

# 一文总结词向量的计算、评估与优化

原创 芙蕖 Datawhale 2020-07-07

↑↑↑关注后"星标"Datawhale  
每日干货 & 每月组队学习, 不错过

---

Datawhale干货

---

作者: 芙蕖, Datawhale优秀学习者, 东北石油大学

---

为了处理语言, 需要将文本信息用向量的形式表达。词向量 (Word Vector) 或称为词嵌入 (Word Embedding) 就是将词语向量化。常见的生成词向量的神经网络模型有NNLM模型,C&W模型,CBOW模型和Skip-gram模型。

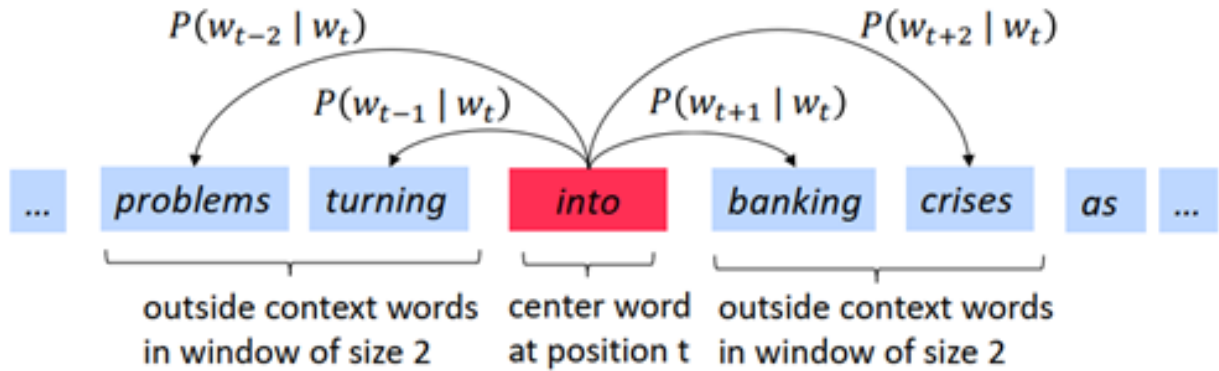
本文目录:

1. 词向量计算方法
  - 1.1 Word2Vec的计算
  - 1.2 Word2Vec中计算方法详解
  - 1.3 高频词 (the) 引起的问题
2. 优化基础
  - 2.1 梯度下降
  - 2.2 随机梯度下降
3. Word Vector优化过程
  - 3.1 SGD引起的稀疏数据
  - 3.2 两种词向量建模方案
  - 3.3 训练效率提升方案
4. 基于统计的单词向量表示
  - 4.1 共现矩阵
  - 4.2 改进思路
5. GloVe模型
  - 5.1 原理
  - 5.2 与Skip-Gram、CBOW模型比较
  - 5.3 步骤
  - 5.4 如何评估词向量的质量

## 一、词向量计算方法

### 1.1 word2vec的计算

对一个中心词, 与窗口内的context词出现的概率:



- $$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$
- Update vectors so you can predict well

通过极大似然方法最大化整个文本出现的概率：

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

损失函数：

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

## 1.2 word2vec中计算方法详解

假设vocabulary包含m个词，每个词向量长度为n，对于每一个词，作为中心词(center)和非中心词(outside)时分别使用v和u两个向量表示。在计算完成后将两个向量平均作为最终词向量表示。

$$U_{m \times n}(\text{outside}) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \quad V_{m \times n}(\text{center}) = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

对每一个词作为中心词时，计算概率分布。这里假定第4个词作为中心词时，有

$$D_{m \times 1} = U_{m \times n} \cdot v_4^T = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix}$$

其中，d为与m个outside词的点积，由于两个向量的点乘可以表示其相似度，进一步可用于表示其出现的概率大小，从而得到概率表示：

$$P_{m \times 1} = \text{softmax}(D_{m \times 1}) = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix}$$

这里原理就很明显了，我们接下来需要做的，就是通过优化问题来更新矩阵U和V，从而使词向量模型需对出现在同一个context中的词赋予较大的概率。

### 1.3 高频词(the)引起的问题

通过以上计算过程可以知道，如果两个词出现在一个context的次数越频繁，那么他们的词向量就会越接近，这样一来像the这样的高频词，就会使它前后的词向量高度集中，从而导致一些问题。

## 二、优化基础

### 2.1 梯度下降

梯度是指多元函数在某个点上升最快的方向，那么梯度的反方向当然就是下降最快的方向。从而得到直观的优化公式：

Update equation (in matrix notation):

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

↑  
α = step size or learning rate

此处 $\nabla_{\theta} J(\theta)$ 为损失函数的梯度， $\alpha$ 为学习率或步长，是一个超参数。以上是对整个问题的矩阵表示，但在计算过程中，需要一个个的更新参数，所以有对单个参数 $\theta_j$ 表示版本：

Update equation (for a single parameter):

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

在高等数学（同济）中关于梯度的定义如下，及梯度是各个自变量的偏导组成的向量。

**定义** 设  $z = f(x, y)$  在平面区域  $D$  内具有一阶连续偏导数，则对于每一点  $(x, y) \in D$ ，  
 $\left( \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right)$  称为  $z = f(x, y)$  在点  $(x, y)$  处的**梯度**，记作 **grad** $z$ ，即 **grad** $z = \left( \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right)$ 。

### 2.2 随机(stochastic)梯度下降 (SGD)

在2.1中提到的梯度下降，为了计算出参数的梯度，需要代入整个数据集，这样一次更新计算量非常大，因此提出随机梯度下降方法，即每一个更新都是从数据及中随机抽样部分数据 (batch)，在词向量计算中对每一个window数据计算一次更新。

- 每次只使用一个window来更新；
- 在一个window中，至多只有 $2m+1$ 个词，所以梯度很稀疏（下图示例为：center word: like, context words: I, learning等）。

- Iteratively take gradients at each such window for SGD
- But in each window, we only have at most  $2m + 1$  words, so  $\nabla_{\theta} J_t(\theta)$  is very sparse!

$$\nabla_{\theta} J_t(\theta) = \begin{bmatrix} 0 \\ \vdots \\ \nabla_{v_{like}} \\ \vdots \\ 0 \\ \nabla_{u_I} \\ \vdots \\ \nabla_{u_{learning}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2dV}$$

q

我们或许只能更新实际出现过的词的词向量

- 解决方法：要么使用稀疏矩阵只更新U和V的特定的行，或者对每个词向量使用hash；
- 若词向量数量很多，并且要做分布式计算，最好不要进行巨大的更新。

### 带有负采样的Skip-grams (HW2)

1) softmax的正则化因子（分母）计算消耗巨大

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

2) 带有负采样(negative sampling)的Skip-grams:

训练一对真词（上下文窗口中的中心词和单词）与几个噪声对（中心词和随机词）的二元逻辑回归（在标准的word2vec和HW2中都使用了负采样）

3) 需要最大化目标函数

- From paper: “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al. 2013)
- Overall objective function (they maximize):  $J(\theta) = \frac{1}{T} \sum_{t=1}^T J_t(\theta)$

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^k \mathbb{E}_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

4) 使用与HW2更类似的符号表示

Notation more similar to class and HW2:

$$J_{neg-sample}(\mathbf{o}, \mathbf{v}_c, \mathbf{U}) = -\log(\sigma(\mathbf{u}_o^\top \mathbf{v}_c)) - \sum_{k=1}^K \log(\sigma(-\mathbf{u}_k^\top \mathbf{v}_c))$$

选取了k个负样例 (选取概率为 $P(w)=U(w)^{3/4}/Z$ )

- the unigram distribution  $U(w)$  raised to the  $3/4$  power;
- The power makes less frequent words be sampled more often.

最大化真实上下文词出现在中心词的概率，最小化随机词出现在中心词的概率

### 三、word vector优化过程

#### 3.1 SGD引起的稀疏数据

由于使用一个窗口更新一次，由于 $\nabla_{\theta} J_t(\theta)$ 各个词向量的偏导组成的向量，如果这个词没有出现，其偏导也就为0，因此梯度将非常稀疏。

对应方案：使用稀疏矩阵或者将词hash映射到具体向量，如果是分布式计算，必须避免大量的中间数据在节点之间的传送

#### 3.2 两种词向量建模方案

- 1) Skip-gram(SG): 给定中心词预测窗口context(outsides)
- 2) Continuous Bag of Words(CBOW): 给定窗口context预测中心词

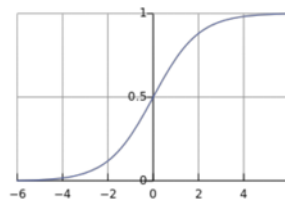
#### 3.3 训练效率提升方案

- 负采样。目前为止仍然以更简单但是计算量大的传统softmax为主要方案，即公式2.1中的分母（正则项）。
- 由于经典方案正则化计算量太大，因此我们在作业二中使用负采样方案。其主要思想为：训练一个logistics regression分类器，判断一个词语对是否来自于同一个context。
- 损失函数：最大化如下函数：

- Overall objective function (they maximize):  $J(\theta) = \frac{1}{T} \sum_{t=1}^T J_t(\theta)$

$$J_t(\theta) = \log \sigma(\mathbf{u}_o^\top \mathbf{v}_c) + \sum_{i=1}^k \mathbb{E}_{j \sim P(w)} [\log \sigma(-\mathbf{u}_j^\top \mathbf{v}_c)]$$

- The sigmoid function:  $\sigma(x) = \frac{1}{1+e^{-x}}$  (we'll become good friends soon)
- So we maximize the probability of two words co-occurring in first log  
→



### 四、基于统计的单词向量表示

## 4.1 共现矩阵 (co-occurrence matrix)

- 统计所有语料当中，任意两个单词出现在同一个窗口中的频率，结果表现为共现矩阵  $X$
- 直接统计得到的原始矩阵大小为  $|V| \times |V|$
- 实践当中通常对原始统计矩阵施加 SVD (Singular Value Decomposition) 来降低矩阵维度

Example corpus:

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

优点:

- 训练速度快
- 充分利用了全局的统计信息

缺点:

- 向量空间结构没有达到最优化，在单词相似度任务上表现不好
- 随着字典的扩充，共现矩阵的大小也会改变
- 矩阵维度十分巨大，需要大量的存储空间
- 共现矩阵十分稀疏，其中大部分区域都为0
- 十分依赖大型的语料进行训练

存在的问题:

- 随着词表的增加而增加
- 维度较高->需要大量存储空间
- 后续分类模型存在稀疏性问题
- 模型缺乏鲁棒性

解决方法:

使用较低纬度的向量

- 想法：将“大多数”重要信息存储在一个固定的、少量的维度中：一个密集的向量
- 通常为25—100维，与word2vec类似
- 如何减小维度，有以下两种方法：

### 1) 奇异值分解 (SVD)





## Count based vs. direct prediction

<ul style="list-style-type: none"> <li>• LSA, HAL (Lund &amp; Burgess),</li> <li>• COALS, Hellinger-PCA (Rohde et al, Lebrete &amp; Collobert)</li> </ul>	<ul style="list-style-type: none"> <li>• Skip-gram/CBOW (Mikolov et al)</li> <li>• NNLM, HLBL, RNN (Bengio et al; Collobert &amp; Weston; Huang et al; Mnih &amp; Hinton)</li> </ul>
<ul style="list-style-type: none"> <li>• Fast training</li> <li>• Efficient usage of statistics</li> <li>• Primarily used to capture word similarity</li> <li>• Disproportionate importance given to large counts</li> </ul>	<ul style="list-style-type: none"> <li>• Scales with corpus size</li> <li>• Inefficient usage of statistics</li> <li>• Generate improved performance on other tasks</li> <li>• Can capture complex patterns beyond word similarity</li> </ul>

- 左边是基于计数的方法的一些特点：训练快、有效利用了统计信息、初步统计了词的相似性
- 右边是基于预测的方法的一些特点：可以捕获超出单词相似度的复杂模式

### 4.2 改进思路

只使用一个大小固定且维度较少的稠密向量来存储最重要的信息。现在的问题是，如何才能有效地降低向量的维度呢？

重要信息：共现概率的比值能够编码单词相似度的信息

**Crucial insight:** Ratios of co-occurrence probabilities can encode meaning components

	$x = \text{solid}$	$x = \text{gas}$	$x = \text{water}$	$x = \text{fashion}$
$P(x \text{ice})$	$1.9 \times 10^{-4}$	$6.6 \times 10^{-5}$	$3.0 \times 10^{-3}$	$1.7 \times 10^{-5}$
$P(x \text{steam})$	$2.2 \times 10^{-5}$	$7.8 \times 10^{-4}$	$2.2 \times 10^{-3}$	$1.8 \times 10^{-5}$
$\frac{P(x \text{ice})}{P(x \text{steam})}$	8.9	$8.5 \times 10^{-2}$	1.36	0.96

从这个例子的最后一行中可以看出， $x$  与 ice 意思更加接近的话，概率比值远大于 1， $x$  与 steam 意思更加接近的话，概率比值远小于 1；如果  $x$  的意思既不与 ice 接近也不与 steam 接近，或者既与 ice 接近又与 steam 接近，那么概率比值在 1 附近。

## 五、GloVe模型

### 5.1 原理

**功能：**基于语料库构建词的共现矩阵，然后基于共现矩阵和GloVe模型对词汇进行向量化表示。



输入：语料库

输出：词向量

5.2 与Skip-Gram、CBOW模型比较

例如：句子为"dogbarked at the mailman"，目标单词为'at'

**Skip-gram模型**：Skip-gram模型只关注单个输入/输出元组中的目标词和上下文中的单个单词，输入为["dog"，"at"]

**CBOW模型**：关注目标单词和单个样本中上下文的所有单词，则输入为：  
[["dog","barked","the","mailman"],"at"]

因此，在给定数据集中，对于指定单词的上下文而言，CBOW比Skip-gram会获取更多的信息。Global Vector融合了矩阵分解的全局统计信息和上下文信息。

5.3 步骤

1) 构建共现矩阵

例如句子为：i love youbut you love him i am sad

包括7个单词：i、love、you、but、him、am、sad

设context= 5，则目标单词的左右长度都为2，以下为统计窗口：

注：中心词为目标单词，窗口内容为目标单词的左右各两个单词。

如：“i”左边无单词，右边有两个单词"love","you",所以窗口内容为["i","love","you"]

窗口标号	中心词	窗口内容
0	i	i love you
1	love	i love you but
2	you	i love you but you
3	but	love you but you love
4	you	you but you love him
5	love	but you love him i
6	him	you love him i am
7	i	love him i am sad
8	am	him i am sad
9	sad	i am sad

窗口0、1长度小于5是因为中心词左侧内容少于2个，同理窗口8、9长度也小于5。

以窗口5为例说明如何构造共现矩阵。中心词为love，语境词为but、you、him、i；则执行：

$$X_{love, but} + = 1$$

$$X_{love, you} + = 1$$

$$X_{love, him} + = 1$$

$$X_{love, i} + = 1$$

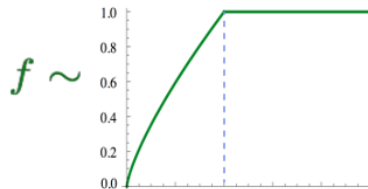
使用窗口将整个语料库遍历一遍，即可得到共现矩阵X。

- LSA和word2vec作为两大类方法的代表，一个是利用了全局特征的矩阵分解方法，一个是利用局部上下文的方法。
- GloVe模型将这两中特征合并到一起，即使用了语料库的全局统计（overall statistics）特征，也使用了局部的上下文特征（即滑动窗口）。为了做到这一点GloVe模型引入了Co-occurrence Probabilities Matrix。
- 目标函数如下：

$$w_i \cdot w_j = \log P(i|j)$$

$$J = \sum_{i,j=1}^V f(X_{ij}) (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors



## 5.4 如何评估词向量的质量

### 5.4.1 Intrinsic (内部评价)

- 在特定的子任务上对词向量进行评估（例如评估词向量时候可以正确预测词性标签，或者评估同义词是否具有相似的向量结构）
- 评估速度快，易于计算
- 能够帮助理解这个系统
- 除非与实际任务建立了关联，否则不清楚是否真正有用

#### (1) 词向量类比 (Word Vector Analogies)

Word Vector Analogies

a:b :: c:?  
man:woman :: king:?



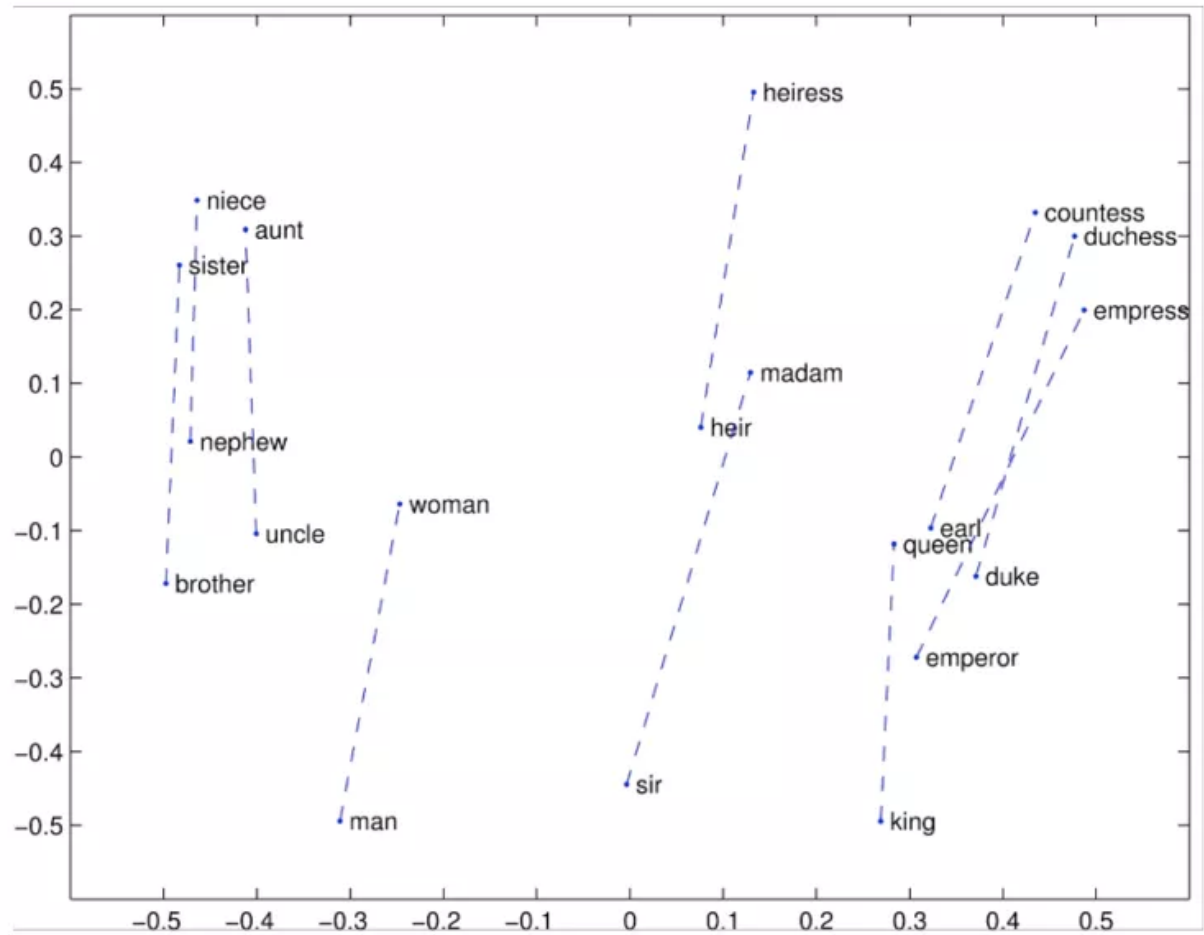
$$d = \arg \max_i \frac{(x_b - x_a + x_c)^T x_i}{\|x_b - x_a + x_c\|}$$

通过捕获直观的语义和句法类比问题之后的余弦距离来评价词向量

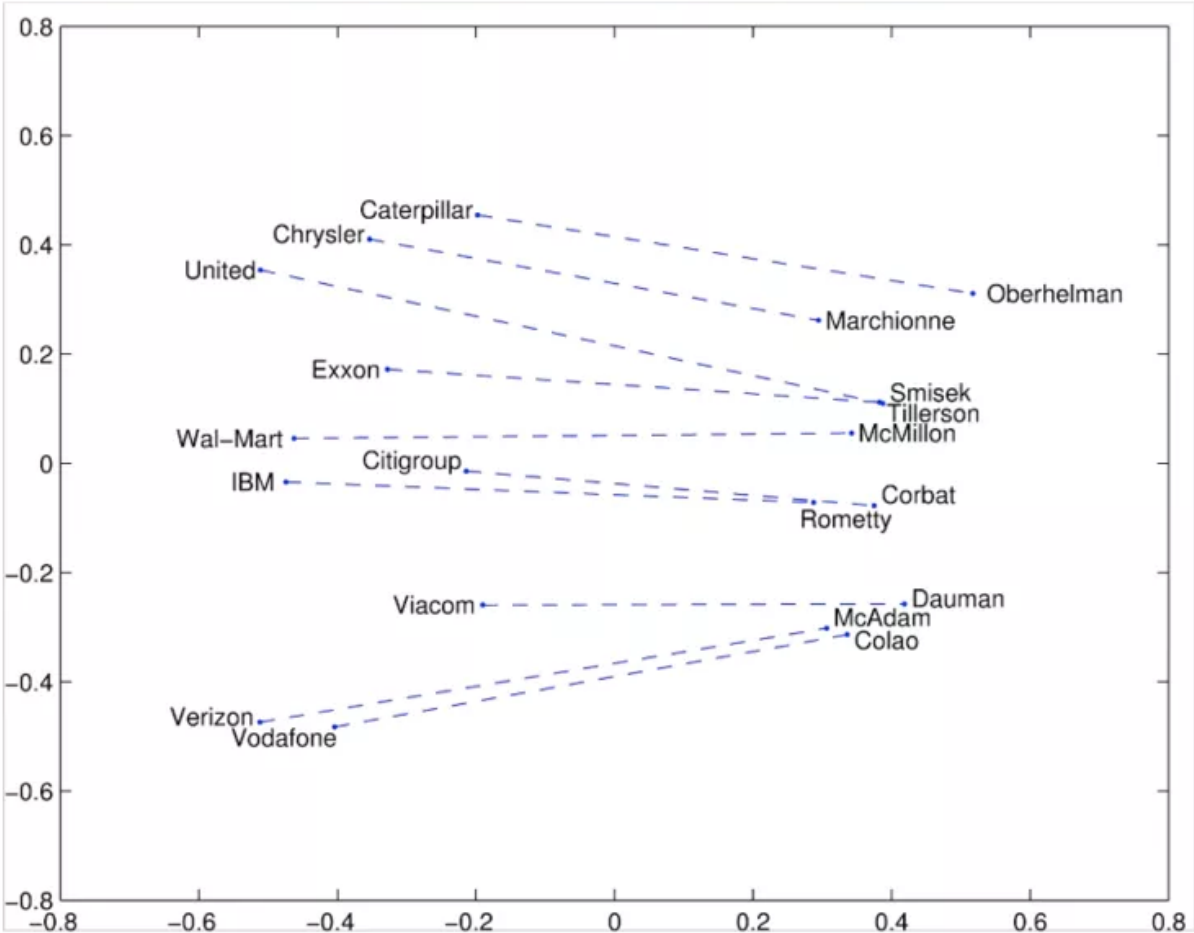
问题：如果信息不是线性的？

GloVe的可视化

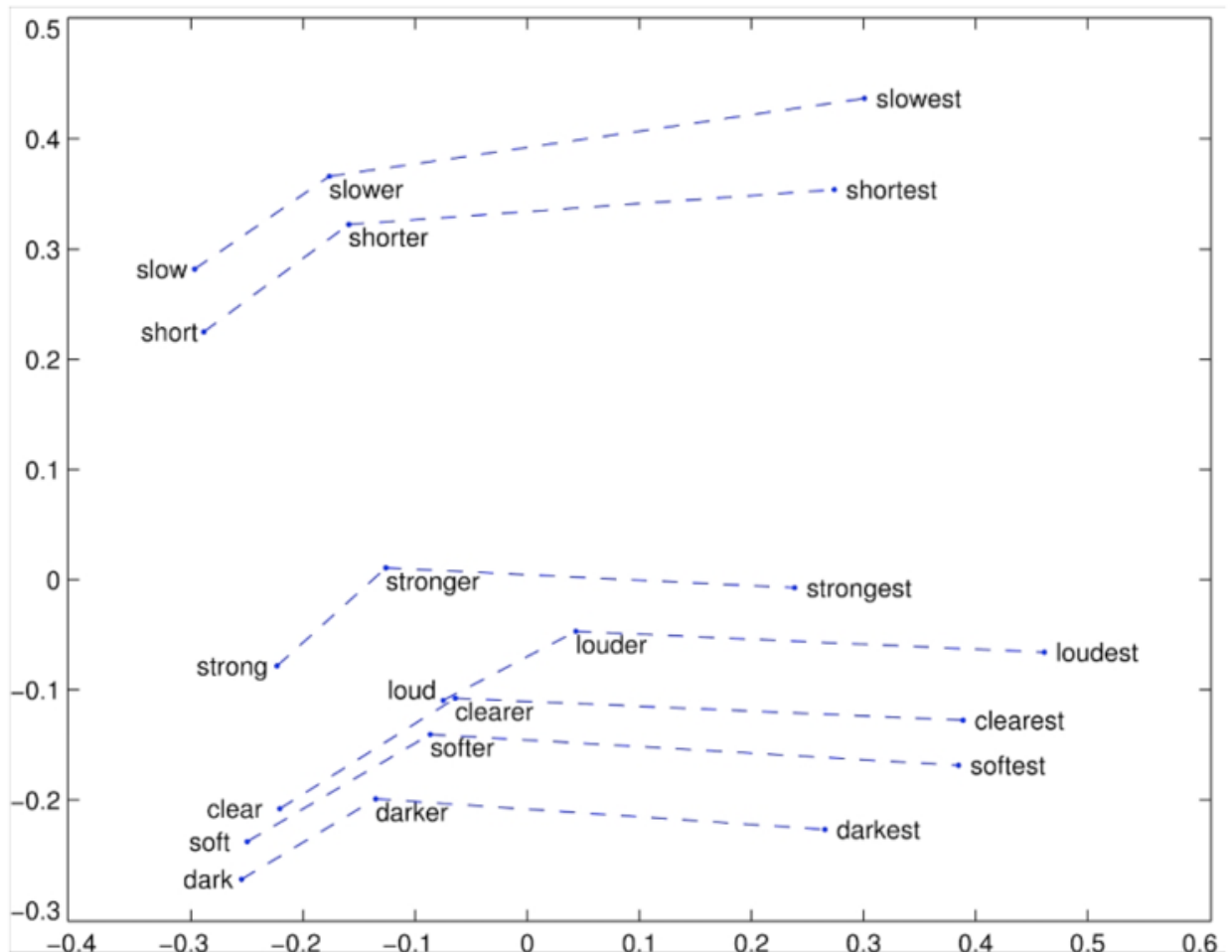
## Glove Visualizations



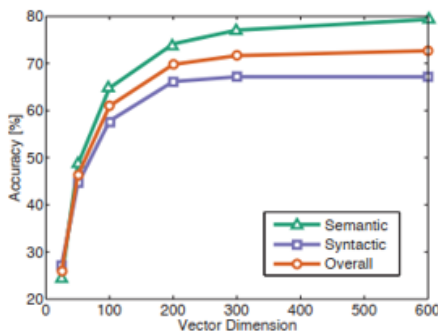
# Glove Visualizations: Company - CEO



# Glove Visualizations: Superlatives

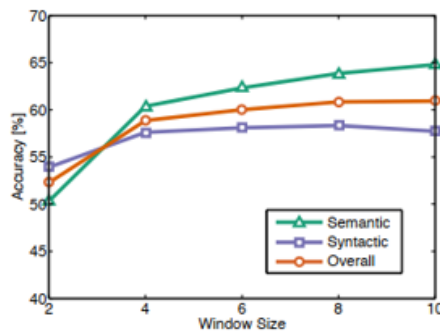


## 类比评价与超参数



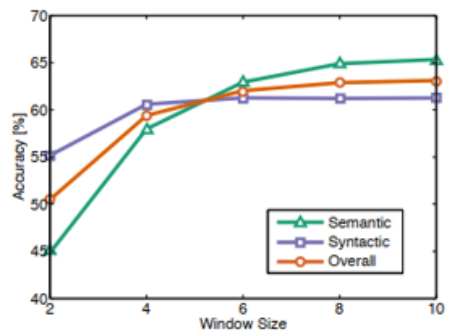
(a) Symmetric context

Dimensionality



(b) Symmetric context

Window size



(c) Asymmetric context

Window size

- Good dimension is ~300
- Asymmetric context (only words to the left) are not as good
- But this might be different for downstream tasks!
- Window size of 8 around each center word is good for GloVe vectors

- 训练次数越多越好
- 数据越多越好

## (2) 另一种内部评价

词向量距离及其与人类判断的关系。

词义与词义歧义

- 大多数单词含有很多意义
- 一个向量能否捕获所有的意义，或者会将意义搞得一团糟

Improving Word Representations Via Global Context AndMultiple Word Prototypes  
(Huang et al. 2012)

想法：将单词窗口聚集在单词周围，重新训练每个单词，并将其分配给多个不同的集群bank1、bank2等

Linear Algebraic Structure of Word Senses, withApplications to Polysemy (Arora, ..., Ma, ..., TACL 2018)

- 单词的不同意义存在于标准单词嵌入（如word2vec）中的线性叠加（加权和）中

Linear Algebraic Structure of Word Senses, with Applications to Polysemy (Arora, ..., Ma, ..., TACL 2018)

- Different senses of a word reside in a linear superposition (weighted sum) in standard word embeddings like word2vec
- $v_{\text{pike}} = \alpha_1 v_{\text{pike}_1} + \alpha_2 v_{\text{pike}_2} + \alpha_3 v_{\text{pike}_3}$
- Where  $\alpha_1 = \frac{f_1}{f_1+f_2+f_3}$ , etc., for frequency  $f$
- Surprising result:
  - Because of ideas from *sparse coding* you can actually separate out the senses (providing they are relatively common)

tie				
trousers	season	scoreline	wires	operatic
blouse	teams	goalless	cables	soprano
waistcoat	winning	equaliser	wiring	mezzo
skirt	league	clinching	electrical	contralto
sleeved	finished	scoreless	wire	baritone
pants	championship	replay	cable	coloratura

5.4.2 Extrinsic (外部评价)

- 在现实任务中进行评测
- 可能需要很长时间才能得到评估结果
- 有时无法确定具体是什么原因导致任务表现出现差异，因此难以合理地对词向量进行评估

下面对Glove模型训练词向量进行实现实战。



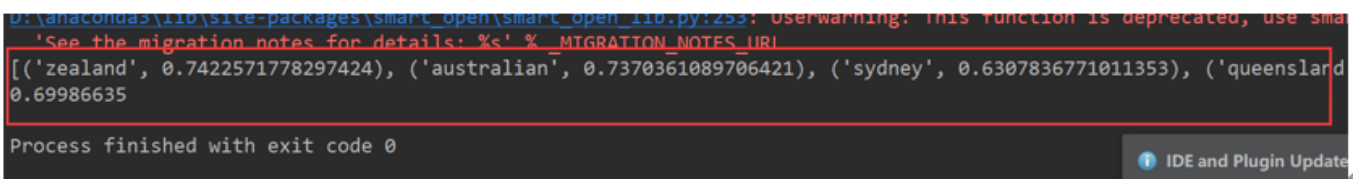
```
1 # -*- coding: utf-8 -*-
2 """
3 word2vec embeddings start with a line with the number of lines (tokens?) and
4 the number of dimensions of the file. This allows gensim to allocate memory
5 accordingly for querying the model. Larger dimensions mean larger memory is
6 held captive. Accordingly, this line has to be inserted into the GloVe
7 embeddings file.
8 """
9
10 import os
11 import shutil
12 import smart_open
13 from sys import platform
14
15 import gensim
16
17
18 def prepend_line(infile, outfile, line):
19     """
20     Function use to prepend lines using bash utilities in Linux.
21     (source: http://stackoverflow.com/a/10850588/610569)
22     """
23     with open(infile, 'r', encoding='UTF-8') as old:
24         with open(outfile, 'w', encoding='UTF-8') as new:
25             new.write(str(line) + "\n")
26             shutil.copyfileobj(old, new)
27
28 def prepend_slow(infile, outfile, line):
29     """
30     Slower way to prepend the line by re-creating the inputfile.
31     """
32     with open(infile, 'r', encoding='UTF-8') as fin:
33         with open(outfile, 'w', encoding='UTF-8') as fout:
34             fout.write(line + "\n")
35             for line in fin:
36                 fout.write(line)
37
38 def get_lines(glove_file_name):
39     """Return the number of vectors and dimensions in a file in GloVe format."""
40     with smart_open.smart_open(glove_file_name, 'r', encoding='UTF-8') as f:
```

```

41     num_lines = sum(1 for line in f)
42     with smart_open.smart_open(glove_file_name, 'r', encoding='UTF-8') as f:
43         num_dims = len(f.readline().split()) - 1
44     return num_lines, num_dims
45
46 # Input: GloVeModel File
47 # More models can be downloaded from http://nlp.stanford.edu/projects/glove/
48 glove_file="glove.6B.300d.txt"
49
50 num_lines, dims = get_lines(glove_file)
51
52 # Output: GensimModel text format.
53 gensim_file='glove_model2.txt'
54 gensim_first_line= "{}{}".format(num_lines, dims)
55
56 # Prepends theline.
57 if platform == "linux" or platform == "linux2":
58     prepend_line(glove_file, gensim_file, gensim_first_line)
59 else:
60     prepend_slow(glove_file, gensim_file, gensim_first_line)
61
62 # Demo: Loads thenewly created glove_model.txt into gensim API.
63 model=gensim.models.KeyedVectors.load_word2vec_format(gensim_file, binary=False)
64
65 print(model.most_similar(positive=['australia'], topn=10))
66 print(model.similarity('woman', 'man'))

```

结果:



```

D:\anaconda3\lib\site-packages\smart_open\smart_open_110.py:253: UserWarning: This function is deprecated, use smart
'See the migration notes for details: %s' % _MIGRATION_NOTES_URL
[('zealand', 0.7422571778297424), ('australian', 0.7370361089706421), ('sydney', 0.6307836771011353), ('queensland',
0.69986635)
Process finished with exit code 0

```

本文电子版 后台回复 **NLP入门** 获取