Social Media & Text Analysis

lecture 10 - Vector Semantics

Follow @cocoweixu

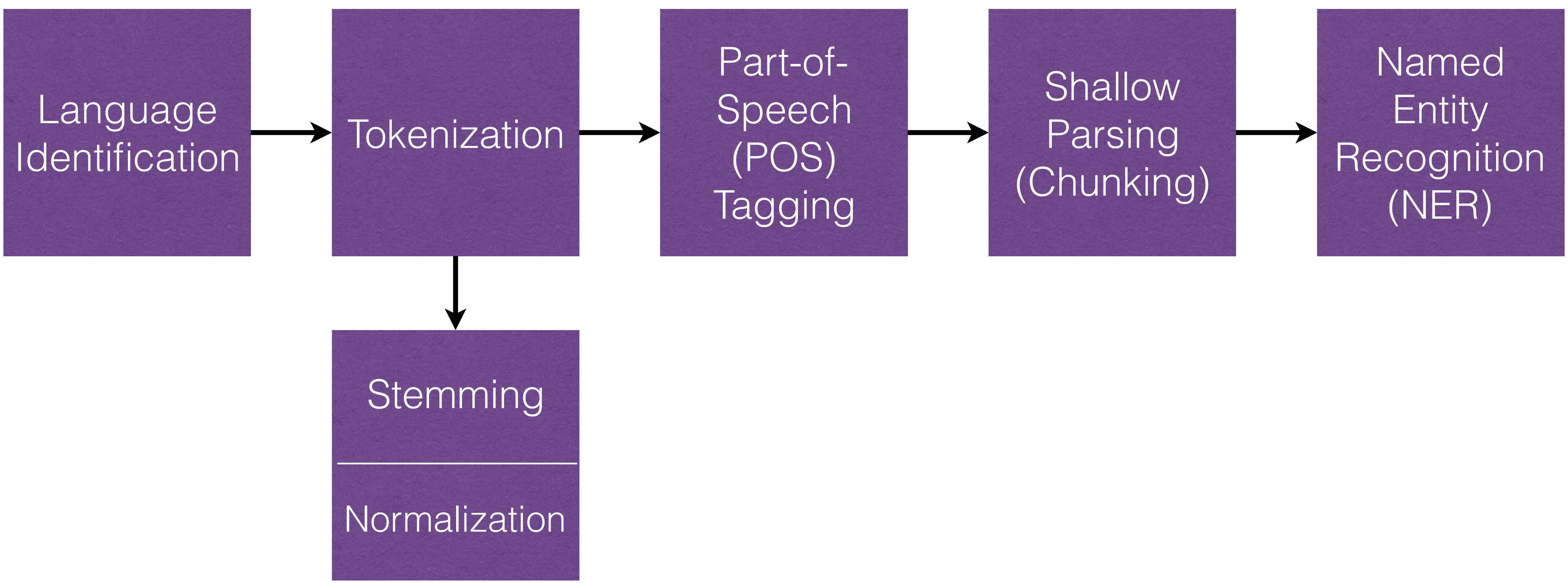
CSE 5539-0010 Ohio State University

Instructor: Wei Xu

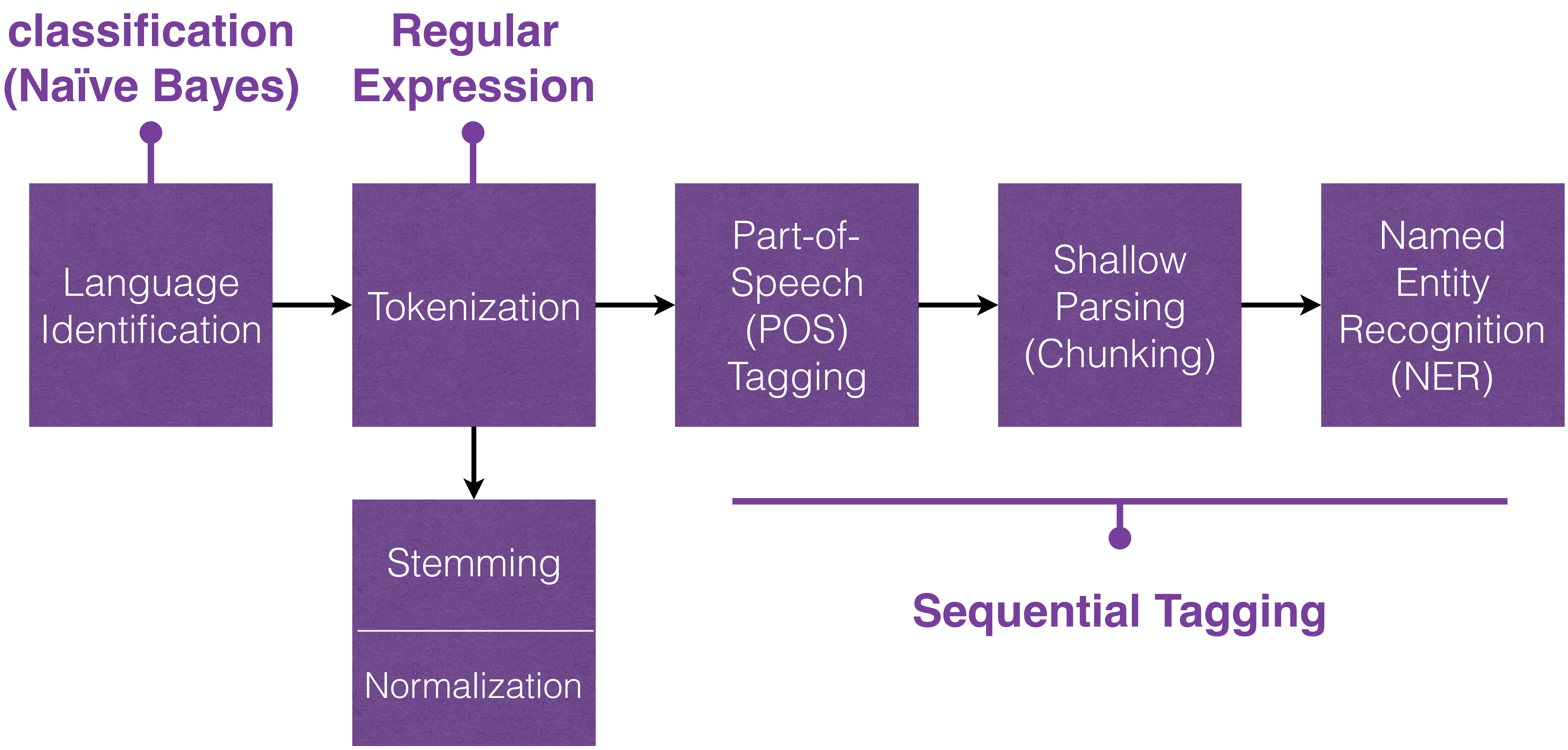
Website: socialmedia-class.org

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

NLP Pipeline

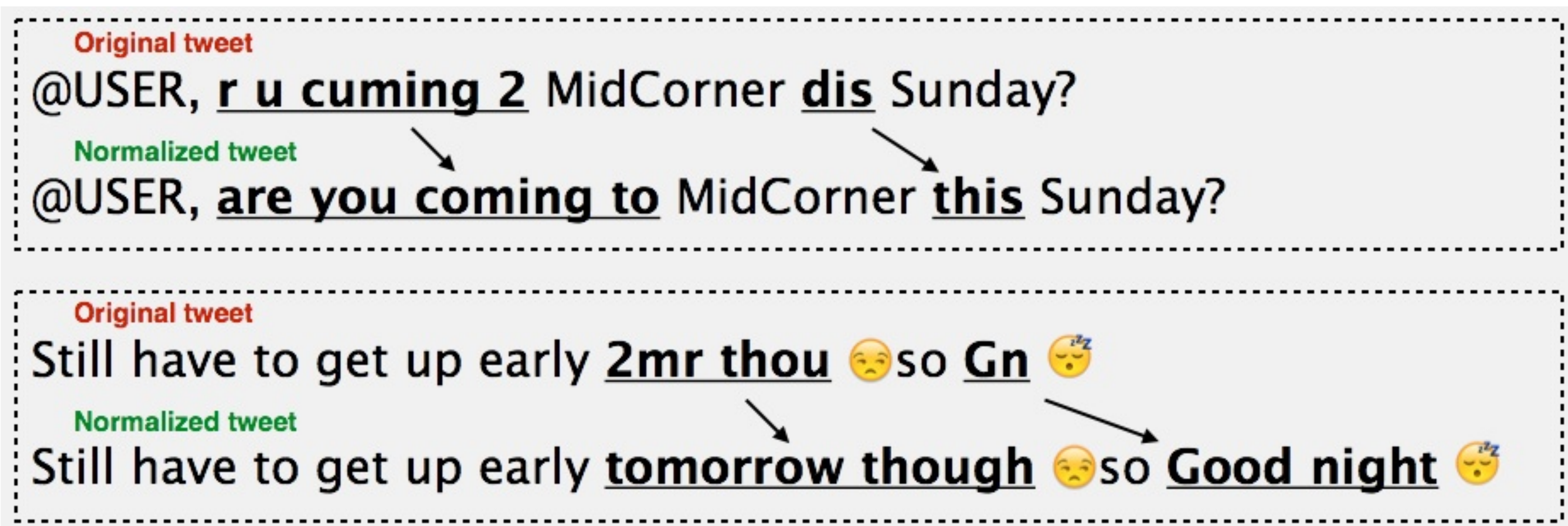


NLP Pipeline



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

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

Source: Strauss, Toma, Ritter, de Marneffe, Xu
Results of the WNUT16 Named Entity Recognition Shared Task (WNUT@COLING 2016)

Timeline of NLP on Microblogs

2010

- **Dialog Modeling (Ritter et al.)**
- First Story Detection (Petrovic et al.)
- Geographic Variation (Eisenstein et al.)

2011

- **Named Entity Recog. (Ritter et al.)**
- **POS Tagging (Gimpel et al.)**
- **Normalization (Han and Baldwin)**
- Summarization (Liu et. al)

2012

- **Open-Domain Event Extraction (Ritter et al.)**
- Censorship Detection (Bamman et al.)

Timeline of NLP on Microblogs

2013

- **Summarization (Xu et al.)**
- **Normalization (Xu et al.)**

- Machine Translation (Ling et. al.)
- POS (Owoputi et al.)

2014

- **Paraphrase Extraction (Xu et. al.)**

- Parsing Weibo (Wang et. al.)
- Parsing Twitter (Kong et. al.)

2015

- **Named Entity Recognition (Cherry and Guo)**

- Dialogue Modeling (Sordoni et al.)

Challenges in Twitter

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

An Unsupervised Learning Method:

(1) Brown Clustering

- Input:
  - a (large) text corpus

- Output:
  1. a partition of words into word clusters
  2. (generalization of 1) a hierarchical word clustering

Clusters in Twitter NER

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

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, $c_1, c_2, \ldots, c_m$
  - repeat for $i = (m+1) \ldots |V|$
    - create a new cluster c_{m+1} for the i'th most frequent word
    - choose two clusters from $c_1, c_2, \ldots, c_{m+1}$ 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

$$Quality(C) = \sum_{i}^{n} \log e(w_i \mid C(w_i)) q(C(w_i) \mid C(w_{i-1}))$$

$$= \sum_{c=1}^{k} \sum_{c'=1}^{k} p(c,c') \log \frac{p(c,c')}{p(c)p(c')} + G$$

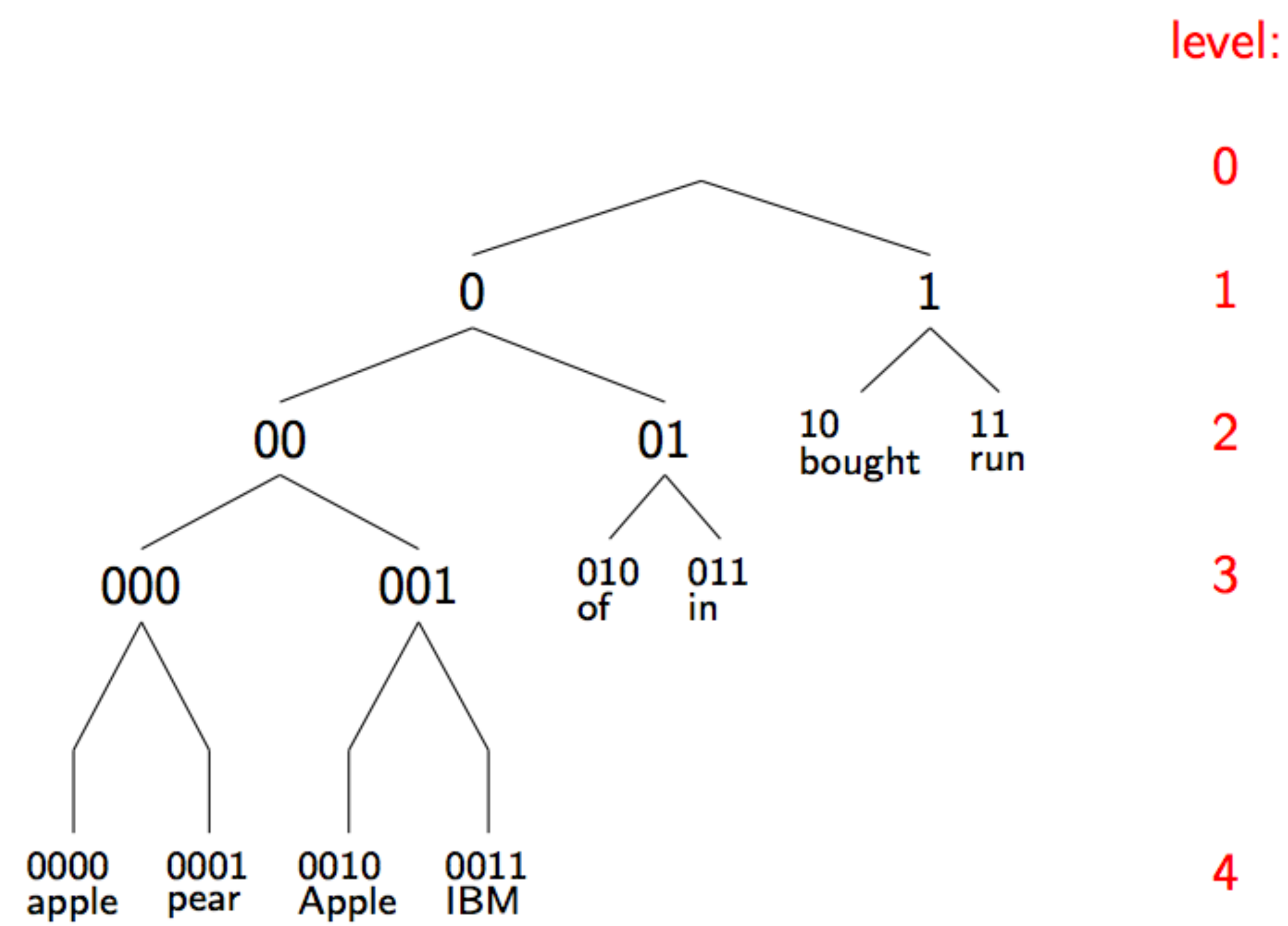
- ***n(c)*** :count of class ***c*** seen in the corpus
- ***n(c,c')*** : counts of ***c'*** seen following ***c***

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

Hierarchical Word Clustering

- Each intermediate node is a cluster:

bit string representation



Brown Clusters as Vectors

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
Ventritex	101101110010010110010
Tractebel	1011011100100101100110
Synopsys	1011011100100101100111
WordPerfect	1011011100100101101000
....	
John	10111001000000000
Consuelo	10111001000000001
Jeffrey	10111001000000010
Kenneth	1011100100000001100
Phillip	10111001000000011010
WILLIAM	10111001000000011011
Timothy	1011100100000001110

- Example Clusters
  (from Miller et al. 2004)

Brown Clusters as Vectors

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
Ventritex	101101110010010110010
Tractebel	1011011100100101100110
Synopsys	1011011100100101100111
WordPerfect	1011011100100101101000
....	
John	101110010000000000
Consuelo	101110010000000001
Jeffrey	101110010000000010
Kenneth	10111001000000001100
Phillip	101110010000000011010
WILLIAM	101110010000000011011
Timothy	10111001000000001110

- Example Clusters (from Miller et al. 2004)

word cluster features
(bit string prefix)

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

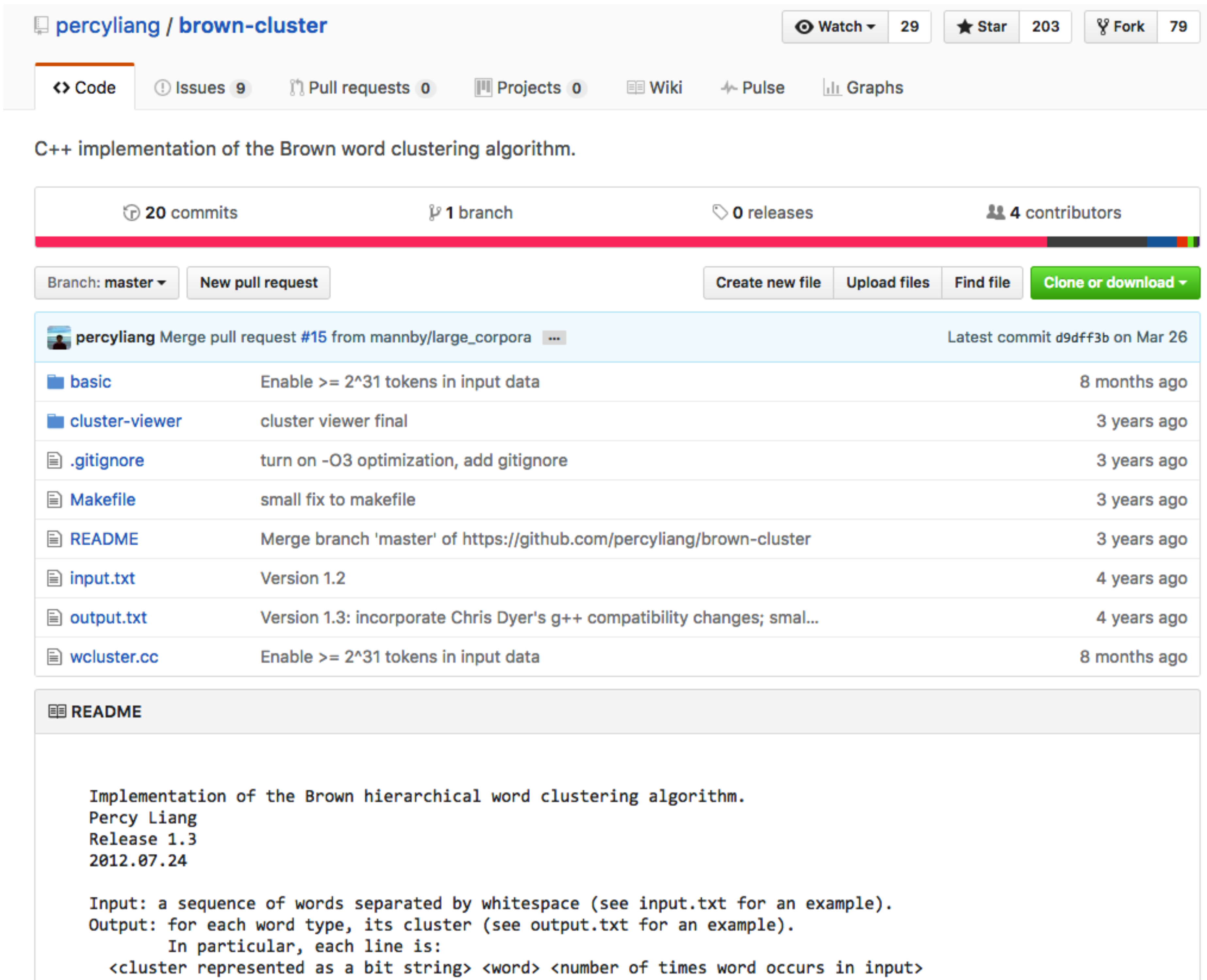
Brown Clustering

- 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

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.

20 commits | 1 branch | 0 releases | 4 contributors

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

dense

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)

sparse

4. Mutual-information weighted word co-occurrence metrics

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(panda.closure(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

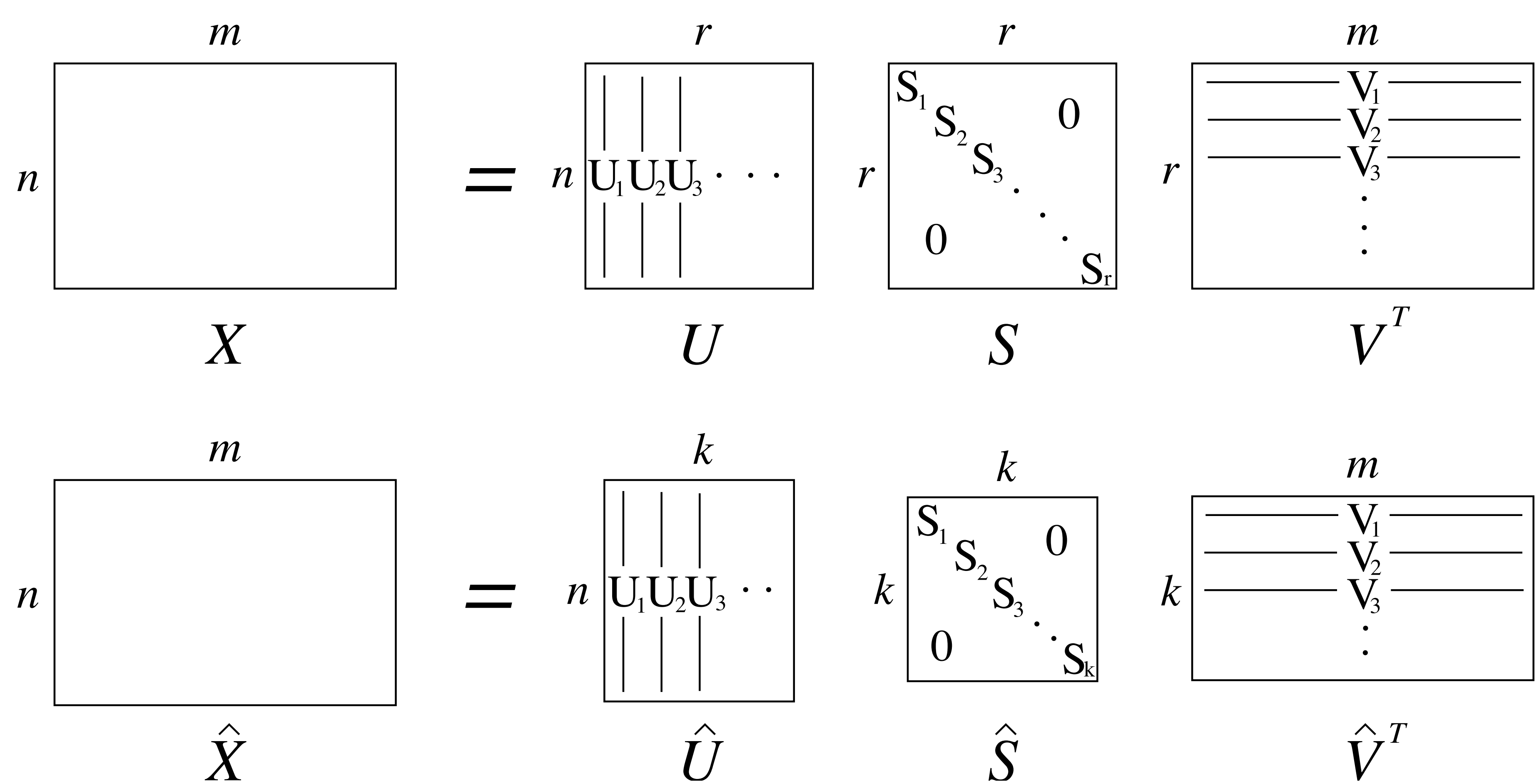
- Word meaning is represented as a **dense** vector

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

How to reduce the dimensionality?

(2) Matrix Factorization

- Singular Value Decomposition (SVD)



$\hat{X}$ 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.

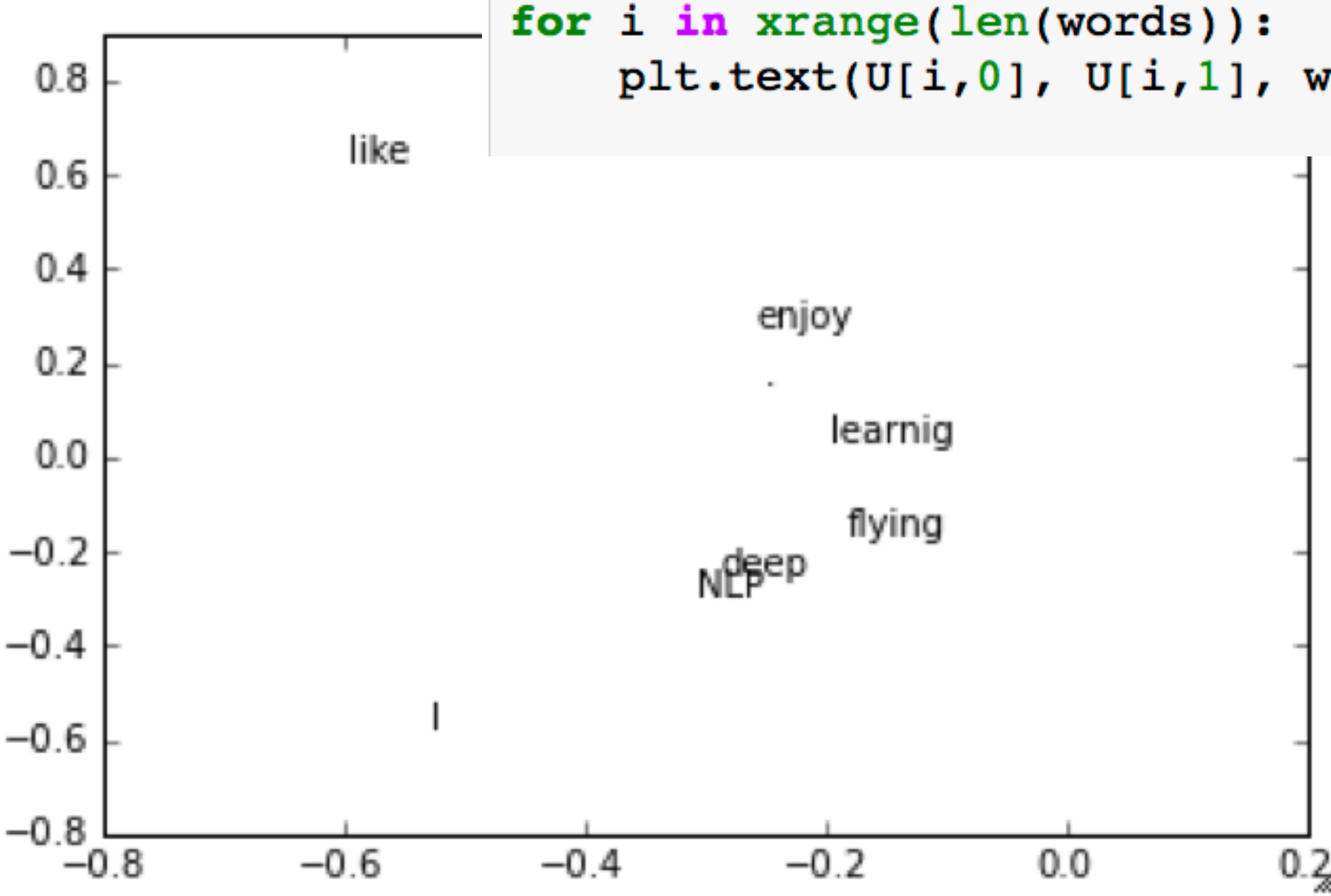
```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],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

```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



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

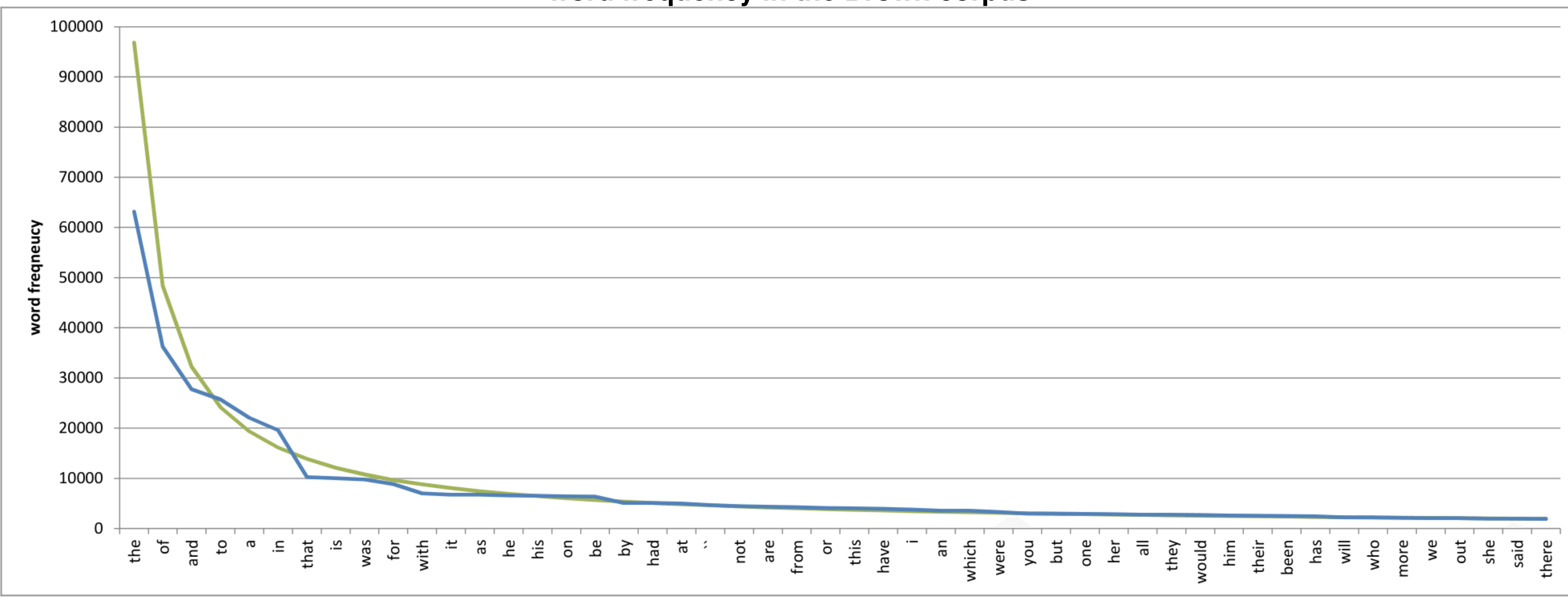
- etc …

Source: Rohde et al. (2005)
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

Zipf’s (Power) Law

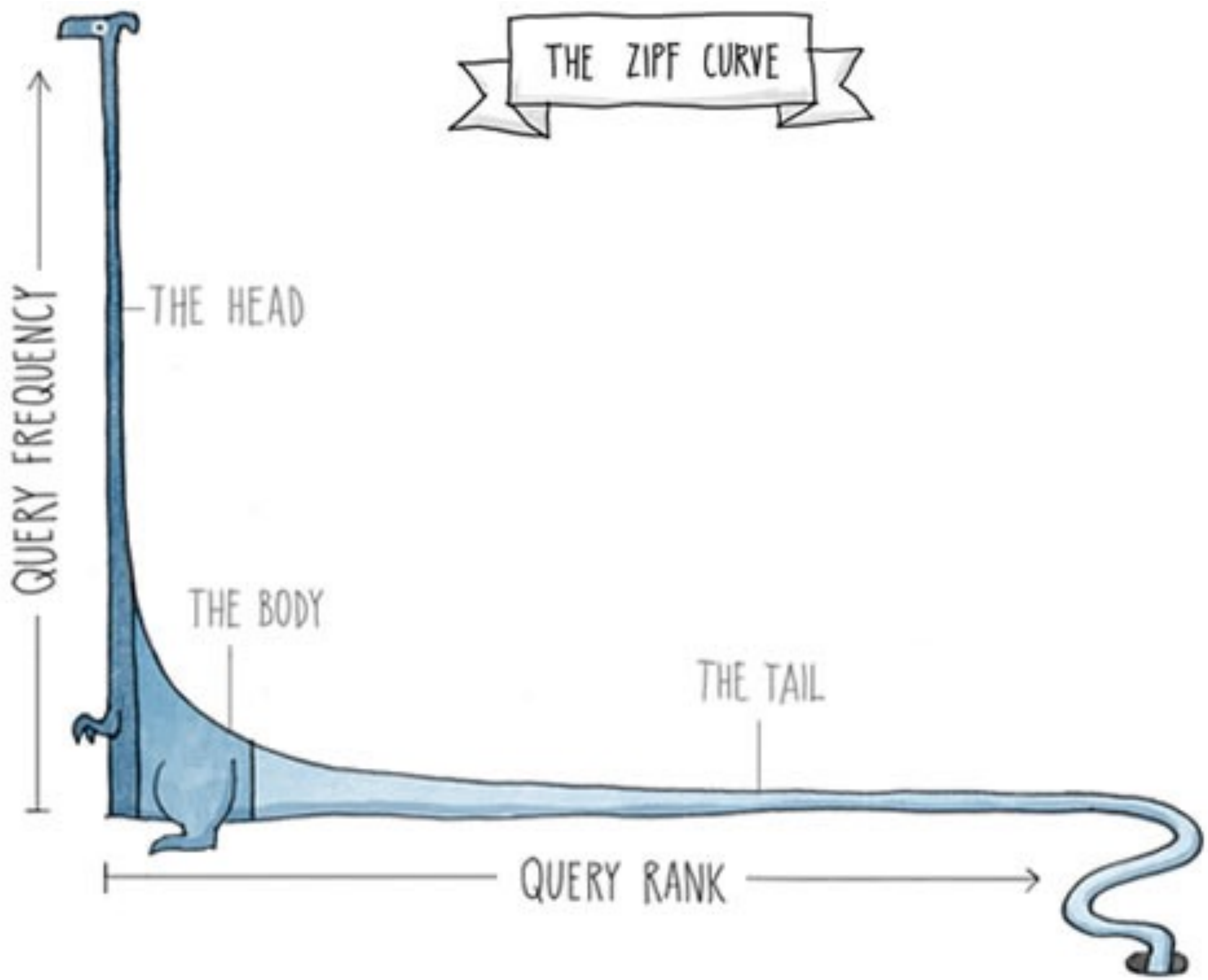
- frequency of word is inversely proportional to its rank in the frequency table

word frequency in the Brown corpus



Source: http://www.shadycharacters.co.uk/

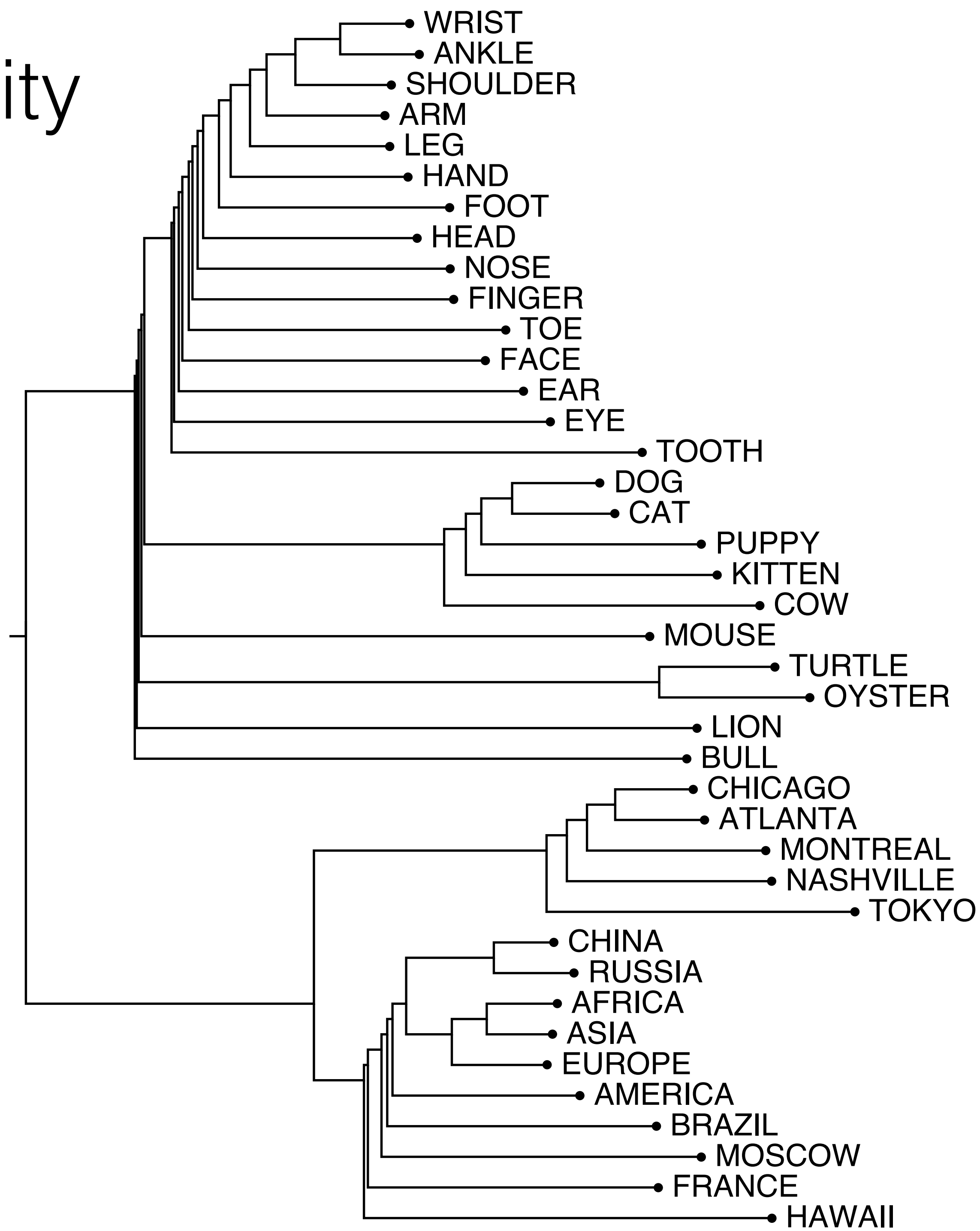
Zipf's (Power) Law



Source: smashing magazine

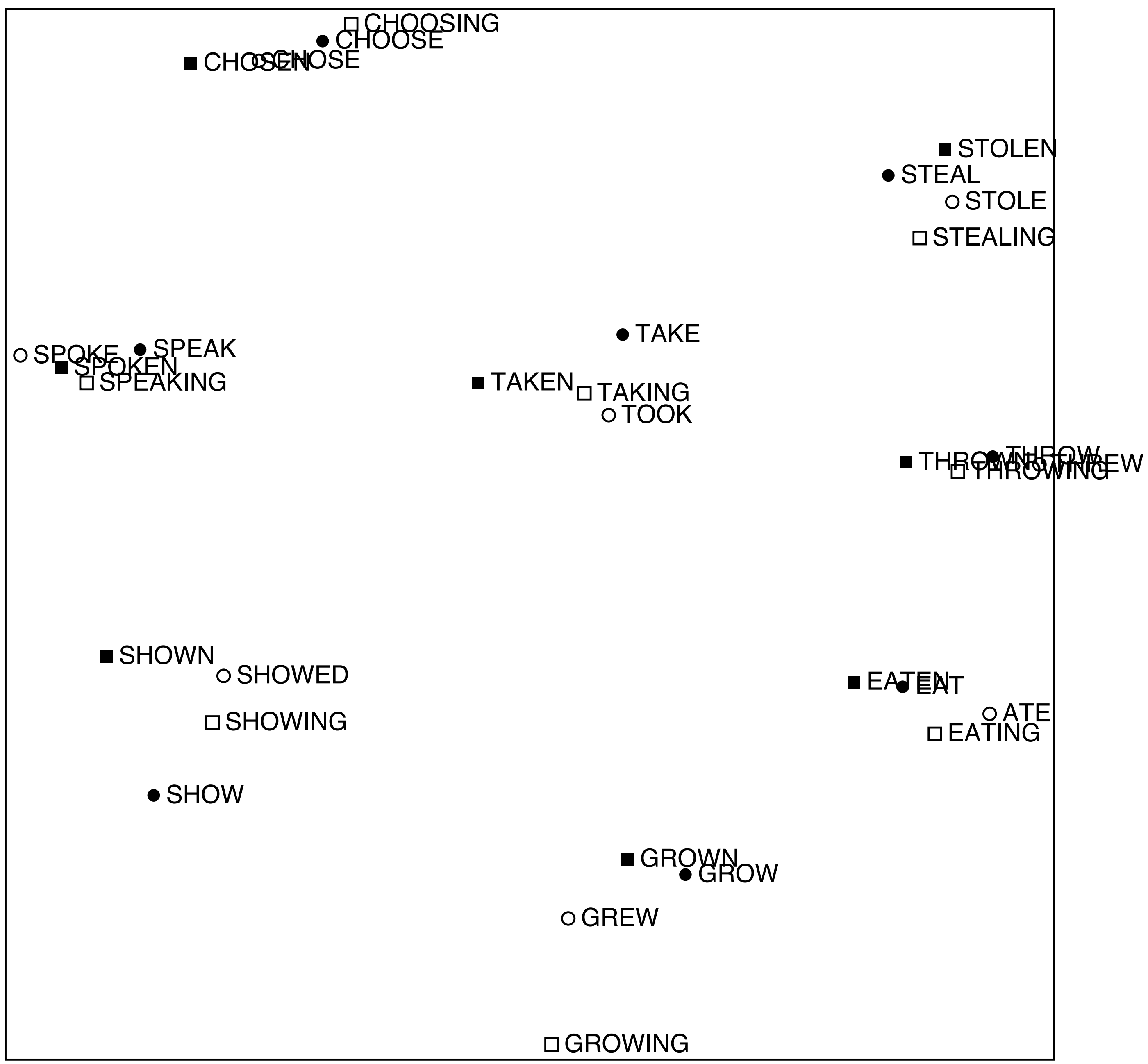
Clustering Vectors

- visualize similarity



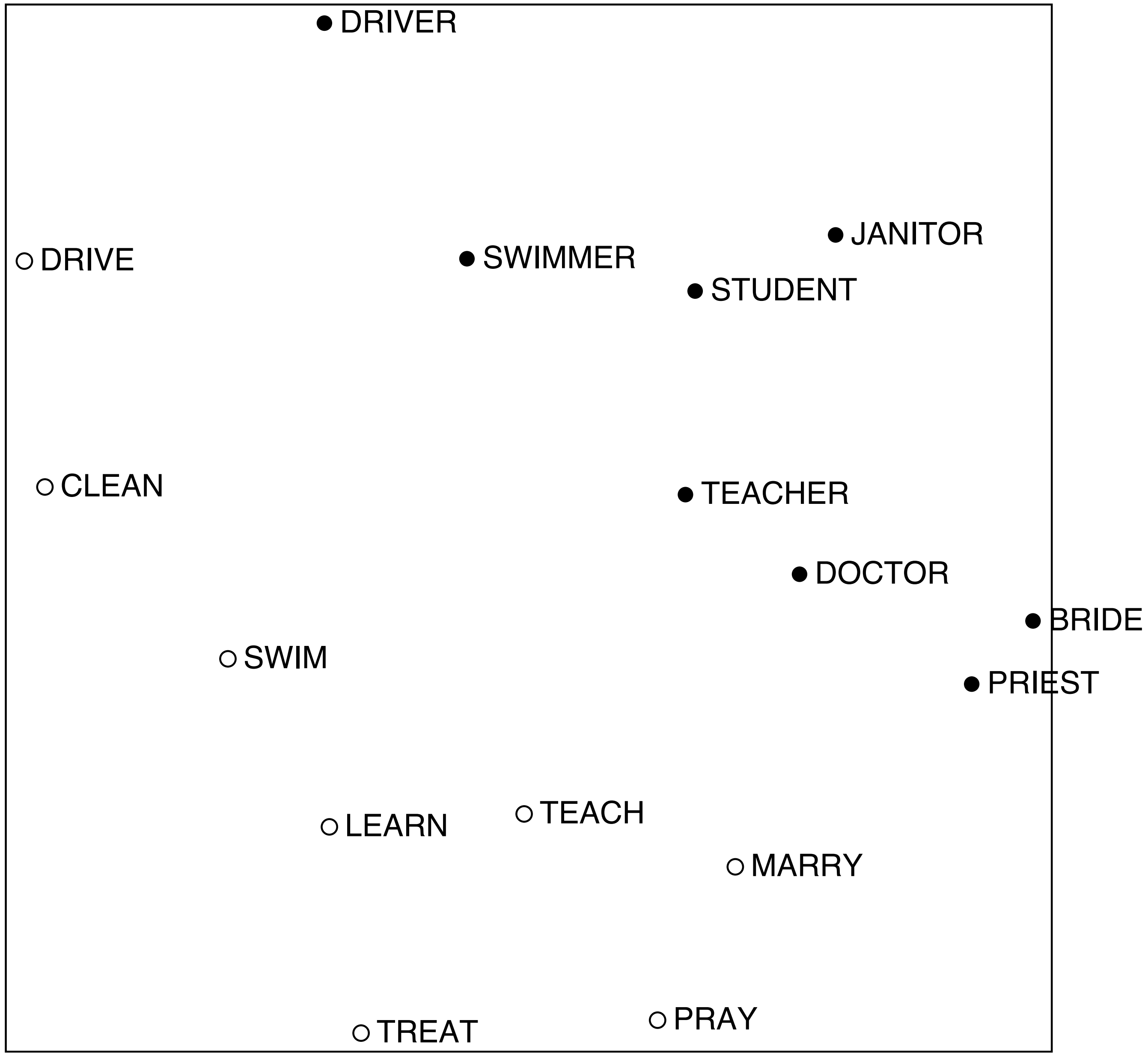
Source: Rohde et al. (2005)
An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

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

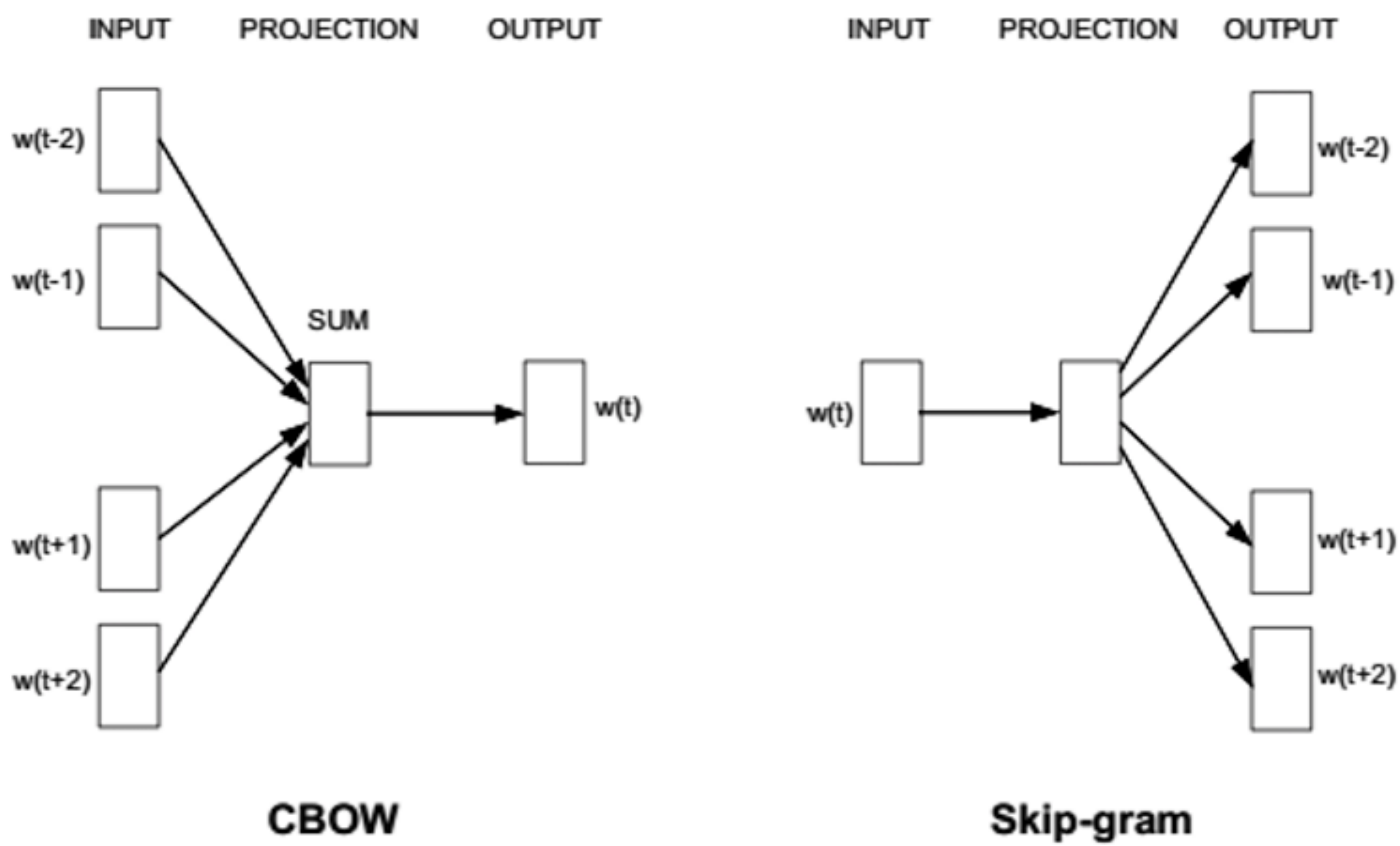
- Still some problems:
  - computational cost scales quadratically for ***m*** x ***n*** matrix — O(mn^2) when n<m
  - hard to use large corpus (and vocabulary)
  - hard to incorporate new words or documents

(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)

Word2vec

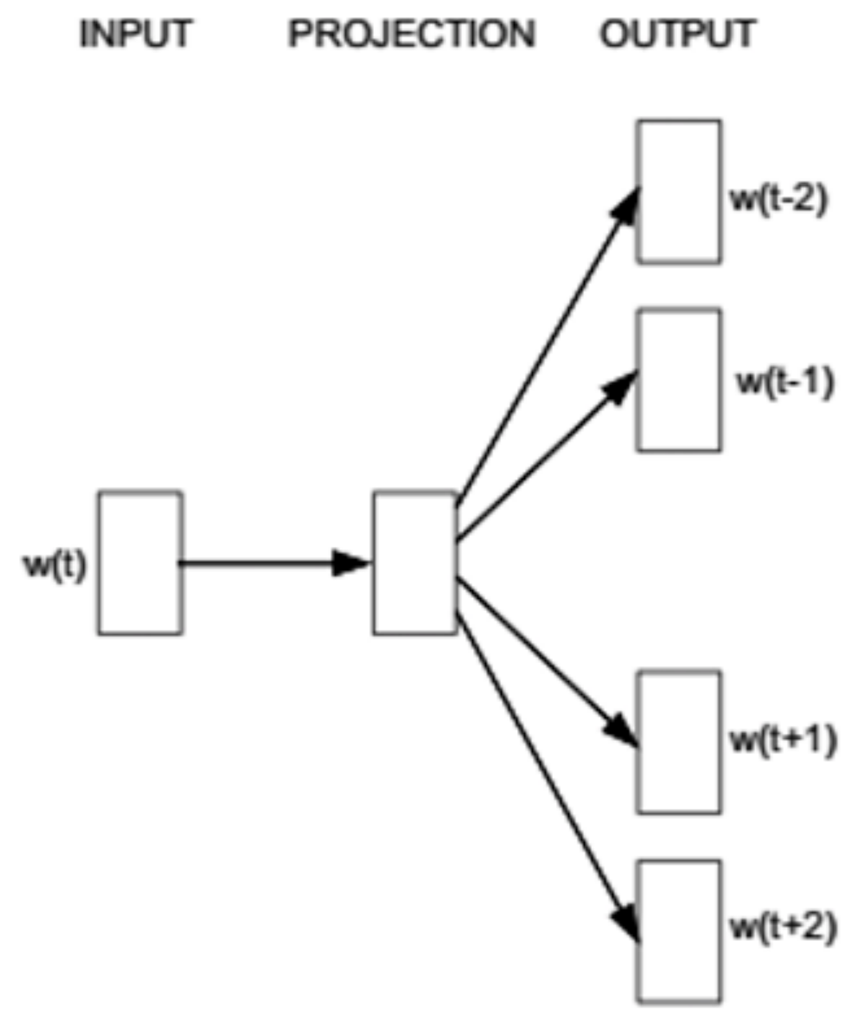
- simple and efficient



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

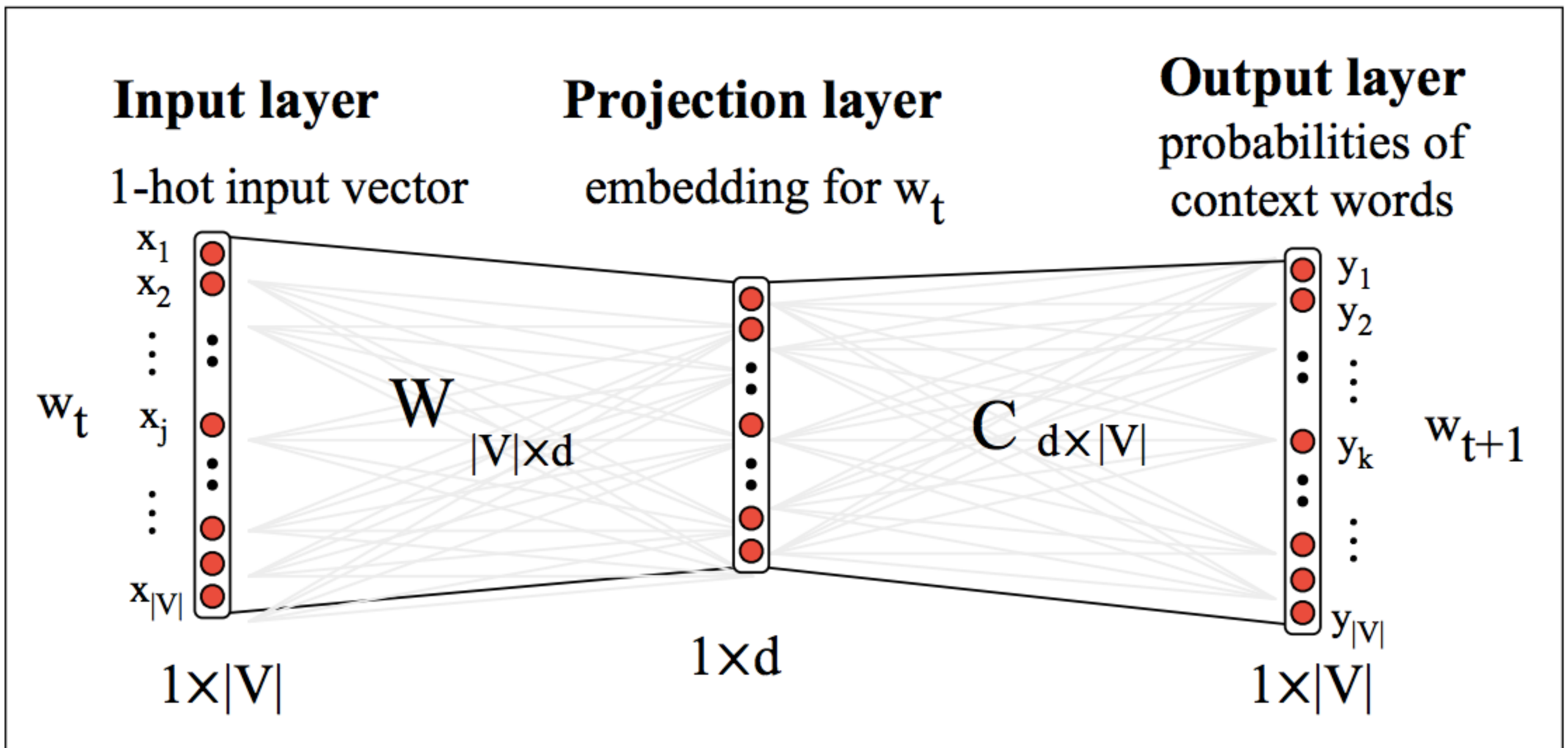
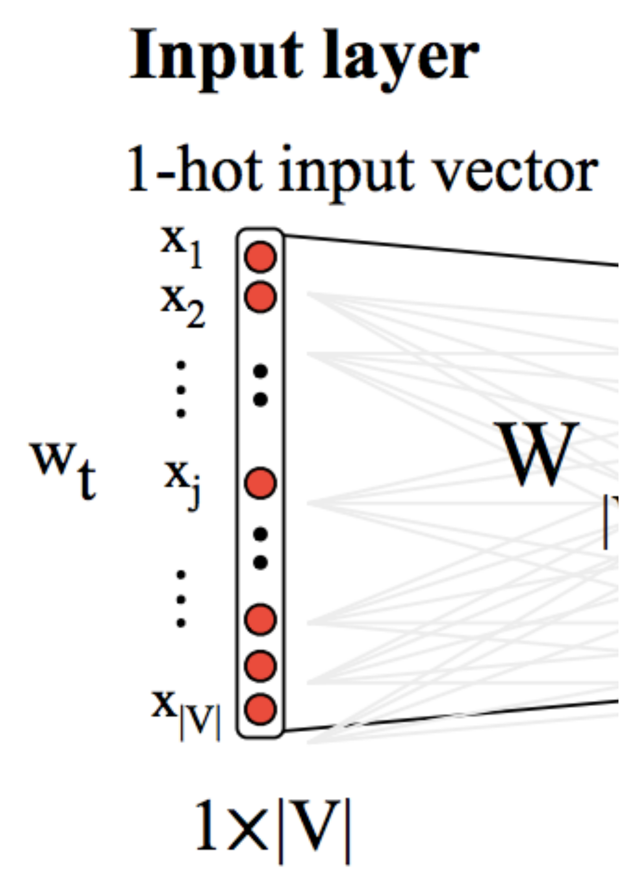


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

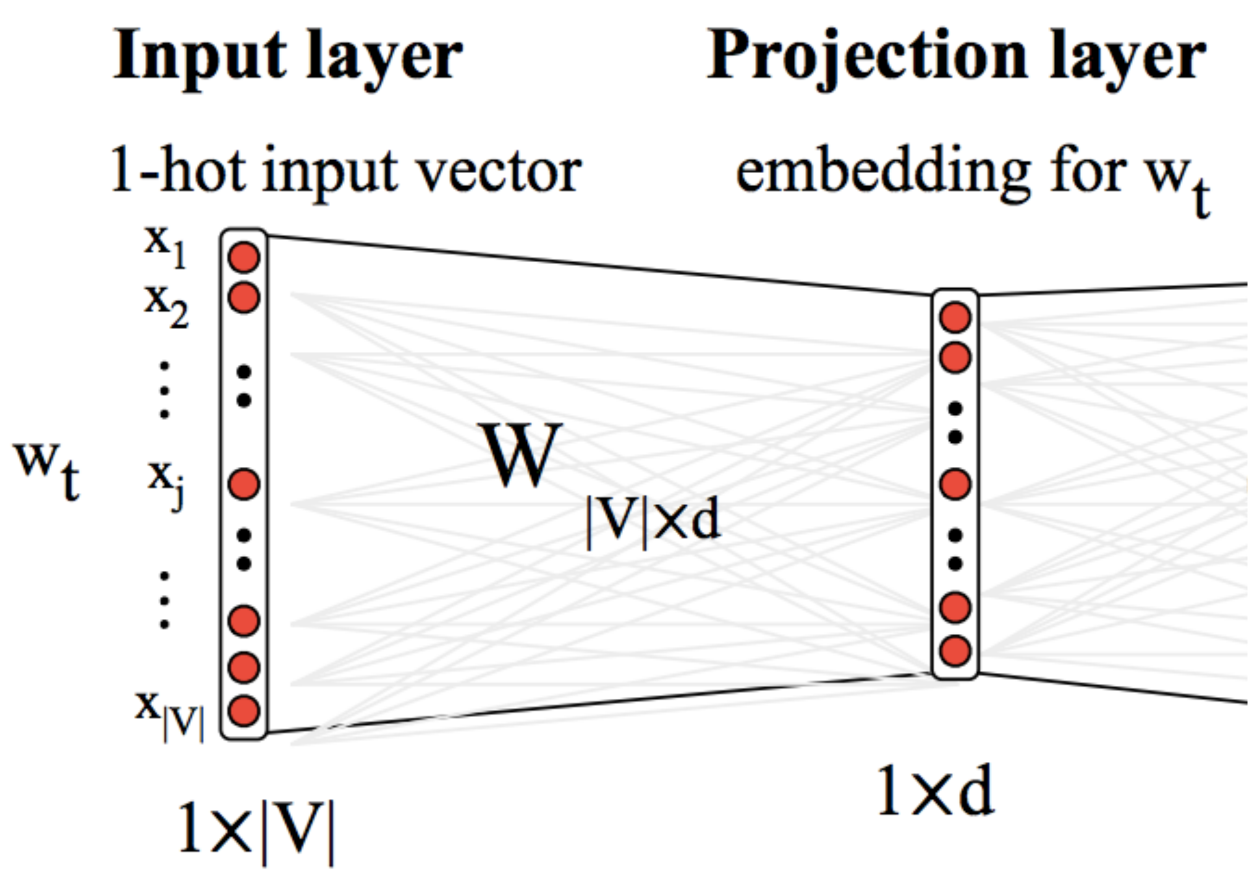
Input Layer

- "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

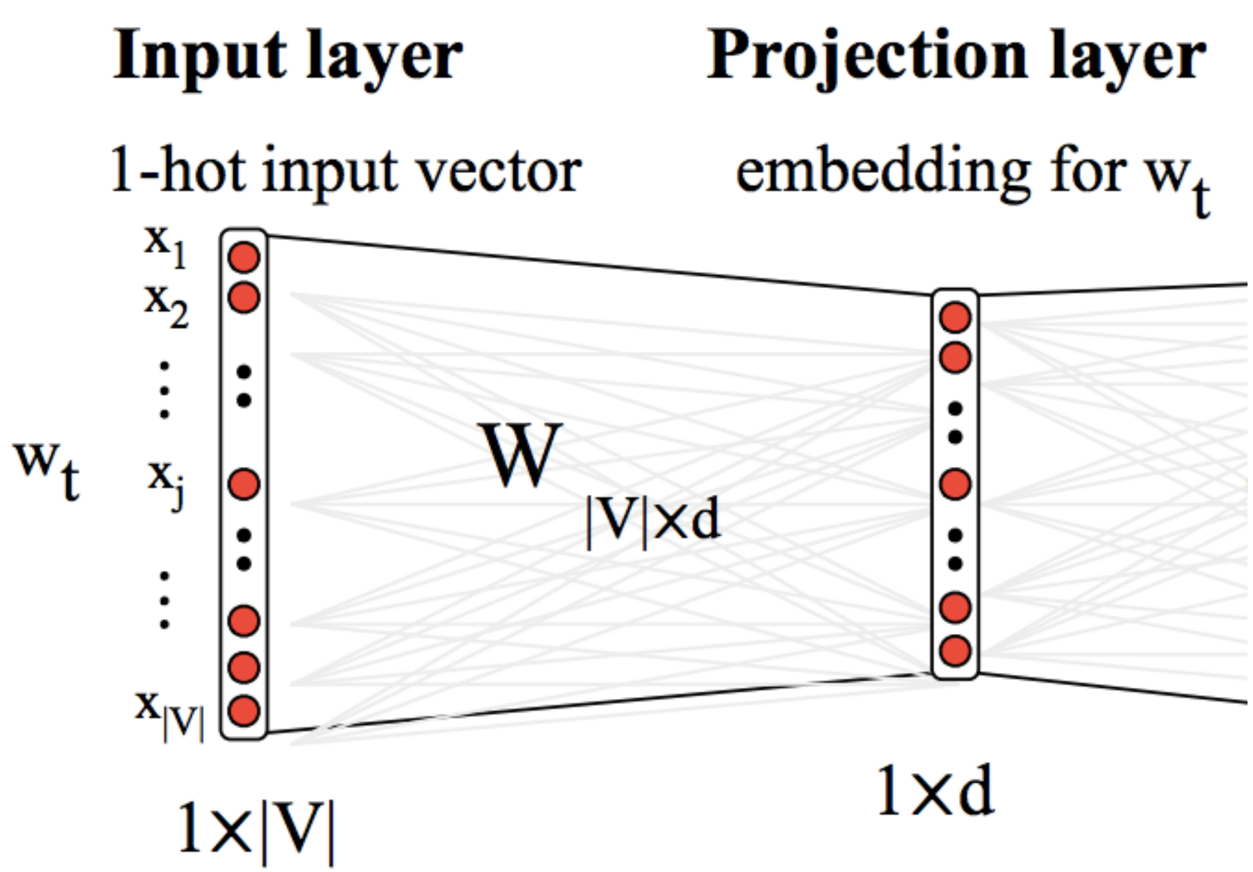
- A simple look up — the rows of this weight matrix are actually "input" word vectors



Hidden (Projection) Layer

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

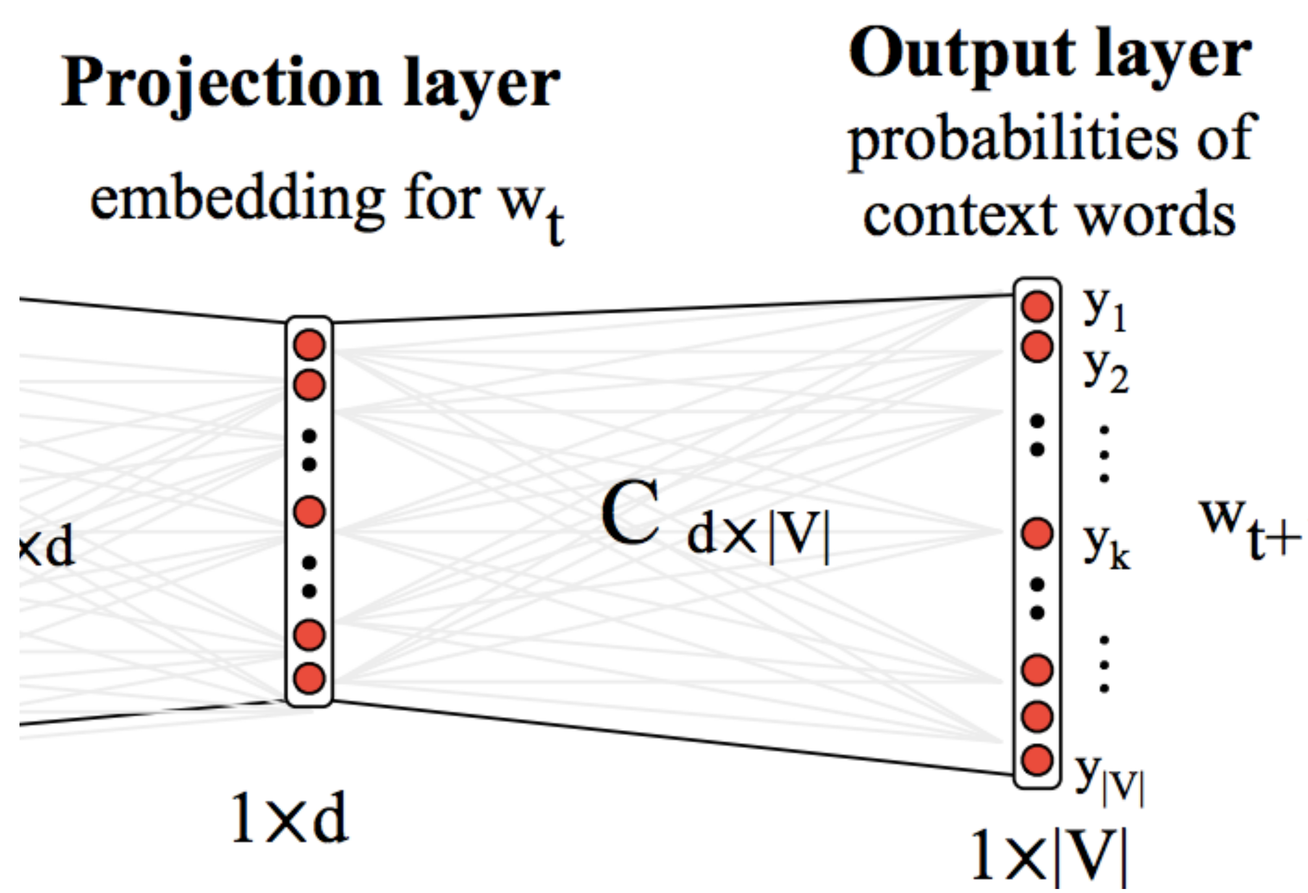
$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$



Output Layer

- predicts surrounding “outside” (context) words given the “center” word ⟶ A classification problem!

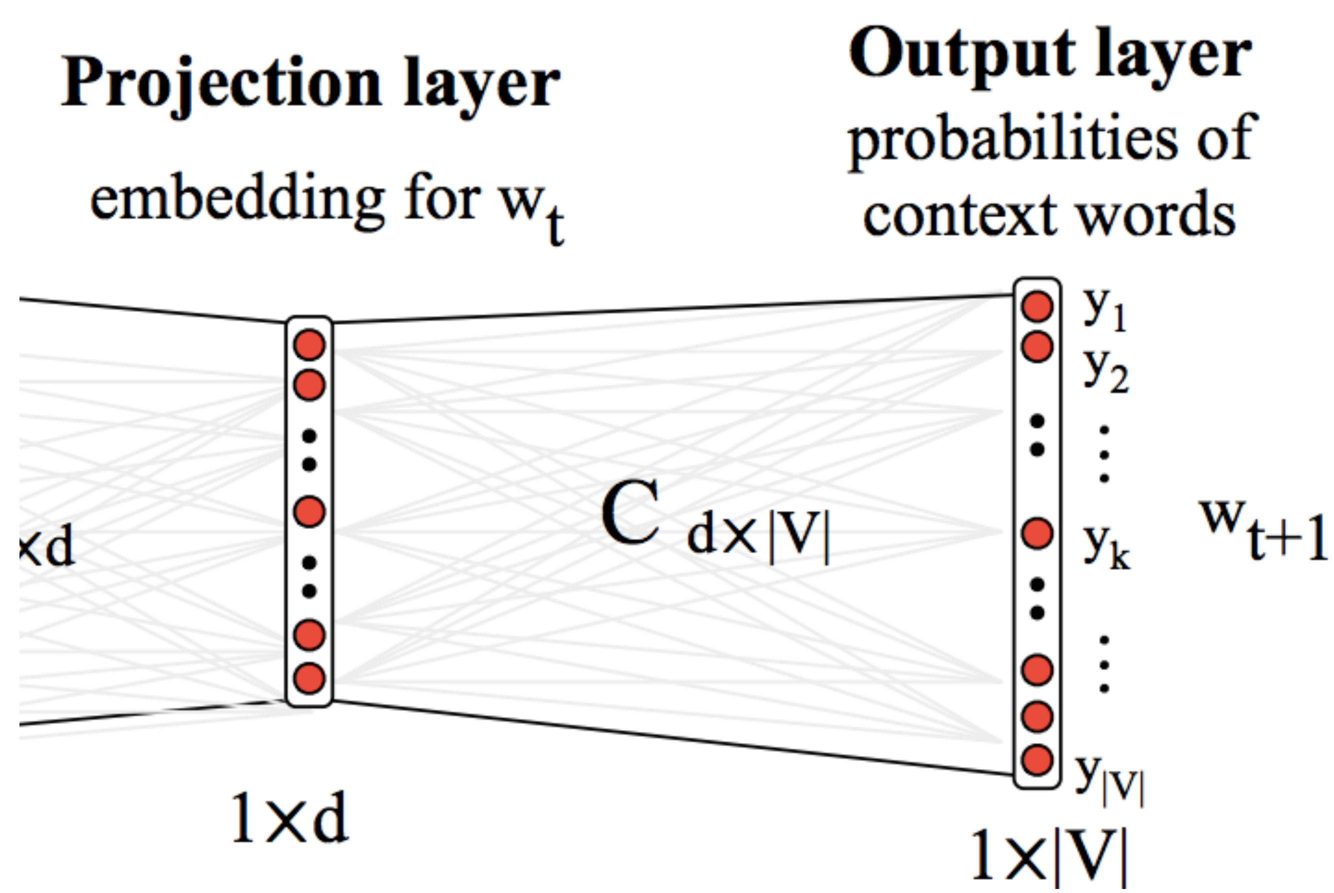
- Softmax Regression = Multi-class Logistic Regression



Output Layer

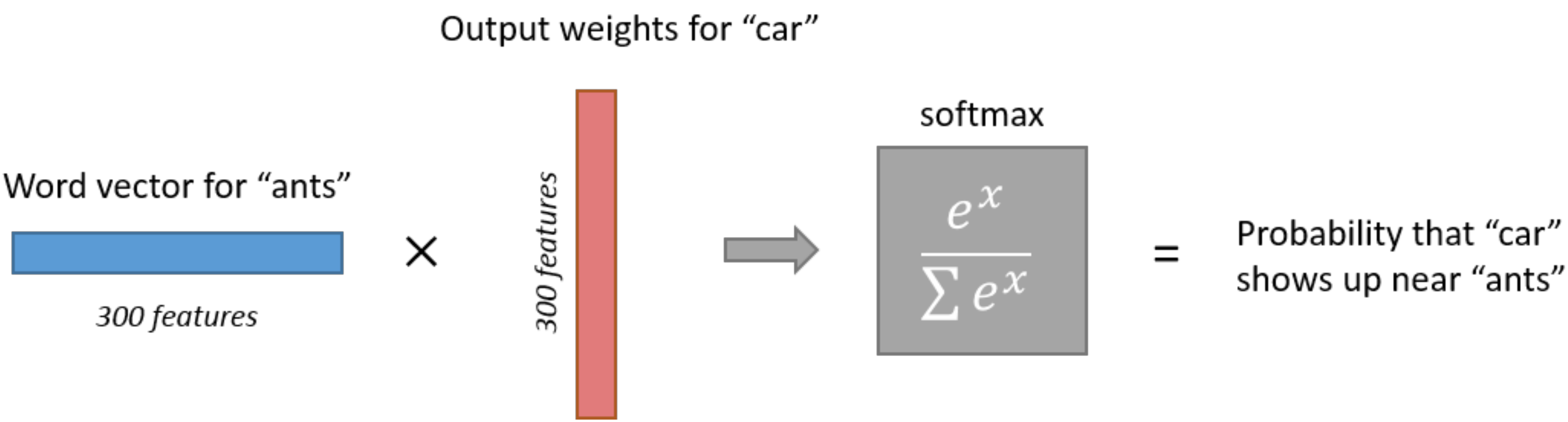
- Objective function: maximize the log probability of any "outside" (context) word given the "center" word

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



Output Layer

- Intuition

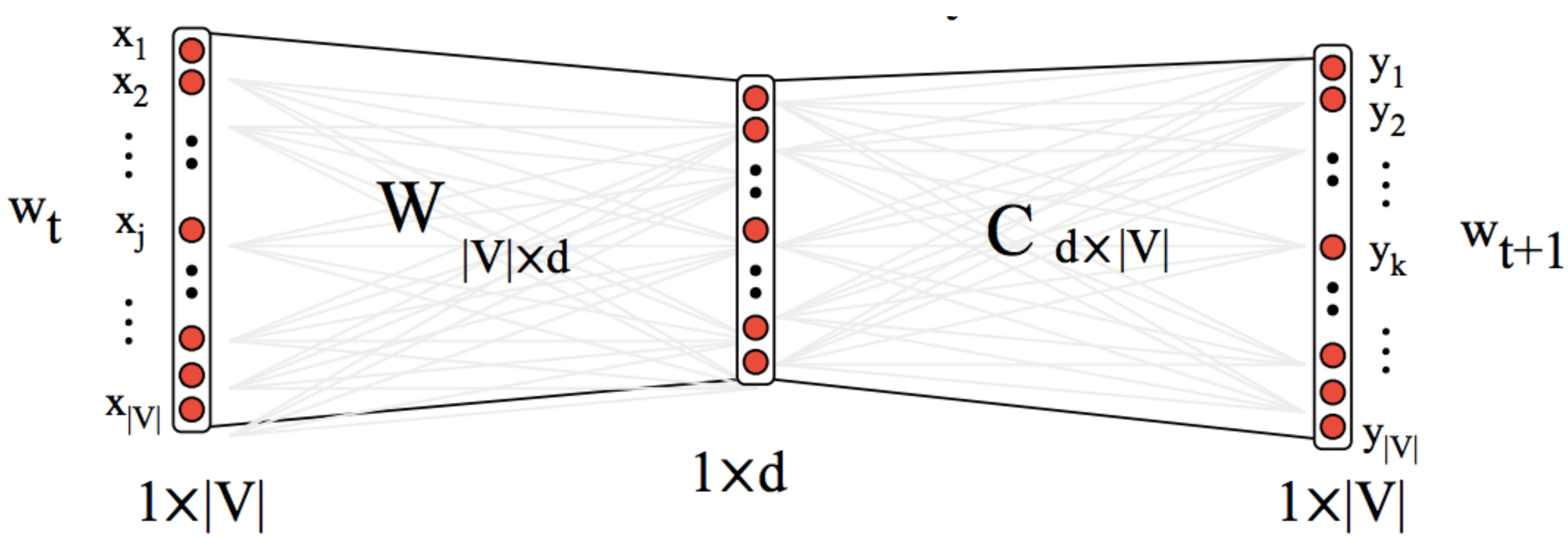


Output Layer

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

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

- so, every word has two vectors!



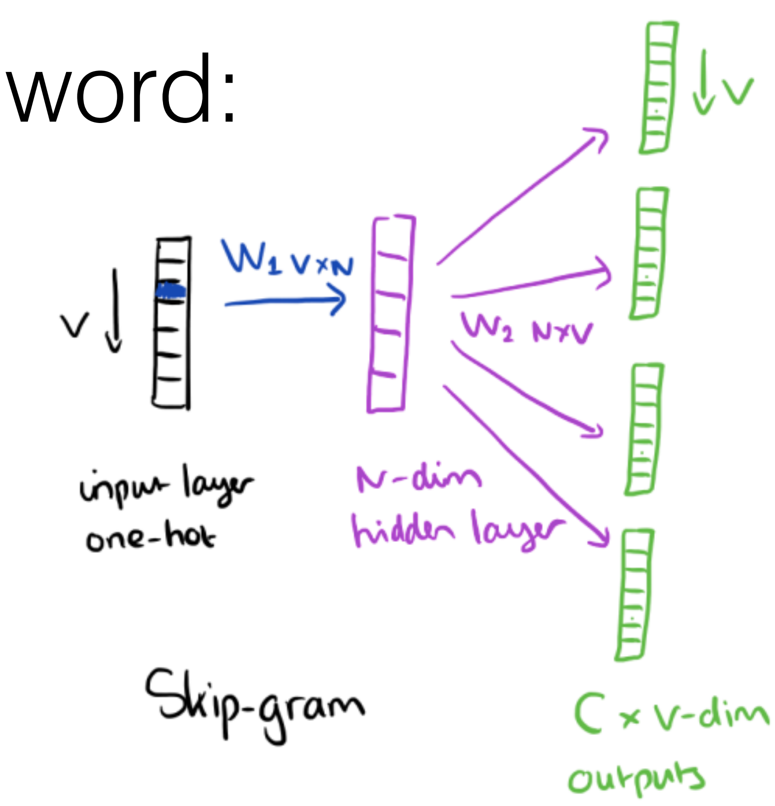
Gradient Descent

- 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:

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



Gradient Descent

- Basics:

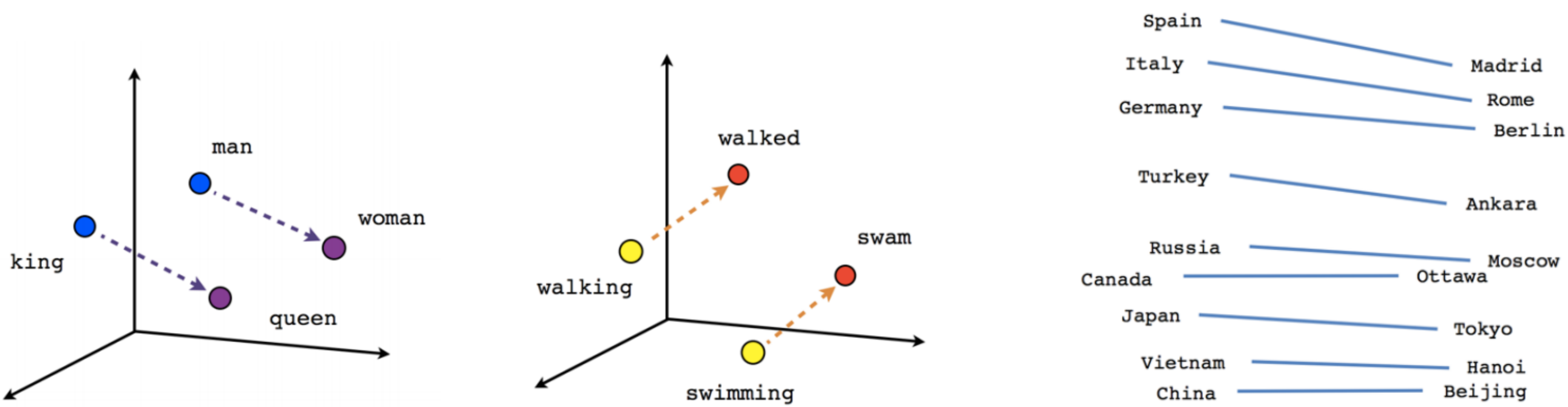
$$\frac{\partial \mathrm{x}^T a}{\partial \mathrm{x}} = \frac{\partial a^T \mathrm{x}}{\partial \mathrm{x}} = a$$

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

- Chain Rule:

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

Visualization



Male-Female Verb tense Country-Capital

Source: tensorflow.org

Visualization

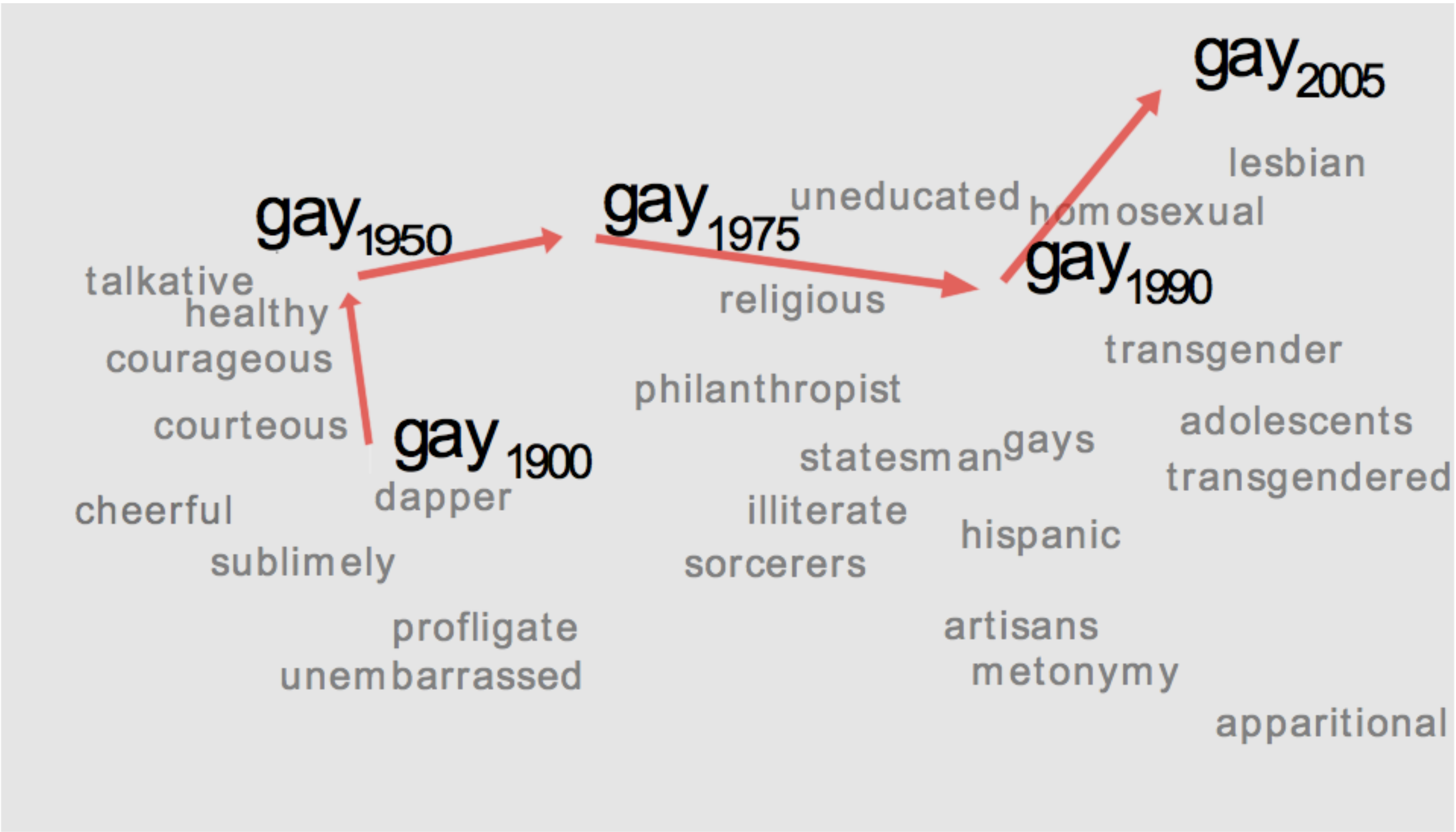
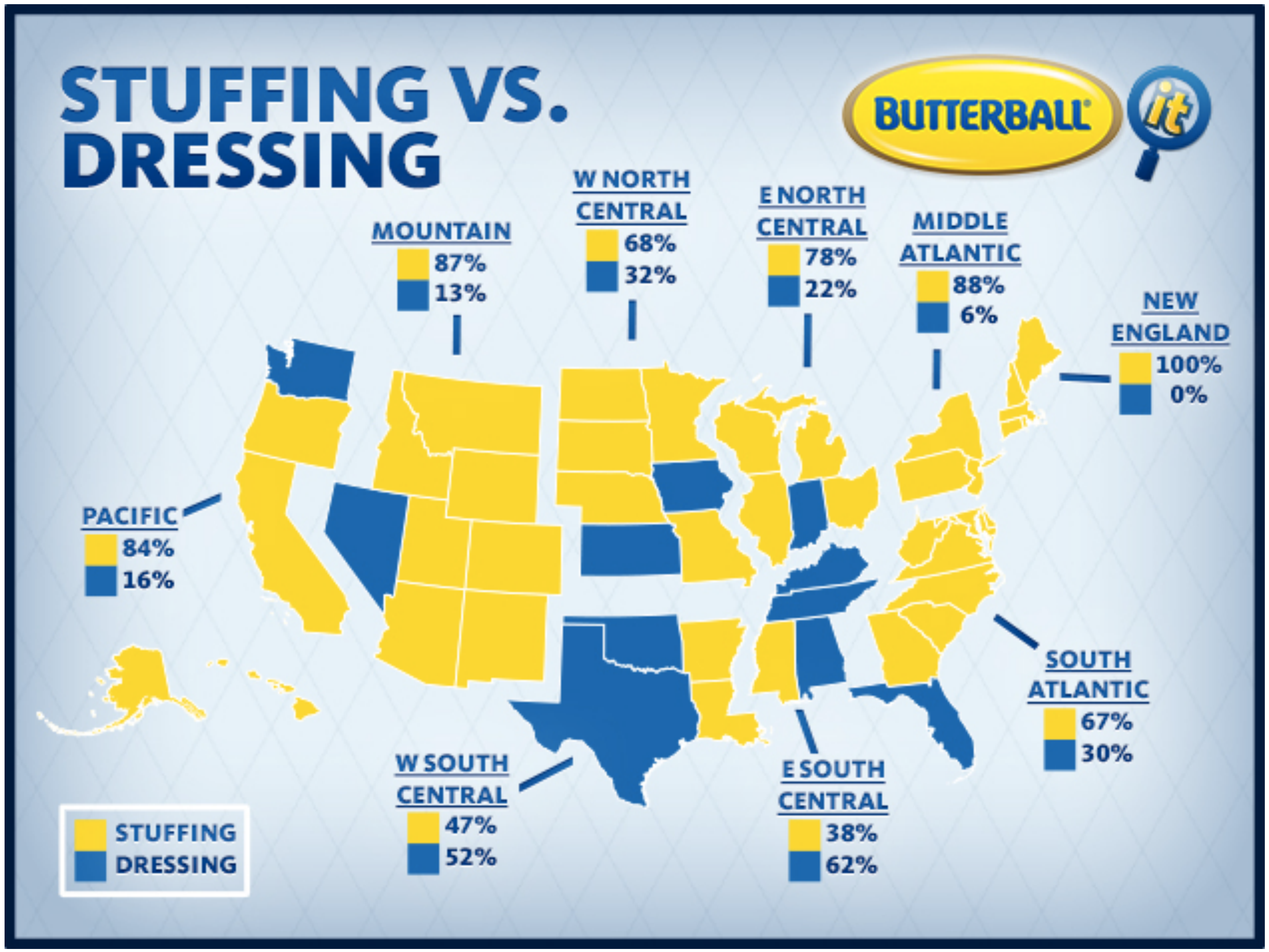


Figure 1: A 2-dimensional projection of the latent semantic space captured by our algorithm. Notice the semantic trajectory of the word **gay** transitioning meaning in the space.

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 CyberPulseTM Advisory Panel. Research was conducted in May 2007. The overall sampling error for the survey is +/-3% at the 95% level of confidence.

Source: http://www.butterball.com/how-tos/stuffing-vs-dressing

Thank You!

Follow @cocoweixu

Instructor: Wei Xu

www.cis.upenn.edu/~xwe/

Course Website: socialmedia-class.org