6.1 Lexical Semantics

不同时期不同地区，类似的环境和气候出现相同结构的生物

分布假说：单词的相似分布和他们的相似意义具有关系

从语言学分布中学习向量语义

representation learning表示学习：自监督方法学习输入表征

不是简单用大小写表示这种无意义表示，要得到同义词/反义词/积极含义/消极含义

词根和词义：

- lemma 词根-引用形式 mouse
- wordform mice

mouse：有不同的word sense 需要**词义消歧**

语言学基本原则：**对比原则** H2O water

同义词不多，但**相似词**很多：cat dog

我们只需处理词，不需要处理词义

相关词：coffee cup

语义场和主题模型：LDA Latent Dirichlet Allocation 从大量文本中自监督学习获得相关文本的集合。医院语义场（外科医生，手术刀，护士，医院） 具体的语义领域，词语之间有结构化关系

语义框架和角色：表现特定视角或者参与一个特定类型的事件，金融交易（购买，售卖，买家，卖家）

情感：积极和消极的评估语言 二分类

connotations（**隐含意义**）：和作者读者情感有关的词意义，三个维度：

- valence: the pleasantness of the stimulus
- arousal: the intensity of emotion provoked by the stimulus
- dominance: the degree of control exerted by the stimulus

6.2 Vector Semantics

语言学家提出的使用单词分布定义词义，以及情感三维度定义

相同上下文中相同位置的词意义相同

embedding: mapping from one space or structure to another

6.3 Words and Vectors

6.3.1 Vectors and documents

向量模型和分布模型大多基于共现矩阵：

- term-document matrix：词语和文档
- term-term matrix：词语和词语

vector space model：用几个词语的出现次数来组成文档向量 使用这个向量进行信息检索

Information retrieval (IR)： is the task of finding the document d from the D documents in some collection that best matches a query q

6.3.2 term-document matrix

V rows:词语数量

D columns:文档数量

文档向量：将每列作为向量

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

词语向量：将每行作为向量

	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.

6.3.3 term-term matrix

两个词语在某种上下文中共现概率（可以是文档或者段落），一般是一个语境窗口（前4词-后四词）

V通常在10000-50000之间，保留最频繁的。一般50000名之后的词语保留也无太大意义

	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.

6.4 Cosine for measuring similarity

内积/点积：$(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = v_1w_1 + \ldots + v_Nw_N$

需要标准化，否则数值范围不统一 $cos\theta = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|}$

一般先预先标准化为单位向量，再直接计算cos相似度

原始频率值为正值，因此之前的方法计算出的相似度范围为$[0,1]$，广义的范围为$[-1,1]$

6.5 TF-IDF: Weighing terms in the vector

原始频率有偏，不够有判别力。对于the it they等出现太频繁的词语无法给出特定的意义提示。一些出现比较频繁的比如strawberry等又比较有用

TF-IDF：文档维度的加权策略

PPMI：词语维度的加权策略

Term Frequency **TF**：某个词在某个文档中出现的次数

- $tf_{t,d} = count(t,d)$ ==> $log_{10}(count(t,d)+1)$

Collection Frequency: 某个词在所有文档中出现的总次数

Document Frequency：某个词在多少个文档中出现过

给只出现在一些文档中的词更高权重：逆文件频率 inverse document frequency or **IDF** term weight
N代表文档数量

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

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

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.9 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of $\text{tf} = \log_{10}(20+1) = 1.322$ 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*.

6.6 Pointwise Mutual Information (PMI)

PPMI (positive pointwise mutual information) 利用了一种直觉，即衡量两个单词之间关联的最好方法是，询问这两个单词在**我们的语料库中同时出现**的次数比我们**预测它们出现的次数多多少**

点互信息：$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$

词语和上下文词之间：$I(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$

- 分子：同时出现的概率
- 分母：假设词语和上下文词之间相互独立，同时出现的概率

对于寻找**强相关词汇**非常有用

由于不确定能否评估不相关的分数（即负数值），且如果PMI分数为负可信往往需要大数据集（比如两个分数都是1e-6，相乘1e-12，则需要至少1e12大小语料库？），一般使用**正点互信息**

$PPMI(w,c) = max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0) = [\log_2 \frac{P(w,c)}{P(w)P(c)}]_+$

	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.

W rows C columns

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \tag{6.19}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0) \tag{6.20}$$

解决罕见词PPMI分数高:

1. P(c)乘以因子 $\alpha = 0.75$比较合适

$$\text{PPMI}_{\alpha}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_{\alpha}(c)}, 0) \tag{6.21}$$

$$P_{\alpha}(c) = \frac{count(c)^{\alpha}}{\sum_{c} count(c)^{\alpha}} \tag{6.22}$$

2. Laplace平滑 每个数值在计算前被加上一个常量，则更小的非零值就会分到更小的权重

6.7 Applications of the tf-idf or PPMI vector models

文档相似度基本应用：信息检索，抄袭监测，新闻推荐系统

- 计算文档质心向量后计算cos：$d = \frac{w_1+...+w_k}{k}$

词语相似度：查找词语释义，跟踪单词含义变化，自动发现不同语料库中单词含义

6.8 Word2vec

embeddings： short dense vectors d-50~100

TF-IDF和PPMI的向量是**稀疏**的

嵌入是**稠密的实数值**，没有那么多0向量，可以为负

为什么稠密向量效果更好？两个直觉推测：更少的向量可以用**更少的分类器参数**进行学习，且有助于**泛化**和**防止过拟合**

同义词捕捉上也更好

基于负采样的跳跃图模型：skip-gram with negative sampling: **SGNS** word2vec算法库中的一个算法，也可以粗略被称作word2vec

- 静态嵌入方法：每个词分配一个固定的向量

直观理解：不计算某个词附近词语出现的频率，训练一个二分类任务—**词语c是否可能出现在词w的附近**？我们不关心预测任务，相反我们将学习到的**分类器权重**作为词嵌入

任意的文本都可以作为隐式监督数据，比如出现在w附近的单词c都可以作为二分类任务的正例，即这是个**自监督任务。**最开始在神经语言模型提出（基于先前词语预测下一个词语，通过这个任务学习词嵌入表示）

基本步骤：

1. Treat the **target word** and a **neighboring context word** as **positive examples**.
2. Randomly sample other words in the lexicon to **get negative samples**.
3. Use **logistic regression** to train a classifier to distinguish those two cases.
4. Use the **learned weights** as the embeddings.

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

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

简化假设：所有上下文词相互独立，因此可以直接将所有词的概率相乘

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

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

对于V个词，需要V个w向量组成**W**矩阵，V个c向量组称**C**矩阵

训练实例

输入：一个文本语料库+所选择的词表V

先初始化各个向量参数

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

选择出正例和**k倍**的负例（即对每个正样本选取k个负样本），负例根据其**一元语法频率权重**随机获取，**不能是上下文词**

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

一元语法频率权重：设置$\alpha = 0.75$更合适因为这样会给稀有噪声词汇更多的概率

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

目标：最大化正例和目标词相似度，最小化目标词和负样本相似度

使得正样本对相似度接近1，负样本对相似度接近0

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

随机梯度下降进行优化，**分别对每个向量求导**（链式求导即可）

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

最终的优化表达式：

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

最终将中心词向量和上下文词向量相加得到词表示。也可以每个词直接使用一个向量表示

上下文窗口大小L需要使用开发集调优

其他静态嵌入方法

fasttext：考虑了未知词。由于有单词稀疏性问题，很多单词的形态（名词/动词等）出现太少，因此使用子词模型来解决问题，比如将 `<where>` 表示成 `<wh, whe, her, ere, re>`

GloVe: Global Vectors 捕捉全局语料统计信息 GloVe基于词-词矩阵的概率，结合PPMI等基于计数的模型，同时也使用了word2vec等方法的线性结构

word2vec和稀疏编码PPMI具有优雅数学关系，因此word2vec可以被看作隐式优化PPMI矩阵

6.9 Visualizing Embeddings

最简单方式：列举和每个词最相关的一些词

另一种：分层聚类算法来展示相似单词的层次化嵌入

比较常用的：将多维向量转为2维 **t-SNE**，从而方便画图

6.10 Semantic properties of embeddings

不同类型的相似度和联系：

- 窗口小：词性相同。霍格沃茨和其他小说中的魔法学校
- 窗口大：主题相关但并不相似。霍格沃茨和邓布利多，马尔福

一级共现：组合关联，直接靠近彼此。比如：书和诗歌

二级共现：范式关联，他们有相似的邻居。比如：said remarked

类比/关系相似度

- 早期认知模型，四边形模型。a is to b as c is to ?
- word2vec or GloVe 在这个任务表现好

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

然而有时候也会倾向于输出另外三个单词或者三个单词的变体

四边形方法对于其他关系表现可能很差，而且被证明相比人类认知过于简单

6.11 Bias and Embeddings

偏见和刻板印象：

- 男人-程序员=女人-家庭主妇
- 男人-医生=女人-护士
- allocational harm 分配伤害 系统不公平分配资源给不同群体

放大偏见：

- 性别偏见方法，比实际的劳动力就业数据更夸张

嵌入还编码了人类推理属性的隐式关联

- 不同人种，不同年龄，不同性别可能和不同情绪/领域词汇相连
- 代表性伤害 representational harm 系统贬低或歧视一些群体

转变编码空间从而消除性别偏见但保留性别定义或改变训练流程，仍然是开放问题

也可以调查历史上的偏见

6.12 Evaluating Vector Models

最重要的外部评估：在任务中进行外部评估 在NLP任务中使用向量观察是否提升性能

内在评价：比较词相似度分数和人类打分

不包含上下文：WordSim-353 SimLex-999 TOEFL dataset

包含上下文：

- Stanford Contextual Word Similarity (SCWS) dataset
- Word-in-Context (WiC) dataset
- semantic textual similarity task
- analogy 类比任务（平行四边形任务）：形态学，词典关系，百科关系

初始化参数和随机负采样导致固有的不确定性太大，集合中单个文档可能会强烈影响整个嵌入

所以最好使用引导采样再取平均的方式