

Social Media & Text Analysis

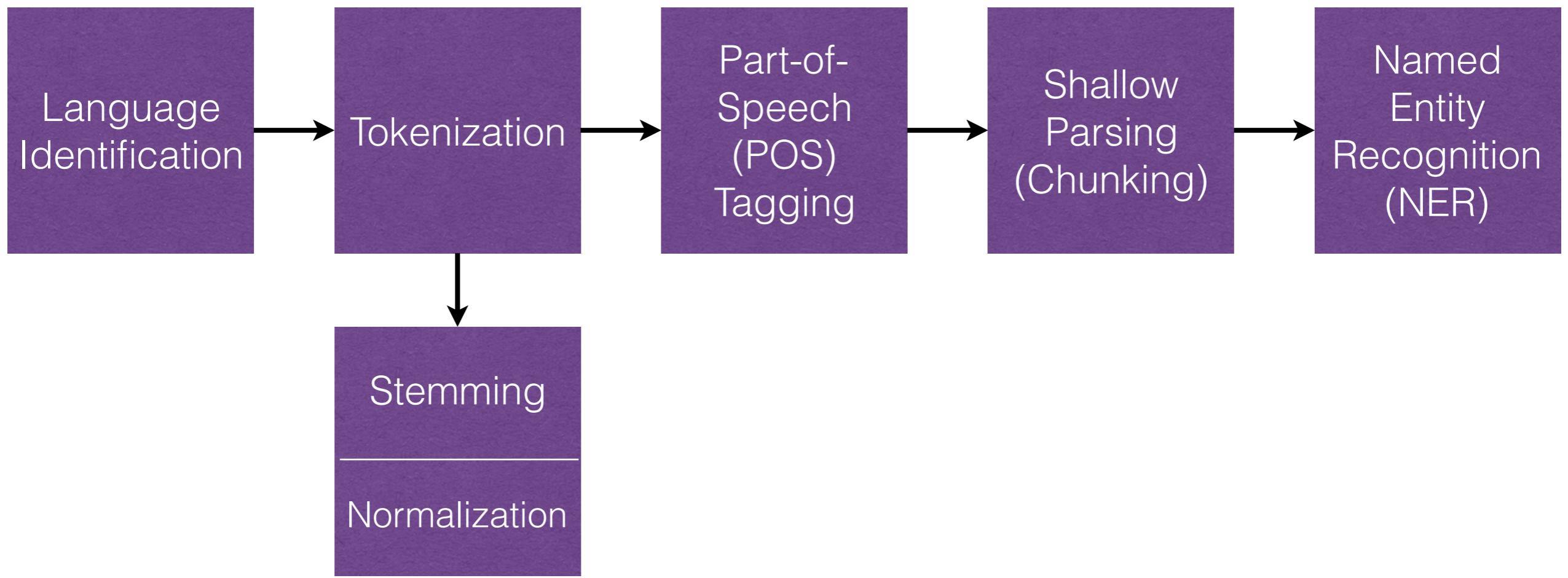
lecture 8 - Vector Semantics



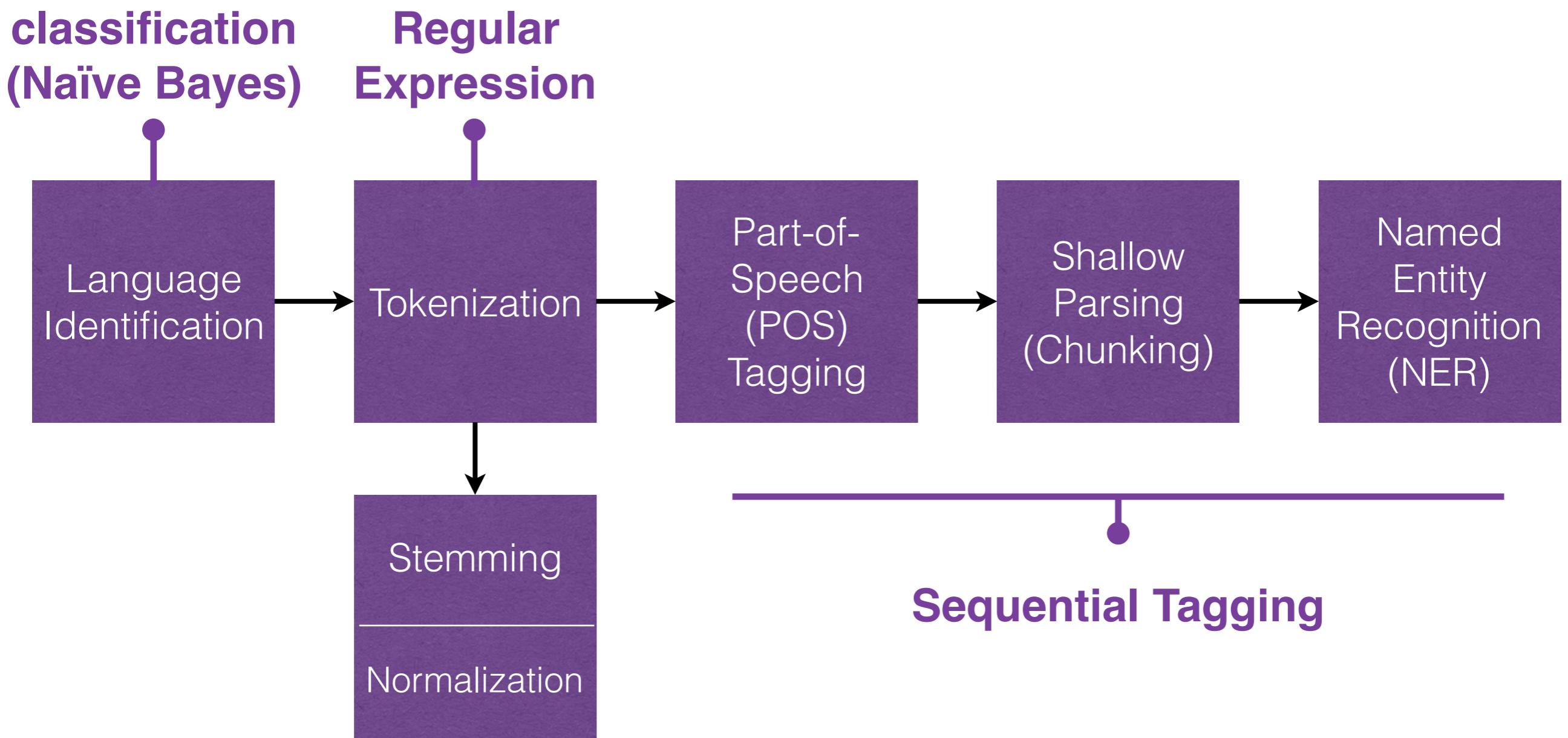
CSE 5539-0010 Ohio State University
Instructor: Wei Xu
Website: socialmedia-class.org

some slides are adapted from Michael Collins, Dan Jurafsky, Richard Socher,, Chris Manning

NLP Pipeline



NLP Pipeline



Part-of-Speech (POS) Tagging

| | |
|----------|-----|
| Cant | MD |
| wait | VB |
| for | IN |
| the | DT |
| ravens | NNP |
| game | NN |
| tomorrow | NN |
| ... | : |
| go | VB |
| ray | NNP |
| rice | NNP |
| !!!!!! | . |



Named Entity Recognition

India vs Australia 2014-15 , 4th Test in Sydney

Samsung to launch Galaxy S6 in March

New Suits and Brooklyn Nine-Nine tomorrow ... Happy days

sportsteam sportsteam geo-loc

India vs Australia 2014-15 , 4th Test in Sydney

company product

Samsung to launch Galaxy S6 in March

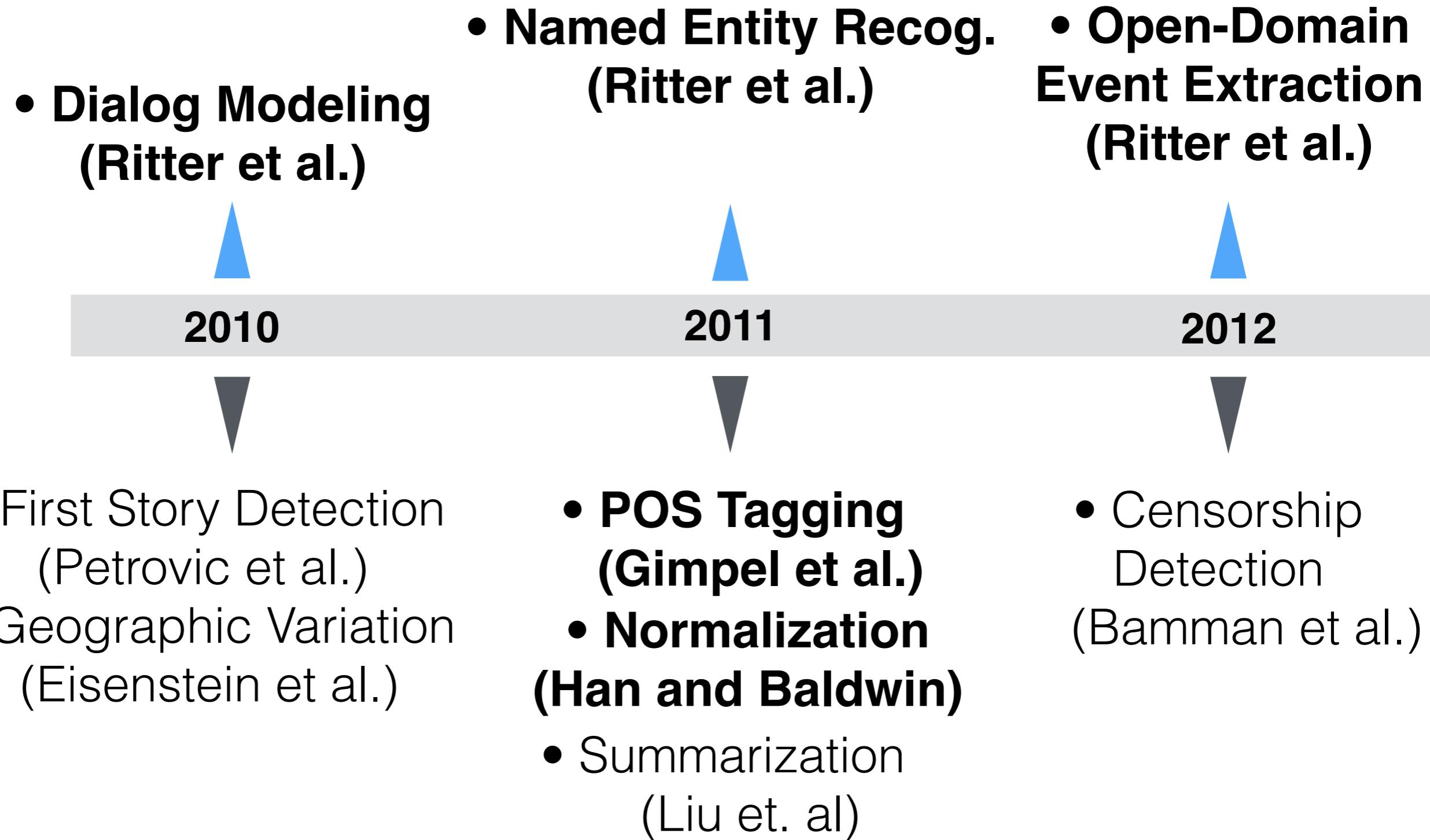
tvshow tvshow

New Suits and Brooklyn Nine-Nine tomorrow ... Happy days

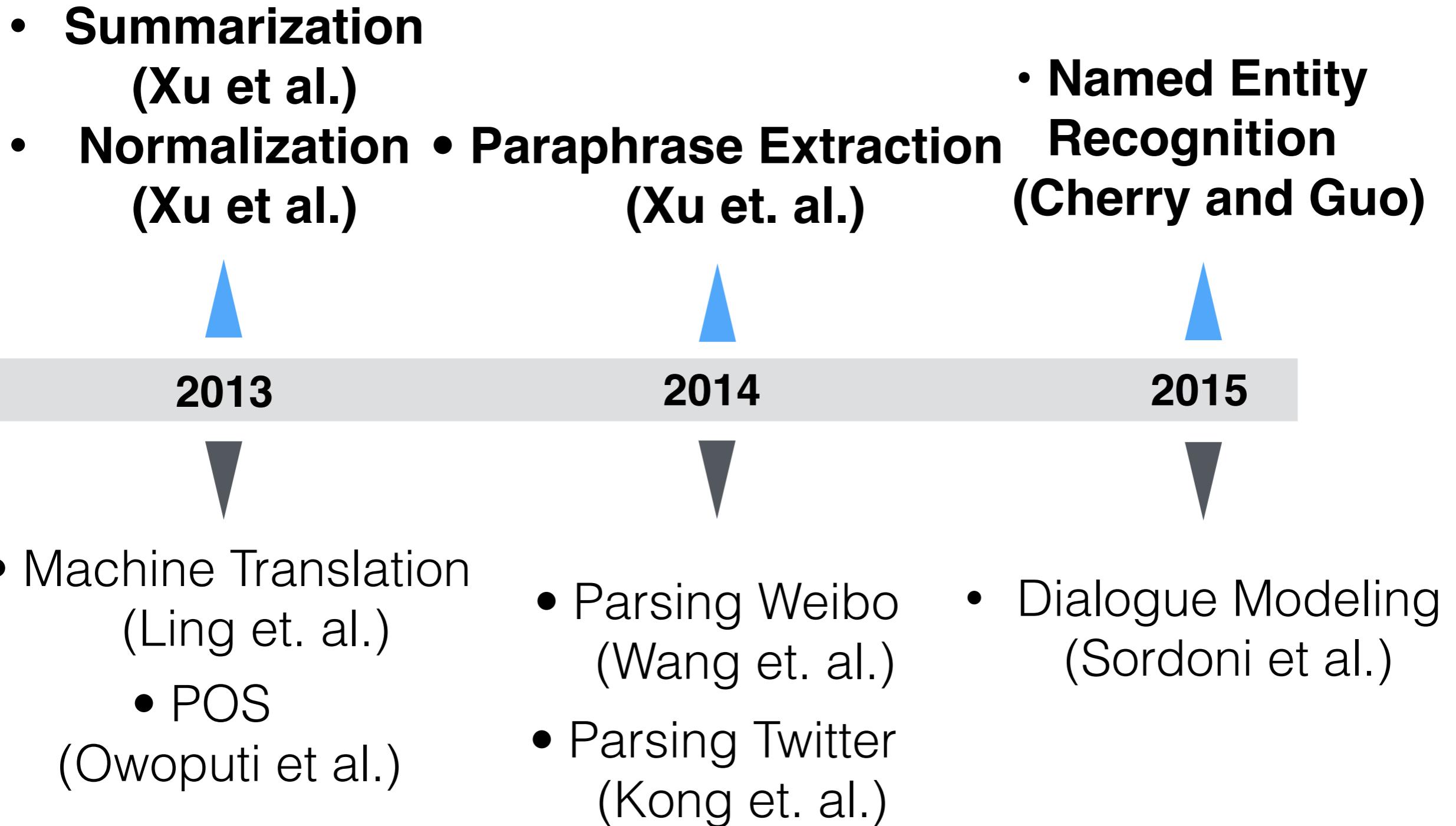
Source: Strauss, Toma, Ritter, de Marneffe, Xu

Results of the WNUT16 Named Entity Recognition Shared Task (WNUT@COLING 2016)

Timeline of NLP on Microblogs

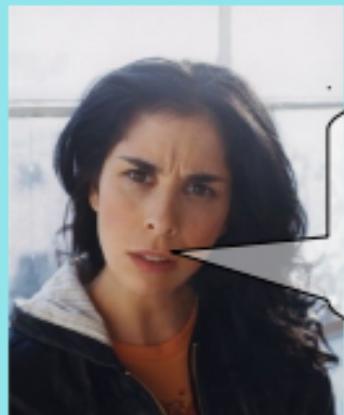


Timeline of NLP on Microblogs



BAD LANGUAGE!

...on the INTERNET!!



Boom! Ya ur
website suxx bro

...dats why pluto is pluto
it can neva be a star



michelle obama great.
job. and. whit all my.
respect she. look. great.
congrats. to. her.



I now h v an iphone

What can we do about it?

*Why don't they just write **NORMALLY**??*

*Can our software ever **ADAPT**???*

Jacob EISENSTEIN
GEORGIA Institute of **TECH**nology

How does language go bad?

Illiteracy? No.

(Tagliamonte and Denis 2008;
Drouin and Davis 2009)

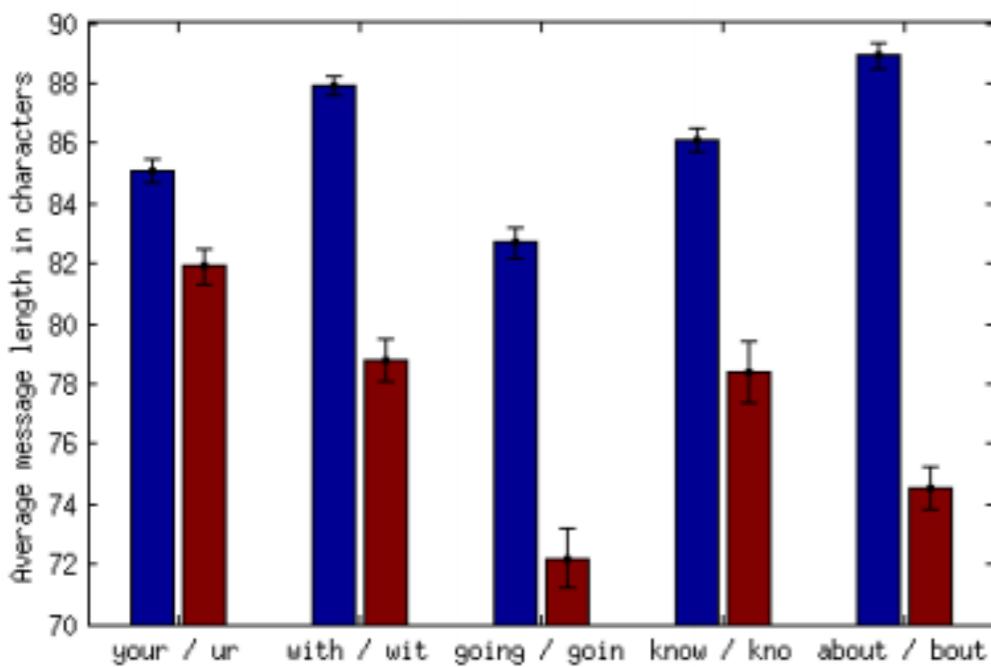


rob delaney @robdelaney

1 Jun

Great. Now a bunch of illiterate teens claim to be "powning" me with their insults. Heads up jerks my wife & children love me & are proud of
[Expand](#) [Reply](#) [Classic RT](#) [Retweet](#) [Favorite](#) [More](#)

Length limits? (probably not)

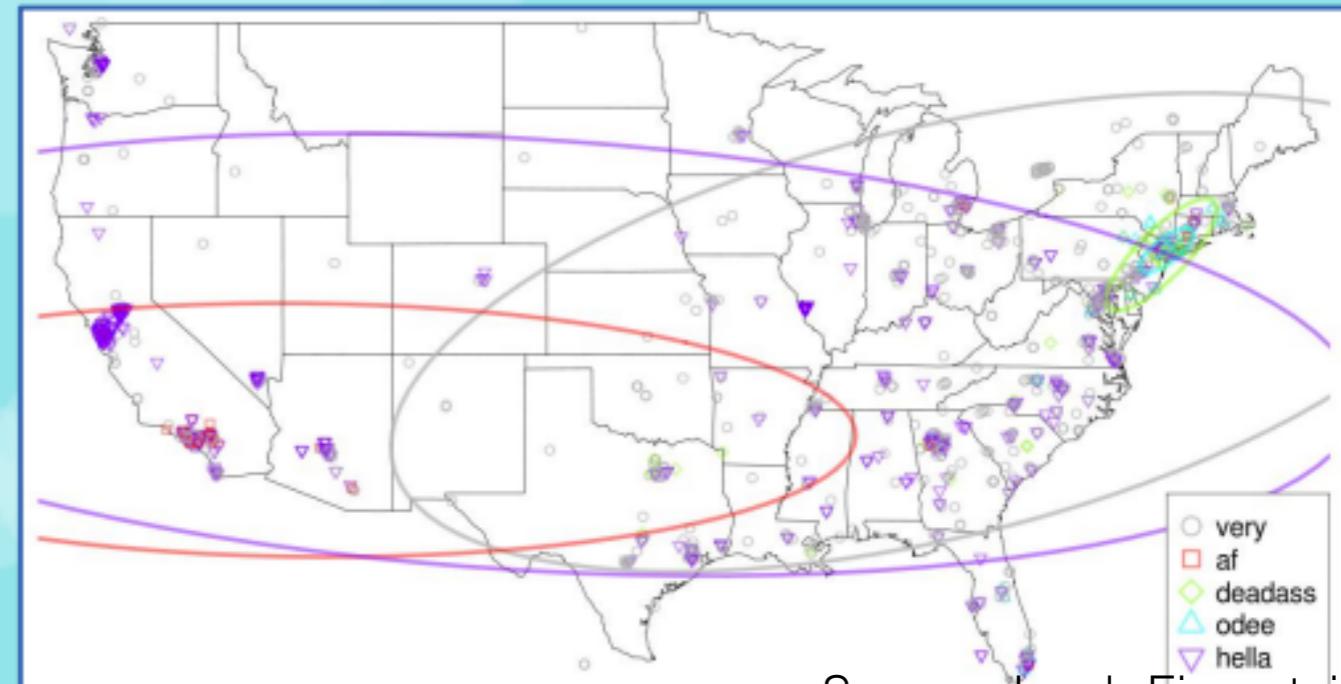


Hardware input constraints? (Gouws et al 2011)



Social variables

- Non-standard language does *identity work*, signaling authenticity, solidarity, etc.
- Social variation is usually inhibited in written language, but social media is less regulated than other written genres.



Source: Jacob Eisenstein

Why is Social Media Text “Bad”?

- Lack of literacy? no [Drouin and Davis, 2009]
- Length restrictions? not primarily [Eisenstein, 2013]
- Text input method? to some degree, yes [Gouws et al., 2011]
- Pragmatics (mimicking prosodic effects etc. in speech)? yeeeess [Eisenstein, 2013]
- Social variables/markers of social identity? blood oath! [Eisenstein, 2013]

Why is Social Media Text “Bad”?

- Pragmatics (mimicking prosodic effects etc. in speech)? [yeeeess](#) [Eisenstein, 2013]

HELLA 🔊

Derived from "hell of a lot". Similar to "very, really, a lot," etc.

Used mostly in Northern California though has been heard in other parts of CA and even in the media such as an infamous "hella" South Park episode. (Cartman used it outside of its meaning to annoy Kyle.)

Before: There's a hell of a lot of beer in that fridge.

After: There's hella beer in that fridge.

As "very" or "really":

"That's hella far away!"

Why is Social Media Text “Bad”?

- Social variables/markers of social identity? **blood oath!**
[Eisenstein, 2013]



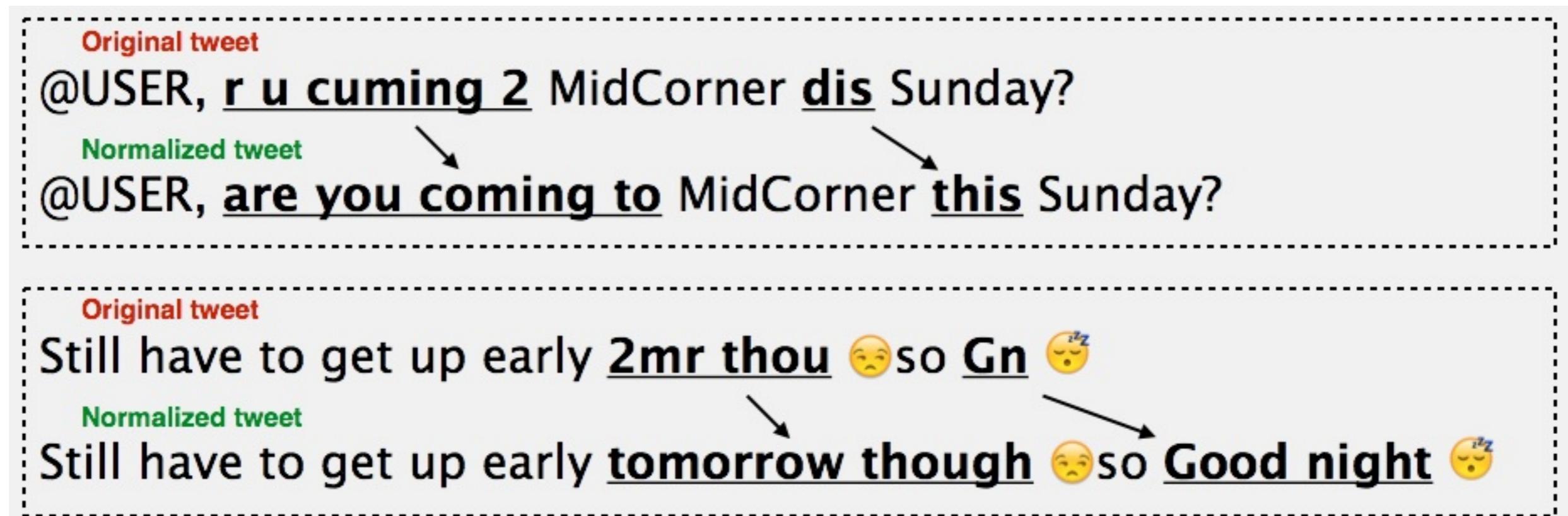
“I would like to believe he’s **sick** rather than just mean and evil.”



“You could’ve been getting down to this **sick** beat.”

Text Normalization

- convert non-standard words to standard



Source: Tim Baldwin, Marie de Marneffe, Han Bo, Young-Bum Kim, Alan Ritter, Wei Xu
Shared Tasks of the 2015 Workshop on Noisy User-generated Text:
Twitter Lexical Normalization and Named Entity Recognition

An Unsupervised Learning Method: **(1) Brown Clustering**

- Input:
 - a (large) text corpus
- Output:
 1. a partition of words into word clusters
 2. or a hierarchical word clustering (generalization of 1)

Brown Clustering

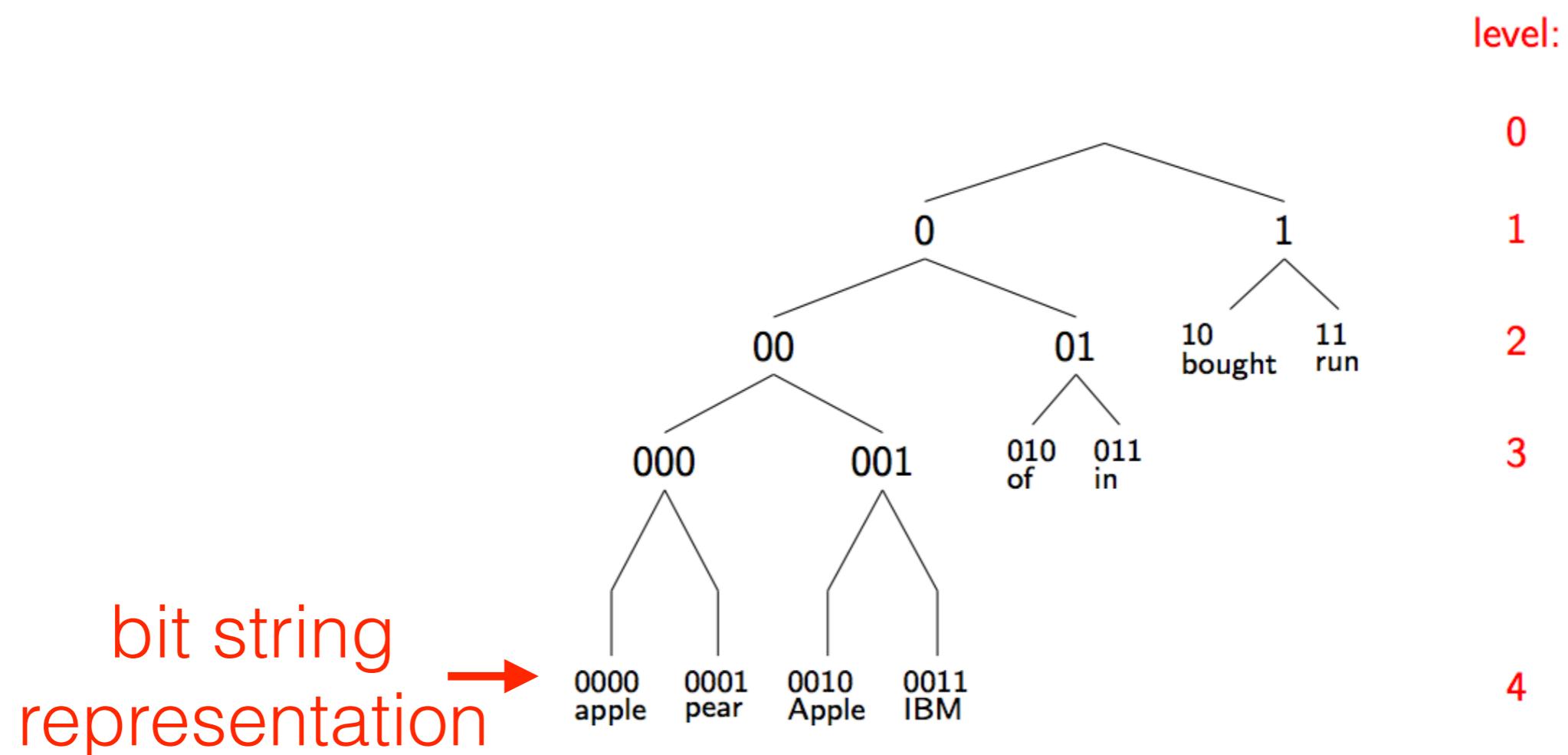
- Example Clusters (from Brown et al. 1992)

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody

Source: Miller, Guinness, Zamanian (NAACL 2004)
Name Tagging with Word Clusters and Discriminative Training

Hierarchical Word Clustering

- Each intermediate node is a cluster:



Hierarchical Word Clustering

| | |
|-----------------|--------------------------|
| mailman | 10000011010111 |
| salesman | 100000110110000 |
| bookkeeper | 1000001101100010 |
| troubleshooter | 10000011011000110 |
| bouncer | 10000011011000111 |
| technician | 1000001101100100 |
| janitor | 1000001101100101 |
| saleswoman | 1000001101100110 |
| ... | |
| Nike | 1011011100100101011100 |
| Maytag | 10110111001001010111010 |
| Generali | 10110111001001010111011 |
| Gap | 1011011100100101011110 |
| Harley-Davidson | 10110111001001010111110 |
| Enfield | 101101110010010101111110 |
| genus | 101101110010010101111111 |
| Microsoft | 10110111001001011000 |
| Ventrity | 101101110010010110010 |
| Tractebel | 1011011100100101100110 |
| Synopsys | 1011011100100101100111 |
| WordPerfect | 1011011100100101101000 |
| | |
| John | 1011100100000000000 |
| Consuelo | 1011100100000000001 |
| Jeffrey | 1011100100000000010 |
| Kenneth | 10111001000000001100 |
| Phillip | 101110010000000011010 |
| WILLIAM | 101110010000000011011 |
| Timothy | 10111001000000001110 |

- Example Clusters
(from Miller et al. 2004)

Hierarchical Word Clustering

mailman
salesman
bookkeeper
troubleshooter
bouncer
technician
janitor
saleswoman

10000011010111
100000110110000
1000001101100010
10000011011000110
10000011011000111
1000001101100100
1000001101100101
1000001101100110

...
Nike
Maytag
Generali
Gap
Harley-Davidson
Enfield

1011011100100101011100
10110111001001010111010
10110111001001010111011
1011011100100101011110
10110111001001010111110
101101110010010101111110
1011011100100101011111110
1011011100100101011111111
10110111001001011000
101101110010010110010
1011011100100101100110
1011011100100101100111
1011011100100101101000

....

John
Consuelo
Jeffrey
Kenneth
Phillip
WILLIAM
Timothy

101110010000000000
101110010000000001
101110010000000010
10111001000000001100
101110010000000011010
101110010000000011011
10111001000000001110

- Example Clusters
(from Miller et al. 2004)

word cluster features
(bit string prefix)

Challenges in Twitter

2m 2ma 2mar 2mara 2maro 2marrow 2mor 2mora
2moro 2morow 2morr 2morro 2orrow 2moz 2mr
2mro 2mrrw 2mrw 2mw tmmrw tmo tmoro tmorrow
tmoz tmr tmro tmrow tmrrow tmrrw tmrw tmrww tmw
tomaro tomarow tomarro tomorrow tomm
tommarow tommarrow tommoro tommorow
tommorrow tommorw tommrow tomo tomolo tomoro
tomorow tomorro tomorrw tomoz tomrw tomz

Clusters in Twitter NER

| System | Fin10Dev | Rit11 | Fro14 | Avg |
|-----------------------------------------------------------|----------|-------|-------|------|
| CoNLL + Brown + Vector + Reps | 27.3 | 27.1 | 29.5 | 28.0 |
| | 38.4 | 39.4 | 42.5 | 40.1 |
| | 40.8 | 40.4 | 42.9 | 41.4 |
| | 42.4 | 42.2 | 46.2 | 43.6 |
| Fin10 + Brown + Vector + Reps | 36.7 | 29.0 | 30.4 | 32.0 |
| | 59.9 | 53.9 | 56.3 | 56.7 |
| | 61.5 | 56.4 | 58.4 | 58.8 |
| | 64.0 | 58.5 | 60.2 | 60.9 |
| CoNLL+Fin10 + Brown + Vector + Reps + Weights | 44.7 | 39.9 | 44.2 | 42.9 |
| | 54.9 | 52.9 | 58.5 | 55.4 |
| | 58.9 | 55.2 | 59.9 | 58.0 |
| | 58.9 | 56.4 | 61.8 | 59.0 |
| | 64.4 | 59.6 | 63.3 | 62.4 |

Table 5: Impact of our components on Twitter NER performance, as measured by F1, under 3 data scenarios.

Source: Colin Cherry, Hongyu Guo (NAACL 2015)

The Unreasonable Effectiveness of Word Representations for Twitter Named Entity Recognition

Clusters in Twitter NER

Brown clusters, for each i s.t. $s \leq i < t$:

$$\{[y_j, brn(n, x_i), n]\}_{n \in \{2, 4, 8, 12\}},$$
$$\{[y_j, er_{s,t}(i), brn(n, x_i), n]\}_{n \in \{2, 4, 8, 12\}}$$

Word vectors, for each i s.t. $s \leq i < t$:

$$\{[y_j, n] = w2v(n, x_i)\}_{n=1}^{300},$$
$$\{[y_j, er_{s,t}(i), n] = w2v(n, x_i)\}_{n=1}^{300}$$

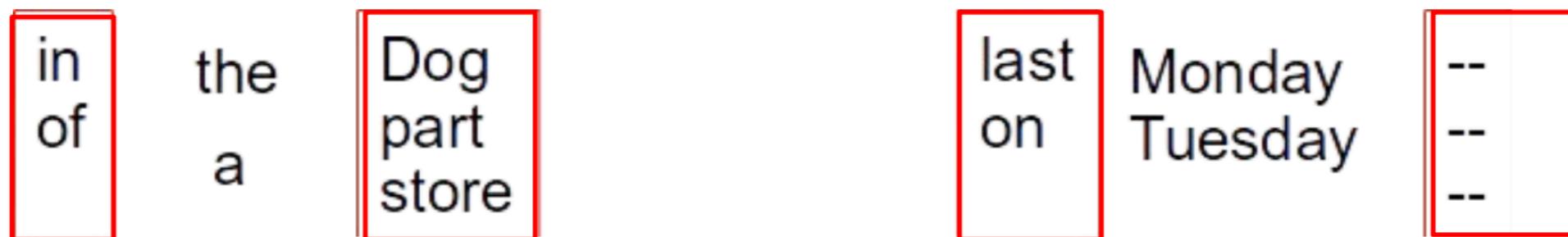
Table 2: Word representation features in $\phi(s, t, y_j, x)$.
 $brn(n, x_i)$ maps a word x_i to the first n bits of its Brown cluster bit sequence. $w2v(n, x_i)$ maps x_i to the n^{th} component of its word vector, and $[str] = v$ stands for a real-valued feature with name str and value v .

Source: Colin Cherry, Hongyu Guo (NAACL 2015)

The Unreasonable Effectiveness of Word Representations for Twitter Named Entity Recognition

Brown Clustering

- The Intuition:
 - similar words appear in similar contexts
 - more precisely: similar words have similar distributions of words to their immediate left and right



Brown Clustering Algorithm

- An agglomerative clustering algorithm:
 - take the top m most frequent words, put each into its own cluster, $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_m$
 - repeat for $i = (m+1) \dots |V|$
 - create a new cluster \mathbf{c}_{m+1} for the i 'th most frequent word
 - choose two clusters from $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{m+1}$ to be merged, which give the highest **Quality** based on a *training corpus*

Brown Clustering Algorithm

- maximize the **Quality** function that score a given partitioning **C**: **parameters**

$$\begin{aligned} \text{Quality}(C) &= \sum_i^n \log e(w_i | C(w_i)) q(C(w_i) | C(w_{i-1})) \\ &= \sum_{c=1}^k \sum_{c'=1}^k p(c, c') \log \frac{p(c, c')}{p(c)p(c')} + G \end{aligned}$$

- **n(c)** : count of class **c** seen in the corpus
- **n(c,c')** : counts of **c'** seen following **c**

$$p(c, c') = \frac{n(c, c')}{\sum_{c, c'} n(c, c')}$$

$$p(c) = \frac{n(c)}{\sum_c n(c)}$$

Brown Clustering

 [percyliang / brown-cluster](#) Watch ▾ 29 Star 203 Fork 79

[Code](#) [Issues 9](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Pulse](#) [Graphs](#)

C++ implementation of the Brown word clustering algorithm.



Branch: master ▾ [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download ▾](#)

| percyliang Merge pull request #15 from mannby/large_corpora | ... | Latest commit d9dff3b on Mar 26 |
|-------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------|
| basic | Enable >= 2^31 tokens in input data | 8 months ago |
| cluster-viewer | cluster viewer final | 3 years ago |
| .gitignore | turn on -O3 optimization, add gitignore | 3 years ago |
| Makefile | small fix to makefile | 3 years ago |
| README | Merge branch 'master' of https://github.com/percyliang/brown-cluster | 3 years ago |
| input.txt | Version 1.2 | 4 years ago |
| output.txt | Version 1.3: incorporate Chris Dyer's g++ compatibility changes; smal... | 4 years ago |
| wcluster.cc | Enable >= 2^31 tokens in input data | 8 months ago |

[README](#)

Implementation of the Brown hierarchical word clustering algorithm.
Percy Liang
Release 1.3
2012.07.24

Input: a sequence of words separated by whitespace (see `input.txt` for an example).
Output: for each word type, its cluster (see `output.txt` for an example).
In particular, each line is:
`<cluster represented as a bit string> <word> <number of times word occurs in input>`

Word Vector Representations

(a.k.a. “word embeddings”)

- 4 kinds of vector semantic models
 1. Hard clustering (e.g. Brown clustering)
 2. Soft clustering (e.g. SVD, LSA, LDA)
 3. Neural Network inspired models
(e.g. skip-grams and CBOW in word2vec)
 4. Mutual-information weighted word co-occurrence metrics

dense



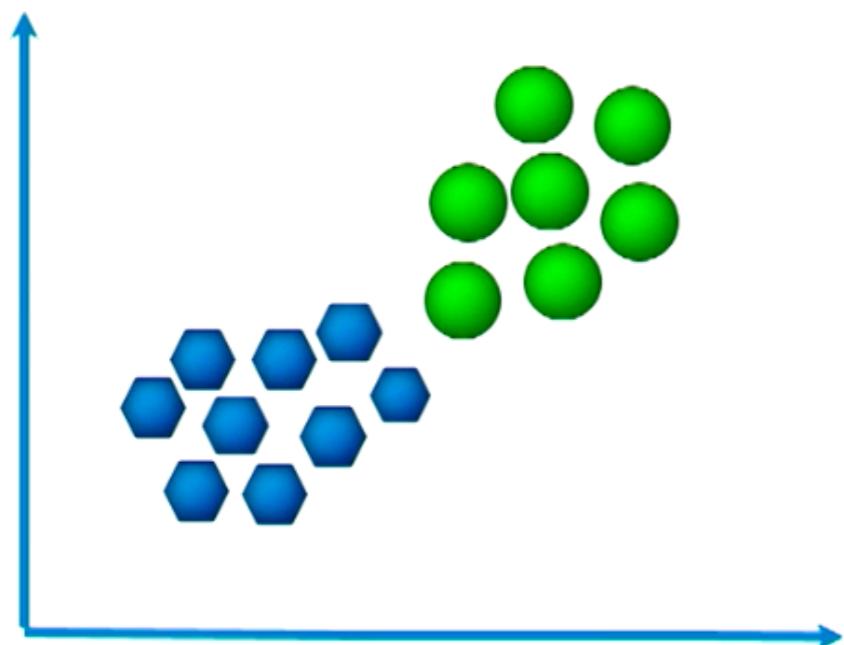
sparse



Hard vs. Soft Clustering

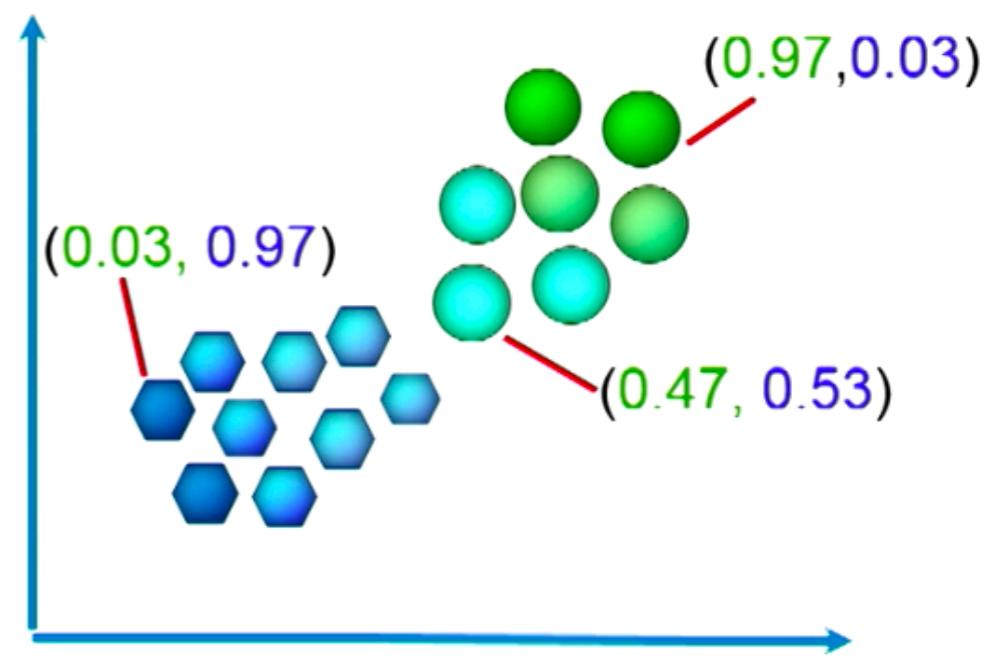
Hard clustering

Each observation belongs to exactly one cluster



Soft clustering

An observation can belong to more than one cluster to a certain degree (e.g. likelihood of belonging to the cluster)



In Contrast To

represent word meaning by a taxonomy like WordNet

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(pandaclosure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

synonym sets (good):

S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced, proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
...
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good

In Contrast To

represent word meaning by a taxonomy like WordNet

- problems with this discrete representation:
 - missing new words (impossible to keep up-to-date): *wicked, badass, nifty, crack, ace, wizard, genius, ninja*
 - requires human labor to create and adapt
 - hard to compute accurate word similarity
 - and apparently not enough to handle social media data!

Distributional Intuition

- From context words, human can guess a word's meaning:

A bottle of ***tesgüino*** is on the table

Everybody likes ***tesgüino***

Tesgüino makes you drunk

We make ***tesgüino*** out of corn.

“You shall know a word by the company it keeps”

— J. R. Firth 1957

Distributional Intuition

- From context words, human can guess a word's meaning:

A bottle of ***tesgüino*** is on the table

Everybody likes ***tesgüino***

Tesgüino makes you drunk

We make ***tesgüino*** out of corn.

- similar words = similar contexts = similar vectors
- word meaning is represented by a vector of numbers

Simple Co-occurrence Vectors

- Option #1: word-document co-occurrence counts

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---------|----------------|---------------|---------------|---------|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

this will give general topics (e.g. sports terms will have similar entries), leading to **Latent Semantic Analysis**

Simple Co-occurrence Vectors

- Option #2: use a sliding window over a big corpus of text and count word co-occurrences:

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 |

this captures both syntactic (POS) and semantic information

Simple Co-occurrence Vectors

- Problems with this representation of raw counts:
 - increase in size with vocabulary
 - high dimensionality and very sparse!
 - not a great measure of association between words:

“the” and “of” are very frequent, but maybe not the most discriminative

- unable to capture word order

“I like NLP” and “NLP like I” will have same representation

Lower Dimensional Vectors

- **The Idea:** use dense vectors to store “most” of the important information in a fixed, small number of dimensions
- usually around 25 ~1000 dimensions

Lower Dimensional Vectors

- Word meaning is represented as a **dense** vector

“linguistic” =

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

How to reduce the dimensionality?

(2) Matrix Factorization

- Singular Value Decomposition (SVD)

$$\begin{array}{c} m \\ \hline n \end{array} X \quad = \quad n \begin{array}{c} r \\ \hline U_1 U_2 U_3 \cdots \end{array} \quad r \begin{array}{c} r \\ \hline S_1 S_2 S_3 \cdots 0 \\ 0 \quad \ddots \quad S_r \end{array} \quad r \begin{array}{c} m \\ \hline V_1 V_2 V_3 \cdots \\ \vdots \quad \vdots \end{array}$$
$$\begin{array}{c} m \\ \hline n \end{array} \hat{X} \quad = \quad n \begin{array}{c} k \\ \hline \hat{U}_1 \hat{U}_2 \hat{U}_3 \cdots \end{array} \quad k \begin{array}{c} k \\ \hline \hat{S}_1 \hat{S}_2 \hat{S}_3 \cdots 0 \\ 0 \quad \ddots \quad \hat{S}_k \end{array} \quad k \begin{array}{c} m \\ \hline \hat{V}_1 \hat{V}_2 \hat{V}_3 \cdots \\ \vdots \quad \vdots \end{array}$$

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

SVD Word Vectors

example corpus:

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

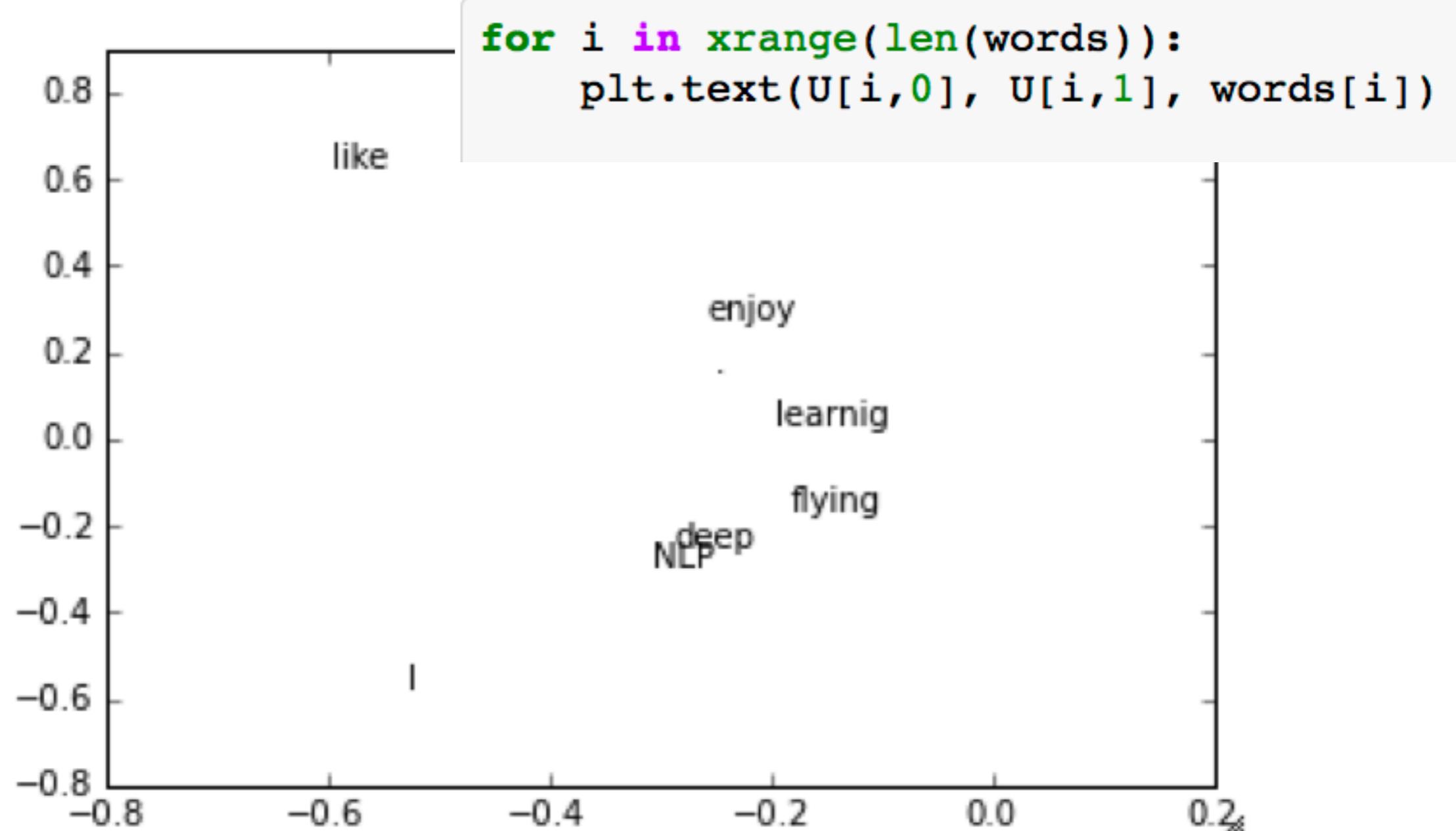
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
          "deep", "learnig", "NLP", "flying", ".."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [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)
```

SVD Word Vectors

- plot first 2 columns of U corresponding to the 2 biggest singular values:



Some Hacks

- Problem: function words (“the”, “he”, “has”) are too frequent → syntax has too much impact.
 - fixes: cap the counts, or ignore them all
- ramped windows that count closer words more
- etc ...

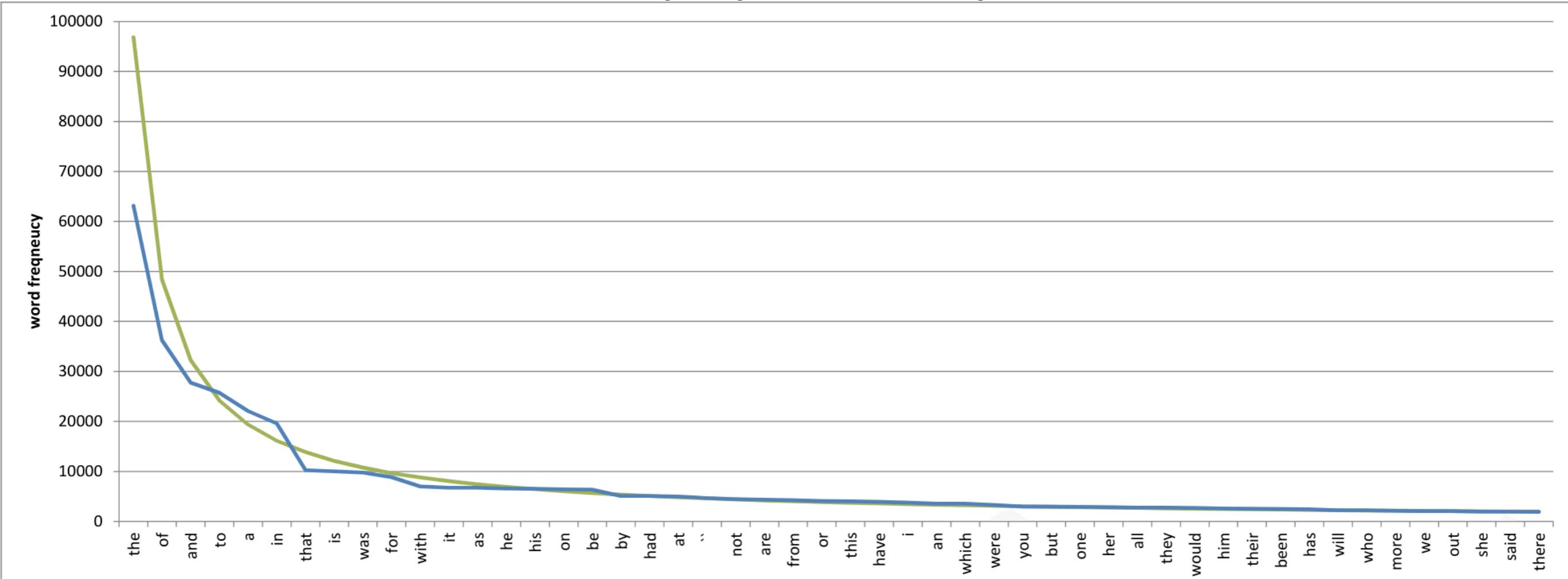
Source: Rohde et al. (2005)

An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

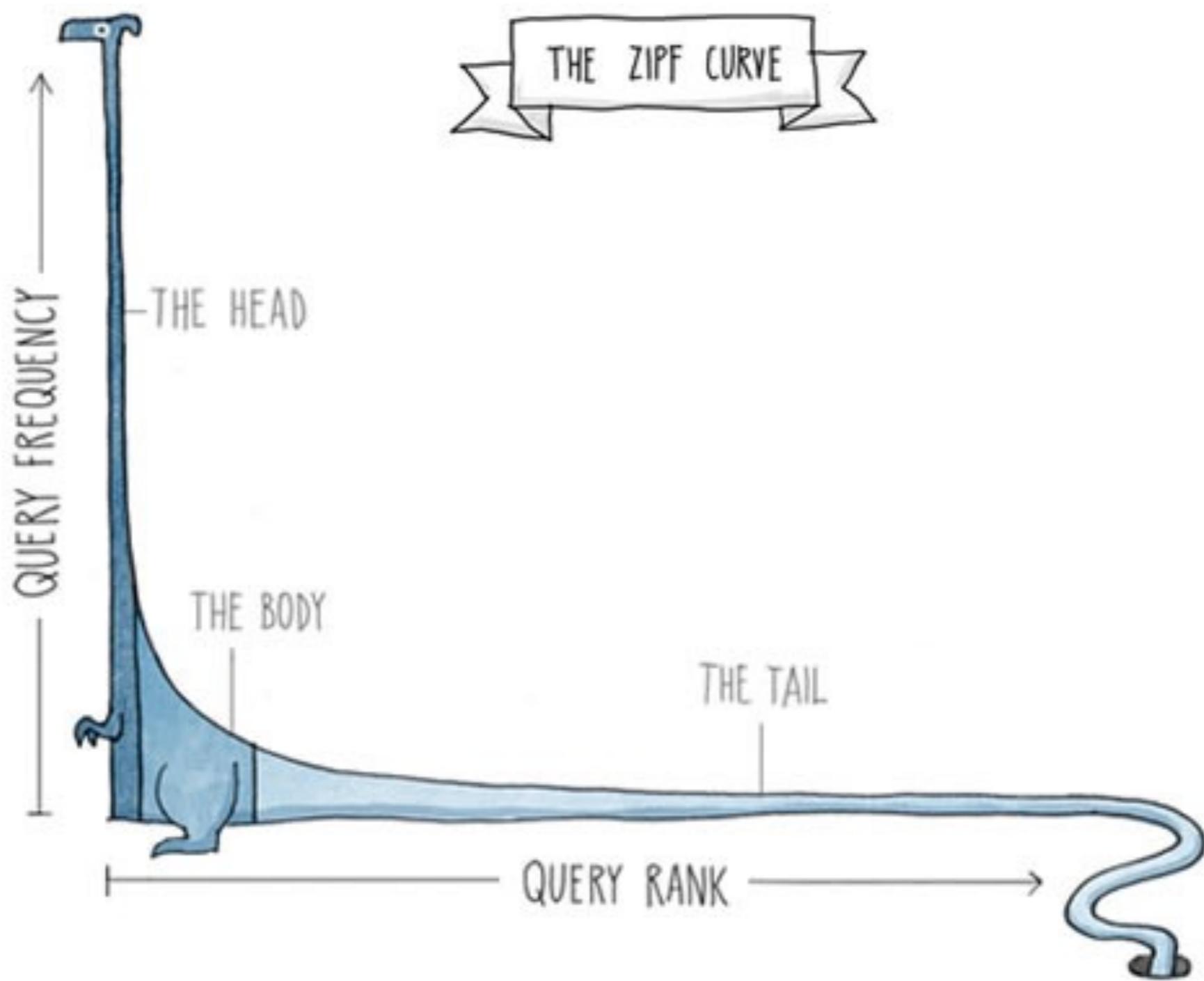
Zipf's (Power) Law

- frequency of word is inversely proportional to its rank in the frequency table

word frequency in the Brown corpus

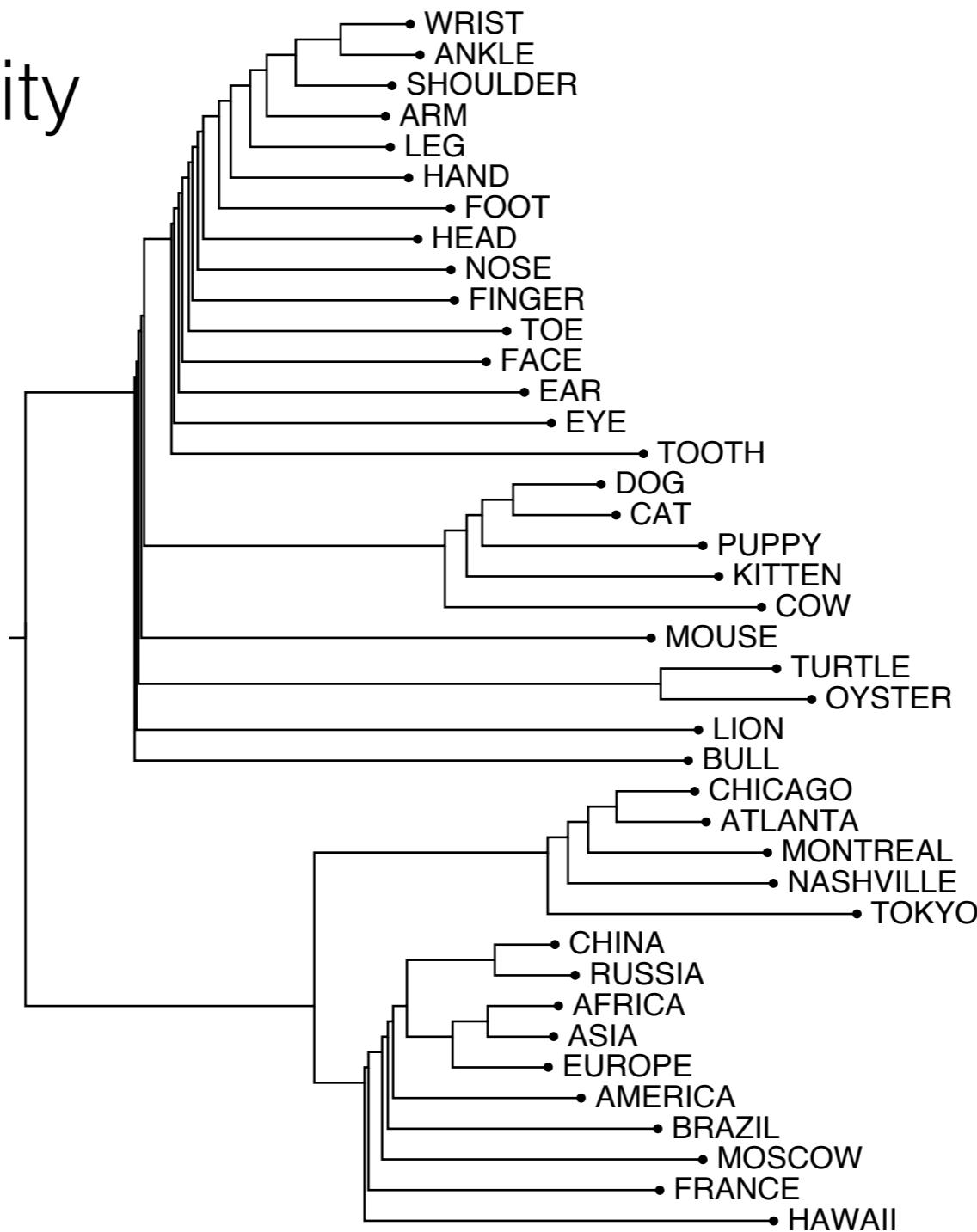


Zipf's (Power) Law



Clustering Vectors

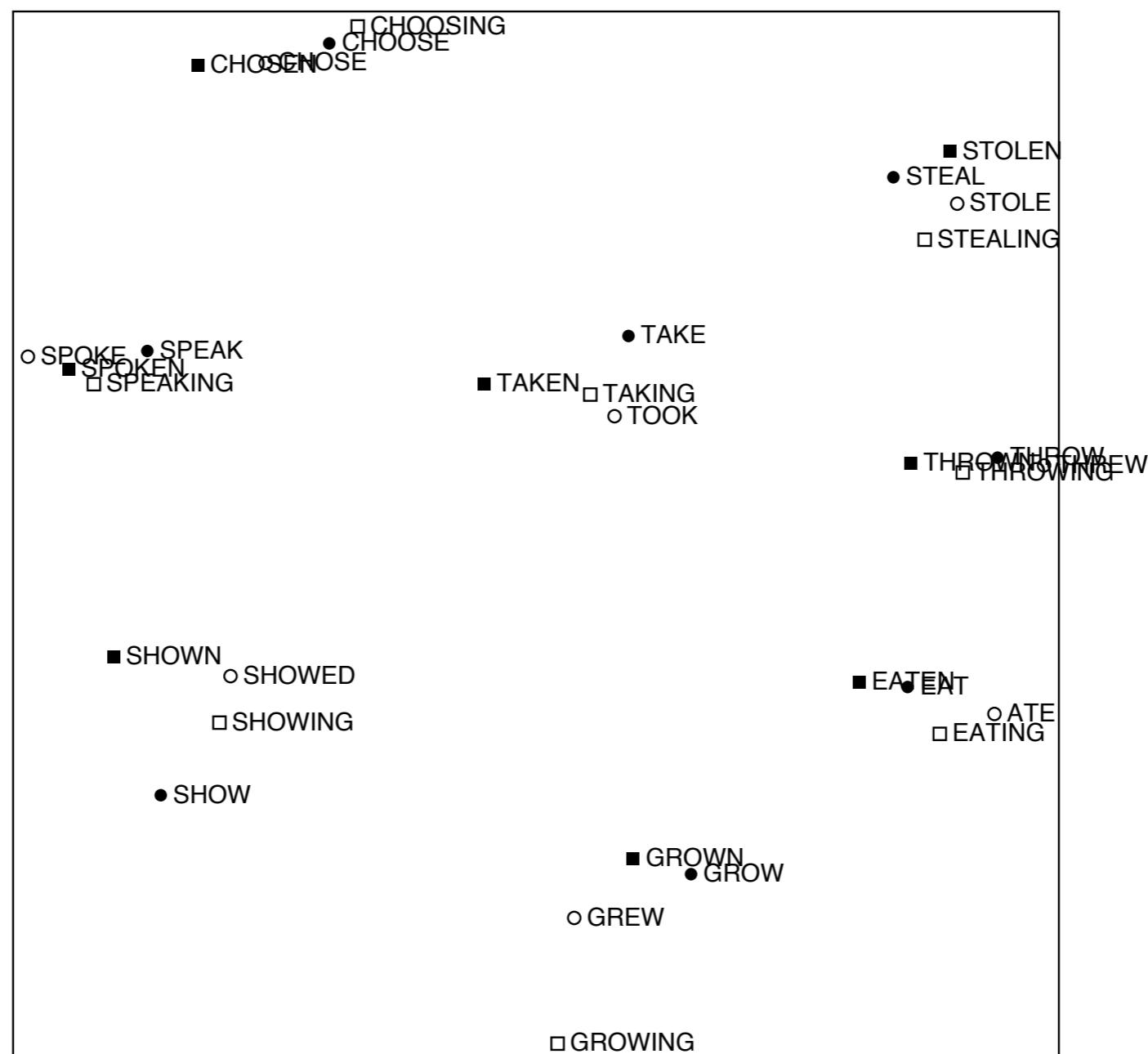
- visualize similarity



Source: Rohde et al. (2005)

An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

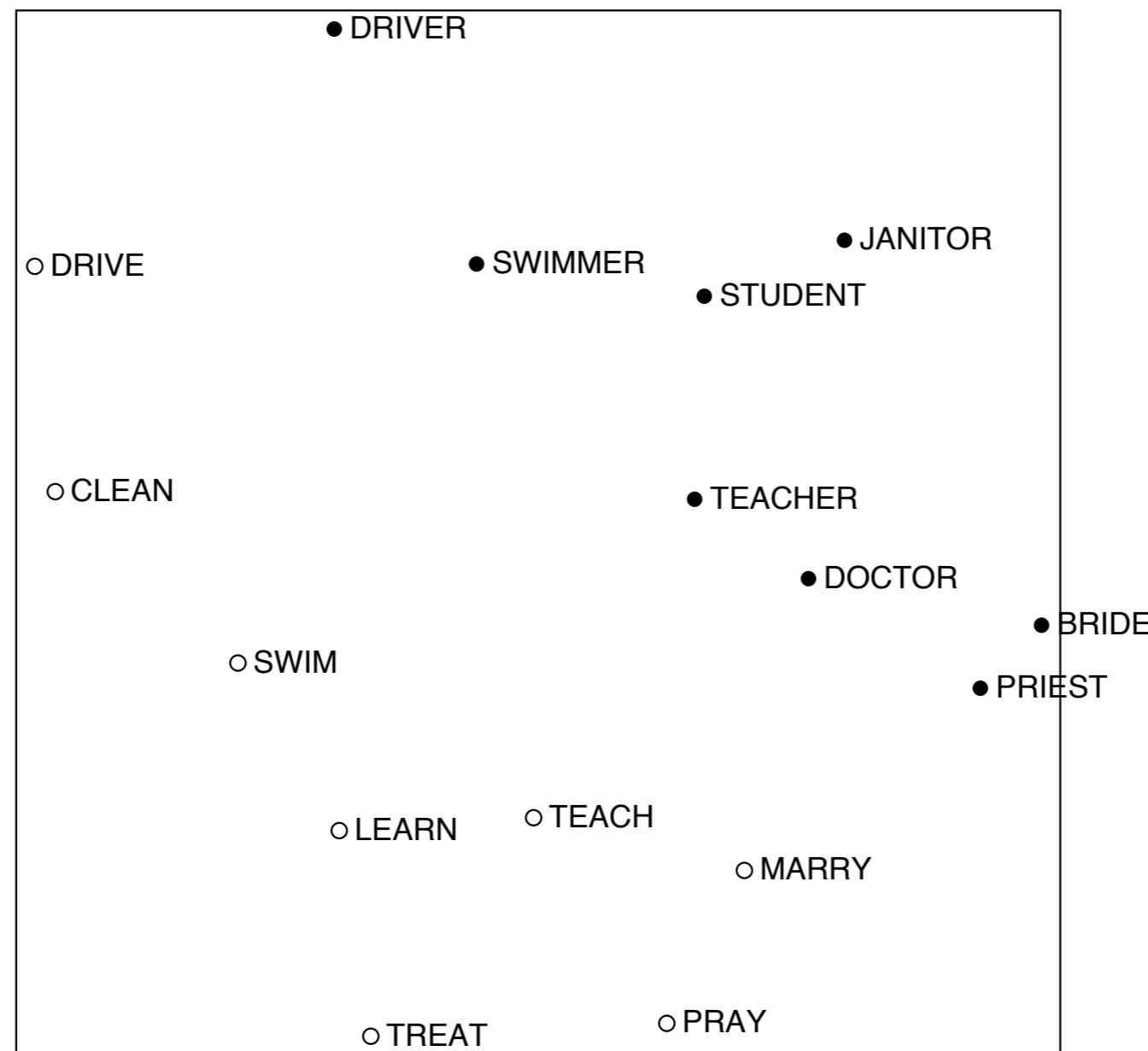
Interesting Syntactic Patterns



Source: Rohde et al. (2005)

An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

Interesting Semantic Patterns



Source: Rohde et al. (2005)

An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

SVD Word Vectors

- Still some problems:
 - computational cost scales quadratically for $m \times n$ matrix — $O(mn^2)$ when $n < m$
 - hard to use large corpus (and vocabulary)
 - hard to incorporate new words or documents

(3) Neural Word Embeddings

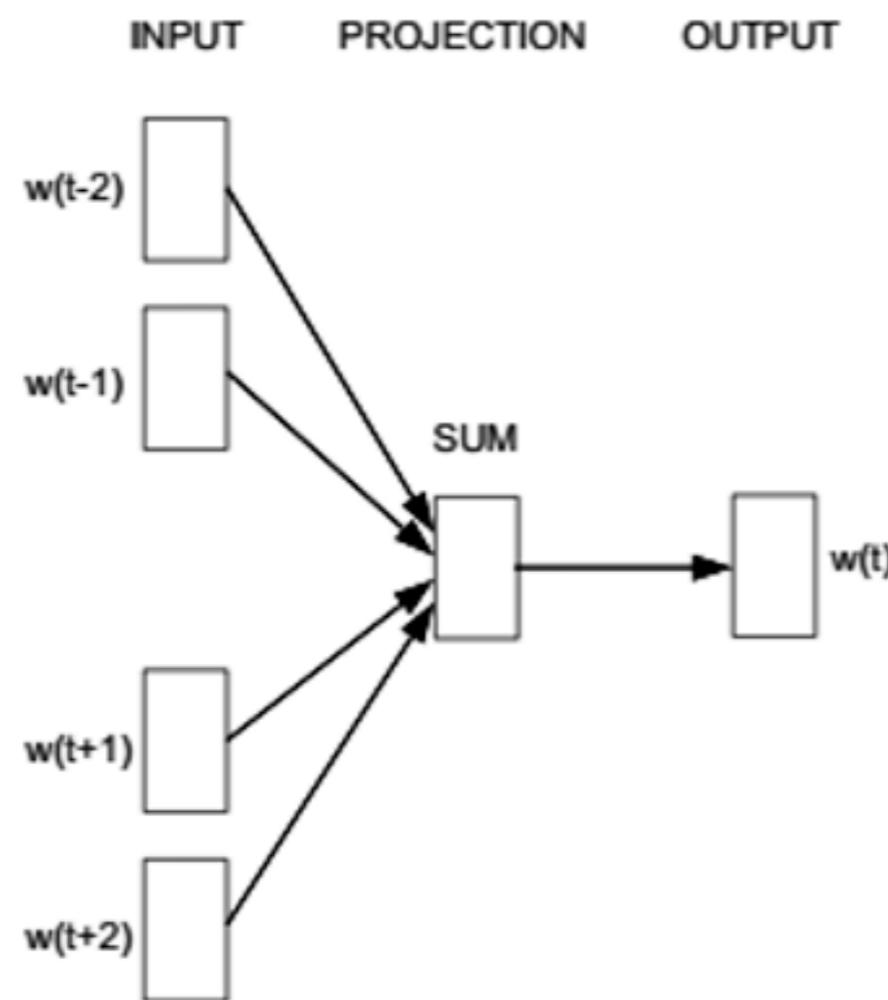
- **The Idea:** directly learn low-dimensional word vectors
- ... can go back to 1980s:
 - Learning Representations by Back-Propagating Errors (Rumelhart et al., 1986)
 - **A Neural Probabilistic Language Model** (Bengio et al., 2003)
 - NLP from Scratch (Collobert & Weston, 2008)
 - **Word2vec** (Mikolov et al. 2013)

Neural Word Embeddings

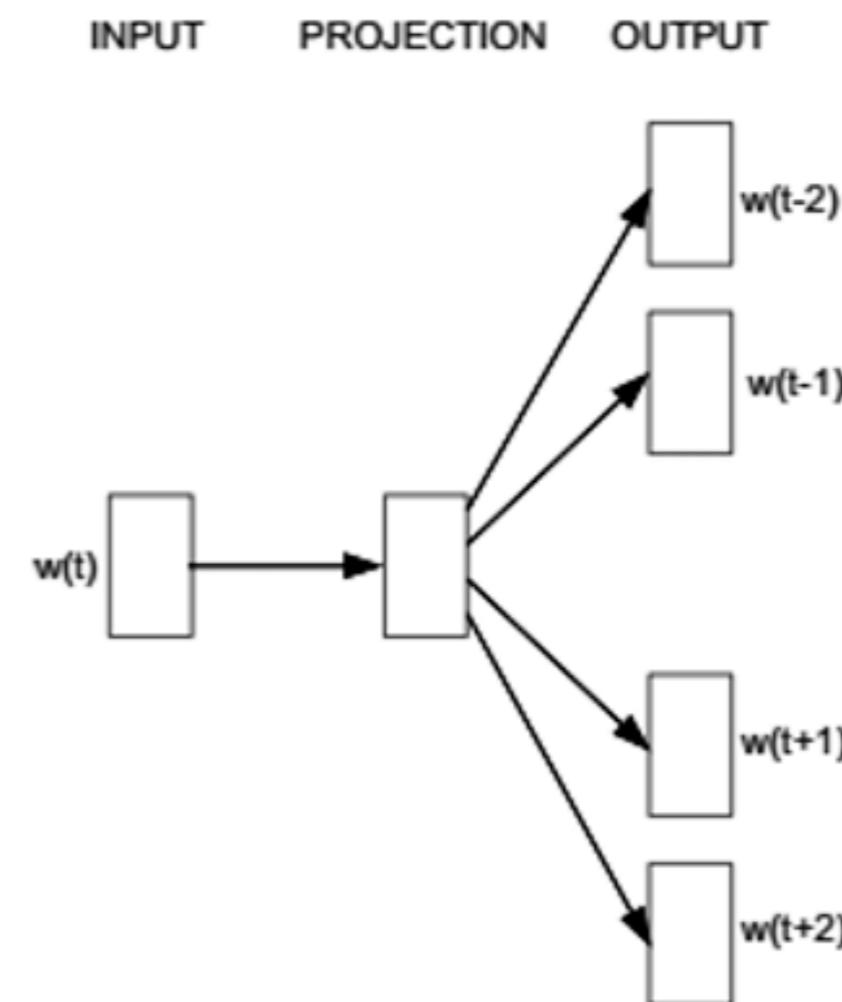
- **The Basic Idea:**
 - We define a model that aims to predict a word given its context words (word vectors), which has a loss function, e.g. $J = 1 - P(\text{context} \mid w_t)$
 - We look at many positions of t in a big text corpus,
 - and keep adjusting the word vectors to minimize this loss.

Word2vec

- simple and efficient



CBOW



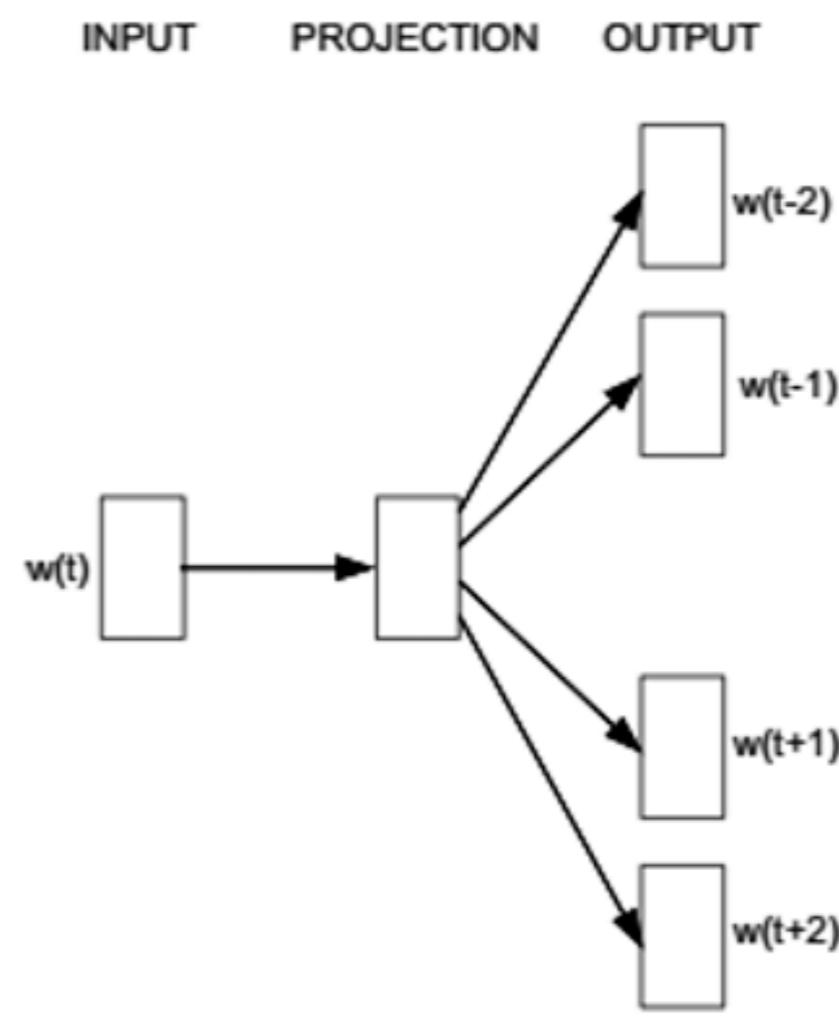
Skip-gram

Source: Mikolov et al. (NIPS 2013)

Distributed Representations of Words and Phrases and their Compositionality

Word2vec

- Skip-gram — predicts surrounding “outside” words given the “center” word



Skip-gram

Word2vec

- Skip-gram — predicts surrounding “outside” words given the “center” word

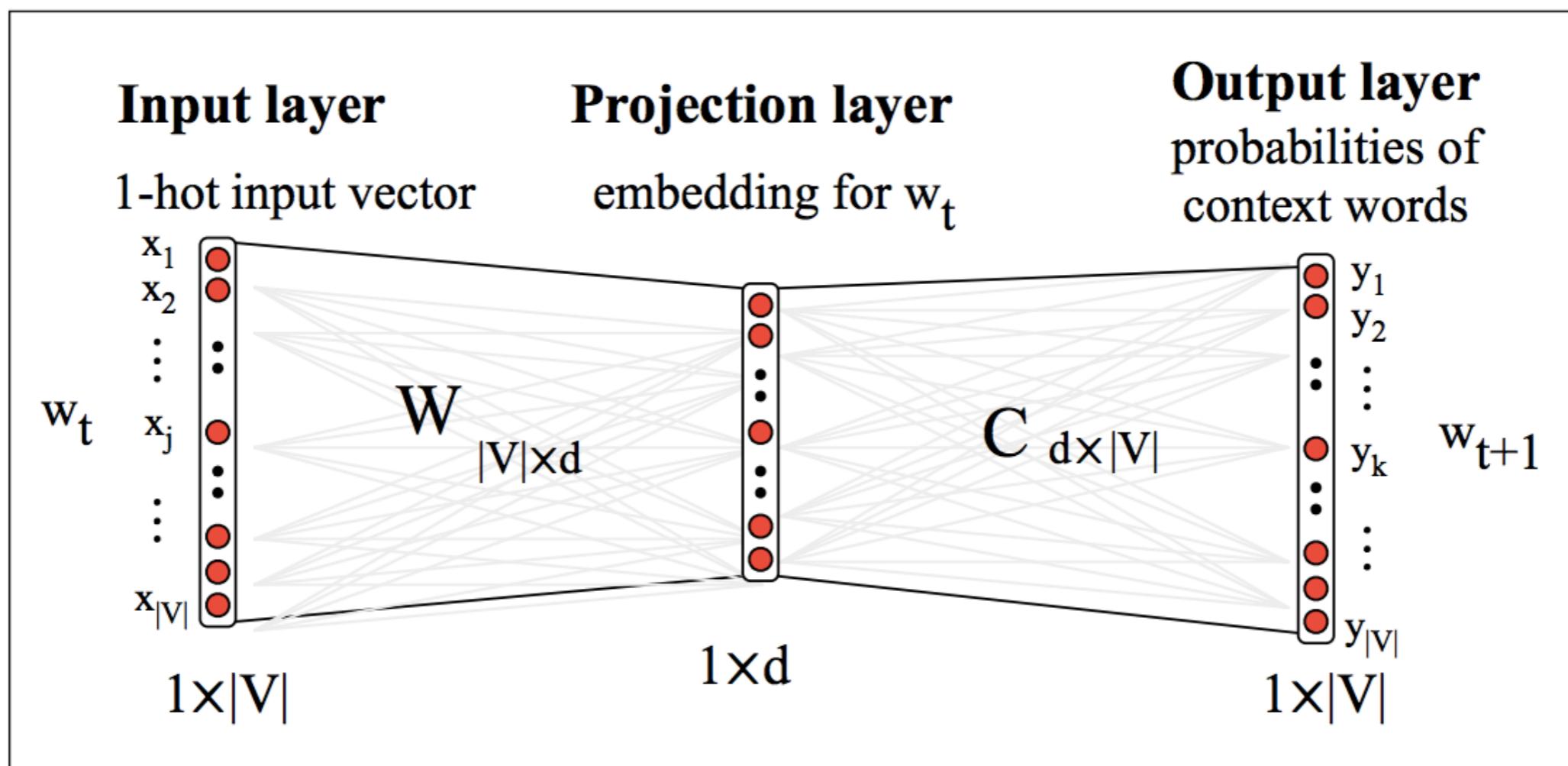
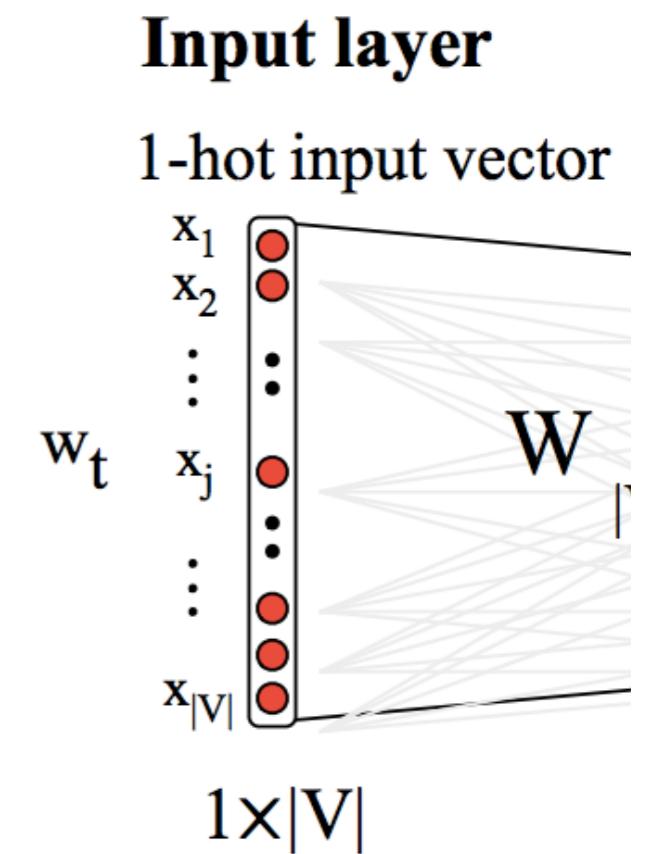


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

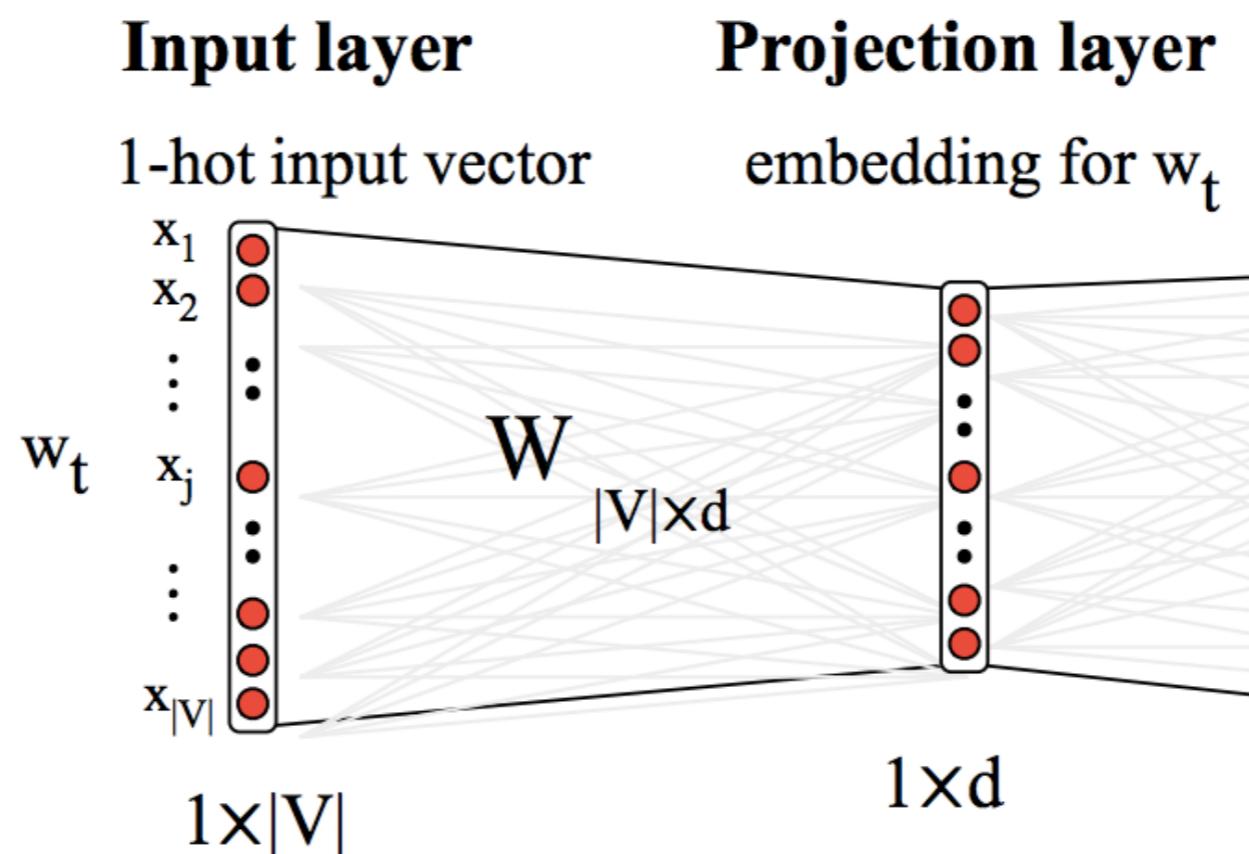
Input Layer

- “one-hot” word vectors
 - a vector of dimension $|V|$ (size of vocabulary)
 - all “0”s expect a single “1” in the vector
 - different positions of that “1” represent different words



Hidden (Projection) Layer

- A simple look up — the rows of this weight matrix are actually “input” word vectors



Hidden (Projection) Layer

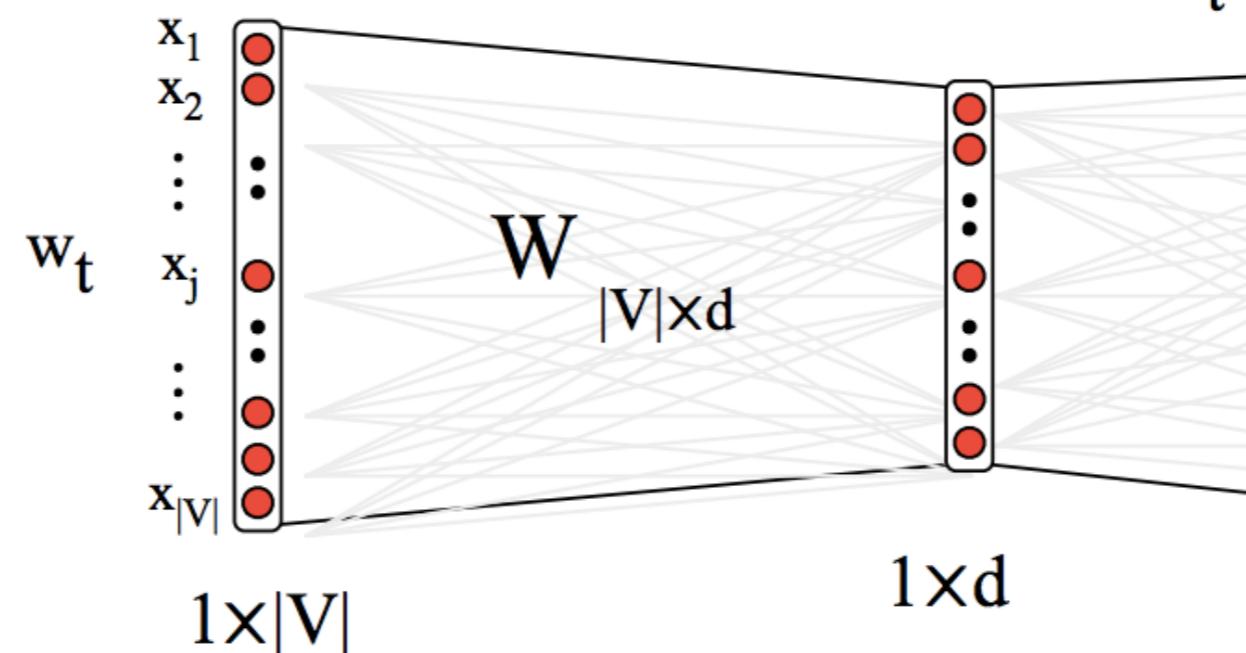
- A simple look up — the rows of this weight matrix are actually “input” word vectors

$$\begin{bmatrix} 0 & 0 & 0 & \boxed{1} & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & \boxed{12} & \boxed{19} \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

Input layer **Projection layer**

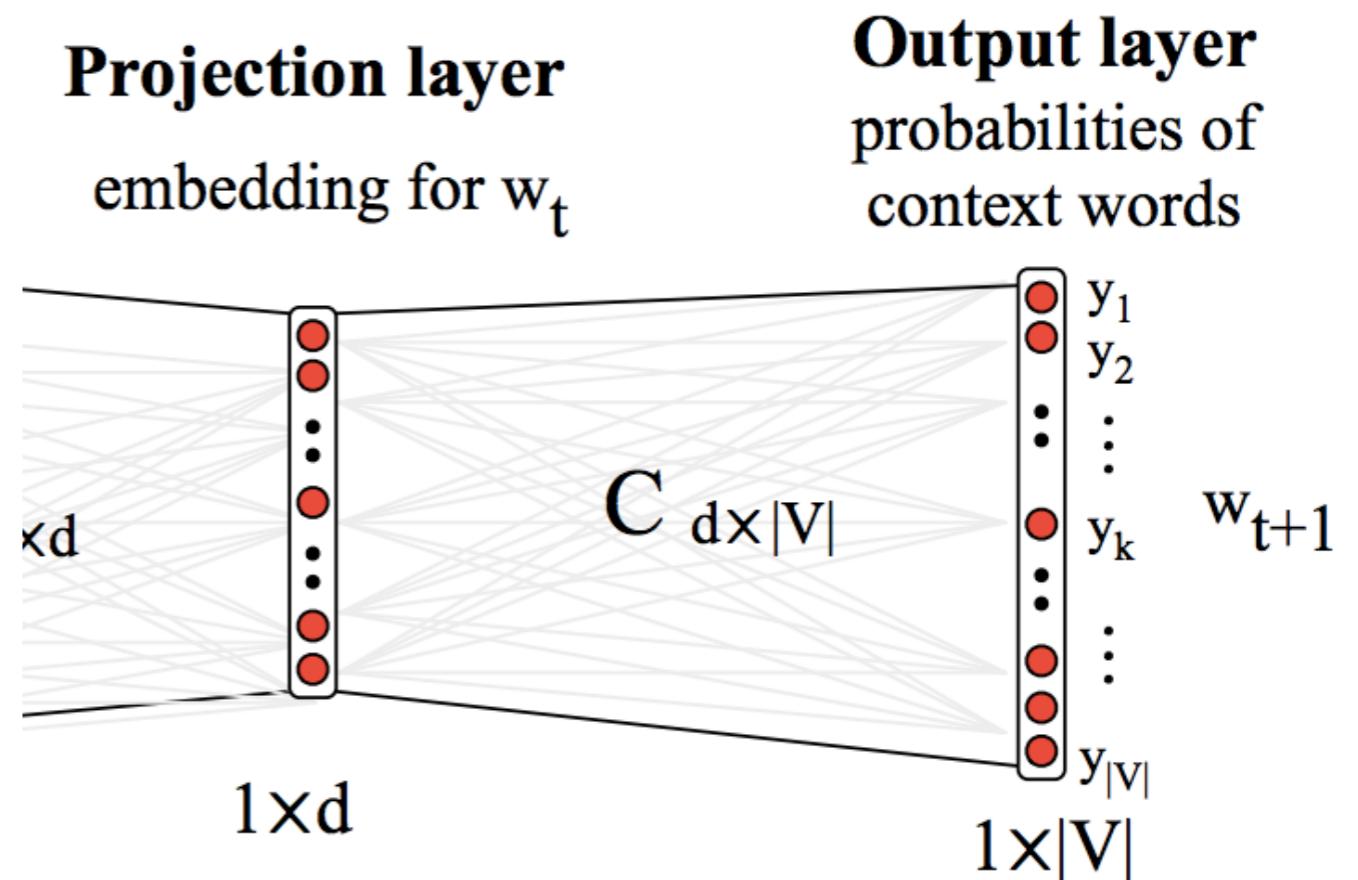
1-hot input vector

embedding for w_t



Output Layer

- predicts surrounding “outside” (context) words given the “center” word → A classification problem!
- Softmax Regression = Multi-class Logistic Regression



Softmax Function

- Softmax function is a generalization of logistic function

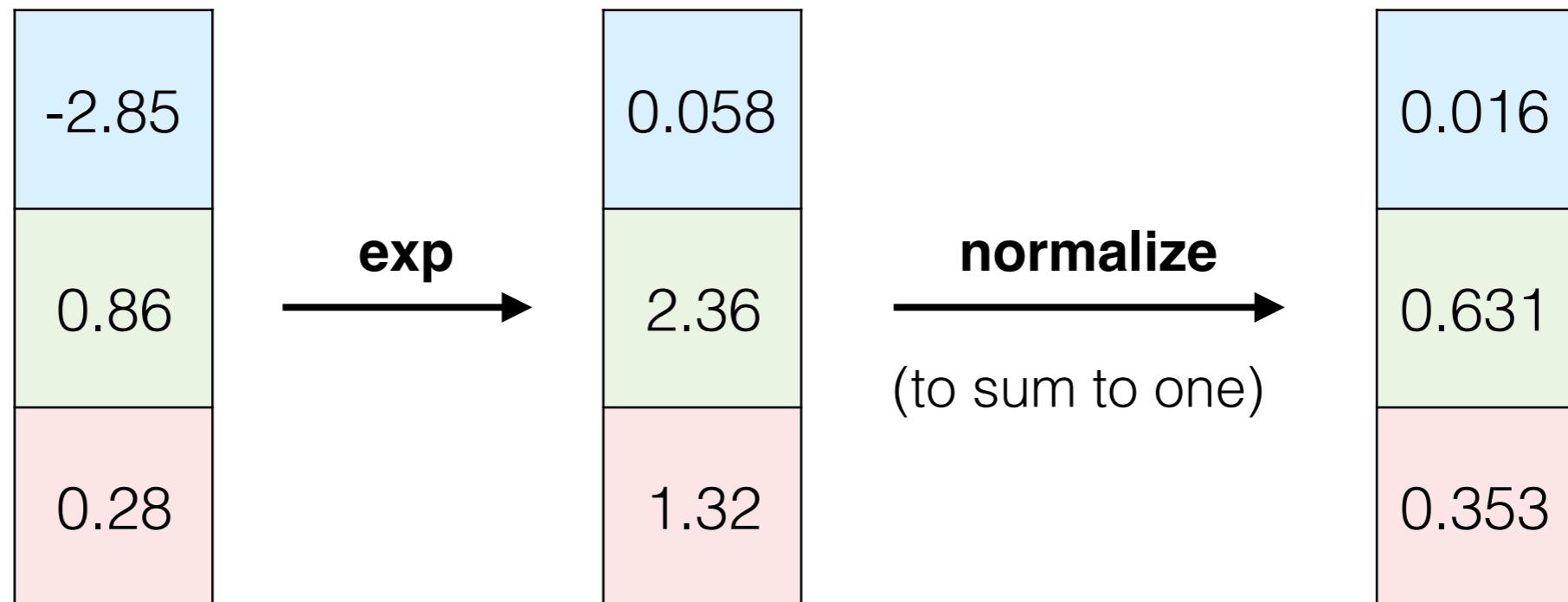
$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

exponentiate to make positive ← **normalized to give probability**

Softmax Function

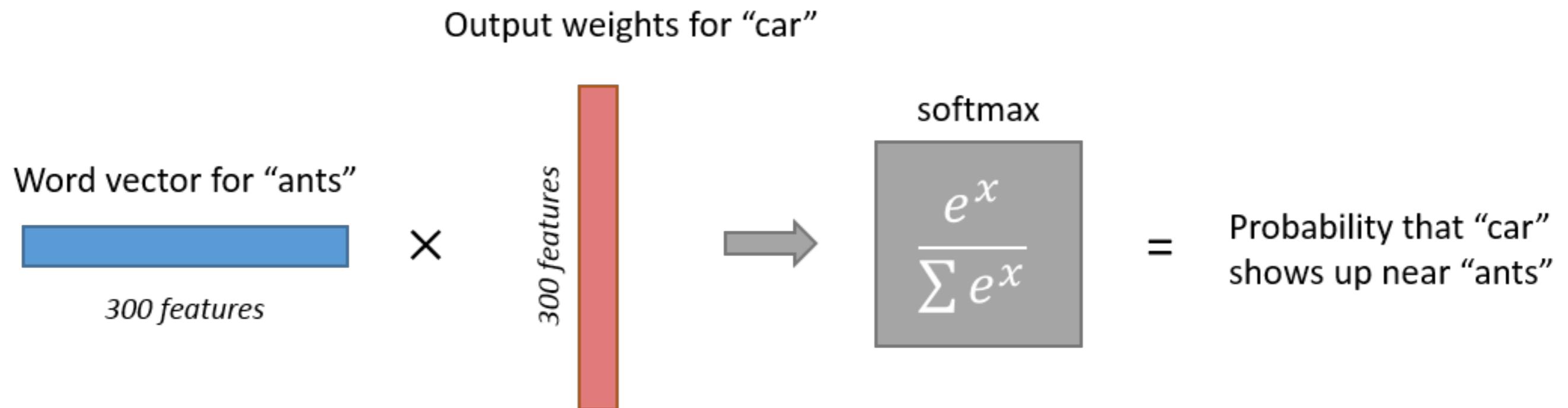
- Softmax function is a generalization of logistic function

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



Output Layer

- Intuition



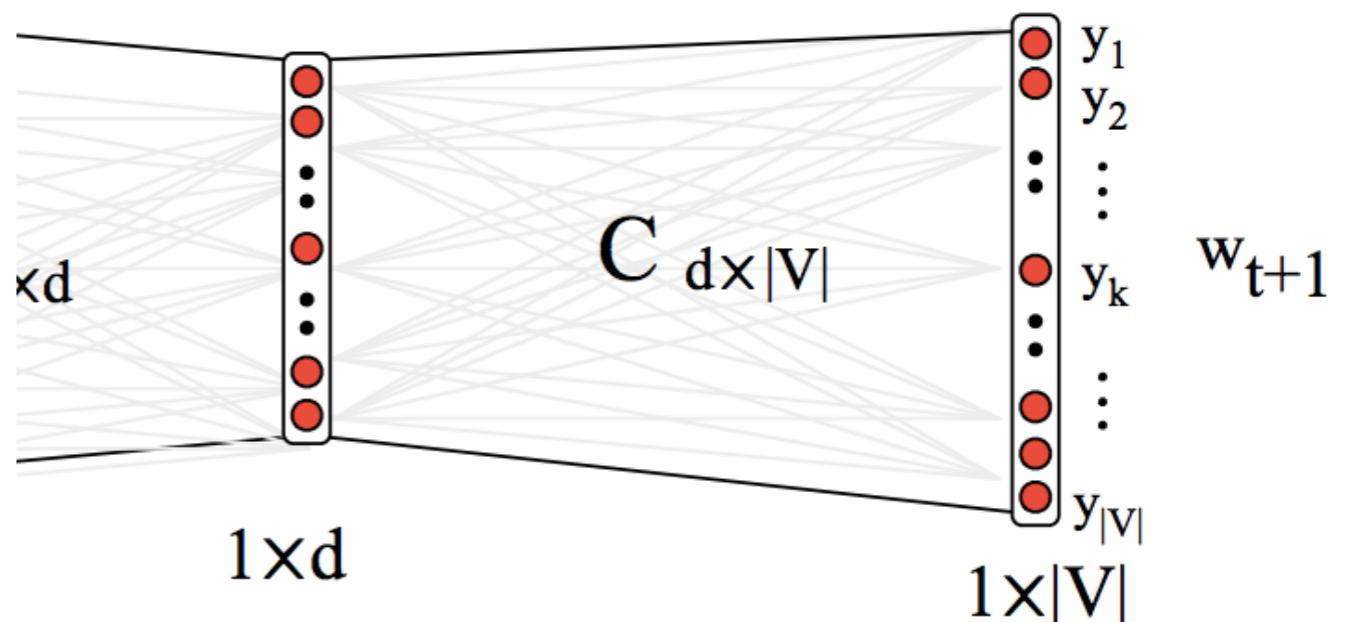
Output Layer

- Objective function: maximize the log probability of any “outside” (context) word given the “center” word

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

Projection layer
embedding for w_t

Output layer
probabilities of
context words

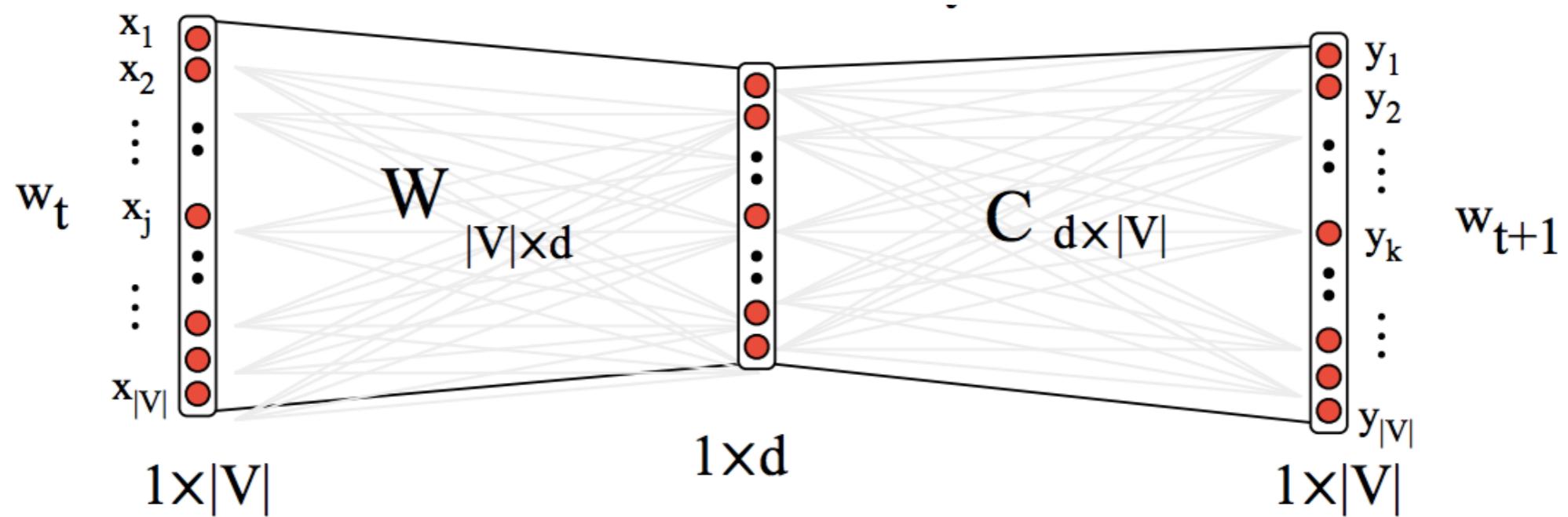


Output Layer

- predicts surrounding “outside” (context) words given the “center” word

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$

- so, every word has two vectors!



Gradient Descent

- Cost/Objective function:

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

- For a “center” word and an “outside” word:

$$\log p(o|c) = \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$

Gradient Descent

- Basics:

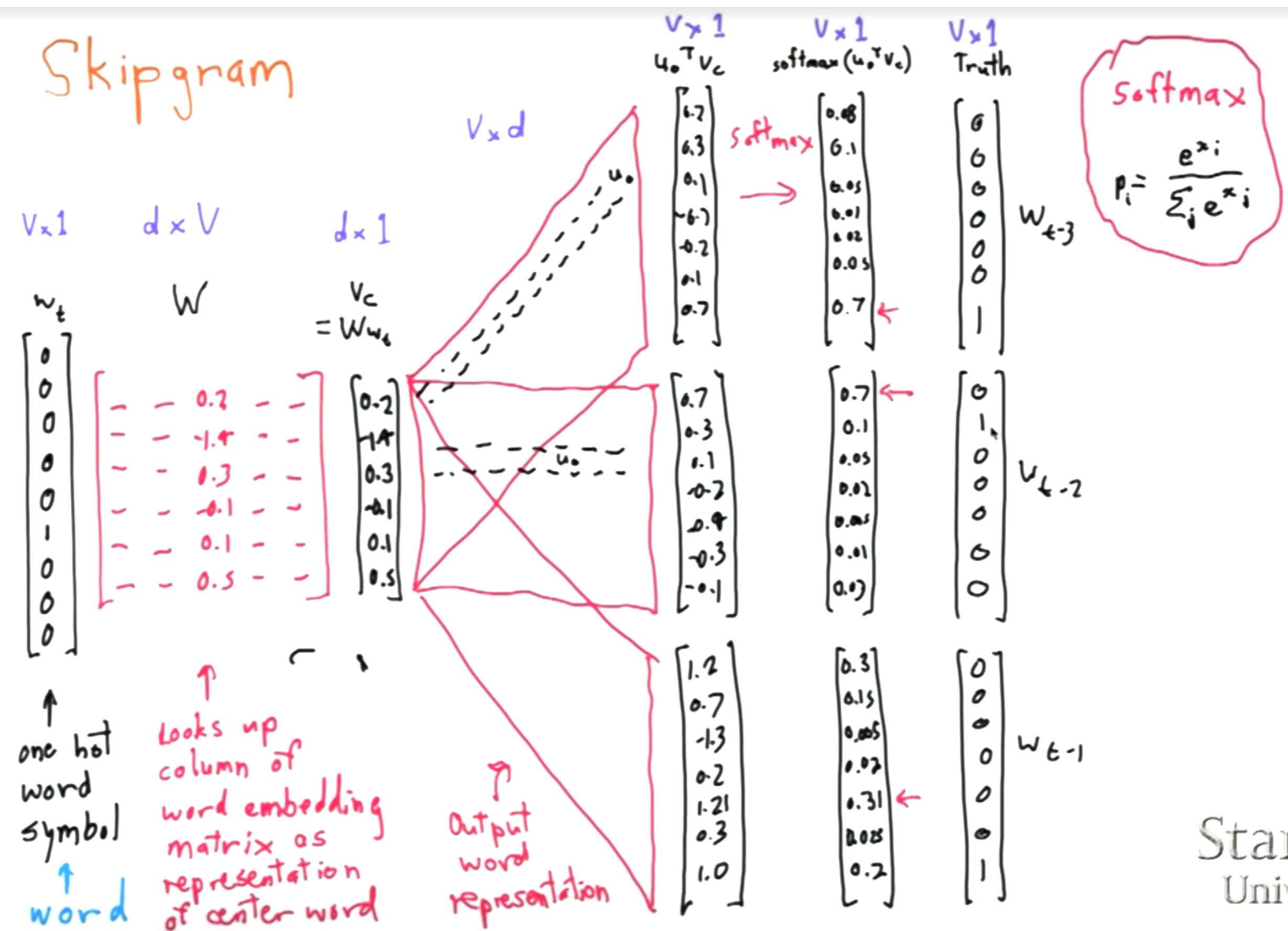
$$\frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$$

$$\frac{\partial e^x}{\partial x} = e^x \qquad \qquad \frac{\partial \log x}{\partial x} = \frac{1}{x}$$

- Chain Rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x} = \frac{\partial f(g)}{\partial g} \frac{\partial g(x)}{\partial x}$$

Skipgram



Stanford
University

Word Vector Representations

(a.k.a. “word embeddings”)

- 4 kinds of vector semantic models
 1. Hard clustering (e.g. Brown clustering)
 2. Soft clustering (e.g. SVD, LSA, LDA)
 3. Neural Network inspired models
(e.g. skip-grams and CBOW in word2vec)
 4. Mutual-information weighted word co-occurrence metrics

dense



sparse



Word2vec

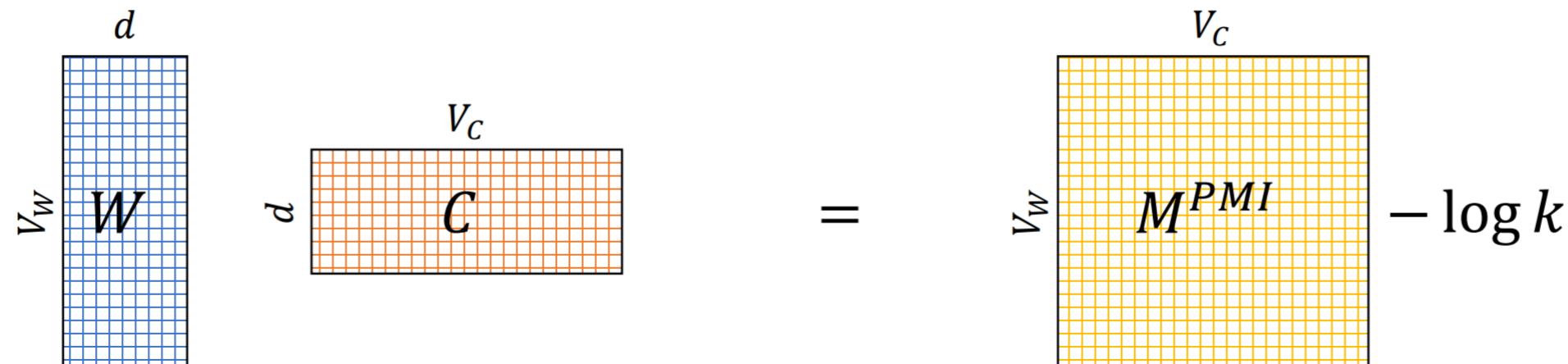
- Word2vec is not a single algorithm, but a toolkit
 - which contains two distinct algorithms (Skip-gram & CBOW), two training methods (negative sampling & hierarchical softmax)
- Word2vec is not deep learning, but neural-inspired
 - only one hidden layer followed by softmax, no non-linear activation function

Learn more: Omer Levy's answer on Quora

Relation between Skip-gram and SVD

- Levy and Goldberg (2014) show that skip-gram is factorizing (a shifted version of) the traditional word-context PMI matrix:

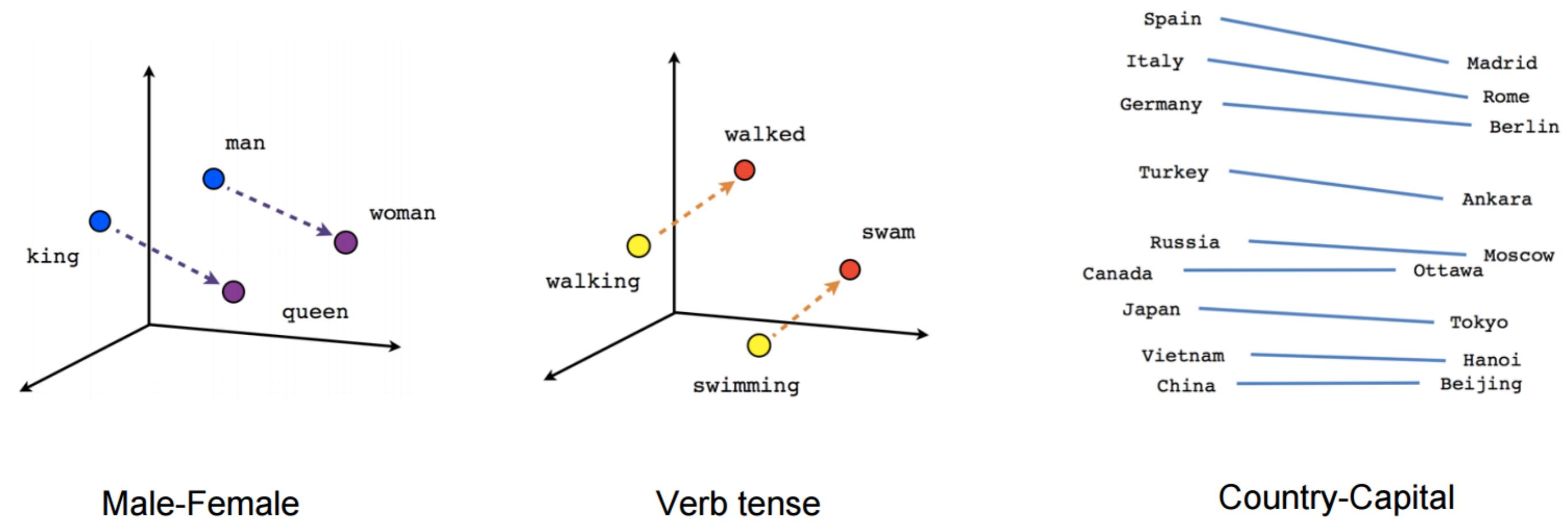
$$Opt(\vec{w} \cdot \vec{c}) = PMI(w, c) - \log k$$



- So does SVD!

Source: Omer Levy and Yoav Goldberg (NIPS 2014)
Neural Word Embedding as Implicit Matrix Factorization

Visualization



Male-Female

Verb tense

Country-Capital

Visualization

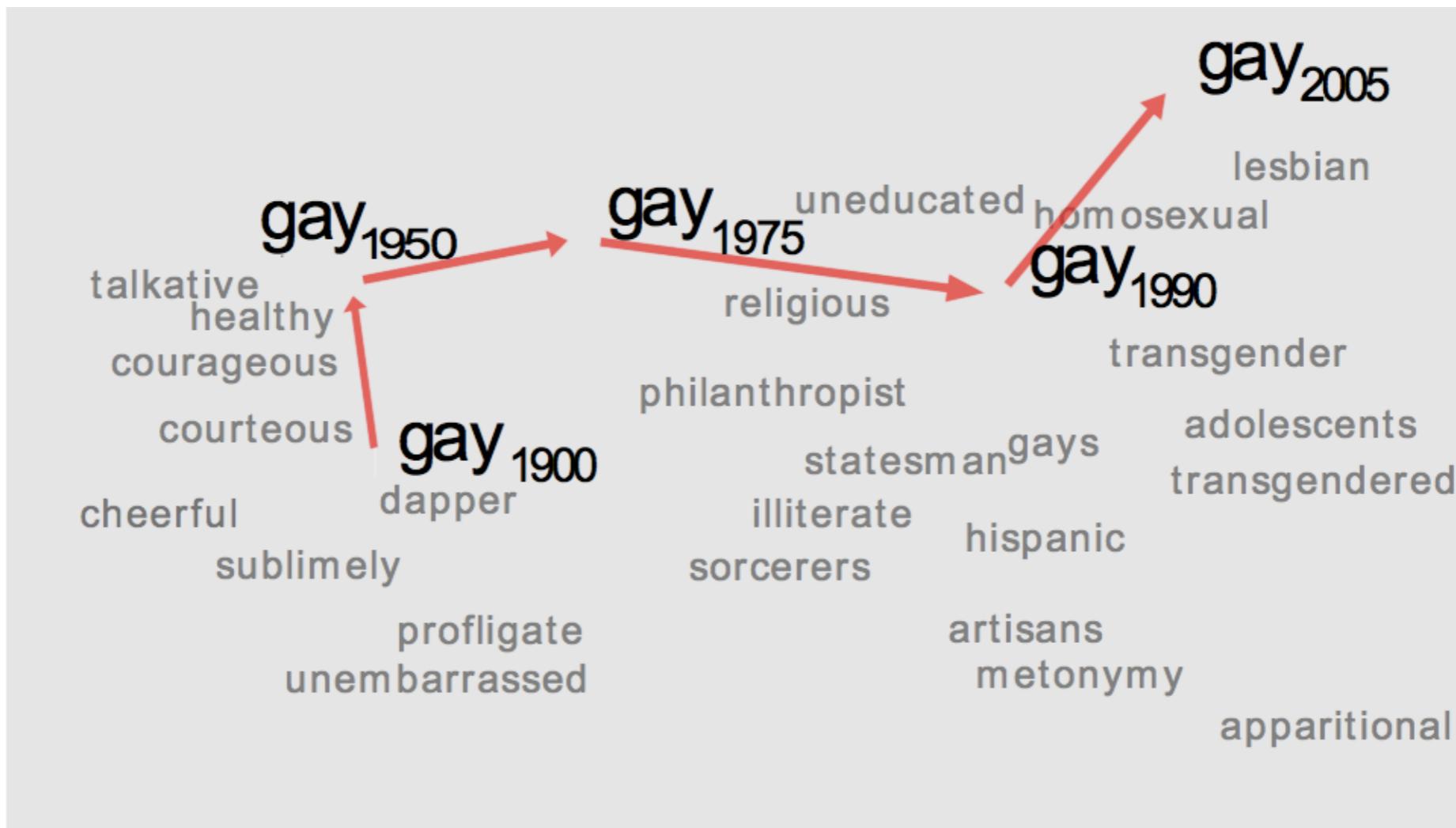
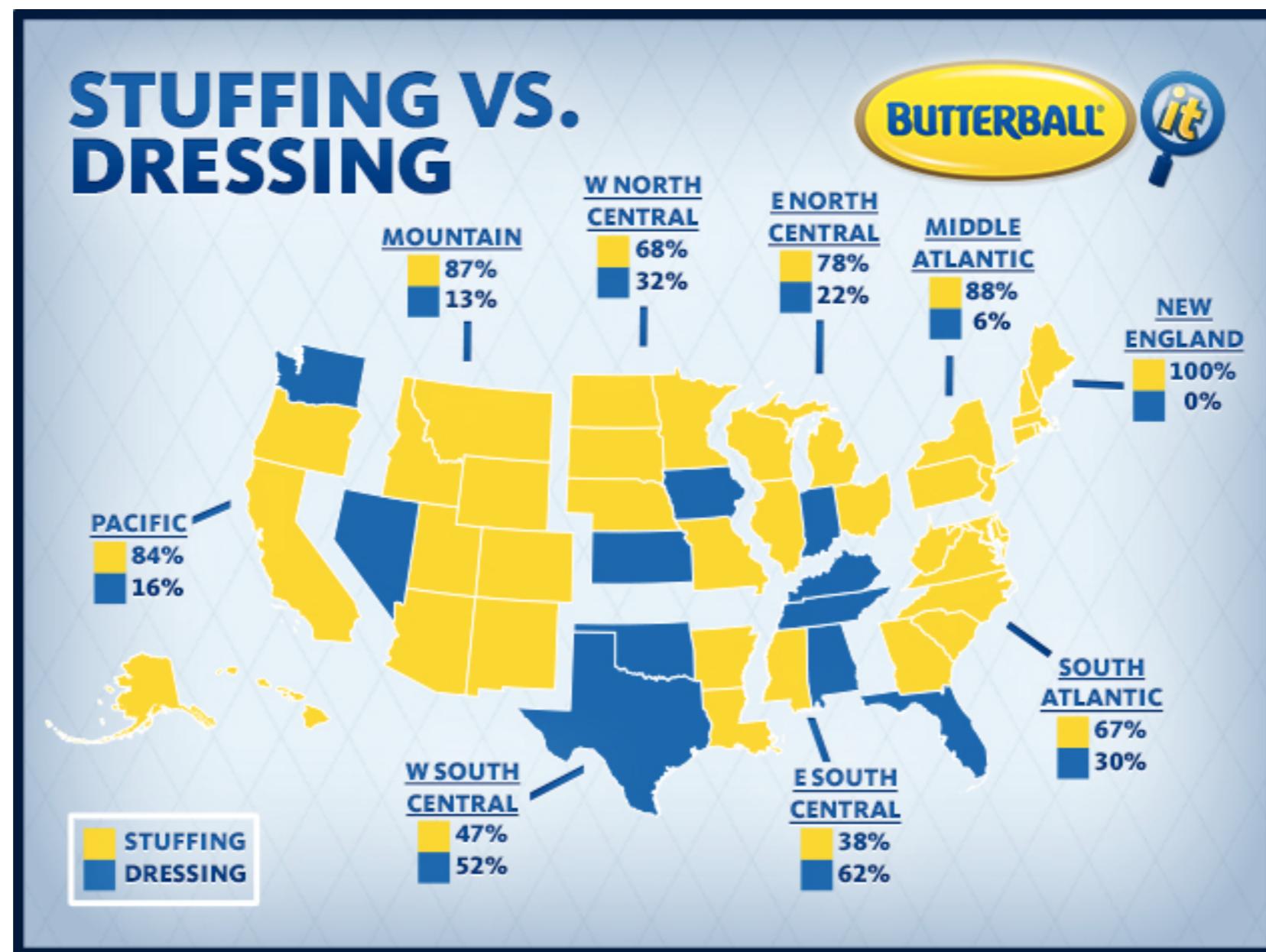


Figure 1: A 2-dimensional projection of the latent semantic space captured by our algorithm. Notice the semantic trajectory of the word **gay** transitioning meaning in the space.

Source: Kulkarni et al. (WWW 2015)
Statistically Significant Detection of Linguistic Change

Happy Turkey Day!



*This survey was conducted online with a random sample 1,000 men and women in 9 regions – all members of the CyberPulse™ Advisory Panel. Research was conducted in May 2007. The overall sampling error for the survey is +/-3% at the 95% level of confidence.

Thank You!

 Follow @cocoweixu

Instructor: Wei Xu

www.cis.upenn.edu/~xwe/

Course Website: socialmedia-class.org