

# Natural Language Processing with Deep Learning

## CS224N/Ling284



Christopher Manning  
Lecture 2: Word Vectors and Word Senses



# Lecture Plan

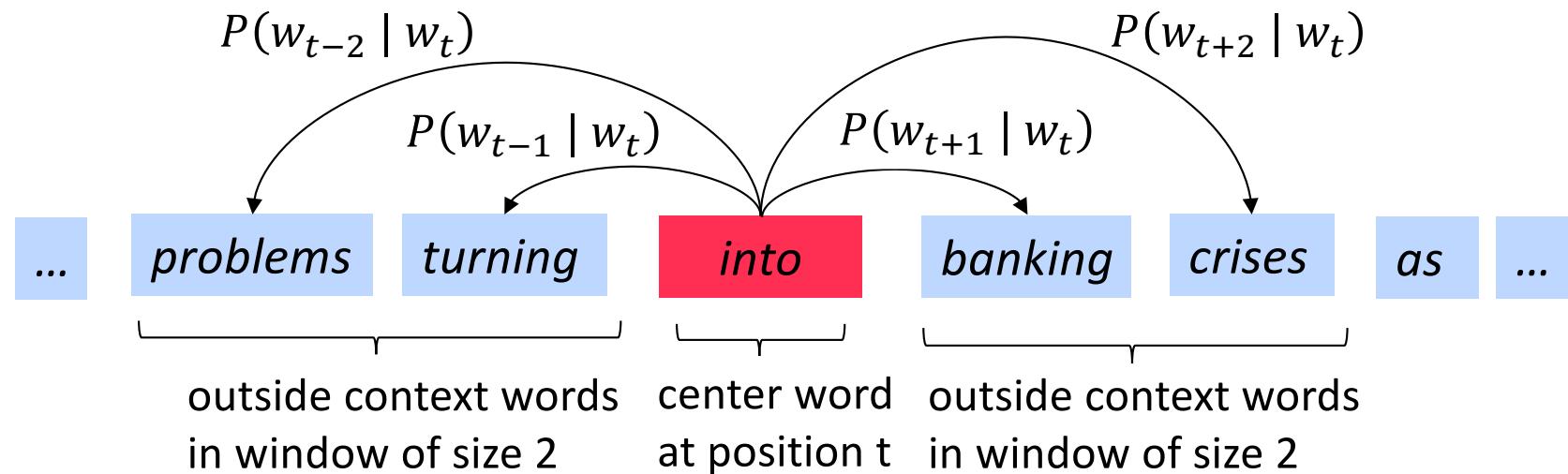
## Lecture 2: Word Vectors and Word Senses

1. Finish looking at word vectors and word2vec (12 mins)
2. Optimization basics (8 mins)
3. Can we capture this essence more effectively by counting? (15m)
4. The GloVe model of word vectors (10 min)
5. Evaluating word vectors (15 mins)
6. Word senses (5 mins)
7. The course (5 mins)

Goal: be able to read word embeddings papers by the end of class

# 1. Review: Main idea of word2vec

- Iterate through each word of the whole corpus
- Predict surrounding words using word vectors



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

# Word2vec parameters and computations

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

$U$

outside

6 words  $\times$  5 dim

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

$V$

center

!

high frequency effect:

词比如and/or/of，作为center的时候出来的概率分布中每个分量都很大（有论文试图解决，得到更好的语义相似度）

$$\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

$U \cdot v_4^T$

dot product

tensorflow和PyTorch的词向量都以行向量表示

$$\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

$\text{softmax}(U \cdot v_4^T)$

probabilities

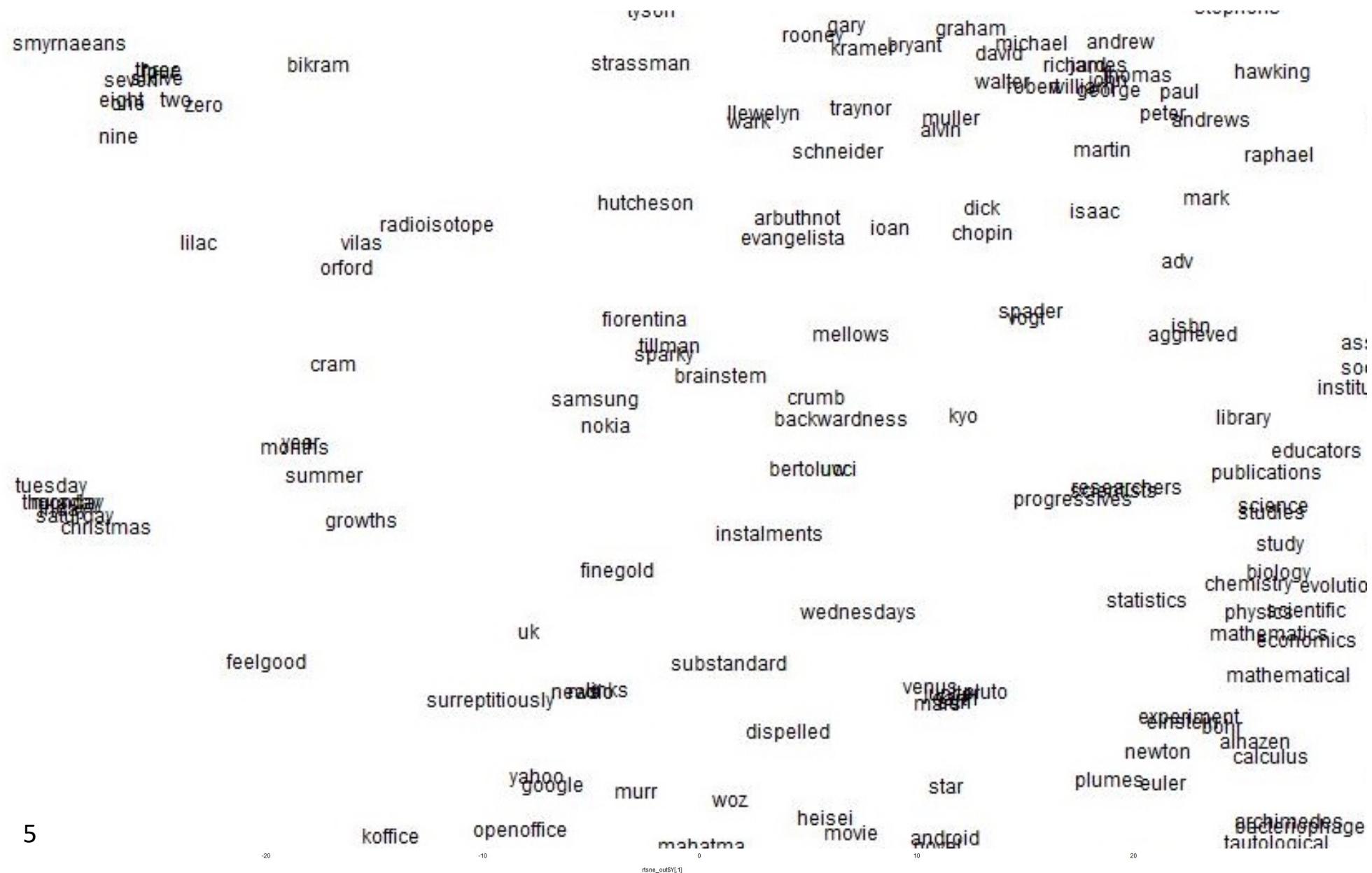
Same predictions at each position

We want a model that gives a reasonably high probability estimate to *all* words that occur in the context (fairly often)

也就是不考虑位置

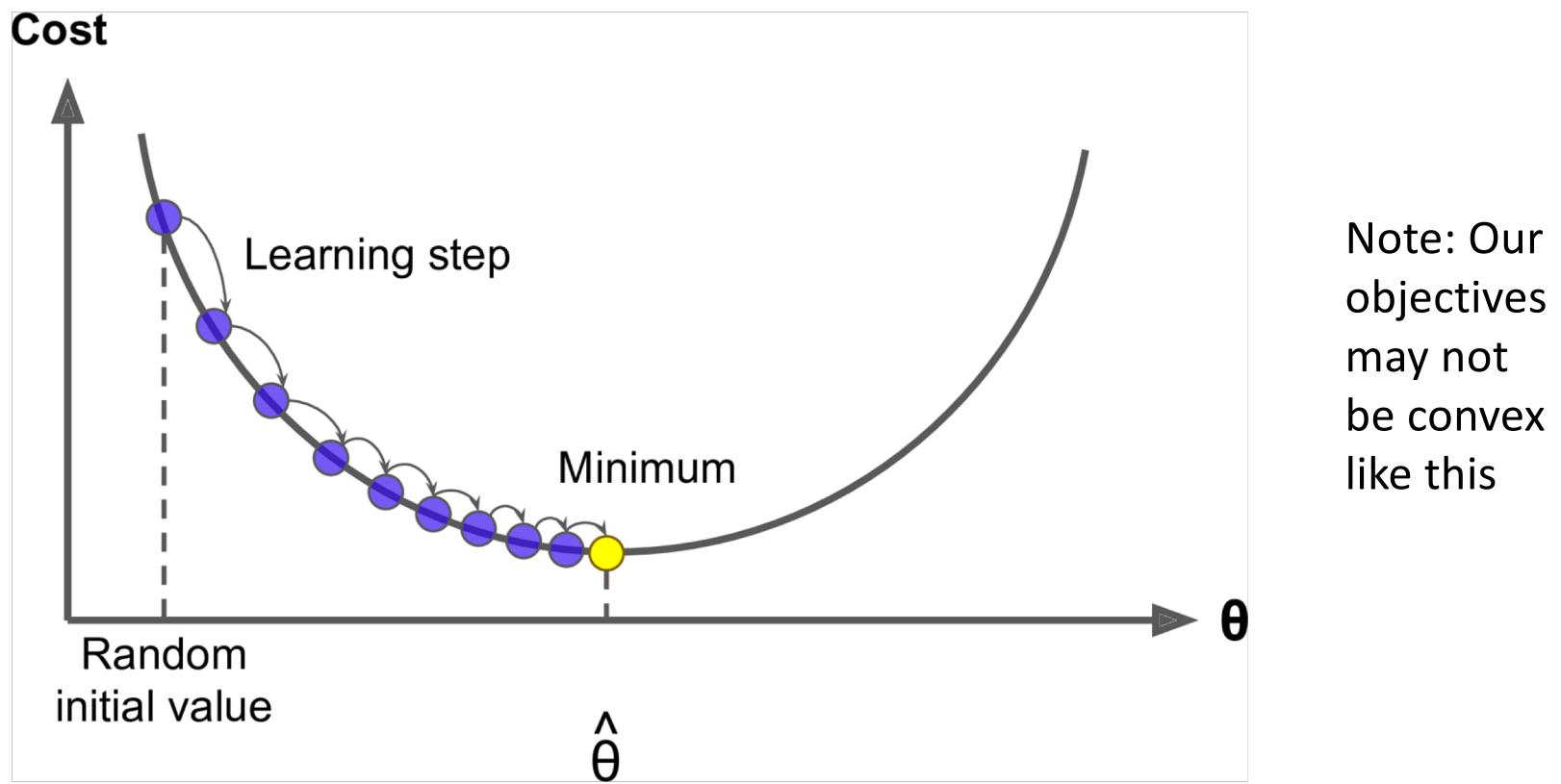
实际上压缩到2D的话很多高维的接近就不直观了，难以看出

# Word2vec maximizes objective function by putting similar words nearby in space



## 2. Optimization: Gradient Descent

- We have a cost function  $J(\theta)$  we want to minimize
- **Gradient Descent** is an algorithm to minimize  $J(\theta)$
- Idea: for current value of  $\theta$ , calculate gradient of  $J(\theta)$ , then take small step in the direction of negative gradient. Repeat.



# Gradient Descent

- Update equation (in matrix notation):

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

$\alpha$  = step size or learning rate

- Update equation (for a single parameter):

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

- Algorithm:

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

# Stochastic Gradient Descent

是整个corpus上的参数

- Problem:  $J(\theta)$  is a function of **all** windows in the corpus (potentially billions!)
  - So  $\nabla_{\theta} J(\theta)$  is **very expensive to compute**
- You would wait a very long time before making a single update!
- **Very** bad idea for pretty much all neural nets!
- Solution: **Stochastic gradient descent (SGD)**
  - Repeatedly sample windows, and update after each one
- Algorithm: noisy ,但是迭代多次就好

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

一般取样一个mini-batch, 32/64个，可以average降噪，也便于GPU并行计算

# Stochastic gradients with word vectors!

- Iteratively take gradients at each such window for SGD
- But in each <sup>采样窗口</sup> window, we only have at most  $2m + 1$  words,  
so  $\nabla_{\theta} J_t(\theta)$  is very sparse! ( 和CV不同 ) m是context window半径

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

# Stochastic gradients with word vectors!

- We might only update the word vectors that actually appear!  
只更新出现的（因此会变的）
- Solution: either you need sparse matrix update operations to only update certain rows of full embedding matrices  $U$  and  $V$ , or you need to keep around a hash for word vectors

【 ? 】

$$|V| \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}^d$$

- If you have millions of word vectors and do distributed computing, it is important to not have to send gigantic updates around!

## 1b. Word2vec: More details

计算上求导推导更方便

Why two vectors? → Easier optimization. Average both at the end

- But can do it with just one vector per word

Two model variants:

1. Skip-grams (SG) 之前讲的是这个，从中心词预测context

Predict context (“outside”) words (position independent) given center word

2. Continuous Bag of Words (CBOW) 反过来，给出context预测中心词

Predict center word from (bag of) context words

We presented: **Skip-gram model**

Additional efficiency in training:

1. Negative sampling

So far: Focus on **naïve softmax** (simpler, but expensive, training method)

点积直接做element-wise softmax求概率分布

## The skip-gram model with negative sampling (HW2)

- The normalization factor is too computationally expensive.

【优化】 分子中是实际观察到的，应该有高概率
- $$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

分母计算代价太大了
- Hence, in standard word2vec and HW2 you implement the skip-gram model with **negative sampling**
- Main idea: train binary logistic regressions for a true pair (center word and word in its context window) versus several noise pairs (the center word paired with a random word)

随机取样一些，认为这些不是分子中的，概率应该低

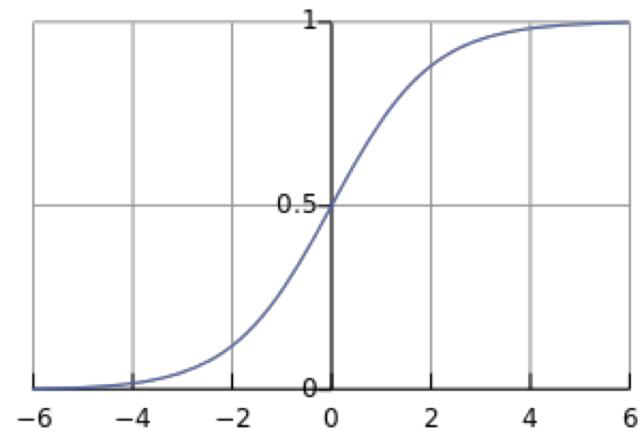
## The skip-gram model with negative sampling (HW2)

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

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

【 ? 】

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



# The skip-gram model with negative sampling (HW2)

- Notation more similar to class and HW2:

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

是奇对称函数  
观察到的context      随机采样K个词作为负样本，可以帮助减轻

- We take  $k$  negative samples (using word probabilities) **high frequency**
  - Maximize probability that real outside word appears,  
minimize prob. that random words appear around center word

## 被选作负采样的概率计算

- $P(w) = U(w)^{3/4} / Z$ , 一元模型分布，幂次 < 1 有助于均衡采样  
the unigram distribution  $U(w)$  raised to the  $3/4$  power  
(We provide this function in the starter code).  $3/4$ 是一个hyper-param  
( $k$ 也是)
  - The power makes less frequent words be sampled more often  
 $Z$ 是normalization term (=sum), 变成概率  
(QA: 训练的时候会在每个epoch做data shuffle, 然后顺序取, 这样有更好的locality)

### 3. But why not capture co-occurrence counts directly?

即使有了分布式语义，也可以考虑词语共现计数，而不是word2vec的模型，为什么不？

With a co-occurrence matrix  $X$

- 2 options: windows vs. full document
- Window: Similar to word2vec, use window around each word → captures both syntactic (POS) and semantic information
- Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis”

LSA，可以用于语义检索

## Example: Window based co-occurrence matrix

- Window length 1 (more common: 5–10)
- Symmetric (irrelevant whether left or right context)
- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

# Window based co-occurrence matrix

- 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

like和en-  
joy相似

# Problems with simple co-occurrence vectors

Increase in size with vocabulary

矩阵尺寸为 $|V| \times |V|$

Very high dimensional: requires a lot of storage

Subsequent classification models have sparsity issues

→ Models are less robust

## Solution: Low dimensional vectors

- Idea: store “most” of the important information in a fixed, small number of dimensions: a dense vector
- Usually 25–1000 dimensions, similar to word2vec
- How to reduce the dimensionality?

# Method 1: Dimensionality Reduction on X (HW1)

Singular Value Decomposition of co-occurrence matrix  $X$

Factorizes  $X$  into  $U\Sigma V^T$ , where  $U$  and  $V$  are orthonormal

(相似对角化的延伸)

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{X^k} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

共现矩阵是对称阵，所以可以对称对角化， $U=V$

Retain only  $k$  singular values, in order to generalize.

$\hat{X}$  is the best rank  $k$  approximation to  $X$ , in terms of least squares.

Classic linear algebra result. Expensive to compute for large matrices.

# Simple SVD word vectors in Python

Corpus:

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

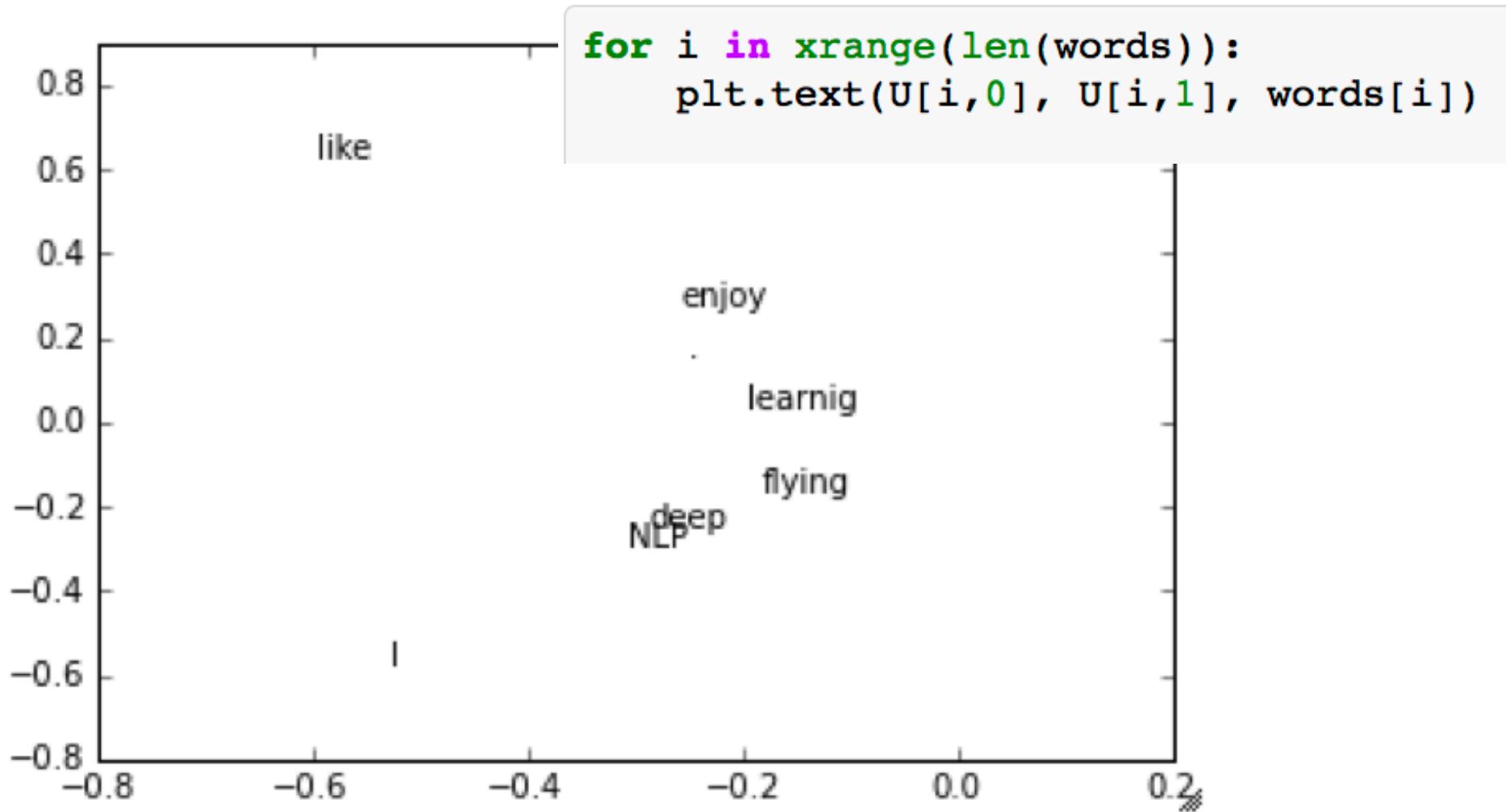
```
import numpy as np
la = np.linalg linear algebra
words = ["I", "like", "enjoy",
          "deep", "learnig", "NLP", "flying", ".."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])  
  
U, s, Vh = la.svd(X, full_matrices=False)
```

# Simple SVD word vectors in Python

21世纪初的主流做法

Corpus: I like deep learning. I like NLP. I enjoy flying.

Printing first two columns of U corresponding to the 2 biggest singular values

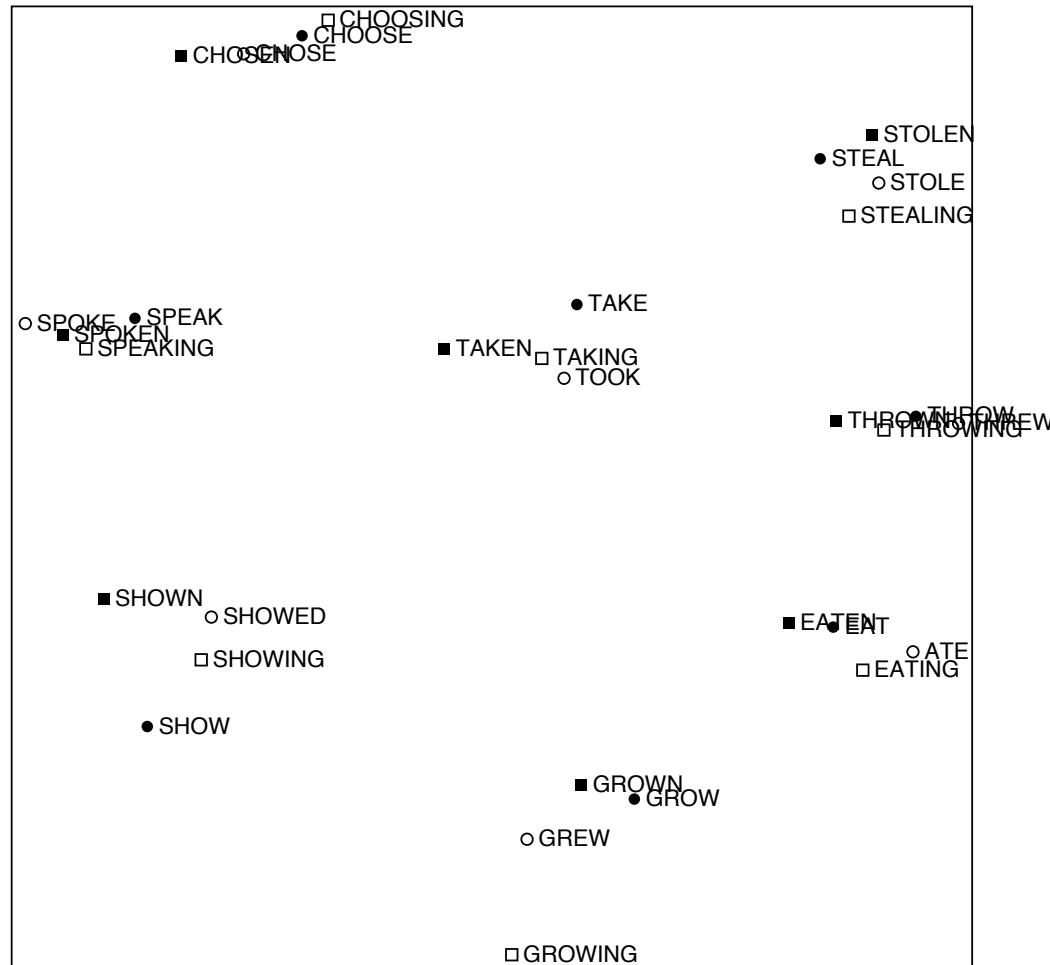


# Hacks to X (several used in Rohde et al. 2005)

Scaling the counts in the cells can help *a lot*

- Problem: function words (*the, he, has*) are too frequent → syntax has too much impact. Some fixes:
  - $\log$  (IR经常做，信息抽取)
    - $\min(X, t)$ , with  $t \approx 100$
    - Ignore them all
- Ramped windows that count closer words more
- Use Pearson correlations instead of counts, then set negative values to 0
- Etc.

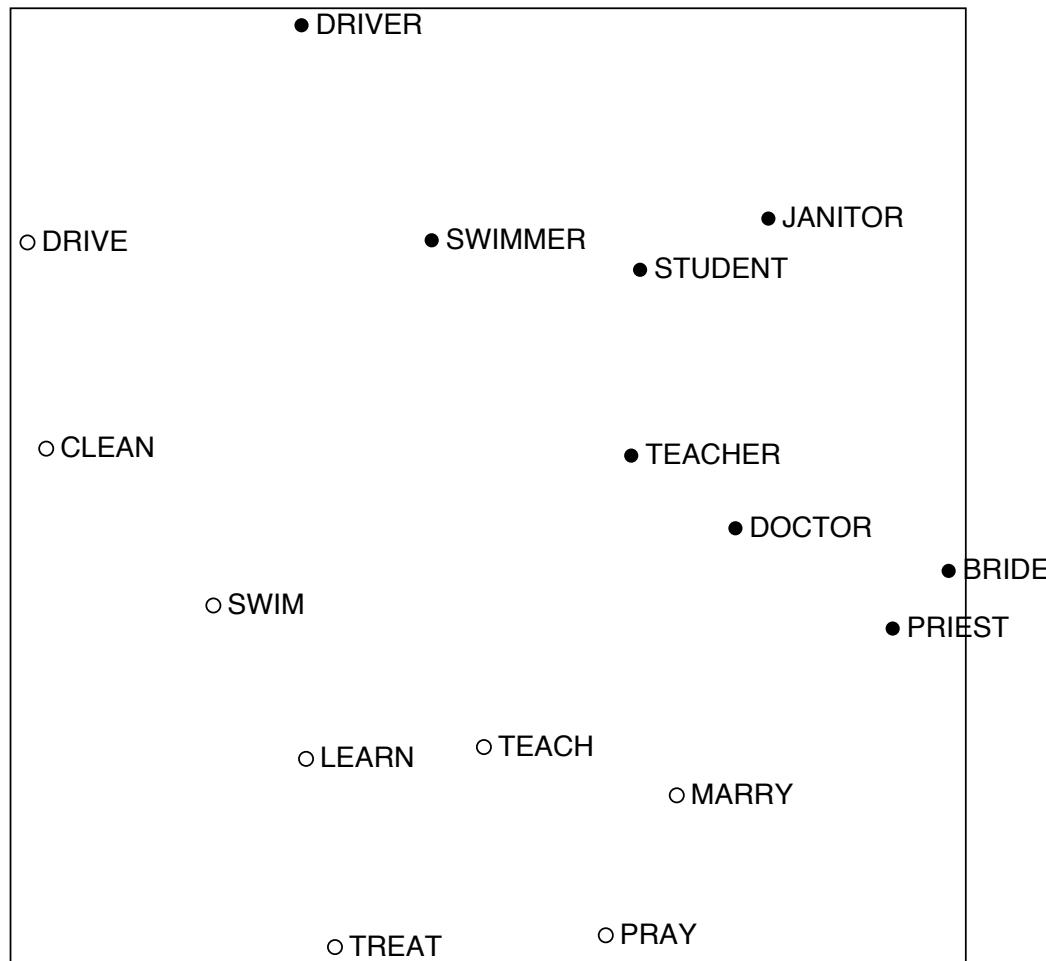
# Interesting syntactic patterns emerge in the vectors



COALS model from  
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence  
Rohde et al. ms., 2005

发现了线性型，虽然没有提出analogy的测评方式，但是隐含在其中了（analogy即king-man +woman=queen, Mikolov提出）

# Interesting semantic patterns emerge in the vectors



COALS model from  
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence  
Rohde et al. ms., 2005

# Count based vs. direct prediction

## 基于计数 ( word-based counts )

- LSA, HAL (Lund & Burgess),
- COALS, Hellinger-PCA (Rohde et al, Lebret & Collobert)

- Fast training
- Efficient usage of statistics  
用了全局data
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- Skip-gram/CBOW (Mikolov et al)
- NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton)

- Scales with corpus size
- Inefficient usage of statistics  
忽略了counts，而且是每次采样
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity



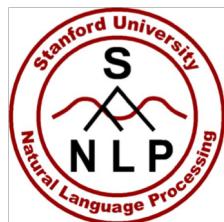
结合count matrix和word2vec

# Encoding meaning in vector differences

## [Pennington, Socher, and Manning, EMNLP 2014]

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

	$x = \text{solid}$	$x = \text{gas}$	$x = \text{water}$	$x = \text{random}$
$P(x \text{ice})$	large	small	large	small
$P(x \text{steam})$	small	large	large	small
$\frac{P(x \text{ice})}{P(x \text{steam})}$	large	small	$\sim 1$	$\sim 1$



# Encoding meaning in vector differences

[Pennington, Socher, and Manning, EMNLP 2014]

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



# Encoding meaning in vector differences

Q: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

A: Log-bilinear model:

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

with vector differences

$$w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$$

比值为1，则 $\log 1 = 0$ ，表明向量正交（线性无关）



count方法+prediction方法(iterative loss)

# Combining the best of both worlds

## GloVe [Pennington et al., EMNLP 2014]



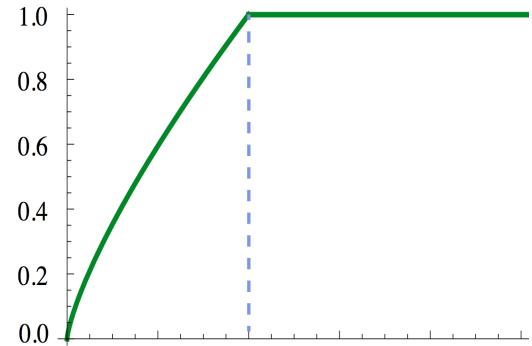
$$w_i \cdot w_j = \log P(i|j)$$

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

截断高频词  
内积      加上了bias

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

$f \sim$



# GloVe results

Nearest words to  
[frog](#):

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus



[litoria](#)



[leptodactylidae](#)



[rana](#)



[eleutherodactylus](#)

# How to evaluate word vectors?

- Related to general evaluation in NLP: Intrinsic vs extrinsic
- Intrinsic: 例如word similarity , 比较快
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems 比如加了东西但性能没有提升 , 可能换个模型就好了
  - If replacing exactly one subsystem with another improves accuracy → Winning!

# Intrinsic word vector evaluation

- Word Vector Analogies

$$\boxed{a:b :: c: ?}$$

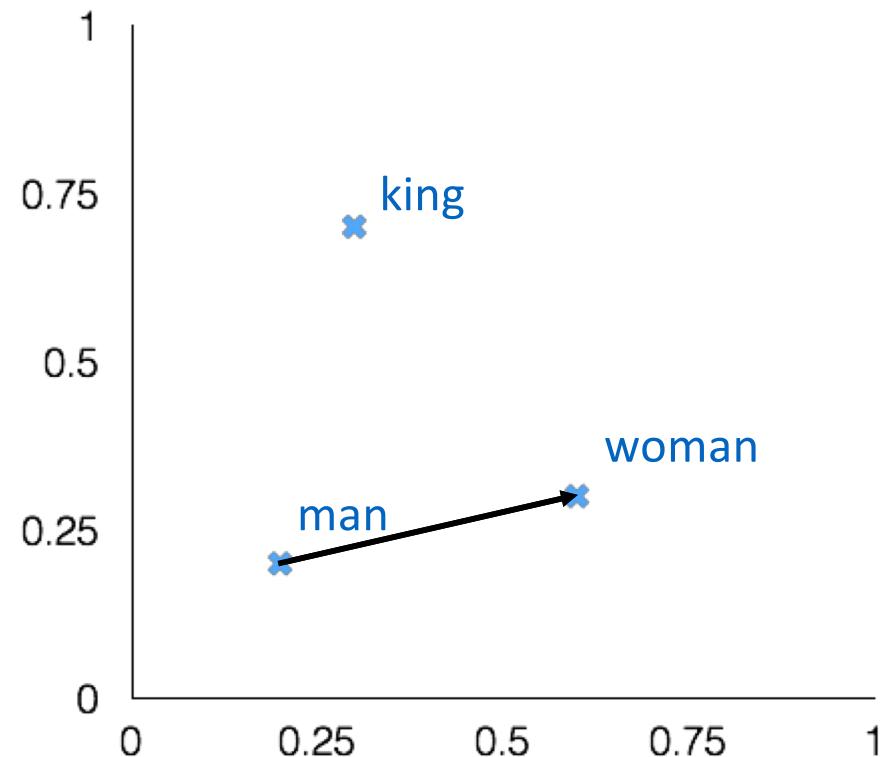
man:woman :: king:?



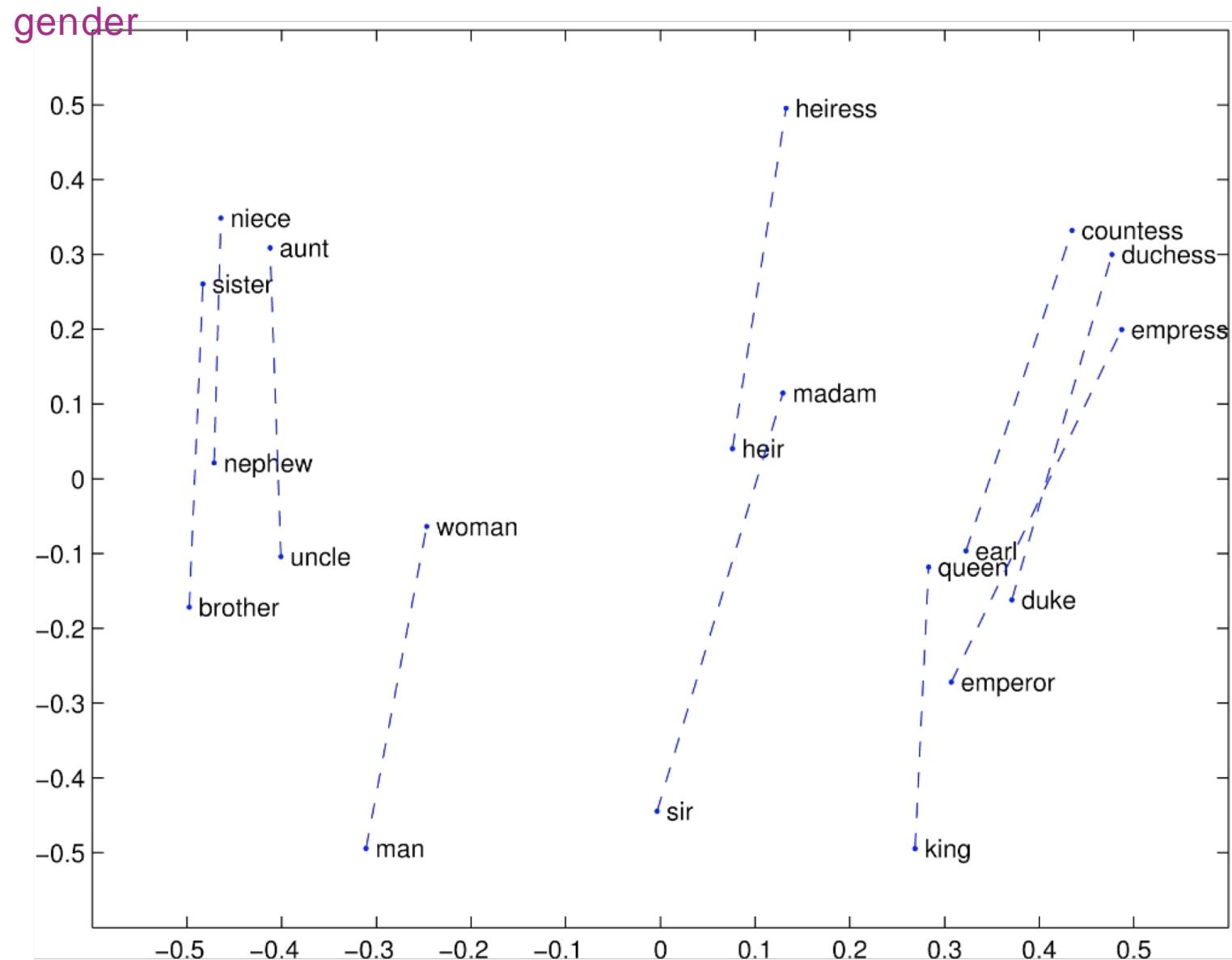
$$d = \arg \max_i \frac{(x_b - x_a + x_c)^T x_i}{\|x_b - x_a + x_c\|}$$

余弦距离

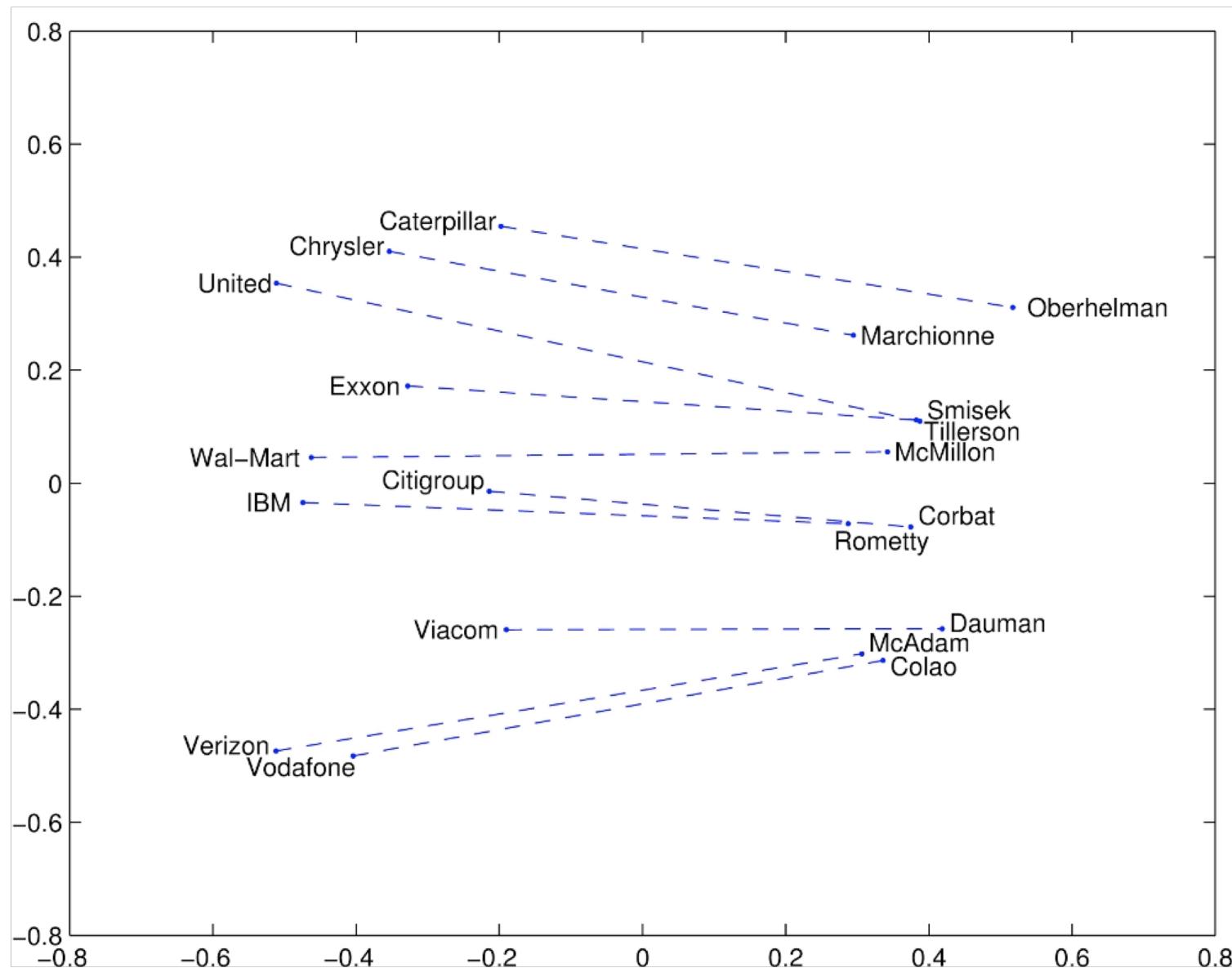
- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



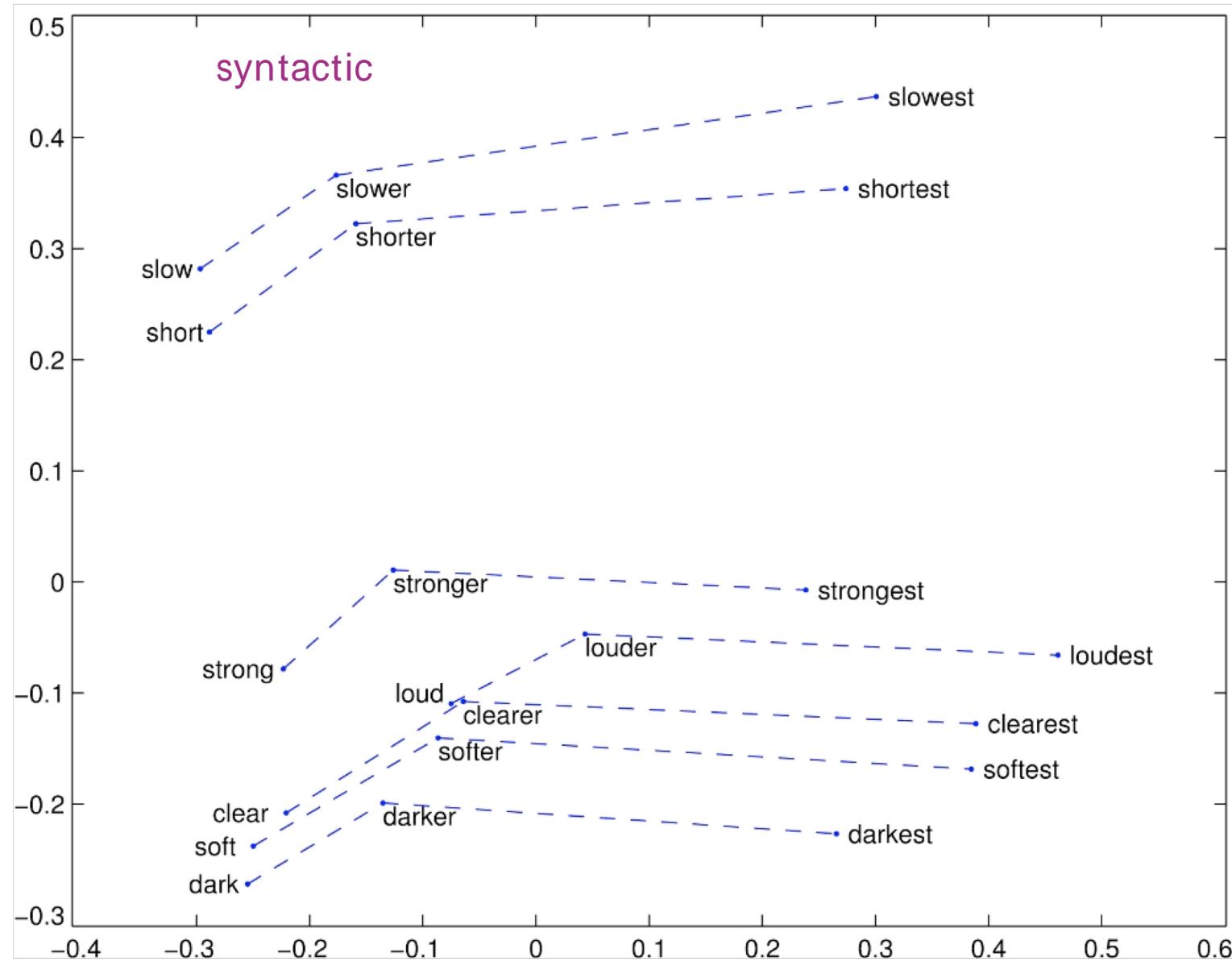
# Glove Visualizations



# Glove Visualizations: Company - CEO



# Glove Visualizations: Superlatives



# Details of intrinsic word vector evaluation

- Word Vector Analogies: Syntactic and **Semantic** examples from  
<http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt>

: city-in-state

Chicago Illinois Houston Texas

Chicago Illinois Philadelphia Pennsylvania

Chicago Illinois Phoenix Arizona

Chicago Illinois Dallas Texas

Chicago Illinois Jacksonville Florida

Chicago Illinois Indianapolis Indiana

Chicago Illinois Austin Texas

Chicago Illinois Detroit Michigan

Chicago Illinois Memphis Tennessee

Chicago Illinois Boston Massachusetts

problem: different cities  
may have same name

# Details of intrinsic word vector evaluation

- Word Vector Analogies: **Syntactic** and Semantic examples from

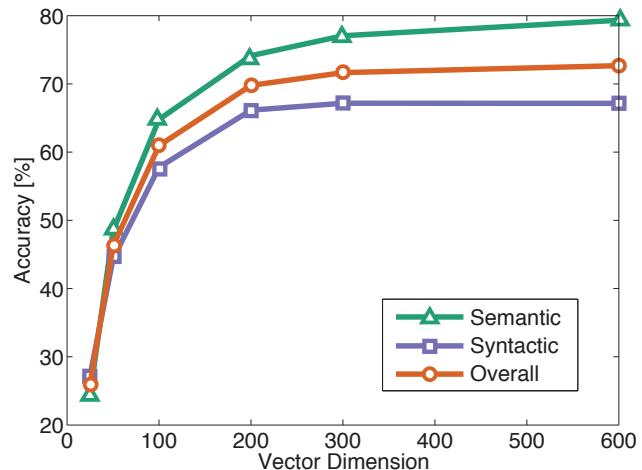
: gram4-superlative  
bad worst big biggest  
bad worst bright brightest  
bad worst cold coldest  
bad worst cool coolest  
bad worst dark darkest  
bad worst easy easiest  
bad worst fast fastest  
bad worst good best  
bad worst great greatest

# Analogy evaluation and hyperparameters

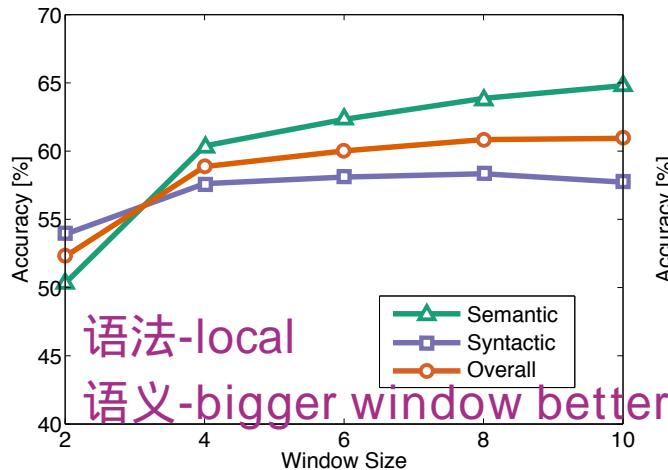
- Glove word vectors evaluation

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	<u>67.5</u>	<u>54.3</u>	<u>60.3</u>
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	<u>64.8</u>	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	<u>80.8</u>	61.5	<u>70.3</u>
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
<b>SVD-L</b>	300	6B	56.6	63.0	60.1
CBOW <sup>†</sup>	300	6B	63.6	<u>67.4</u>	65.7
SG <sup>†</sup>	300	6B	73.0	66.0	69.1
GloVe	300	6B	<u>77.4</u>	67.0	<u>71.7</u>
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	<b><u>81.9</u></b>	<b><u>69.3</u></b>	<b><u>75.0</u></b>

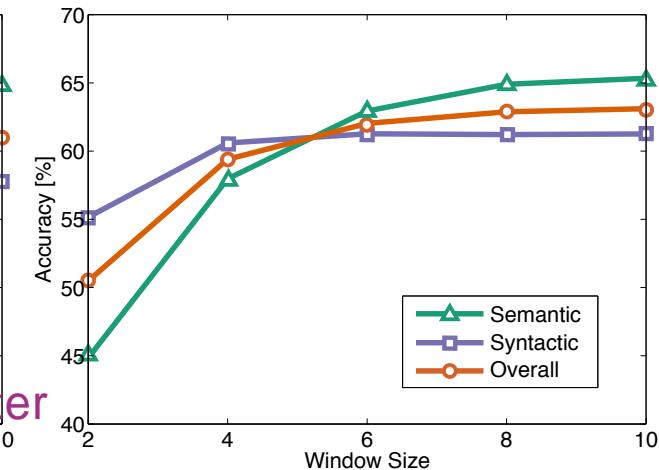
# Analogy evaluation and hyperparameters



(a) Symmetric context



(b) Symmetric context



(c) Asymmetric context

Dimensionality

Window size

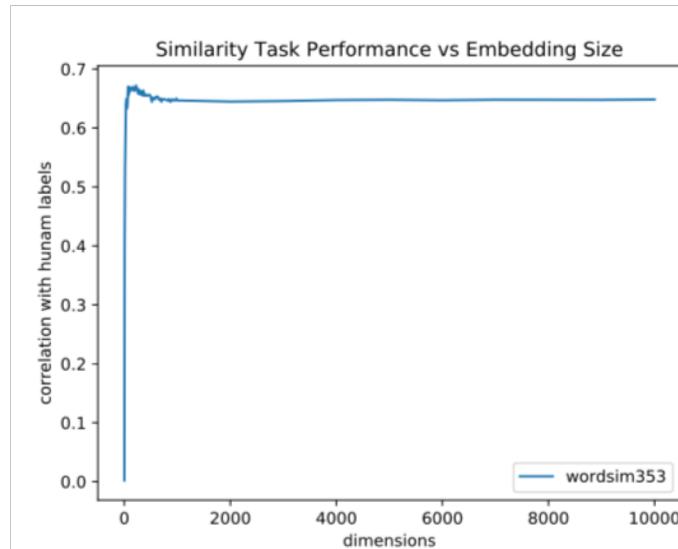
Window size

- Good dimension is  $\sim 300$
- 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



# On the Dimensionality of Word Embedding

## [Zi Yin and Yuanyuan Shen, NeurIPS 2018]



(b) WordSim353 Test



<https://papers.nips.cc/paper/7368-on-the-dimensionality-of-word-embedding.pdf>

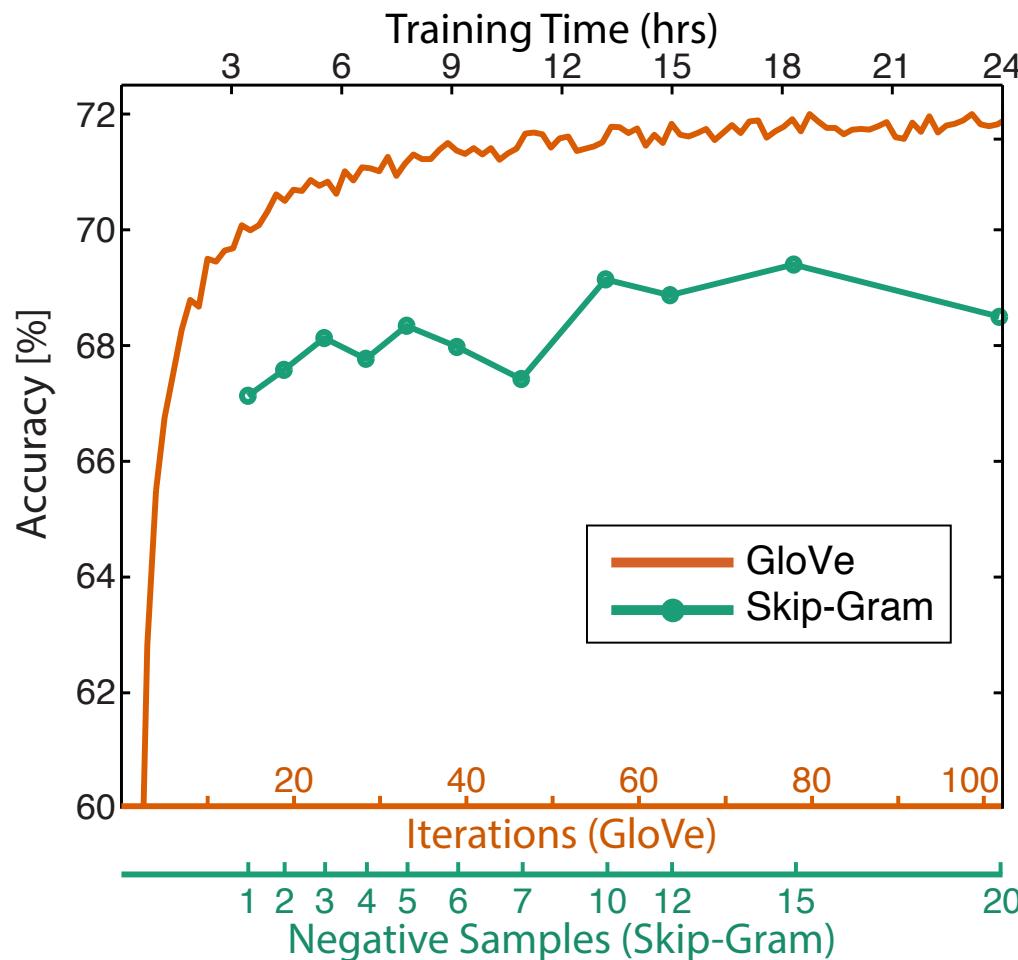
Using matrix perturbation theory, reveal a fundamental bias-variance trade-off in dimensionality selection for word embeddings

Table 3: PIP loss minimizing dimensionalities and intervals for GloVe on Text8 corpus

Surrogate Matrix	$\arg \min$	+5% interval	+10% interval	+20% interval	+50% interval	WS353	MT771	Analogy
GloVe (log-count)	719	[290,1286]	[160,1663]	[55,2426]	[5,2426]	220	860	560

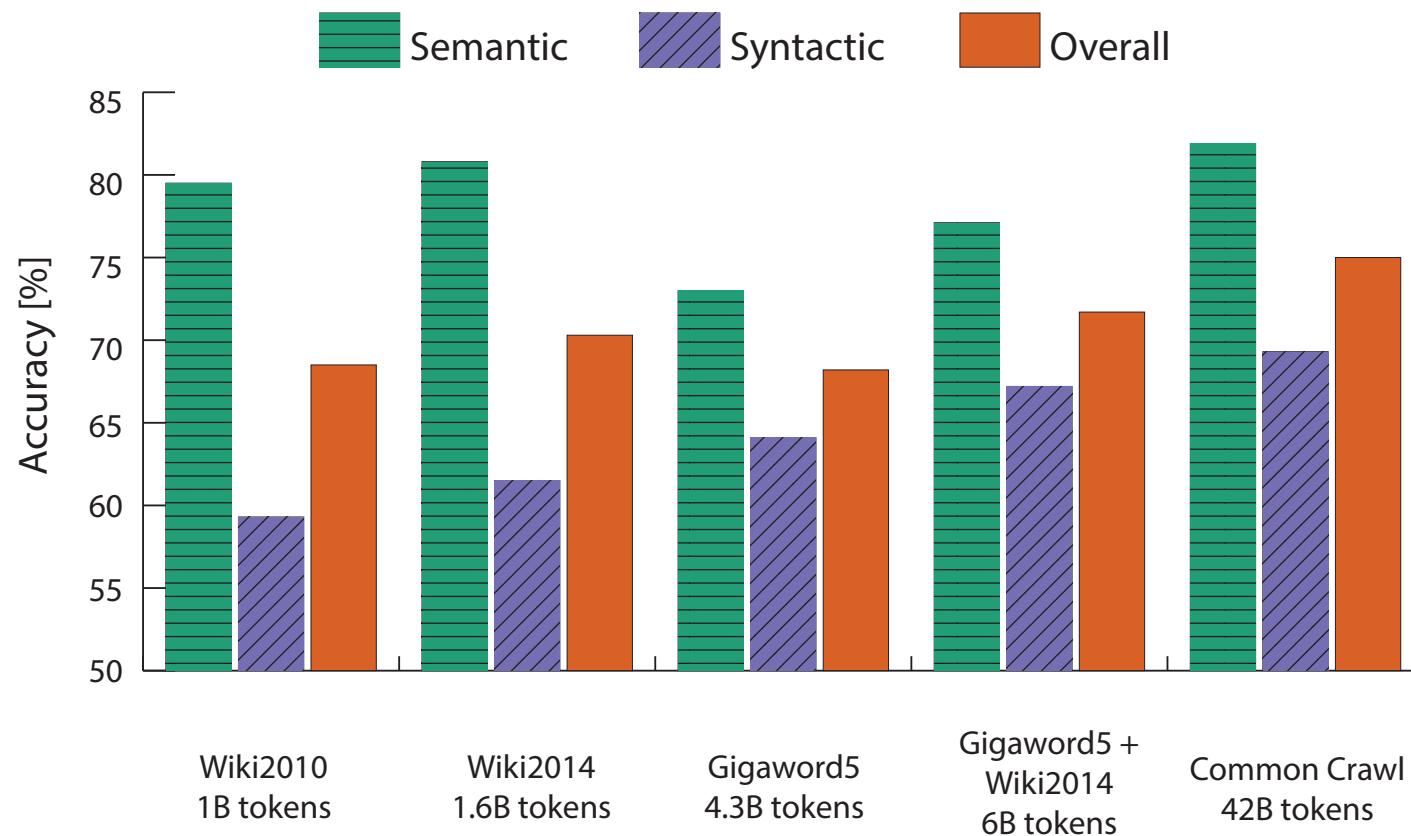
# Analogy evaluation and hyperparameters

- More training time helps



# Analogy evaluation and hyperparameters

- More data helps, Wikipedia is better than news text!



# Another intrinsic word vector evaluation

- Word vector distances and their correlation with human judgments
- Example dataset: WordSim353 和心理学有关（锚定效应）  
<http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/>

Word 1	Word 2	Human (mean)	打分，1~10分
tiger	cat	7.35	
tiger	tiger	10	
book	paper	7.46	
computer	internet	7.58	
plane	car	5.77	
professor	doctor	6.62	
stock	phone	1.62	
stock	CD	1.31	
stock	jaguar	0.92	

## Closest words to “Sweden” (cosine similarity)

Word	Cosine distance
-----	
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

# Correlation evaluation

- Word vector distances and their correlation with human judgments

对相关性建模

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW <sup>†</sup>	6B	57.2	65.6	68.2	57.0	32.5
SG <sup>†</sup>	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	<u>75.9</u>	<u>83.6</u>	<u>82.9</u>	<u>59.6</u>	<u>47.8</u>
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

- Some ideas from Glove paper have been shown to improve skip-gram (SG) model also (e.g. sum both vectors)

# Word senses and word sense ambiguity

- Most words have lots of meanings!
  - Especially common words
  - Especially words that have existed for a long time  
有很多指代项
- Example: pike
  - fish / road / spear ...
- Does one vector capture all these meanings or do we have a mess?

# pike

- A sharp point or staff
- A type of elongated fish
- A railroad line or system
- A type of road
- The future (coming down the pike)
- A type of body position (as in diving)
- To kill or pierce with a pike
- To make one's way (pike along)
- In Australian English, pike means to pull out from doing something: I reckon he could have climbed that cliff, but he piked!

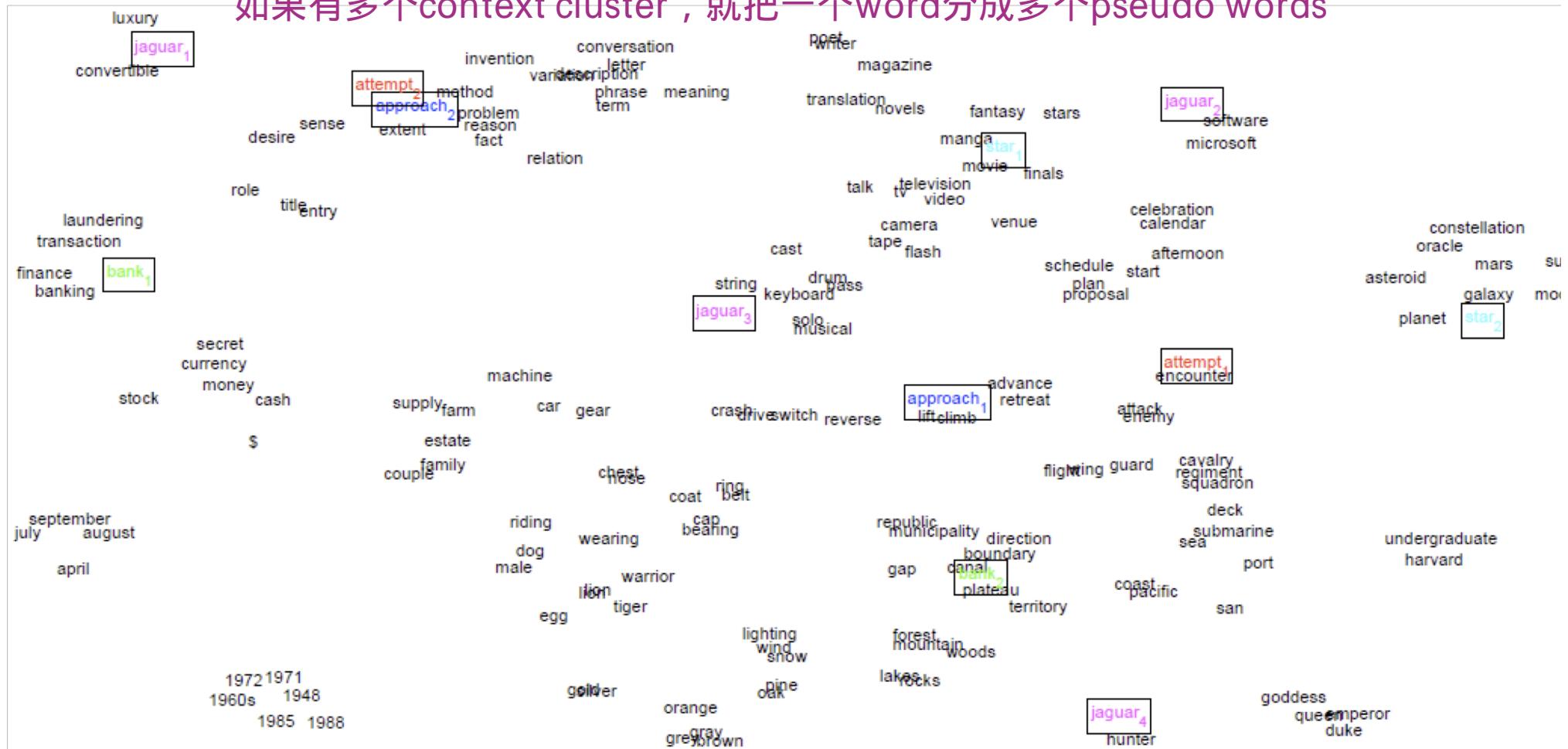
想法：为什么只能有一个word sensor，而不是多个？

# Improving Word Representations Via Global Context And Multiple Word Prototypes (Huang et al. 2012)

## ( context )

- Idea: Cluster word windows around words, retrain with each word assigned to multiple different clusters  $\text{bank}_1$ ,  $\text{bank}_2$ , etc

如果有多个context cluster，就把一个word分成多个pseudo words



如果还是只有一个sensor...

# 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 加权平均
- $v_{\text{pike}} = \alpha_1 v_{\text{pike}_1} + \alpha_2 v_{\text{pike}_2} + \alpha_3 v_{\text{pike}_3}$
- 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

# Extrinsic word vector evaluation

- Extrinsic evaluation of word vectors: All subsequent tasks in this class  
不管做什么NLP任务，放一个word embedding进去，分又涨了！
- One example where good word vectors should help directly: named entity recognition: finding a person, organization or location

NER	Model	Dev	Test	ACE	MUC7
	Discrete	91.0	85.4	77.4	73.4
	SVD	90.8	85.7	77.3	73.7
	SVD-S	91.0	85.5	77.6	74.3
	SVD-L	90.5	84.8	73.6	71.5
	HPCA	92.6	<b>88.7</b>	81.7	80.7
	HSMN	90.5	85.7	78.7	74.7
	CW	92.2	87.4	81.7	80.2
	CBOW	93.1	88.2	82.2	81.1
	GloVe	<b>93.2</b>	88.3	<b>82.9</b>	<b>82.2</b>

- Next: How to use word vectors in neural net models!

# Course plan: coming weeks

## Week 2: Neural Net Fundamentals

- We concentrate on understanding (deep, multi-layer) neural networks and how they can be trained (learned from data) using backpropagation (the judicious application of calculus)
- We'll look at an NLP classifier that adds context by taking in windows around a word and classifies the center word (not just representing it across all windows)!

## Week 3: We learn some natural language processing

- We learn about putting syntactic structure (dependency parses) over sentence (this is HW3!)
- We develop the notion of the probability of a sentence (a probabilistic language model) and why it is really useful



# A note on your experience 😊

“Terrible class”

“Don’t take it”

“Instructors don’t care”

“Too much work”

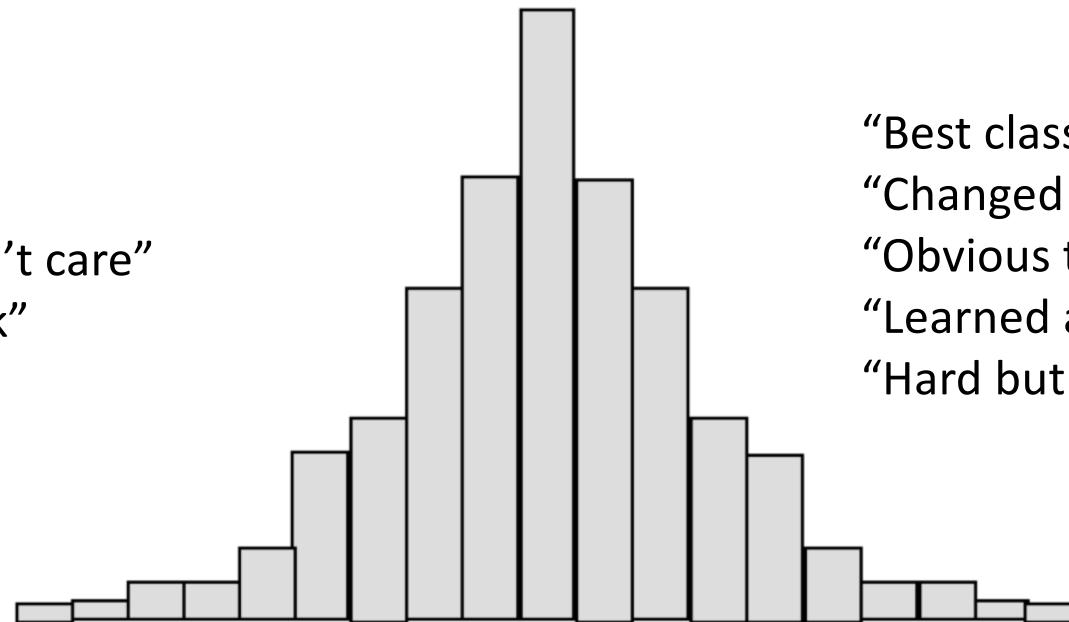
“Best class at Stanford”

“Changed my life”

“Obvious that instructors care”

“Learned a ton”

“Hard but worth it”



- This is a hard, advanced, graduate level class
- I and all the TAs really care about your success in this class
- Give Feedback. Take responsibility for holes in your knowledge
- **Come to office hours/help sessions**

# Office Hours / Help sessions

- **Come to office hours/help sessions!**
  - Come to discuss final project ideas as well as the homeworks
  - Try to come early, often and off-cycle
- **Help sessions:** daily, at various times, see calendar
  - First one is tonight: 6:30–9:00pm
  - Gates Basement B21 (and B30) – bring your student ID
  - Attending in person: Just show up! Our friendly course staff will be on hand to assist you
  - SCPD/remote access: Use queuestatus
- **Chris's office hours:**
  - Mon 1–3pm. Come along this Monday?