应用: 表示短语/句子
  - 两个句子: A含“英国”, “首都”, 不含“伦敦”; B含“伦敦”
  - 所有词的词向量的和表示句子 (“fastText”的做法)
  - 句子向量接近  $\longrightarrow$  两个句子语义相似

# 词嵌入的依据: 邻居词分布

- 中心词: 正在讨论的词
- 邻居词: 语料中在中心词周围某小窗口内出现的词
- 窗口大小 $c$ 由我们定义

我家/猫/是/我/养/的/第一/只/宠物

— “猫”: 中心词. 我家、是、我、养: “猫”的邻居词.  $c = 3$

- 邻居词分布:

猫: 宠物 0.0045 主人 0.0015 喂食 0.002 蹭 0.006 喵 0.007

狗: 宠物 0.005 主人 0.002 喂食 0.001 咬 0.003 汪 0.003

- 邻居词分布是几万/几十万维的离散分布

# 相似词为什么映射到相似方向

- 假设1: 相似词的邻居词分布类似 (出现很多词都一样, 相同词的出现频率类似)
- 倒推: 两个词的邻居词分布类似 → 两个词语义相近
- 假设2: 邻居词的分布间接反映了词的大部分语义
  - 这种语义是粗糙的大略的, 不那么精确
- 诀窍: 映射时不考虑语义, 只考虑邻居词
- 词向量是邻居词分布的一个“缩影”(降维表示)
- 猫, 狗邻居词分布类似 →  $v(\text{“猫”}) \approx v(\text{“狗”})$

# 词嵌入的优点

传统one-hot编码: “天气”: (1, 0, 0..., 0), “气候”: (0, 1, 0, ...0)

## 词嵌入

- 维度低(100 – 500维), 连续向量, 方便各种模型处理
- 天然有聚类效果、模糊匹配 (语义相近 → 向量的方向相近)
- 可方便进行语义组合
  - 大部分短语/句子的语义可用词向量的组合来近似捕捉
  - $v(\text{“私营企业”}) \approx v(\text{“私营”}) + v(\text{“企业”})$
  - $v(\text{“吸引眼球”}) \neq v(\text{“吸引”}) + v(\text{“眼球”})!$
- 无监督学习, 不需去掉停用词 (“的”, “the”等有语法意义)
- 英文不需要做stemming/lemmatization (变成原型)

# 词嵌入的局限

- 表示的语义比较粗糙
  - 上下文只能反映部分语义
  - 多种语义叠加在一个上下文分布里 – 语义变模糊了
  - 微妙的含义、复杂的过程, 难以用一个向量表示
    - “你懂的”、“城会玩”
    - “高抛低吸”(复杂的过程)
- 丧失了词的特定性
  - “上班”, “下班”词向量比较像 – google translate 闹的笑话
  - 可用one-hot作为补充特征 (fastText的做法)
- 抽象语义的叠加性不好
  - $v(\text{"water"}) - v(\text{"fish"}) + v(\text{"plant"}) \approx v(\text{"plants"}) \neq v(\text{"soil"})$

# 词嵌入向量长度反映了语义选择性

- 向量长度反映词的“语义选择性”
- 越长的向量对邻居词越“挑剔”
- “冷”和“低温”是同义词
- 冷
  - 用处广
  - 邻居词多, 每个邻居词概率相对低些
- 低温
  - 多用来描述天气
  - 邻居词少, 每个邻居词概率相对高些
- $|\mathbf{v}(\text{“低温”})| > |\mathbf{v}(\text{“冷”})|$
- 停用词的向量最短

# 现代词嵌入方法—拟合目标 (1)

- 词嵌入是做**回归**
  - 参数: 中心词 $w_i$ 和邻居词 $w_j$ 的向量
- 拟合目标:
  - 条件概率 $P(w_j|w_i)$  – word2vec
  - 二元词(bigram)概率 $P(w_i, w_j)$  – GloVe
    - 和拟合 $P(w_j|w_i)$ 等价
    - $P(w_i, w_j) = P(w_j|w_i)P(w_i)$
    - 一元词(unigram) 概率  $P(w_i)$ 可直接从数据中估计—常数

# 现代词嵌入方法—拟合目标 (2)

- 拟合目标—点互信息(PMI):  $\log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$

- 用于 SVD(Levy), PSDVec
- 和拟合  $P(w_j|w_i)$  等价:

$$\frac{P(w_i, w_j)}{P(w_i)P(w_j)} = \frac{P(w_j|w_i)}{P(w_j)} \quad \text{差一个常数}$$

- **假设**: 如果两个词语义无关, 则分布独立, 互信息为0

- $P(w_i, w_j) \approx P(w_i)P(w_j) \xrightarrow{\text{green arrow}} \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \approx 1 \xrightarrow{\text{green arrow}} \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \approx 0$
- 大部分词之间语义独立  $\xrightarrow{\text{green arrow}}$  PMI矩阵近似稀疏(接近0的多)
- 两个词喜欢一块凑  $\longleftrightarrow$  互信息 > 0



# 互信息矩阵的例子

- 英文Wikipedia语料


	the	owner	cat	dog	edit
the	0	0.5	0.28	0.238	-0.715
owner	0.065	0	0.553	1.486	-0.745
cat	-0.415	0.621	0	3.79	0.341
dog	-0.381	2.029	3.85	0	1.732
edit	-0.251	-3.912	-0.204	0.542	0

- 不完全对称, 因为 $P(w_i|w_j)$ 和 $P(w_j|w_i)$ 用了不同的平滑
- 对角置0, 因为一般 $P(w_i, w_i) \ll P(w_i)P(w_i)$  (重复词比较罕见, 也没拟合的必要)

# 现代词嵌入方法—链接函数

- Mikolov的word2vec — 第一个现代的词嵌入生成方法
- 现代之处: 用两个词向量的**点乘的指数**做链接函数

$$P(w_j|w_i) \propto \exp(\tilde{\mathbf{v}}_{w_j} \cdot \mathbf{v}_{w_i})$$

- 预测 $w_i$ 周围邻居词 $w_j$ 的概率,  $w_j$ 取遍词汇表所有词
- “双线性对数回归”(取对数后是双线性函数)—多项逻辑回归
- 点乘和cosine大致成正比  $\cos(\mathbf{v}_{w_i}, \tilde{\mathbf{v}}_{w_j}) = \frac{\tilde{\mathbf{v}}_{w_j} \cdot \mathbf{v}_{w_i}}{\|\tilde{\mathbf{v}}_{w_j}\| \|\mathbf{v}_{w_i}\|}$
- Cosine相似度衡量两个词向量的夹角  语义相近性
- Cosine相似度变化太平缓, 不能预测好所有邻居词
- 点乘的指数灵活, 但取值  $\in (0, +\infty)$ , 需做概率归一化

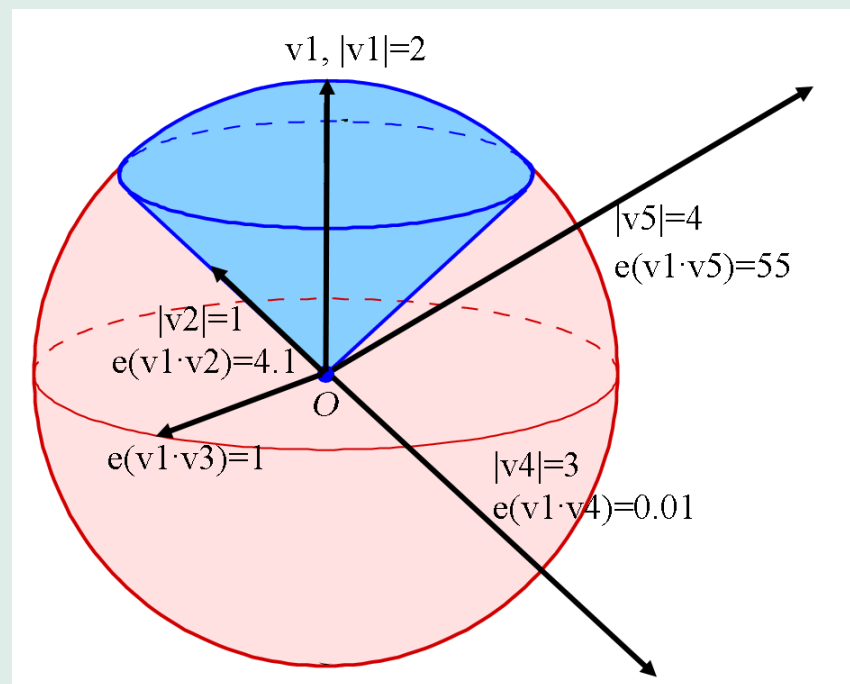
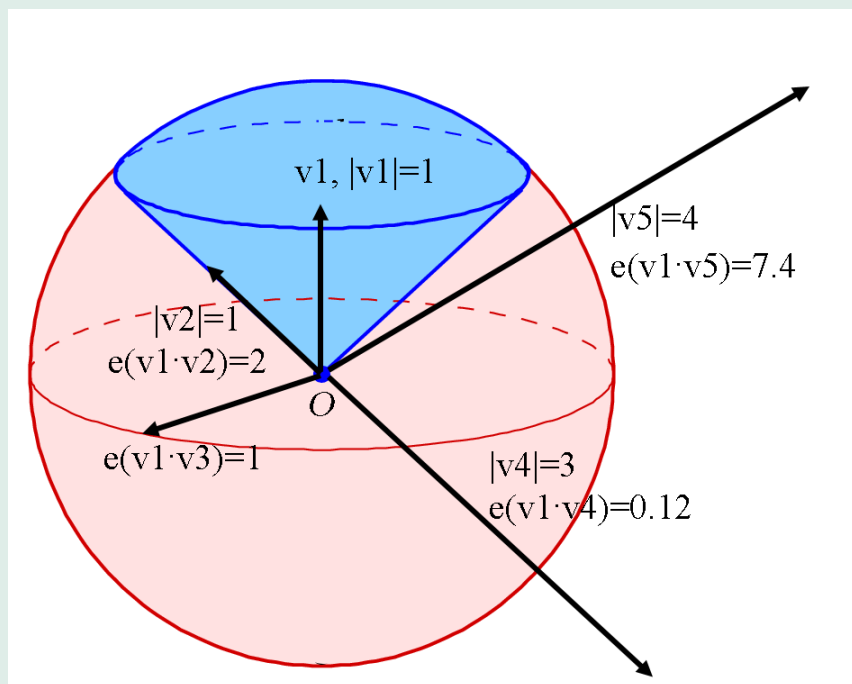
# 点乘的指数—微观解释

$$P(w_j|w_i) \propto \exp(\tilde{v}_{w_j} \cdot v_{w_i}) = \prod_n \exp(\tilde{v}_{w_{j,n}} v_{w_{i,n}})$$

- $\tilde{v}_{w_{j,n}}$  和  $v_{w_{i,n}}$  同号(兼容): 增大  $w_j$  出现在  $w_i$  周围的概率  
异号(不兼容): 减小  $w_j$  出现在  $w_i$  周围的概率
- 有些维表示语法, 有些维表示语义
- 如第  $n$  维对应名词, 则“the”, “cat”第  $n$  维值都大, 且同号
- 但每一维到底对应什么含义, 没人知道

# 向量长度对邻居词分布的影响

- $v_1$ : 中心词的词向量
- 蓝色区域中的词向量: 在  $v_1$  周围经常出现的词
- 两个图中, 向量模长分别取1和2
- 模长为1时, 图中最大和最小拟合值分别为7.4和0.12 (62倍)
- 模长为2时, 图中最大和最小拟合值分别为55和0.01 (5500倍)
- 向量越长, 邻居词分布越两极分化  $\rightarrow$  语义的选择性强





# 词嵌入方法的流程—输入预处理

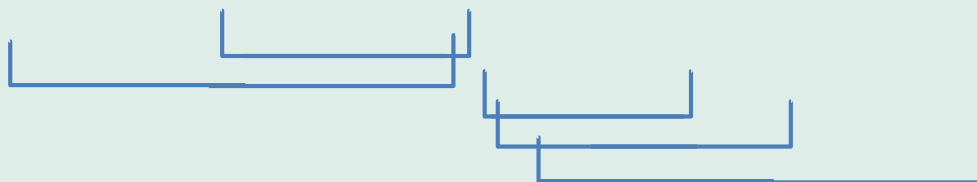
- 输入
  - 参数: 词频阈值 $T$
  - 获得一个含有许多文档的语料, 如Wikipedia
  - 把语料中所有文档并成一个超长的句子
  - 去掉标点, 转化为小写
  - 统计一元词频率, 保留语料中出现了至少 $T$ 次的单词 (删除罕见词)

the final hour of the debate however  
featured some reasonably ~~substantive~~ (罕  
见词) discussions ...

- 句子边界占比例小, 对性能影响很小, 一般可忽略

# Word2vec的流程

- 从训练长句的头走到尾, 轮流视为中心词
- 取上下文窗口大小 $c$
- 计算中心词和每个邻居词的词向量的更新梯度, 更新
- ...the final hour of the debate however...



- 更新  $v(\text{"hour"}), v(\text{"the"}); v(\text{"hour"}), v(\text{"final"}); v(\text{"hour"}), v(\text{"of"})$ .....
- 如用negative sampling: 随机选择 $n$ 个负样本, 更新向量
  - “hour-final”: 选择“google”, “hat”.....
- 迭代若干次(次数预先指定)

# 矩阵分解方法的流程

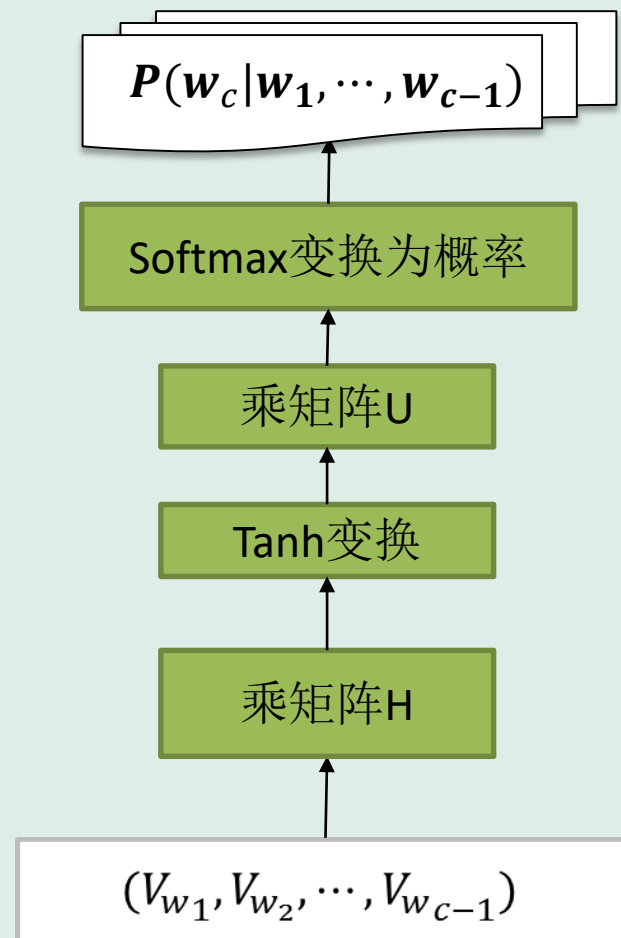
- 统计一元词的频率, 去除罕见词, 计算一元词的概率
- 构建词-邻居词的概率矩阵, 加以合适的变换(平滑、取对数、算PMI、按阈值截断等)

	the	owner	cat	dog	edit
the	0	0.5	0.28	0.238	-0.715
owner	0.065	0	0.553	1.486	-0.745
cat	-0.415	0.621	0	3.79	0.341
dog	-0.381	2.029	3.85	0	1.732
edit	-0.251	-3.912	-0.204	0.542	0

- 矩阵分解(SVD或特征值分解)
- 也可用梯度递降优化 – GloVe

# 最早的词嵌入模型

- Bengio 2003, A Neural Probabilistic Language Model
- 输入: 中心词前 $c-1$ 个词
- 输出: 中心词的概率
- 目标: 根据前面 $c-1$ 个词的词向量, 最大化预测 $w_c$ 的概率
- 两次矩阵乘, 两次非线性变换
- 缺点:
  - 参数多, 容易过拟合
  - 矩阵乘耗时, 不适合大语料
  - 词与词间向量的加权和预测概率, 不能充分抓住词向量的相似性







# Word2vec 简介

- Mikolov 2013, Distributed Representations of Words and Phrases and their Compositionality
- 使用最广泛的词嵌入方法
- 速度快, 效果好, 容易扩展
- 速度快的原因:
  - 简单(Less is more) – 去掉所有矩阵乘, 只有点乘, 很快
- 效果好的原因:
  - 点乘很好的反映了词向量间的相似性
  - 点乘足够灵活

# Word2vec 模型

- 回归链接函数: 
$$P(w_j|w_i) = \frac{\exp(\tilde{\mathbf{v}}_{w_j} \cdot \mathbf{v}_{w_i})}{\sum_{w_k \in V} \exp(\tilde{\mathbf{v}}_{w_k} \cdot \mathbf{v}_{w_i})}$$
- $P(w_j|w_i) \propto \exp(\tilde{\mathbf{v}}_{w_j} \cdot \mathbf{v}_{w_i}) = \prod_n \exp(\tilde{v}_{w_j,n} v_{w_i,n})$
- $\tilde{v}_{w_j,n}$  和  $v_{w_i,n}$  同号-兼容, 异号-不兼容
- 分母保证  $\sum_{w_j \in V} P(w_j|w_i) = 1$ : 分布归一化
- 两套词向量  $\{\tilde{\mathbf{v}}_{w_i}\}, \{\mathbf{v}_{w_i}\}$ 
  - 训练时两套都要更新
  - 输出时只保留其中一套



# Word2vec优化目标

- $w_t$ : 扫描语料, 词轮流做中心词
- $w_{t+j}$ : 中心词 $w_t$ 的邻居词.  $c$ : 上下文窗口大小

$$\max \sum_t \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j} | w_t)$$

- 最大化交叉熵
- 优化方法: 随机梯度递降(SGD)
  - 对每个词对 $w_i, w_j$ , 计算  $\log P(w_j | w_i)$  的梯度, 并更新
- SGD和“批处理”相对:
  - 批处理优化: 算梯度总和, 然后更新
  - SGD: 每碰到一项, 就算个梯度, 立即更新

# Word2vec梯度更新公式 (精确解)

$$F = \log P(w_j|w_i) = \tilde{\mathbf{v}}_{w_j}^\top \mathbf{v}_{w_i} - \log \sum_{w_k \in V} \exp(\tilde{\mathbf{v}}_{w_k}^\top \mathbf{v}_{w_i})$$

$$\frac{\partial F}{\partial \tilde{\mathbf{v}}_{w_j}} = \mathbf{v}_{w_i} - \frac{\mathbf{v}_{w_i} \exp(\tilde{\mathbf{v}}_{w_j}^\top \mathbf{v}_{w_i})}{\sum_{w_k} \exp(\tilde{\mathbf{v}}_{w_k}^\top \mathbf{v}_{w_i})}$$

$$\frac{\partial F}{\partial \mathbf{v}_{w_i}} = \tilde{\mathbf{v}}_{w_j} - \frac{\sum_{w_k} \tilde{\mathbf{v}}_{w_k} \exp(\tilde{\mathbf{v}}_{w_k}^\top \mathbf{v}_{w_i})}{\sum_{w_k} \exp(\tilde{\mathbf{v}}_{w_k}^\top \mathbf{v}_{w_i})}$$

$$\forall w_k \in V, \frac{\partial F}{\partial \tilde{\mathbf{v}}_{w_k}} = - \frac{\mathbf{v}_{w_i} \exp(\tilde{\mathbf{v}}_{w_k}^\top \mathbf{v}_{w_i})}{\sum_{w_k} \exp(\tilde{\mathbf{v}}_{w_k}^\top \mathbf{v}_{w_i})}$$

- $\sum_{w_k \in V} \exp(\tilde{\mathbf{v}}_{w_k} \cdot \mathbf{v}_{w_i})$  项数太多, 时间复杂度很高
- Negative sampling: 从词汇表 $V$ 随机抽取若干 $w_k$ , 计算梯度



# Word2vec 优化特点

- 收敛很快。大语料: 1、2个pass, 小语料: ~10个pass
  - 收敛是指词向量的质量稳定了
- Negative sampling: 近似计算梯度, 提高效率
- Hierarchical softmax: 提高效率, 效果较差—不推荐
- CBOW和skip-gram效果差不多

## Word2vec 的启发

$$P(w_j|w_i) = \frac{\exp(\tilde{\mathbf{v}}_{w_j} \cdot \mathbf{v}_{w_i})}{\sum_{w_k \in V} \exp(\tilde{\mathbf{v}}_{w_k} \cdot \mathbf{v}_{w_i})}$$

$$\rightarrow P(w_j|w_i) = \exp(\tilde{\mathbf{v}}_{w_j} \cdot \mathbf{v}_{w_i}) / Z_i$$

- 两边取对数:

$$\log p(w_j|w_i) + \log Z_i = \tilde{\mathbf{v}}_{w_j}^T \mathbf{v}_{w_i}$$

- 对每一对  $w_i, w_j$  都成立
- 把左右都写成矩阵形式 – 像什么问题?

# Word2vec 和矩阵分解的等价性

- $\log p(w_j|w_i) + \log Z_i = \tilde{\mathbf{v}}_{w_j}^T \mathbf{v}_{w_i}$
- 右边两个低维矩阵的乘积拟合左边矩阵
- 左边:  $V \times V$  ( $V$ : ~10万). 右边:  $D \times V$  ( $D$ : ~500)
- Neural Word Embedding as Implicit Matrix Factorization, Levy 2014
- Word2vec 理论上等价于分解 Pointwise Mutual Information (PMI) 矩阵

$$\text{PMI}(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$$

- 启发了多种用矩阵分解求词嵌入的方法

# GloVe: 矩阵分解求词嵌入

- 回归的链接函数:


$$P(w_i, w_j) = \exp\{\tilde{\mathbf{v}}_{w_j}^\top \mathbf{v}_{w_i} + a_i + \tilde{a}_j\}$$

- 如果  $a_i \approx \log P(w_i)$ ,  $a_j \approx \log P(w_j)$ :

$$\tilde{\mathbf{v}}_{w_j}^\top \mathbf{v}_{w_i} = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \quad \text{—拟合PMI}$$

- 优化目标:

$$\sum_{i,j=1}^V f(P(w_i, w_j)) \left( \tilde{\mathbf{v}}_{w_j}^\top \mathbf{v}_{w_i} + a_i + \tilde{a}_j - \log P(w_i, w_j) \right)^2$$

- $f(P(w_i, w_j))$ : 词对  $w_i, w_j$  的频率越大, 权重  $f$  越大
- 低频二元词对噪音大, 容易产生大的误差 → 主导优化目标 
- 高频词才是最重要的, 数据噪音也小





# 词嵌入算法的评估

- 词相似性任务
  - 哪一对更接近？  
tiger cat  
street children
- 类比任务
  - Beijing to China is like Berlin to \_\_\_\_\_
- 传统NLP任务
  - 命名实体识别 (NER)
  - 名词短语识别 (Noun phrase chunking)

# 不同算法性能比较

	Similarity Tasks				Analogy Tasks		NLP Tasks	
Method	WS	WR	MEN	Turk	Google	MSR	NER	Chunk
word2vec	74.1	54.8	73.2	<b>68.0</b>	<b>72.3</b>	<b>63.0</b>	<b>84.8</b>	94.8
SVD	69.2	60.2	70.7	49.1	24.0	11.3	81.2	94.1
GloVe	75.9	63.0	75.6	64.1	54.4	43.5	84.5	94.6
Singular	76.3	<b>68.4</b>	74.7	58.1	50.8	39.9	83.8	94.8
Sparse	74.8	56.5	74.2	67.6	71.6	61.9	78.8	<b>94.9</b>
PSDVec	<b>79.2</b>	67.9	<b>76.4</b>	67.6	62.3	50.7	84.7	94.7

- NER: 命名实体识别, Chunk: 名词短语识别
- 数据来自本人实验



# Word2vec 为何比大部分MF方法好

- SGD每次只更新碰到的词对
  - 碰到的大部分是高频词对, 更新的多
  - 低频词对很少碰到, 更新的少
  - 高频词对主导优化
  - 低频词对的优化是次要的
- 矩阵分解(MF)方法
  - 高频词对更重要, 频率更能反映真实分布
  - 但低频词对往往造成大的误差
  - 低频词对的频率的随机性更大, 噪音大
  - 低频词对误导了优化的方向
  - 对词对按频率加权, 不能完全修复这个缺陷

# 影响词嵌入效果的因素

- 语料大小; 语料和应用是否在同一领域
  - 语料越大越好, 领域越接近越好
- 向量维度: 语料大 → 维度选大些。过大容易过拟合
  - Wikipedia (25G): 300-500
  - 几M字节的语料: 30-50
- 上下文窗口: 3~7
  - 过小: 捕捉模式太少
  - 过大: 噪音太多
  - 按距离加权: 窗口可大些
- word2vec特有的参数:
  - Iteration: 语料大 → 迭代次数少
  - Negative sampling: 适中, 如5~10
  - Alpha: 初始学习率。缺省值就行

# “多词义”词嵌入

- 多词义, 单向量 = 不同词向量的加权和
- 为每个词义学习单独的词向量—看上去很美
- $\approx$  对邻居词做软聚类. 非参贝叶斯自动选择聚类数
- 学出的多词义并不准确
  - 一个意思可能被打散成几类
- 用于学习词向量的数据变得更稀疏—词向量不够准确
- Jiwei Li, 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding?
- 结论: 多词义向量的好处, 大都可以通过提高向量维度来达到

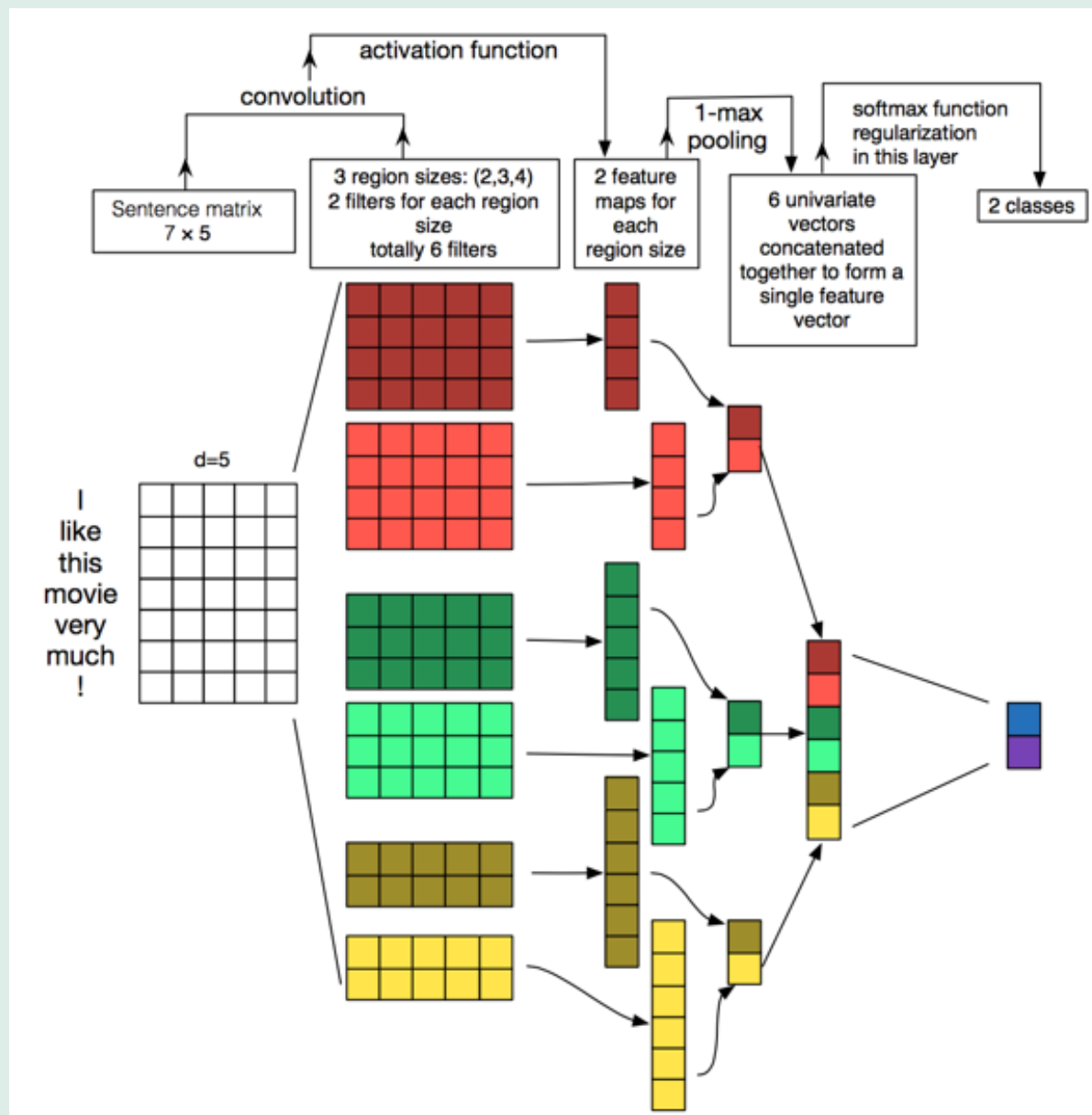
# CNN句子分类

- 激活方程:

Tanh, ReLU...

<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

- 可以用预先训练的词向量, 也可以反向传播训练词向量(速度慢, 需要语料多, 但据说效果稍好)



# 词嵌入 vs. 矩阵分解做推荐系统 – 相似

- 都从成对的数据的分布中学到隐表示
  - 词嵌入: 词-邻居词矩阵
  - 推荐: **user-item**矩阵
- 输入矩阵都是高度稀疏的
  - 绝大部分词的邻居词, 只占词汇表的一少部分
  - 用户购买的商品只占有所有商品的很少一部分
- 无监督学到的隐表示, 人不知道每一维的含义
  - 词嵌入: 不知道每一维对应什么语义
  - 推荐: 不知道每个隐含因子对应了什么

# 词嵌入 vs. 矩阵分解做推荐系统 – 差别

- 词-邻居词矩阵是对称的
- user-item矩阵不对称
- 词-词矩阵一般做对数变换后分解(对数空间)
  - 频繁词对和不频繁词对的频率差别非常大
  - 取对数之后变平滑

of the: 28207850, cat neck: 2

取log后: 17.2, 0.7
- user-item矩阵一般直接分解(线性空间)
- 最常用的算法word2vec基于随机梯度递降(SGD)
  - 理论上等价于某些矩阵分解算法



# 词嵌入和主题模型的区别

- 二者都可以表示文档
- 词嵌入：
  - 利用每个词和它的**小邻域**里其他词的共现模式
  - 得到词的表示, 叠加得到文档的表示
  - 小邻域共现模式更多 — 优势
  - 每个词向量定义了一个邻居词分布
- LDA:
  - 利用两个词在同一组**文档**中的共现模式
  - 得到词的“软聚类”—主题。每个文档主题比例用来表示文档
  - 文档级共现模式相对稀少 — 缺点
  - 主题分布直接定义为多项分布
- 小邻域的共现模式和文档级共现模式是互补的
  - 二者可以结合—主题嵌入模型
  - 每个主题用一个向量, 定义一个主题中的词的分布



# 课后作业: 分析唐诗宋诗分布特性

- 全唐诗:

<http://vdisk.weibo.com/s/tmnyH9Z9h6pew>

- 全宋诗:

<http://vdisk.weibo.com/s/v9HGjPQ3ca0z>

- 不需分词, 以字为单位训练词向量
- 找出唐诗中最常用的20个实词, 比较它们在唐诗和宋诗里的相似词有无明显区别? 如有区别, 是否反映了唐诗和宋诗的风格变化?

# 词嵌入相关资源

- [Word2vec](#), C语言实现
- [Gensim](#), Python实现, 接口多
  - `>>trained_model.most_similar(positive=['woman', 'king'], negative=['man'])`  
`[('queen', 0.50882536), ...]`
- [DL4J](#), Deep Learning for Java
- 我自己的[PSDVec](#)
- 分词软件
  - [jieba](#), Python实现
  - [Thulac](#), C++实现, 需申请



# 语料资源

- [英文Wikipedia](#) 12G
- [中文Wikipedia](#) 1.2G
- <http://lafnews.com/corpus/> 只含新闻链接(需自己使用爬虫抓取)
- [人民日报1998年1月词性标注语料](#)
- [500万微博语料](#)
- [分词词库](#)
- [命名实体识别语料\(英文\)和训练代码](#)
- [使用词嵌入的CRF Chunking代码和数据](#)



# 词嵌入参考资料

- 创始篇– NNLM:  
A neural probabilistic language model, Bengio et al. JMLR 2003
- **Word2vec:**  
Distributed representations of words and phrases and their compositionality. NIPS 2013  
Efficient estimation of word representations in vector space. ICLR 2013
- 矩阵分解:  
Neural word embeddings as implicit matrix factorization. NIPS 2014  
Glove: Global vectors for word representation. EMNLP 2014
- 综合:  
Linguistic regularities in sparse and explicit word representations. CoNLL-2014  
Improving distributional similarity with lessons learned from word embeddings. TACL 2015  
Generative Topic Embedding: a Continuous Representation of Documents. ACL 2016



# 自然语言处理参考资料

## 1. 经典教材 Speech and Language Processing

<https://web.stanford.edu/~jurafsky/slp3/>

(第1,2版内容略陈旧, 第三版很入时, 但有些重要章节没放在网上)

## 2. 课程 Stanford CS224d: Deep Learning for Natural Language Processing

- <http://cs224d.stanford.edu/syllabus.html>

## 3. 会议论文:

- ACL 2015/2016:

[http://acl2015.org/accepted\\_papers.html](http://acl2015.org/accepted_papers.html)

[http://acl2016.org/index.php?article\\_id=13#long\\_papers](http://acl2016.org/index.php?article_id=13#long_papers)

- EMNLP 2015/2016:

<http://www.emnlp2015.org/accepted-papers.html>

<http://www.emnlp2016.net/accepted-papers.html>

- 论文很多, 结合引用数选看



谢谢大家！  
欢迎大家批评指正！



# 法律声明

本课件以及大数据文摘网站和QQ群内的所有内容,包括但不限于演示文稿、软件、声音、图片、视频、代码等,由发布者本人和大数据文摘共同享有。未经大数据文摘的明确书面特别授权,任何人不得翻录、发行、播送、转载、复制、重制、改动或利用大数据文摘的局部或全部的内容或服务或在非大数据文摘所属的服务器上作镜像,否则以侵权论,依法追究法律责任。

本网站享有对用户在本网站活动的监督和指导权,对从事网上非法活动的用户,有权终止对其所有服务。

课程详情咨询

微信客服: shujupeixun

QQ客服: 3530548572