Modern Information Retrieval

现代信息检索

第14讲：面向信息检索的分布式词项表示

Distributed Word Representations for Information Retrieval

怎样更鲁棒的匹配用户搜索意图？

我们希望**理解**查询，而不仅仅是字符串匹配

- 如果用户搜索 [Dell notebook battery size]，我们希望匹配关于 “Dell laptop battery capacity”的文档，虽然仅有一半关键字一致
- 如果用户搜索 [Seattle motel], 我们希望匹配包含 “Seattle hotel”的文档

简单的关键字匹配并不能满足上述需求.....

之前课程涉及到的一些简单技术会有一点帮助

- 拼写矫正
- 词干还原/大小写统一

但是我们希望能够更好的**理解**查询/文档匹配

怎样更鲁棒的匹配用户搜索意图？

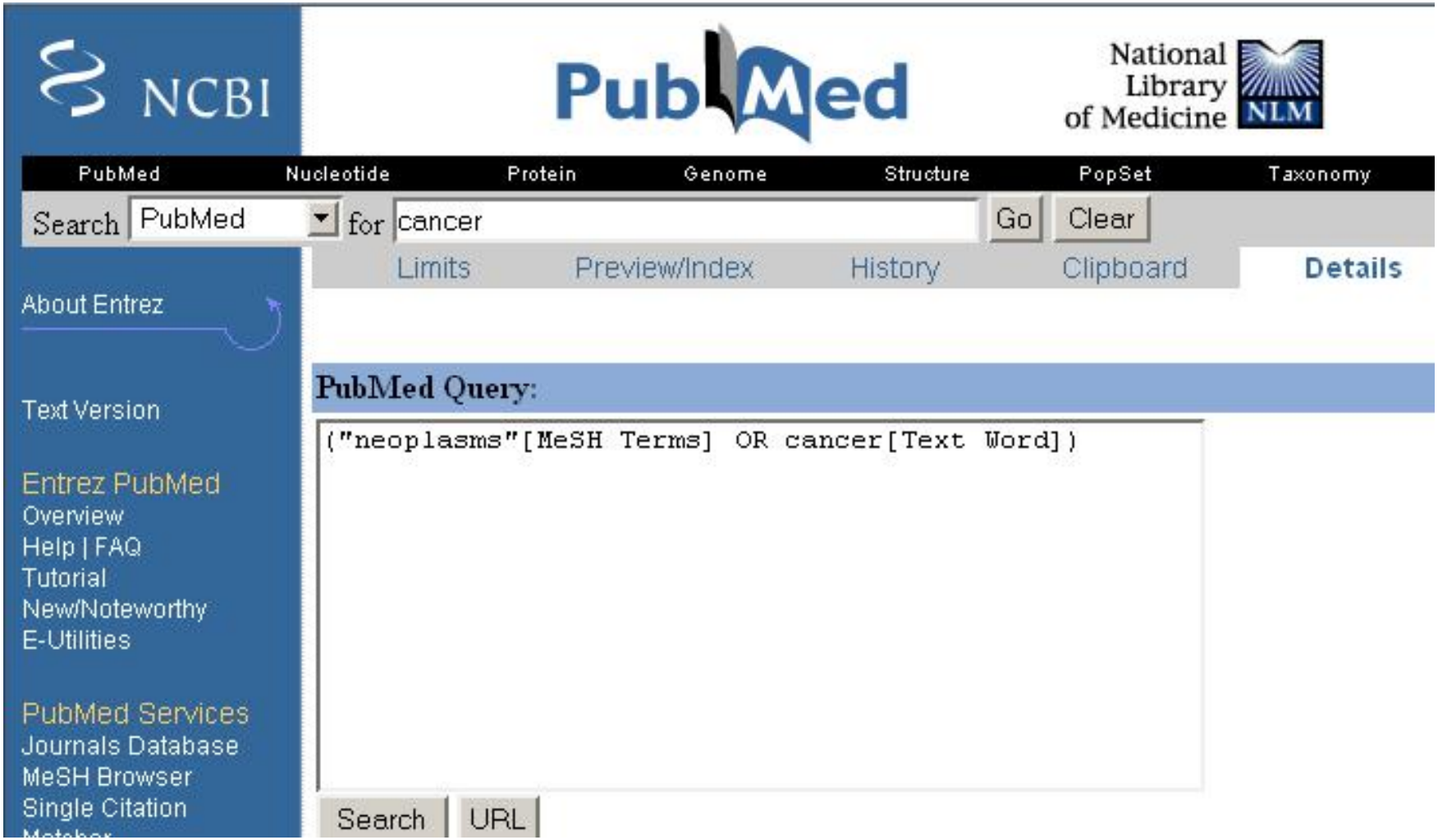
查询扩展/Query expansion:

- 相关反馈/Relevance feedback 能够通过向查询中添加扩展词，从而对捕捉用户搜索意图有所帮助
- 也可以利用 词项相似度/word similarities 信息:
  - 基于人工 同义词表/thesaurus of synonyms 的查询扩展
  - 词项相似度指标
    - 基于大规模文档语料计算
    - 基于查询日志挖掘（Web上的常见做法）计算

文档扩展/Document expansion:

- 使用锚文本/anchor text可以通过提供人工创作的同义词（即锚文本）来解决此问题，但不适用于新的或不太受欢迎的网页（注：链接稀疏，锚文本少）或无超链接的语料

人工同义词典的例子


NCBI
PubMed
National Library of Medicine
NLM
PubMed
Nucleotide
Protein
Genome
Structure
PopSet
Taxonomy
Search
PubMed
for
cancer
Go
Clear
Limits
Preview/Index
History
Clipboard
Details
About Entrez
Text Version
Entrez PubMed
Overview
Help | FAQ
Tutorial
New/Noteworthy
E-Utilities
PubMed Services
Journals Database
MeSH Browser
Single Citation
PubMed Query:
("neoplasms"[MeSH Terms] OR cancer[Text Word])
Search
URL

基于查询日志的查询扩展

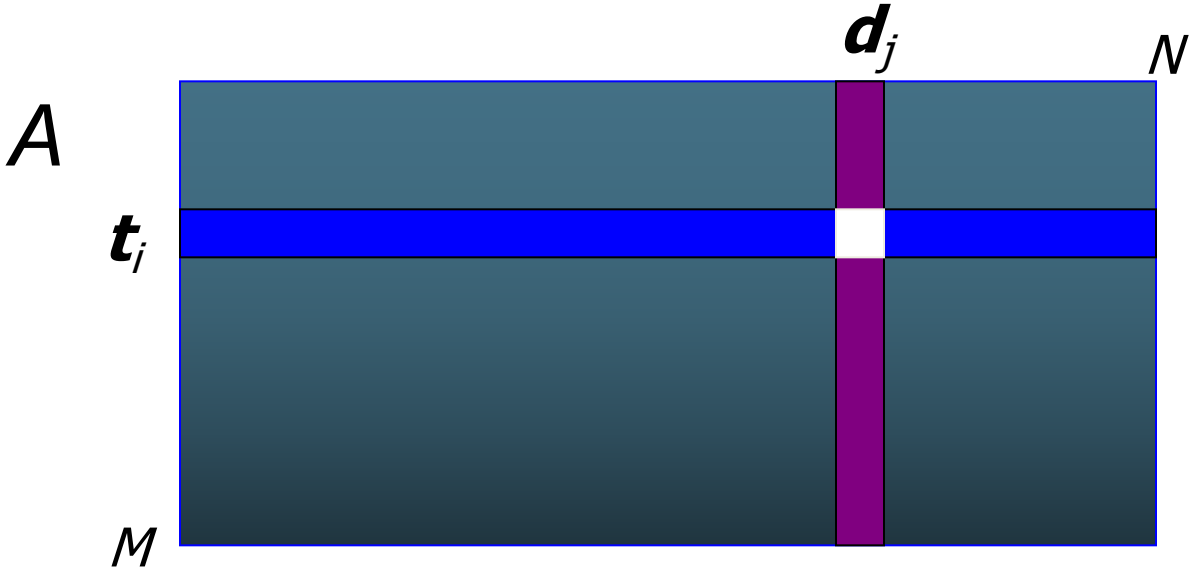
- 不考虑上下文语境的查询扩展可能会导致问题
  - [wet ground] ≈ [wet earth]
  - 于是 expand [ground] ⇒ [ground earth]
  - 但是 [ground coffee] ≠ [earth coffee]
- 从查询日志学习考虑上下文语境的查询重写：识别同一用户基于同一信息需求的多次查询请求
  - [Hinton word vector]
  - [Hinton word embedding]
- 在此上下文语境中，[vector] ≈ [embedding]
  - 但是在其它的上下文中，如 *disease vector* 或 C++ 语言中 [vector] ≠ [embedding]

自动同义词库生成

- 尝试通过分析文档集来自动生成同义词库
- 基本概念：两个词之间的相似性
- 定义1：如果两个单词与相似单词同时出现，则它们是相似的。
  - 我们可以 收获(harvest)、刨皮(peel)、食用(eat)、准备(prepare) 苹果(apple)和梨(pear)，因此苹果和梨相似。
- 定义2：如果两个单词与同一个词在给定的语法关系中出现，则它们是相似的。
  - He eats/harvests/peels/prepares apples/pears
- 基于共现的相似度更鲁棒，基于语法关系的相似度更准确。
  - 前者更全面，后者更准确

简单的基于共现的同义词库

- 最简单的方法就是通过词典-文档矩阵A计算词项-词项的相似度 $C = AA^T$
- $w_{i,j} = (t_i, d_j)$的(归一化)权重



- 对每个t_i，选择C中高权重的词项进行扩展

如果矩阵A是0/1矩阵，那么C的每一项是什么？

基于共现关系的同(近)义词词典样例

单词	同近义词
absolutely	absurd, whatsoever, totally, exactly, nothing
bottomed	dip, copper, drops, topped, slide, trimmed
captivating	shimmer, stunningly, superbly, plucky, witty
doghouse	dog, porch, crawling, beside, downstairs
makeup	repellent, lotion, glossy, sunscreen, skin, gel
mediating	reconciliation, negotiate, cease, conciliation
keeping	hoping, bring, wiping, could, some, would
lithographs	drawings, Picasso, Dali, sculptures, Gauguin
pathogens	toxins, bacteria, organisms, bacterial, parasites
senses	grasp, psyche, truly, clumsy, naïve, innate

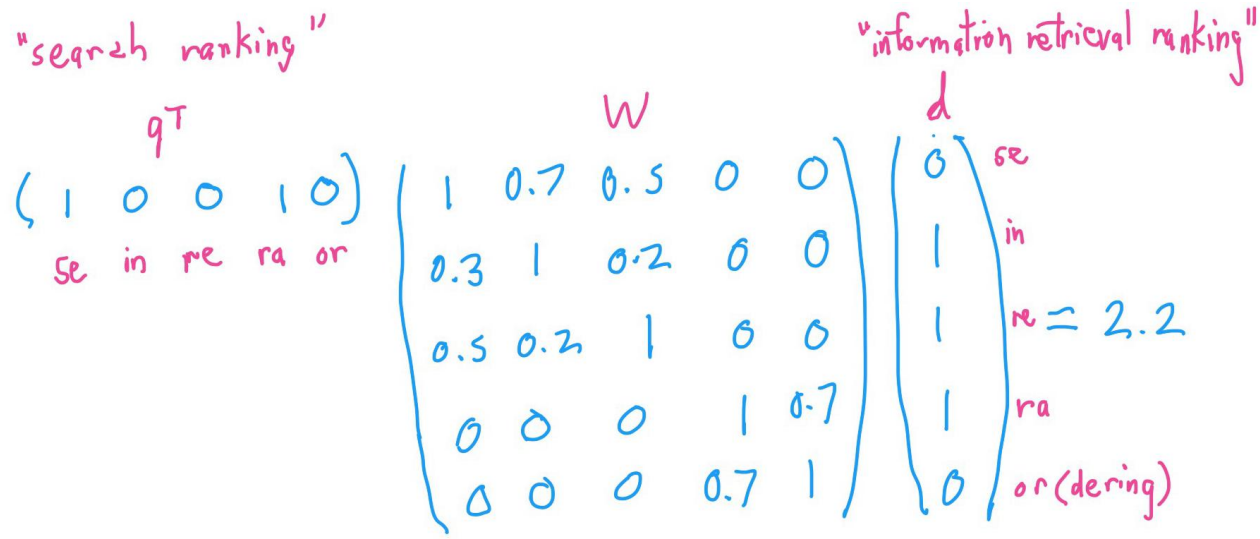
WordSpace demo on web

怎样表示词项之间的关系？

- 使用词项的标准符号编码(standard symbolic encoding of terms)，每个词项都是一个维度
- 不同的词项没有内在的相似性 (inherent similarity)
- motel $[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$
  hotel $[0\ 0\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$
- 如果查询是关于 *hotel* 并且 文档包含 *motel*, 那么查询和文档向量**正交 (orthogonal)**

能否直接学习词项（之间的）关系？

- IR评分基于$q^T d$
- 不考虑同义词，不使用机器学习
- 能否学习参数（矩阵）W 从而可以用 $q^T W d$ *排序*?



- Cf. Query translation models: Berger and Lafferty (1999)
- 存在稀疏性的问题 – W 非常大 > 10^{10}

是否有更好的办法?

- 基本思想:
  - 学习$\mathbb{R}^d$中单词的密集低维表示，从而可以使用点积$u^T v$表示词项相似度
  - 如果想在词汇表之间添加“翻译”矩阵（例如，跨语言），我们仍然可以使用 $u^T W v$ 表示词项相似度
    - 但是 **此时 W 足够小**!
  - 关于有监督的语义索引（Supervised Semantic Indexing）（Bai et al. *Journal of Information Retrieval* 2009）的研究表明我们可以成功的学习面向IR的 W
- 接下来我们将介绍其它直接计算词项相似度的方法

基于分布式相似度的表示/Distributional similarity based representations

- 用相邻词项的意义来表示一个词项，是一种可行的方法
- “You shall know a word by the company it keeps”
  - (J. R. Firth 1957: 11)

- 现代统计自然语言处理最成功的idea之一

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

↖ 使用相邻的词项表示 *banking* ↗

由上下文可知这里bank是“银行”不是“河岸”的意思

解决方案：低维向量

- 文本语料中涉及到的话题（相对词项数量）通常较少
  - 体育，电影，政治，…
- 基本思想：将“大部分的”重要信息存储在一个维度固定的低维向量中 - 即“密集向量”(dense vector)
- 维度通常在 25 - 1000 之间
- 怎样降低维度?
  - 将高维稀疏的共现计数向量转化为低维“词嵌入”

(Go from big, sparse co-occurrence count vector to low dimensional “word embedding”)

传统方法:潜在语义索引/分析 (Latent Semantic Indexing/Analysis, LSI/LSA)

- 使用奇异值分解（Singular Value Decomposition,SVD）－或只是随机投影以找到低维基本向量或正交向量
- 理论上说，相似度被尽可能的保留
- 使用LSA可以获得IR效果上的轻微的提升，这是因为词项变体带来的“噪声”被语义“概念”所替代
- 在90年代有一定的流行 [Deerwester et al. 1990, etc.]
  - 但是 **结果总是不太确定** (… 有时有效)
  - 在IR系统中的实现难以保证效率

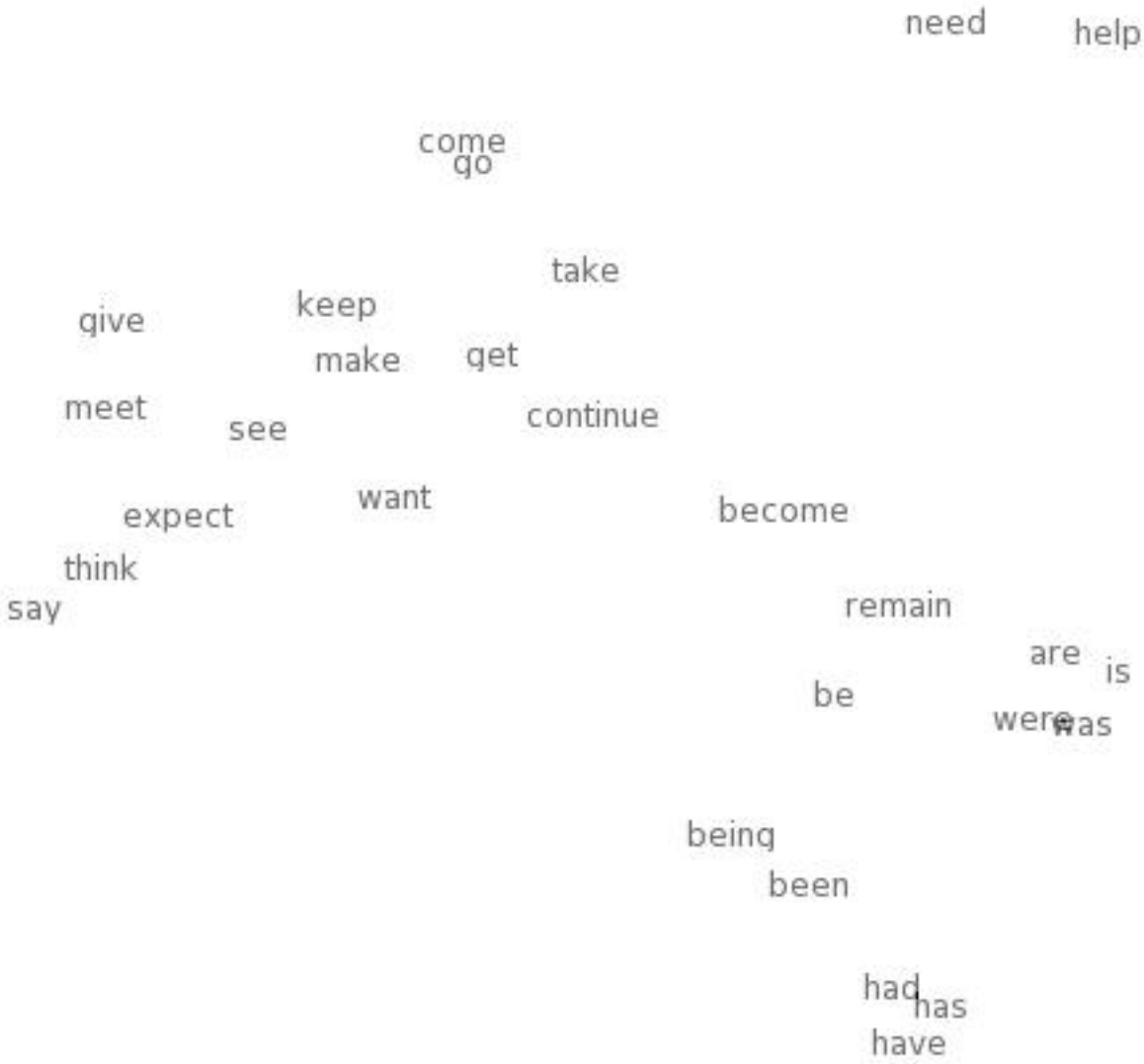
“NEURAL EMBEDDINGS”
神经嵌入

词项的意义由向量表示

- 为每个词类构建一个密集向量，该向量应当能够准确的预测其上下文词项
- 递归的思路：那些上下文词项也由向量表示（并且也可以预测它们自己的上下文）

$$banking = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Neural （network） word embeddings （神经(网络)词嵌入）-图例



学习神经词嵌入：基本思路

- 定义一个向量表示的模型，该模型预测一个中心词 w_t 的 上下文词（或者反过来）
- $\mathrm{p}(\mathrm{context} \mid w_t) = \cdots$
- 同时也有一个损失函数，例如：
- $J = 1 - \mathrm{p}(w_{-t} \mid w_t)$
- 其中 t 是在一个大语料中词项出现的位置
- 不断调整（所有词的）向量表示使得损失最小化
- 最终得到每个词的低维密集向量表示

思路:直接基于预测能力学习低维词向量

- 老方法：基于（神经网络）回传误差的表示学习 (Rumelhart et al., 1986)
- 神经概率语言模型 A neural probabilistic language model (Bengio et al., 2003)
- NLP (almost) from Scratch (Collobert & Weston, 2008)
- 近期更简单、更快的方法: word2vec (Mikolov et al. 2013) → 本章内容
- Stanford GloVe 模型(Pennington, Socher, and Manning 2014) 利用矩阵分解
- 基于词条 (Per-token)的表示，深度上下文词表示（Deep contextual word representations）: ELMo, ULMfit, **BERT**

Non-linear and slow

Fast bilinear models

Current state of the art

Word2Vec包含一组算法

[Mikolov et al. 2013]

预测每个词的上下文（或者反过来）！

两种算法

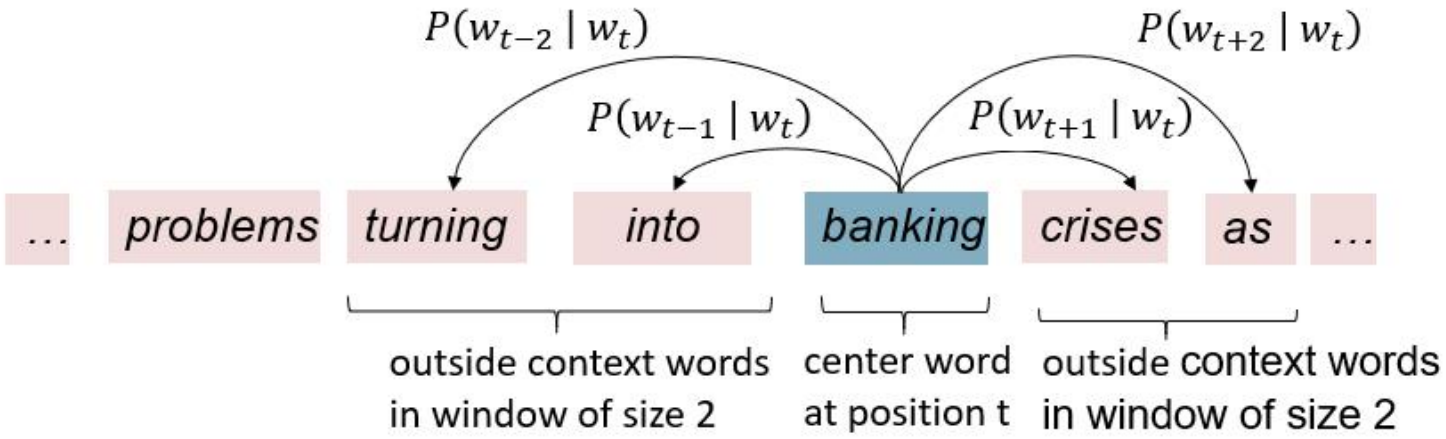
1. Skip-grams (SG)
   根据给定目标词预测上下文（位置独立）
2. Continuous Bag of Words (CBOW)
   根据（基于词袋模型的）上下文预测目标词

两个（中等效率的）训练方法

1. 层次 softmax
2. 负采样

Word2vec Skip-gram 示意图

- 计算概率$P(w_{t+j}/w_t)$的例子
- 给定出现在位置t的中心词 banking，预测上下文概率
- 上下文的范围：window size=2



Word2vec：目标函数

- 对于位置 $t=1,2,\ldots,T$，中心词 w_j，预测固定大小为 m 的文本窗内的上下文词

Likelihood=

$$L(\theta)=\prod_{t=1}^{T}\prod_{\substack{-m\le j\le m\\ j\neq 0}}P(w_{t+j}\mid w_t;\theta)$$

θ：所有要优化的参数

有时也称为成本(cost)或损失(loss)函数

- 目标函数 J(θ)为(平均)负对数似然：

$$J(\theta)=-\frac{1}{T}\log L(\theta)=-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}}\log P(w_{t+j}\mid w_t;\theta)$$

最小化目标函数 <=> 最大化预测精度

Word2vec：目标函数

- 我们希望最小化目标函数

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

- 问题：如何计算$P(w_{t+j}/w_t;\theta)$?
- 答案：对于每个词使用两个向量：
  - v_w：当w被当作中心词
  - u_w：当w被当作上下文词
- 那么对于中心词c与上下文词o：

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

Word2vec：预测函数

指数函数使变量为正

点积计算o与c的相似度.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
更大的点积 = 更大的概率

在整个词汇表上归一，将不同词之间相似度转化为概率分布

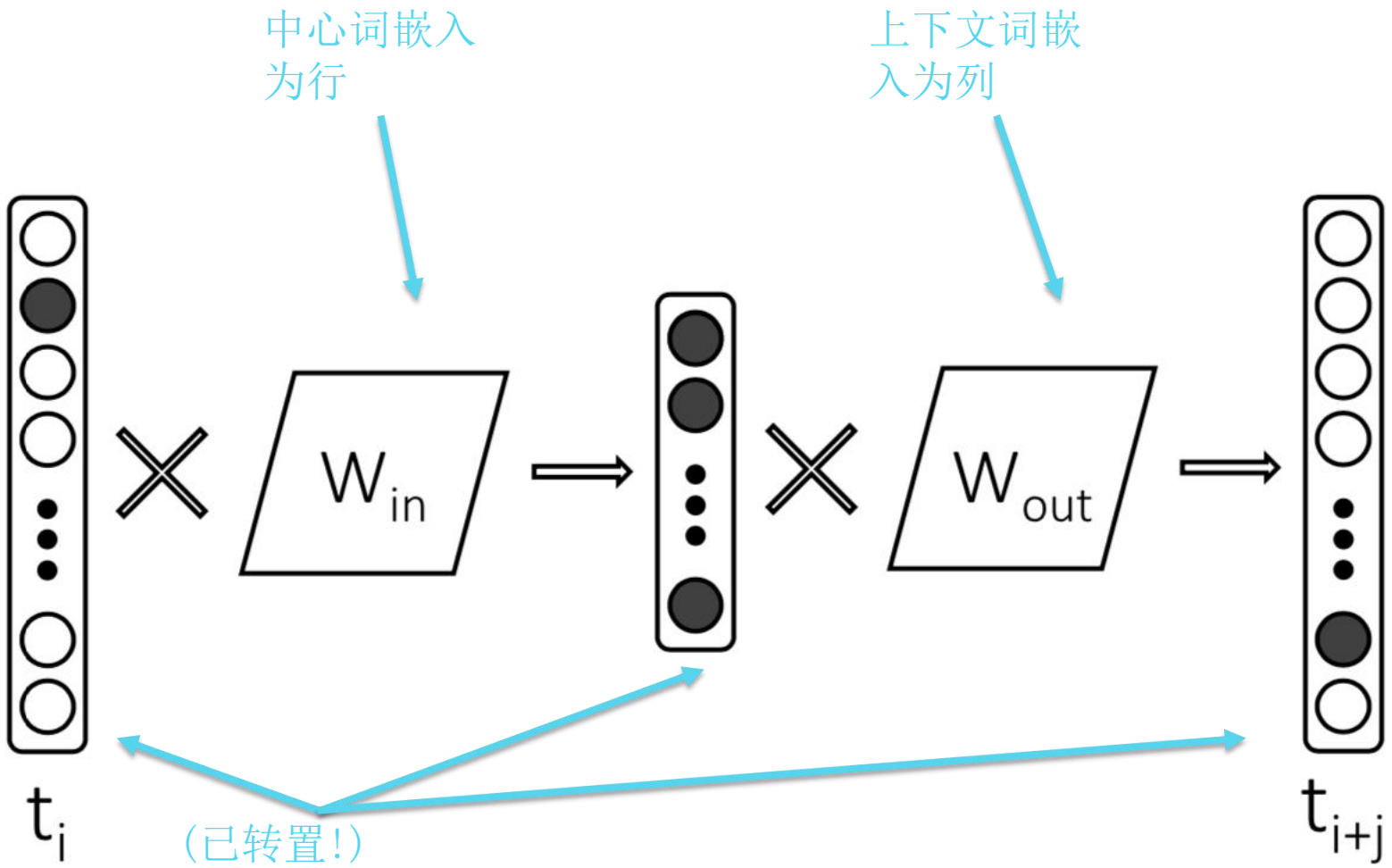
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- 这是一个 **softmax** 函数的例子， $\mathbb{R}^n \to (0,1)^n$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- Softmax函数将任意值x_i映射到概率分布 p_i
  - “max”：对于最大的x_i放大概率
  - “soft”：对于较小的x_i仍然赋一定的概率
  - 在神经网络/深度学习中常用

Word2vec：两个参数矩阵



学习过程：计算所有向量梯度！

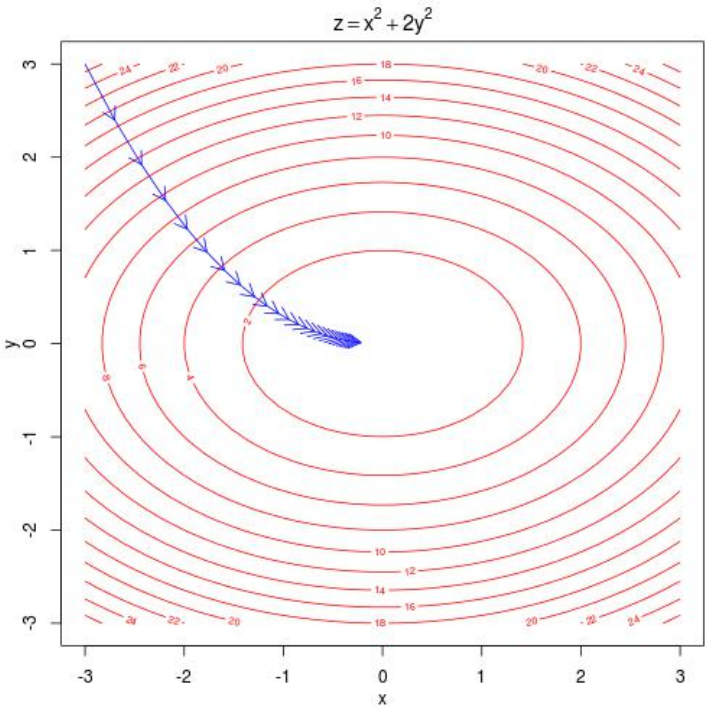
- 我们通常使用一个长向量 θ 定义模型中所有参数的集合
- 也就是 d 维 向量和 V 个词：
- 然后优化这些参数
  - 最小化目标函数

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

注意：每个词有两个向量！需要简化！

在两个参数集上使简单函数的损失最小化的思路

我们从一个随机点开始，然后沿着最陡峭的方向行走，该方向由函数的导数给出

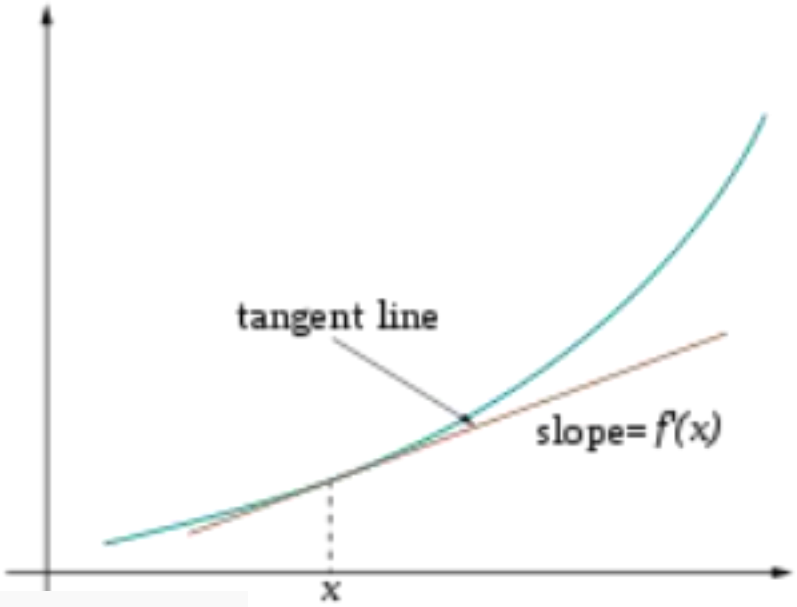


等高线显示目标函数的等值点

使用梯度下降

我们将使用梯度下降(gradient descent)
算法最小化损失函数

一个小例子：(from Wikipedia)
找到以下函数的局部最小值
$f(x) = x^4 - 3x^3 + 2$,
导数： $f'(x) = 4x^3 - 9x^2$




```
x_old = 0
x_new = 6 # The algorithm starts at x=6
eps = 0.01 # step size
precision = 0.00001

def f_derivative(x):
    return 4 * x**3 - 9 * x**2

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new = x_old - eps * f_derivative(x_old)

print("Local minimum occurs at", x_new)
```

减去梯度的一小部分
使函数迈向最小值！

梯度更新

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

Stochastic Gradient Descent（SGD，随机梯度下降）

- 语料可能会有多达400亿的词条和文本窗
- 可能需要很长时间进行一次参数更新
- 对于所有的神经网络来说都是一个非常糟糕的思路！
- 替代思路：每 t 个窗口更新一次参数
  → Stochastic gradient descent (SGD)

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J_t(\theta)$$

```
while True:
    window = sample_window(corpus)
    theta_grad = evaluate_gradient(J,window,theta)
    theta = theta - alpha * theta_grad
```

神经网络的优化：（求导的）链式法则

链式法则： If $y = f(u)$ and $u = g(x)$, i.e. $y = f(g(x))$, then:

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = \frac{df(u)}{du}\frac{dg(x)}{dx}$$

一个简单例子:

$$\frac{dy}{dx} = \frac{d}{dx}5(x^3+7)^4$$

$$y = f(u) = 5u^4 \qquad u = g(x) = x^3 + 7$$

$$\frac{dy}{du} = 20u^3 \qquad \frac{du}{dx} = 3x^2$$

$$\frac{dy}{dx} = 20(x^3+7)^3.3x^2$$

目标函数

$$\text{maximize } J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p(w'_{t+j} \mid w_t; \theta)$$

$$\text{or minimize } J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w'_{t+j} \mid w_t)$$

负对数似然【取负从而最小化；单调对数函数】

文本长度

窗口大小

$$\text{where } p(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

word IDs

每个词汇类型（词汇条目）有两个词表示：作为中心词和上下文词

求导数以得到最小值

$$\frac{\partial}{\partial v_c}\log\frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V}\exp(u_w^T v_c)}$$

$$=\underbrace{\frac{\partial}{\partial v_c}\log\exp(u_o^T v_c)}_{①}-\underbrace{\frac{\partial}{\partial v_c}\log\sum_{w=1}^{V}\exp(u_w^T v_c)}_{②}$$

① $$\frac{\partial}{\partial v_c}\log\exp(u_o^T v_c)=\frac{\partial}{\partial v_c}u_o^T v_c=u_o$$

当计算复杂时，可以一次只考虑一个变量

互为反函数

向量，而非单变量微分

$$\forall j\ \frac{\partial}{\partial (v_c)_j}u_o^T v_c=\frac{\partial}{\partial (v_c)_j}\sum_{i=1}^{d}(u_o)_i(v_c)_i=(u_o)_i$$

除去i=j时，其他项均为0

$$\frac{\partial}{\partial v_c}\underbrace{\log}_{f}\underbrace{\sum_{w=1}^{V}\exp(u_w^T v_c)}_{z=g(v_c)}$$

$$=\frac{1}{\underbrace{\sum_{w=1}^{V}\exp(u_w^T v_c)}_{z}}\cdot\underbrace{\frac{\partial}{\partial v_c}\sum_{x=1}^{V}\exp(u_x^T v_c)}_{\frac{\partial z}{\partial v_c}}$$

重要：更改索引

使用链式法则

$\frac{\partial}{\partial v_c} f(\overbrace{g(v_c)}^{z}) = \frac{\partial f}{\partial z}$

$$=\frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\cdot\underbrace{\left(\sum_{x=1}^{V}\frac{\partial}{\partial v_c}\underbrace{\exp}_{f}\left(\underbrace{u_x^T v_c}_{z=g(v_c)}\right)\right)}$$

移动微分符号到求和符号内

链式法则

$$\left(\sum_{x=1}^{V}\exp(u_x^T v_c)\frac{\partial}{\partial v_c}u_x^T v_c\right)\rightarrow\left(\sum_{x=1}^{V}\exp(u_x^T v_c)u_x\right)$$

$$\frac{\partial}{\partial v_c}\log(\mathrm{p}(\mathrm{o}|c)) = u_o - \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)} \bullet \left(\sum_{x=1}^{V}\exp(u_x^T v_c)u_x\right)$$

$$= u_o - \sum_{x=1}^{V}\frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V}\exp(u_w^T v_c)}u_x$$

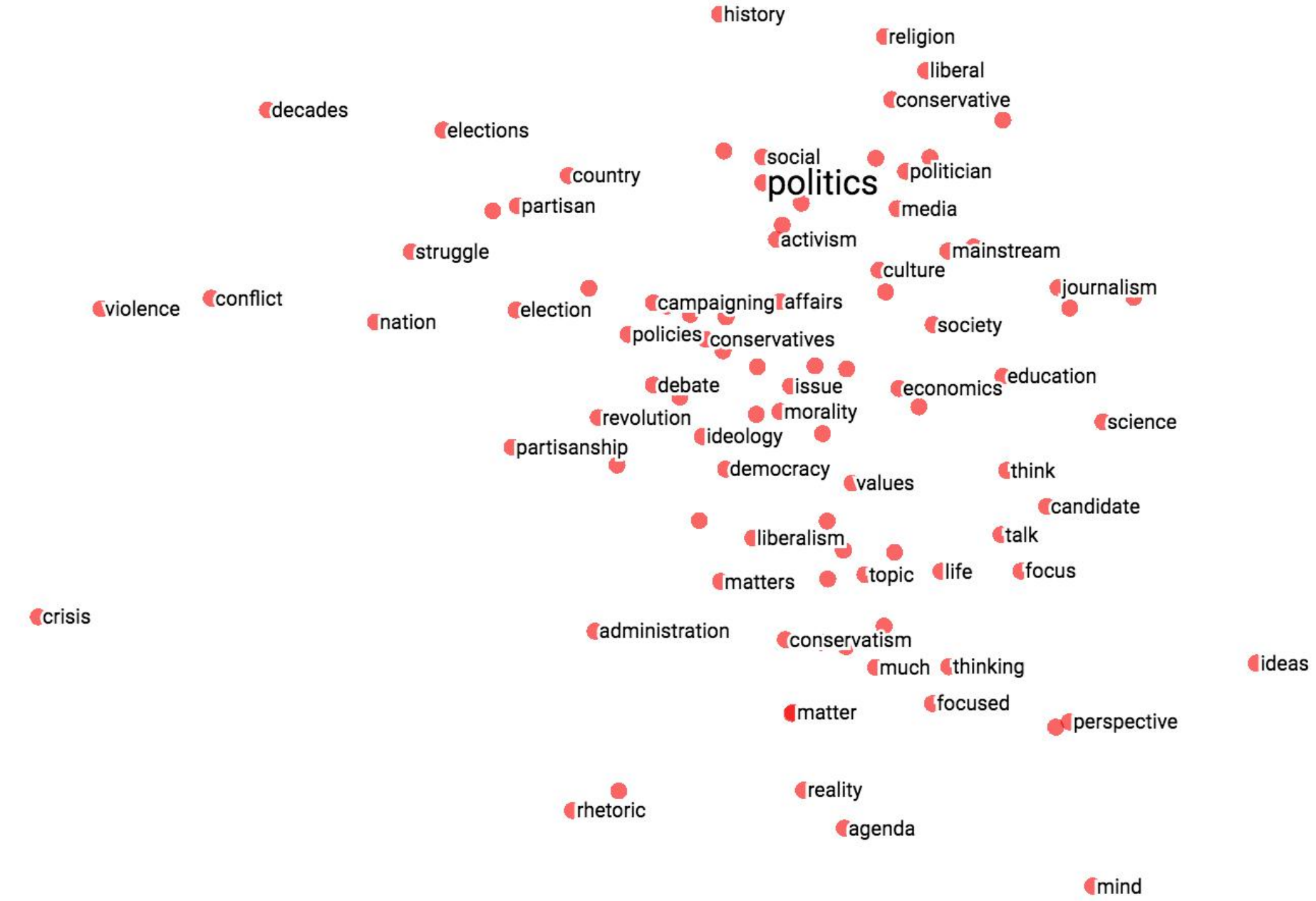
分配求和函数到每一项

$$= u_o - \sum_{x=1}^{V}\mathrm{p}(x|c)u_x$$

这是一个期望值：所有上下文向量的概率加权平均

$$= observed - \exp ected$$

以上只是中心向量参数的导数。
仍需要输出向量参数的导数（计算方法类似）。
这样我们就得到了对于所有参数的导数并且可以被最小化。


history
religion
liberal
conservative
decades
elections
social
politics
politician
country
partisan
media
activism
struggle
mainstream
culture
journalism
violence
conflict
election
campaigning
affairs
nation
society
policies
conservatives
debate
issue
economics
education
revolution
morality
science
ideology
partisanship
democracy
think
values
candidate
liberalism
talk
topic
life
focus
matters
crisis
administration
conservatism
much
thinking
ideas
matter
focused
perspective
reality
rhetoric
agenda
mind

Word2vec里的线性关系

Word2vec的向量表示*非常* 善于对 相似性 和 相似性的维度 编码!

- 仅通过在嵌入空间中进行向量减法就可以很好地解决类比测试相似度的问题
- 从句法上
  - $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
  - 对于动词和形容词形态也是如此
- 从语义上 (Semeval 2012 task 2)
  - $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$
  - $x_{king} - x_{man} \approx x_{queen} - x_{woman}$

词汇类比

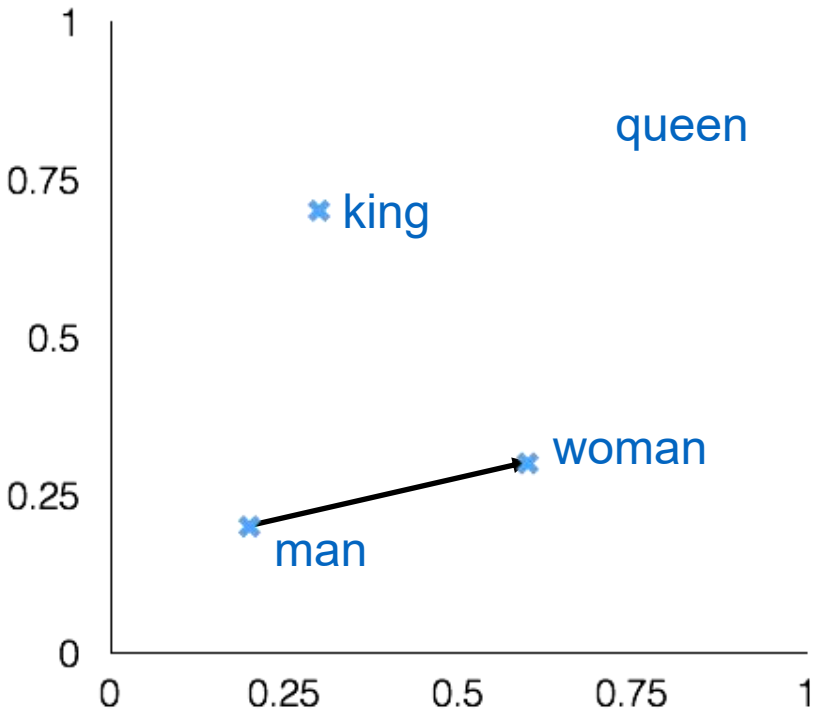
线性关系检验, examined by Mikolov et al.

a:b :: c:?

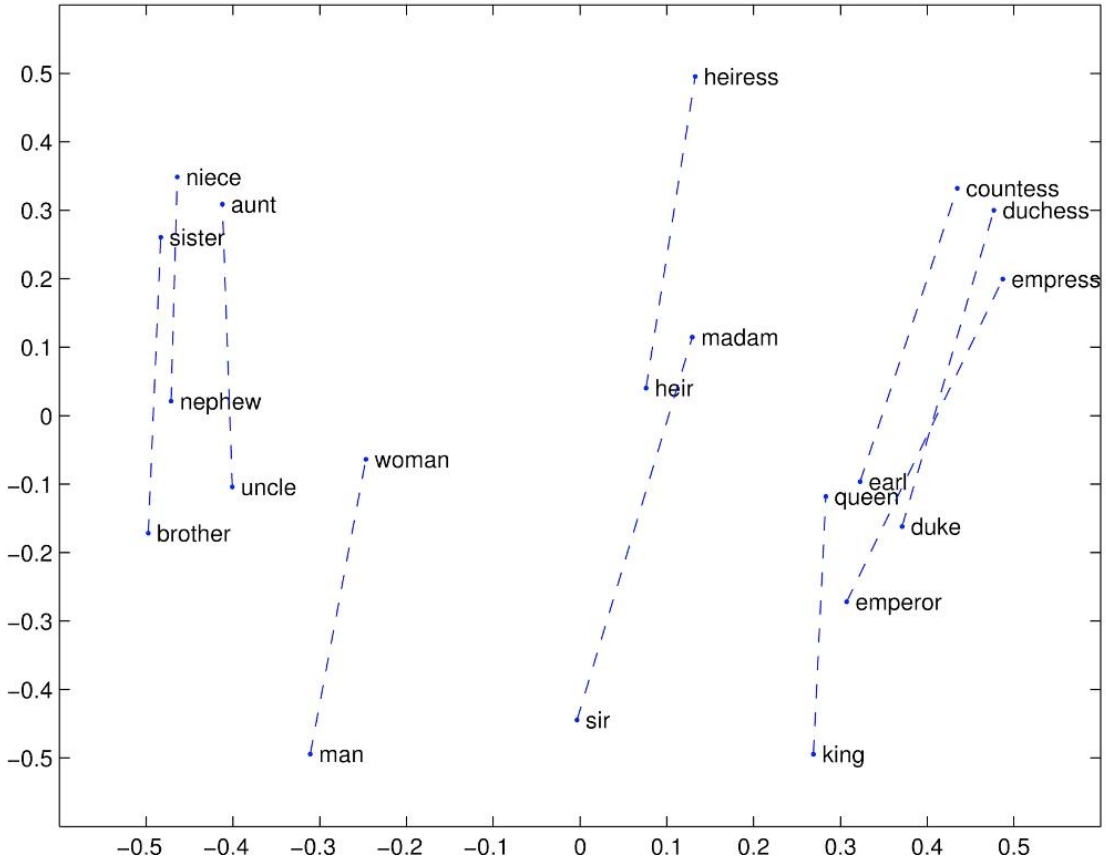
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
−	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

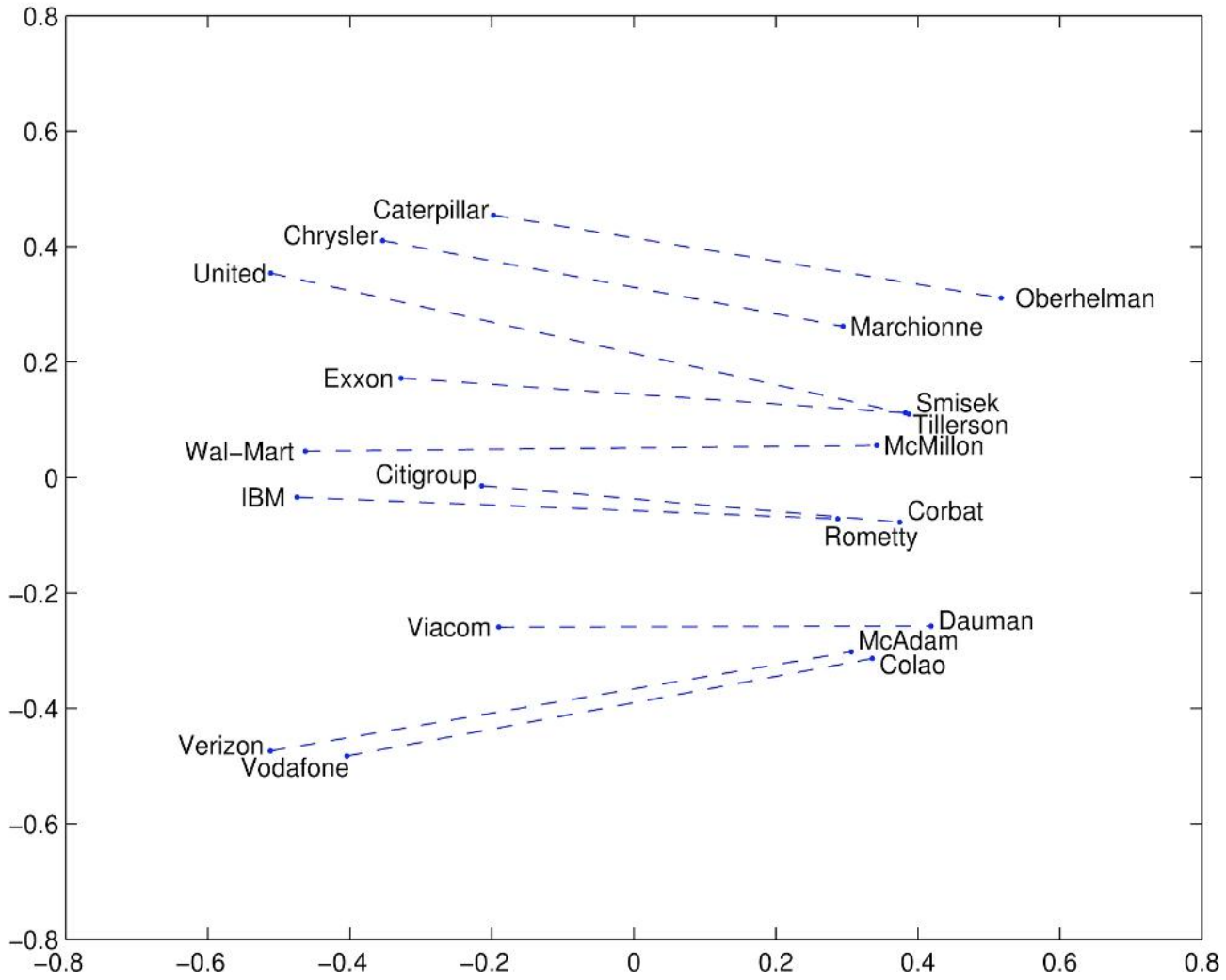


GloVe 可视化

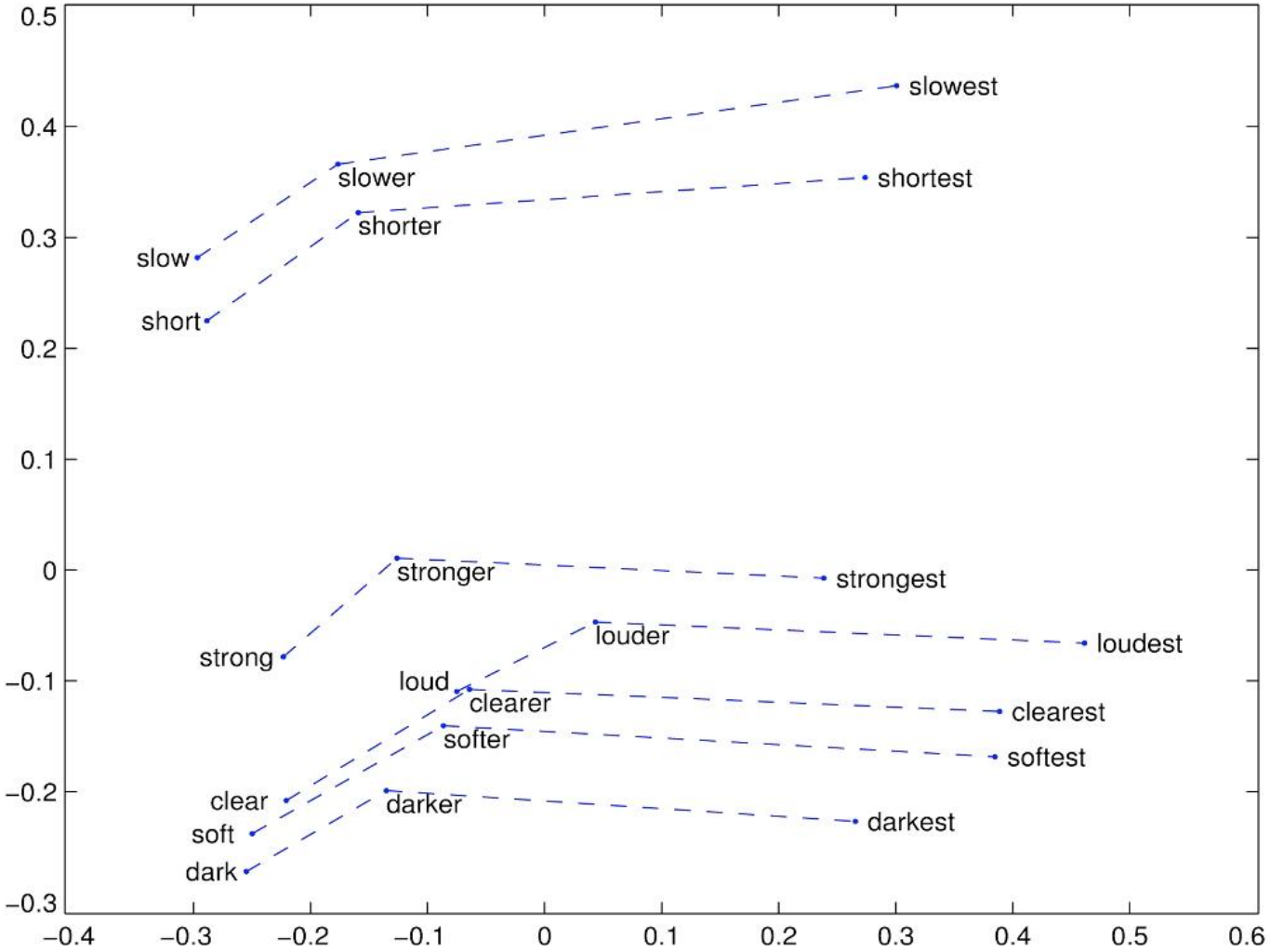


http://nlp.stanford.edu/projects/glove/

Glove 可视化: Company - CEO



Glove 可视化：比较级



信息检索中的应用

相关研究和应用方兴未艾

- Google’s RankBrain – 仅有少量信息为公众所知
  - Bloomberg article by Jack Clark (Oct 26, 2015): http://www.bloomberg.com/news/articles/2015-10-26/google-turning-its-lucrative-web-search-over-to-ai-machines
  - 一个搜索结果重排系统. “排名第三的最有价值的排序信号”
  - 但是需要注意：对于长尾（罕见）查询更有价值?
- New SIGIR Neu-IR workshop series (2016 on)

信息检索应用

Nalisnick, Mitra, Craswell & Caruana. 2016. Improving Document Ranking with Dual Word Embeddings. *WWW 2016 Companion.* http://research.microsoft.com/pubs/260867/pp1291-Nalisnick.pdf

Mitra, Nalisnick, Craswell & Caruana. 2016. A Dual Embedding Space Model for Document Ranking. arXiv:1602.01137 [cs.IR]

基于BM25的“aboutness”（“关联性”）思想构建

- 关联性并非仅仅通过词频体现
- 查询词与文档中*所有*词项的关联性都体现关联性(BM25仅使用查询词，不考虑文档中其它词项)

关联性的计算基于词项的上下文向量，word2vec的CBOW/SGNS (skip-gram with negative sampling)或GloVe

文档“关联性”建模：Albuquerque（新墨西哥州的一个城市）的搜索结果

d_1

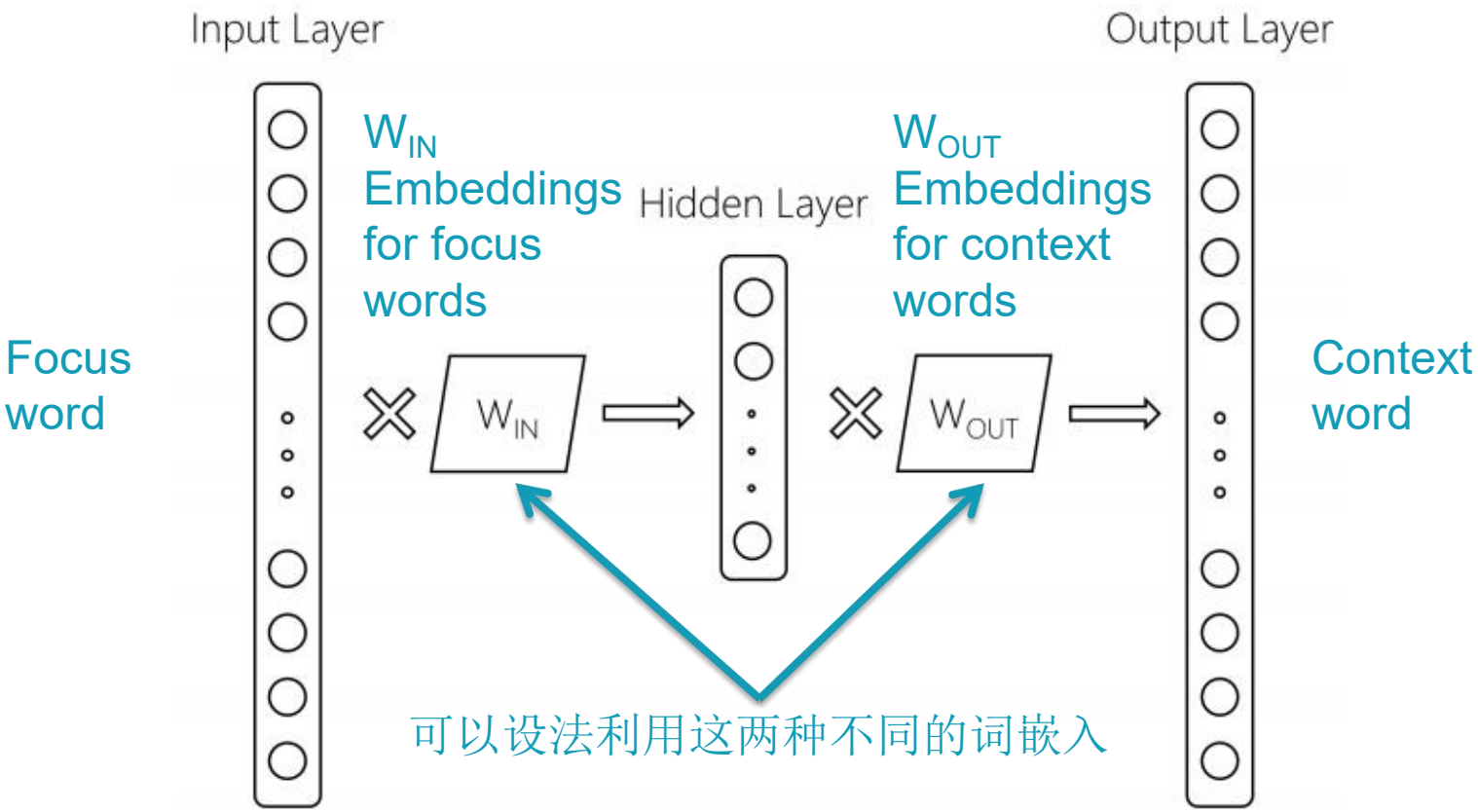
Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in Albuquerque, New Mexico in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

d_2

Albuquerque is the most populous city in the U.S. state of New Mexico. The high-altitude city serves as the county seat of Bernalillo County, and it is situated in the central part of the state, straddling the Rio Grande. The city population is 557,169 as of the July 1, 2014, population estimate from the United States Census Bureau, and ranks as the 32nd-largest city in the U.S. The Metropolitan Statistical Area (or MSA) has a population of 902,797 according to the United States Census Bureau's most recently available estimate for July 1, 2013.

使用两种词嵌入

word2vec 模型



使用两种词嵌入得到的相似词

yale		seahawks	
IN-IN	IN-OUT	IN-IN	IN-OUT
yale	yale	seahawks	seahawks
harvard	faculty	49ers	highlights
nyu	alumni	broncos	jerseys
cornell	orientation	packers	tshirts
tulane	haven	nfl	seattle
tufts	graduate	steelers	hats

Dual Embedding Space Model (DESM)

- 一种简单的利用词嵌入的检索模型
- 文档由其词项嵌入的中心向量表示

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$$

- 查询-文档相似度：查询词向量与文档向量的平均相似度

$$DESM(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{\mathbf{q}_i^T \overline{\mathbf{D}}}{\|\mathbf{q}_i\| \|\overline{\mathbf{D}}\|}$$

Dual Embedding Space Model (DESM)

- 经验表明当使用OUT向量表示文档、IN向量表示查询，结果最好

$$DESM_{IN-OUT}(Q,D) = \frac{1}{|Q|}\sum_{q_i \in Q}\frac{q_{IN,i}^{T}\overline{D_{OUT}}}{\|q_{IN,i}\|\|\overline{D_{OUT}}\|}$$

- 这种相似度（某种程度上）度量*关联性（aboutness）*：与Focus Word相邻的词处于其上下文中，这种表示比分布式语义表示更有用
  - 注：以上仅为一家之言

实验

- 使用以下任一语料训练 word2vec
  - 6亿 Bing 查询
  - 3.42 亿 web 句子
- IR实验数据： 7,741 个随机采样的 Bing 查询
  - 5 级相关性标记 (Perfect, Excellent, Good, Fair, Bad)
- 两种方法
  1. 使用 DESM 重排 BM25 返回的排名靠前的文档
  2. 仅使用 DESM ，或使用 DESM 与 BM25 的插值排序

$$MM(Q,D) = \alpha DESM(Q,D) + (1-\alpha)BM25(Q,D)$$
$$\alpha \in \mathbb{R}, 0 \leq \alpha \leq 1$$

重排前K文档实验结果

	Explicitly Judged Test Set		
	NDCG@1	NDCG@3	NDCG@10
BM25	23.69	29.14	44.77
LSA	22.41*	28.25*	44.24*
DESM (IN-IN, trained on body text)	23.59	29.59	45.51*
DESM (IN-IN, trained on queries)	23.75	29.72	46.36*
DESM (IN-OUT, trained on body text)	24.06	30.32*	46.57*
DESM (IN-OUT, trained on queries)	**25.02***	**31.14***	**47.89***

还不错的提升 - 例如 2% （绝对提升） 的 NDCG@3 提升

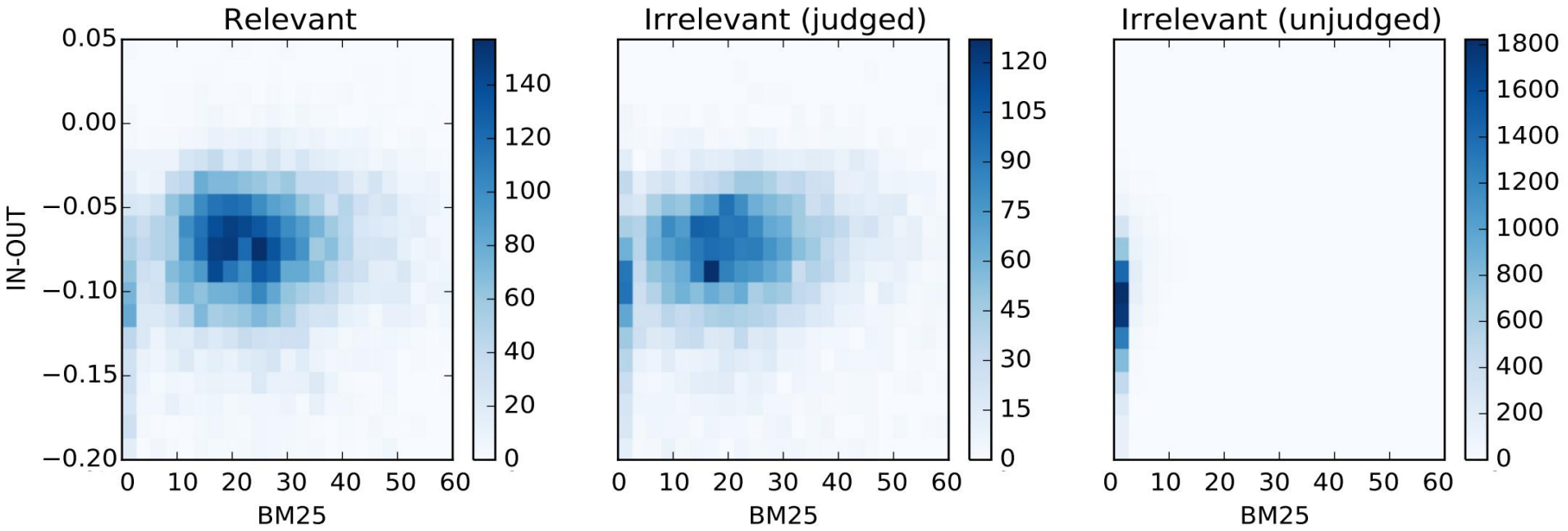
基于查询训练比基于文档训练能得到更大的提升

给全部文档的排序的实验结果

	Explicitly Judged Test Set		
	NDCG@1	NDCG@3	NDCG@10
BM25	21.44	26.09	37.53
LSA	04.61*	04.63*	04.83*
DESM (IN-IN, trained on body text)	06.69*	06.80*	07.39*
DESM (IN-IN, trained on queries)	05.56*	05.59*	06.03*
DESM (IN-OUT, trained on body text)	01.01*	01.16*	01.58*
DESM (IN-OUT, trained on queries)	00.62*	00.58*	00.81*
BM25 + DESM (IN-IN, trained on body text)	21.53	26.16	37.48
BM25 + DESM (IN-IN, trained on queries)	**21.58**	26.20	37.62
BM25 + DESM (IN-OUT, trained on body text)	21.47	26.18	37.55
BM25 + DESM (IN-OUT, trained on queries)	21.54	**26.42***	**37.86***

单独使用DESM或LSA的结果很差，但是DESM与BM25插值可以得到不错的结果

一个可能的解释



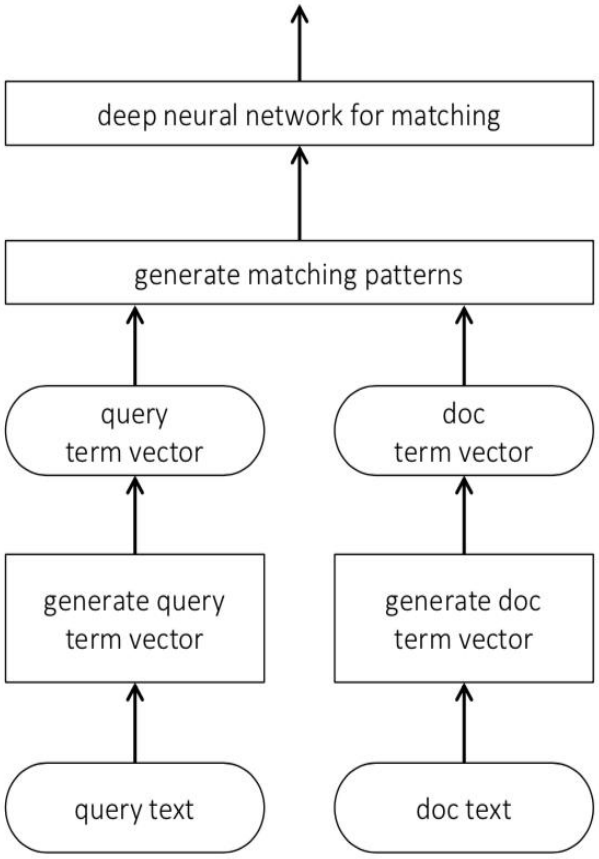
IN-OUT 方法具有一定的提升相关文档、降低不相关文档排名的能力，但是它的评分引入了太多的噪声，所以不能单独使用该评分给所有文档排序

DESM：小结

- DESM 是一个弱排序模型，但是具有发现微妙相似性/关联性的能力
- DESM仅在对一定程度相关的文档列表重排时有效
  - 例如，DESM 会认为 Oxford 与 Cambridge
    - 两者语义上相似，但并不相关
  - 再例如，对于查询“Chinese Food”，DESM会认为关于“Brazilian Cuisine”的文档高度相关
    - 语义相似并不意味着内容相关

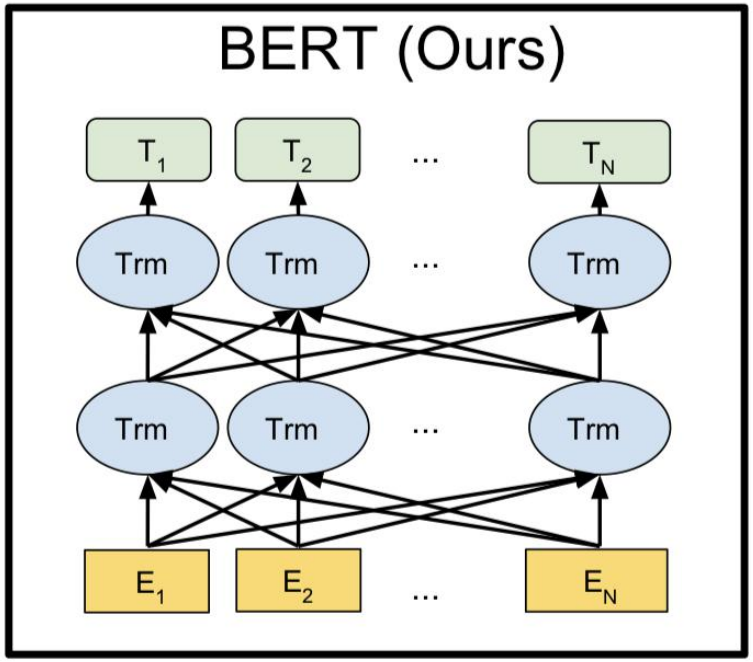
神经网络在IR中的其它应用

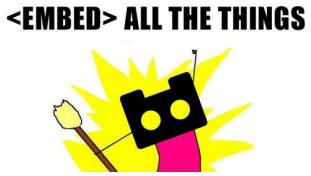
- 将神经网络用作一个有监督排序模型
- 假设一个文档与查询嵌入网络（如前所述）
- 假设有（q, d, rel）相关性标记数据
- （有监督的）学习一个预测(q, d)对相关性的神经网络
- 一个“机器学习相关性”的例子。下一讲将涉及到



神经网络在IR中的其它应用

- BERT: Devlin, Chang, Lee, Toutanova (2018)
- 一个基于Transformer的深度神经网络
- 依据上下文构建逐词条表示
- 也能产生查询/文档表示
- 或（以某种方式）拼接查询与文档的嵌入，由BERT输给相关性评分
- 难以置信的有效!
- https://arxiv.org/abs/1810.04805





小节：对一切构建嵌入!

词嵌入是一种热门新技术

在许多面向上下文或相似度的应用中非常有效：

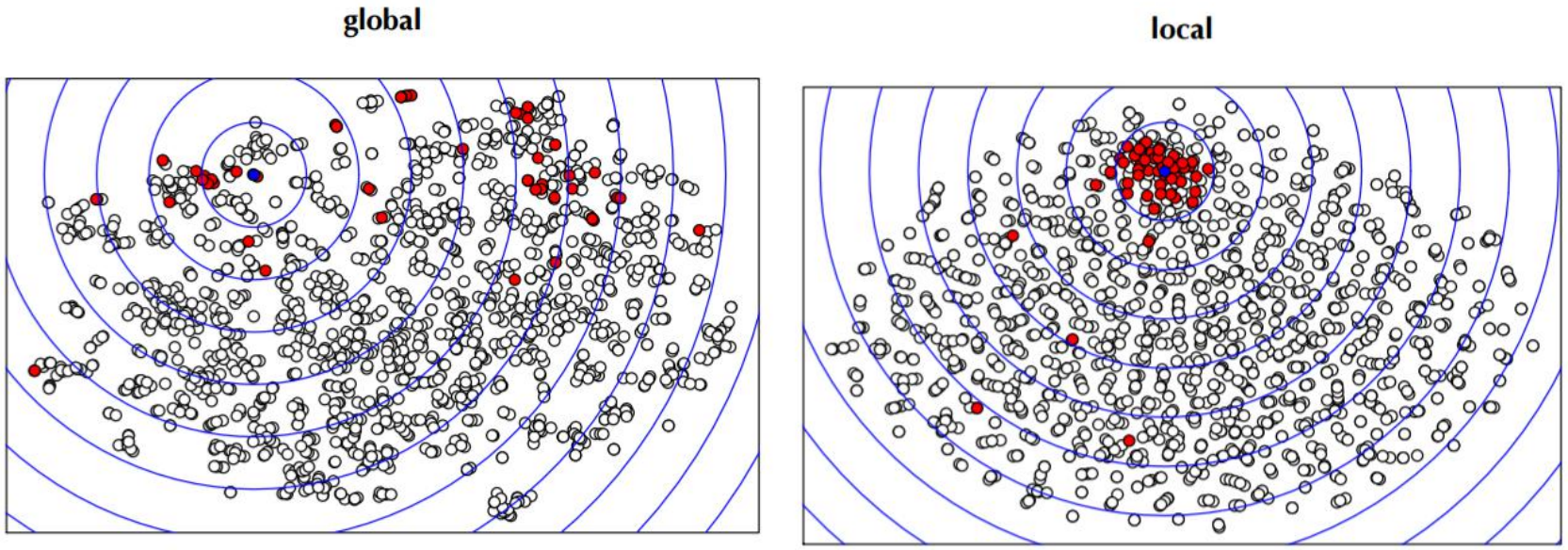
- 搜索中的同义词处理
- 文档“关联性”
- 广告推荐
- 语言模型：从拼写矫正到email应答
- Language models: from spelling correction to email response
- 机器翻译
- 情感分析
- …

全局 vs. 局部嵌入 [Diaz 2016]

global	local
cutting	tax
squeeze	deficit
reduce	vote
slash	budget
reduction	reduction
spend	house
lower	bill
halve	plan
soften	spend
freeze	billion

Figure 3: Terms similar to 'cut' for a word2vec model trained on a general news corpus and another trained only on documents related to 'gasoline tax'.

全局 vs. 局部嵌入 [Diaz 2016]



使用第一轮检索返回的文档训练 word2vec

细粒度的词义消歧

Figure 5: Global versus local embedding of highly relevant terms. Each point represents a candidate expansion term. Red points have high frequency in the relevant set of documents. White points have low or no frequency in the relevant set of documents. The blue point represents the query. Contours indicate distance from the query.

使用局部和分布式表示的Ad-hoc检索

[Mitra et al. 2017]

- 作者认为 “词汇上的（lexical）” 和“语义上的（semantic）” 匹配对于文档排序都重要
- Duet模型是两个DNN的线性组合，使用查询/文档的局部和分布式表示作为输入，并在标记数据上进行联合训练