

# Distributional Semantics, Pt. II

LING 571 — Deep Processing for NLP

November 10, 2021

Shane Steinert-Threlkeld

# Recap

- We can represent words as vectors
  - Each entry in the vector is a score for its correlation with another word
  - If a word occurs frequently with “tall” compared to other words, we might assume height is an important quality of the word
- In these extremely large vectors, most entries are zero

# Roadmap

- Curse of Dimensionality
- Dimensionality Reduction
  - Principle Components Analysis (PCA)
  - Singular Value Decomposition (SVD) / LSA
- Prediction-based Methods
  - CBOW / Skip-gram (word2vec)
- Word Sense Disambiguation

# The Curse of Dimensionality

# The Problem with High Dimensionality

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

# The Problem with High Dimensionality

The cosine similarity for these words will be zero!

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

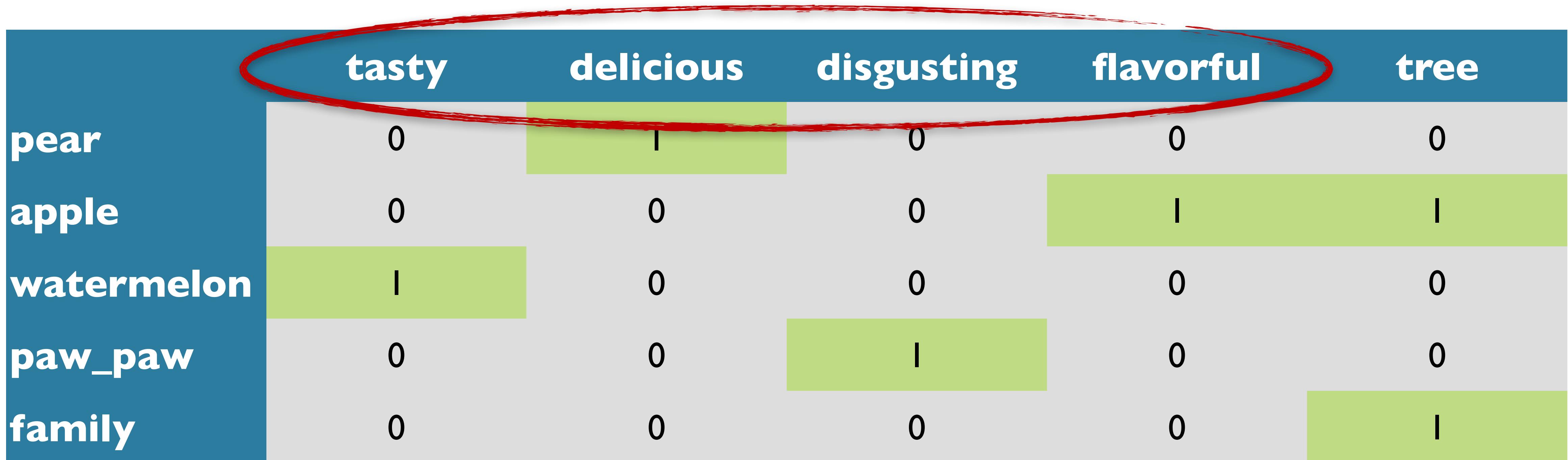
# The Problem with High Dimensionality

The cosine similarity for these words will be >0 (0.293)

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

# The Problem with High Dimensionality

But if we could collapse all of these into one “meta-dimension”...



# The Problem with High Dimensionality

Now, these things have “taste” associated with them as a concept

	<i>&lt;taste&gt;</i>	tree
<b>pear</b>		0
<b>apple</b>		
<b>watermelon</b>		0
<b>paw_paw</b>		0
<b>family</b>	0	

# Curse of Dimensionality

- Vector representations are sparse, very high dimensional
  - # of words in vocabulary
  - # of relations  $\times$  # words, etc

# Curse of Dimensionality

- Vector representations are sparse, very high dimensional
  - # of words in vocabulary
  - # of relations  $\times$  # words, etc
- Google 1T 5-gram corpus:
  - In bigram  $1M \times 1M$  matrix: < 0.05% non-zero values

# Curse of Dimensionality

- Vector representations are sparse, very high dimensional
  - # of words in vocabulary
  - # of relations  $\times$  # words, etc
- Google 1T 5-gram corpus:
  - In bigram  $1M \times 1M$  matrix: < 0.05% non-zero values
  - Computationally hard to manage
    - Lots of zeroes
    - Can miss underlying relations

# Roadmap

- Curse of Dimensionality
- **Dimensionality Reduction**
  - Principle Components Analysis (PCA)
  - Singular Value Decomposition (SVD) / LSA
- Prediction-based Methods
  - CBOW / Skip-gram (word2vec)
- Word Sense Disambiguation

# Reducing Dimensionality

- Can we use *fewer* features to build our matrices?

# Reducing Dimensionality

- Can we use ***fewer*** features to build our matrices?
- Ideally with
  - High ***frequency*** — means fewer zeroes in our matrix
  - High ***variance*** — larger spread over values makes items easier to separate

# Reducing Dimensionality

- One approach – *filter* out features
  - Can exclude terms with too few occurrences
  - Can include only top  $X$  most frequently seen features
  - $\chi^2$  selection

# Reducing Dimensionality

# Reducing Dimensionality

- Things to watch out for:
  - Feature correlation – if features strongly correlated, give redundant information
  - Joint feature selection complex, computationally expensive

# Reducing Dimensionality

- Approaches to project into lower-dimensional spaces
  - Principal Components Analysis (PCA)
  - Locality Preserving Projections (LPP) [[link](#)]
  - Singular Value Decomposition (SVD)

# Reducing Dimensionality

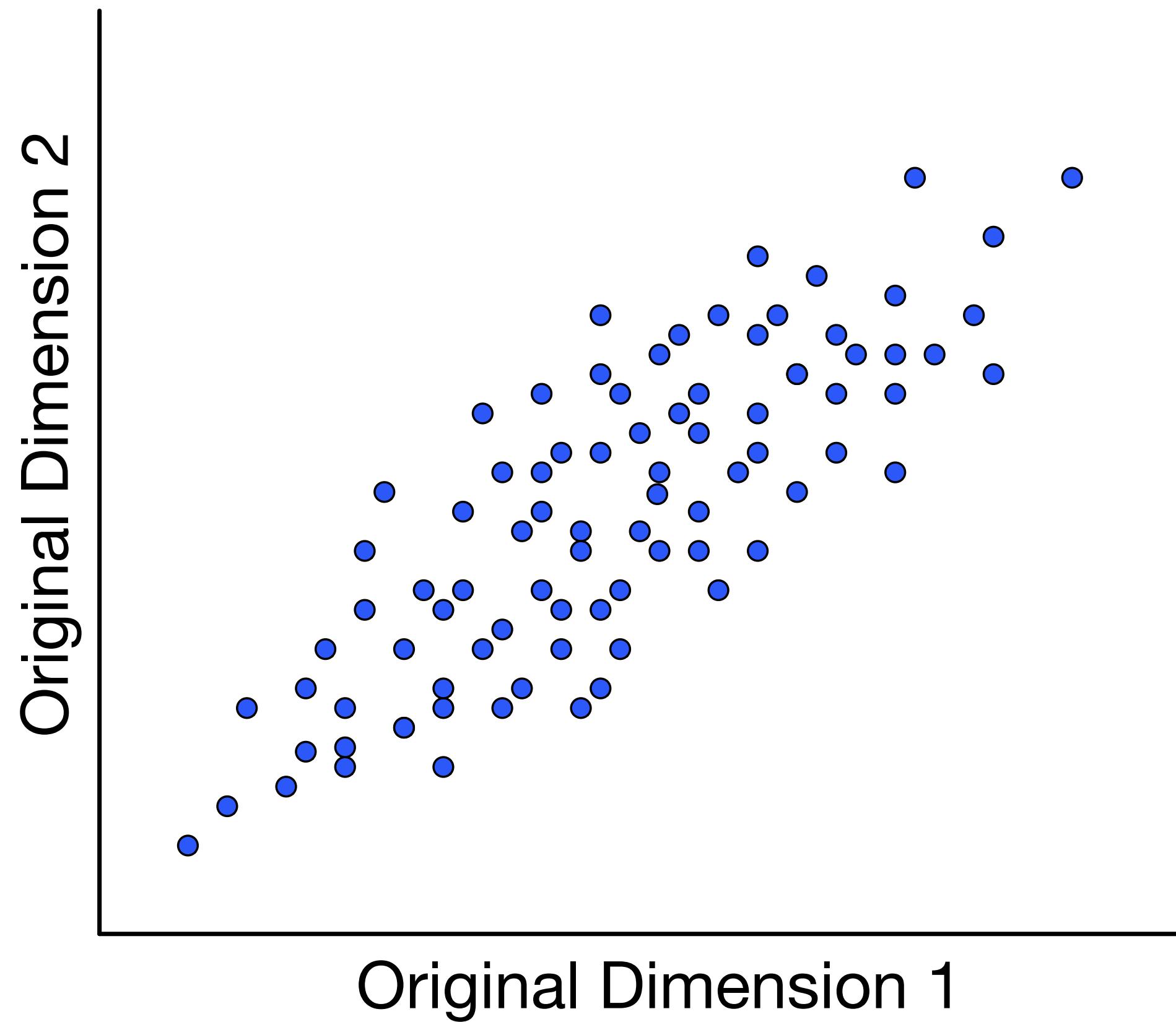
# Reducing Dimensionality

- All approaches create new lower dimensional space that
  - Preserves distances between data points
    - (Keep like with like)

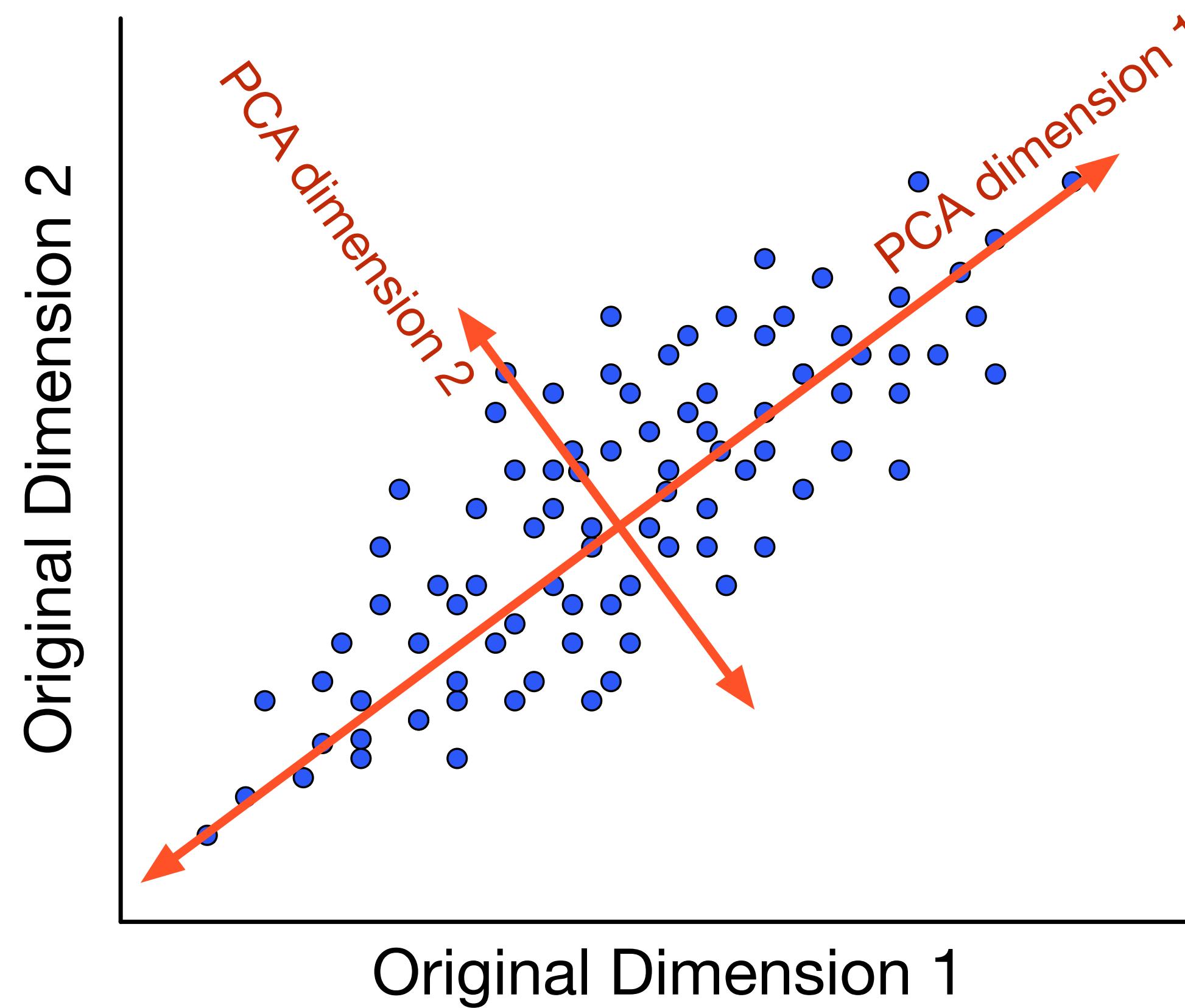
# Reducing Dimensionality

- All approaches create new lower dimensional space that
  - Preserves distances between data points
    - (Keep like with like)
  - Approaches differ on exactly what is preserved

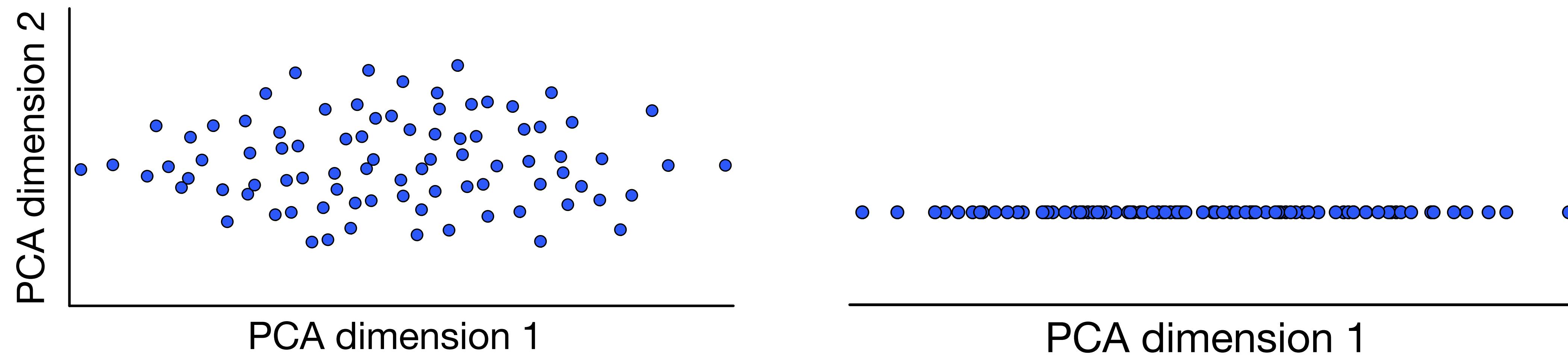
# Principal Component Analysis (PCA)



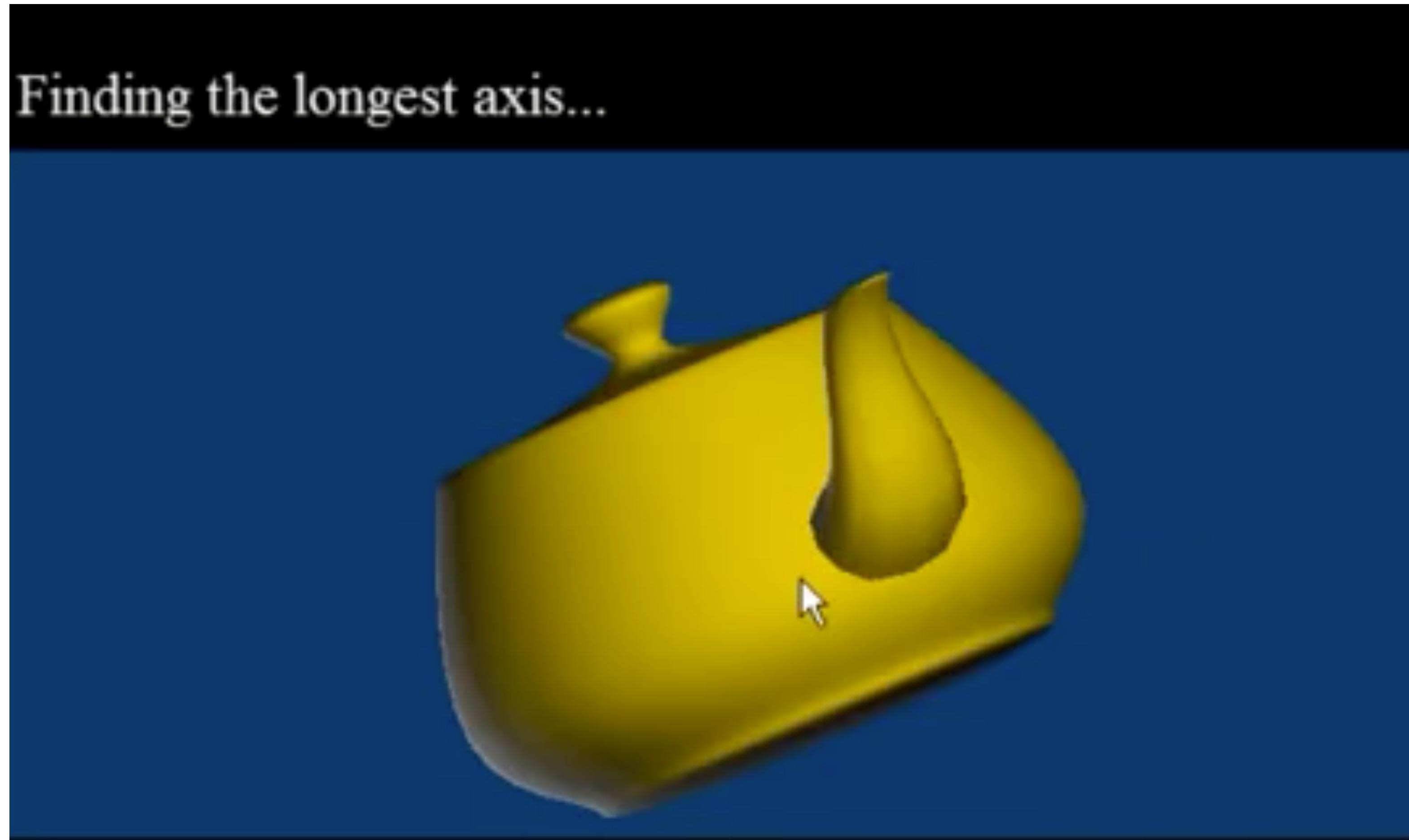
# Principal Component Analysis (PCA)



# Principal Component Analysis (PCA)

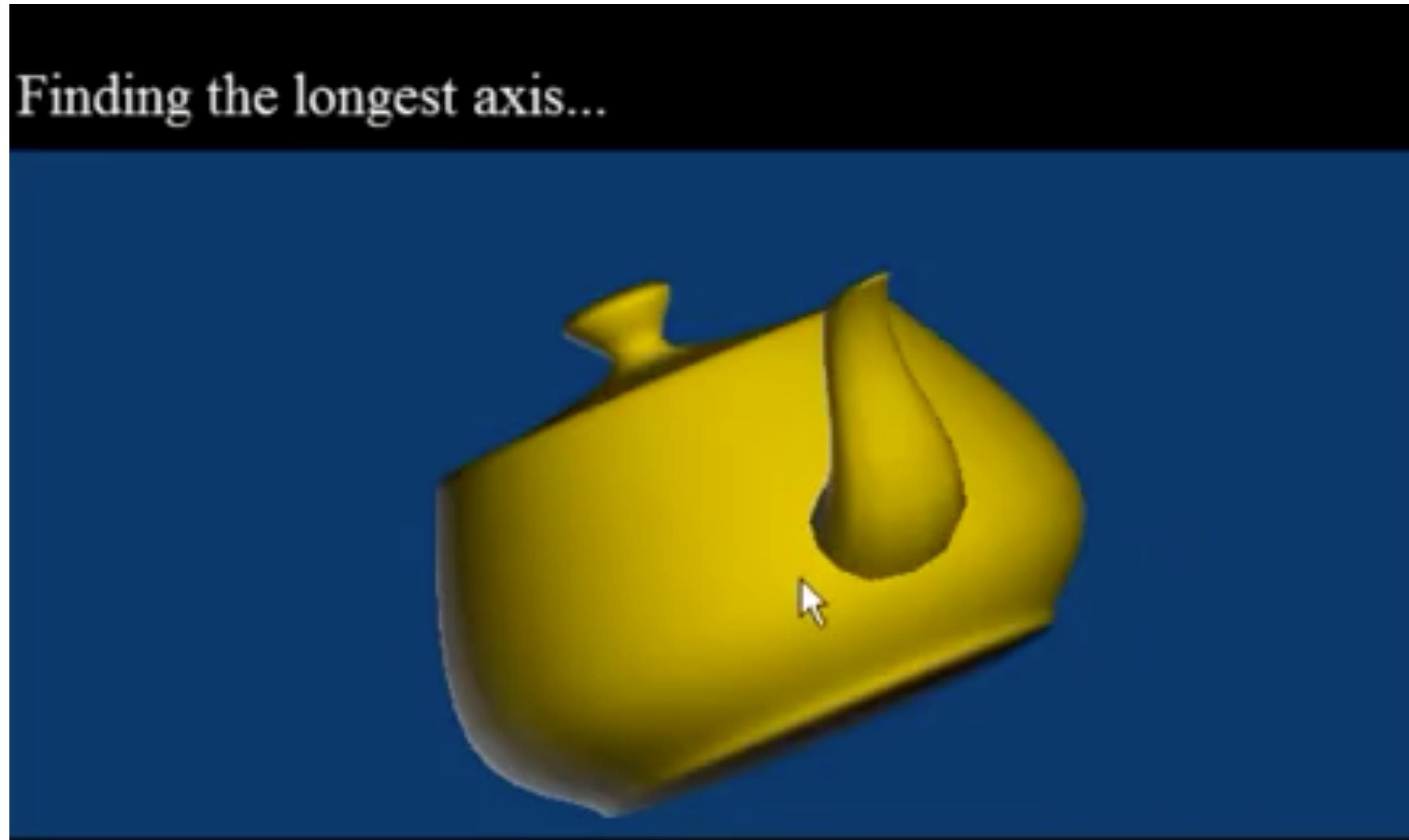


# Principal Component Analysis (PCA)



via [[A layman's introduction to PCA](#)]

# Principal Component Analysis (PCA)



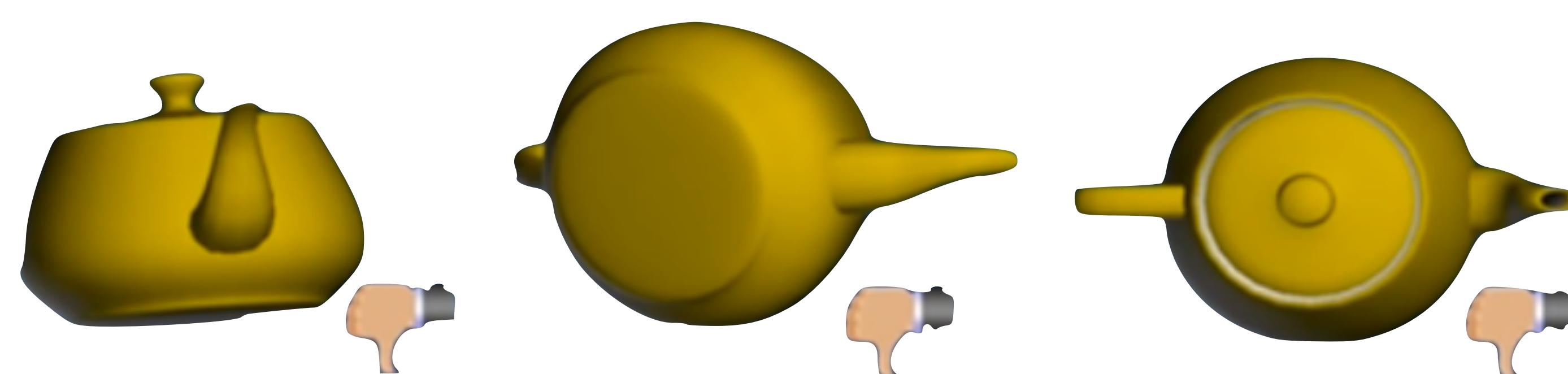
via [[A layman's introduction to PCA](#)]

# Principal Component Analysis (PCA)

This →



These →



via [[A layman's introduction to PCA](#)]

# Singular Value Decomposition (SVD)

- Enables creation of reduced dimension model
  - Low rank approximation of original matrix
  - Best-fit at that rank (in least-squares sense)

# Singular Value Decomposition (SVD)

- Original matrix: high dimensional, sparse
  - Similarities missed due to word choice, etc
- Create new, projected space
  - More compact, better captures important variation
- Landauer et al (1998) argue identifies underlying “concepts”
  - Across words with related meanings

# Latent Semantic Analysis (LSA)

- Apply SVD to  $|V| \times c$  term-document matrix  $X$ 
  - $V \rightarrow$  Vocabulary
  - $c \rightarrow$  documents
  - $X$ 
    - row  $\rightarrow$  word
    - column  $\rightarrow$  document
    - cell  $\rightarrow$  count of word/document

# Latent Semantic Analysis (LSA)

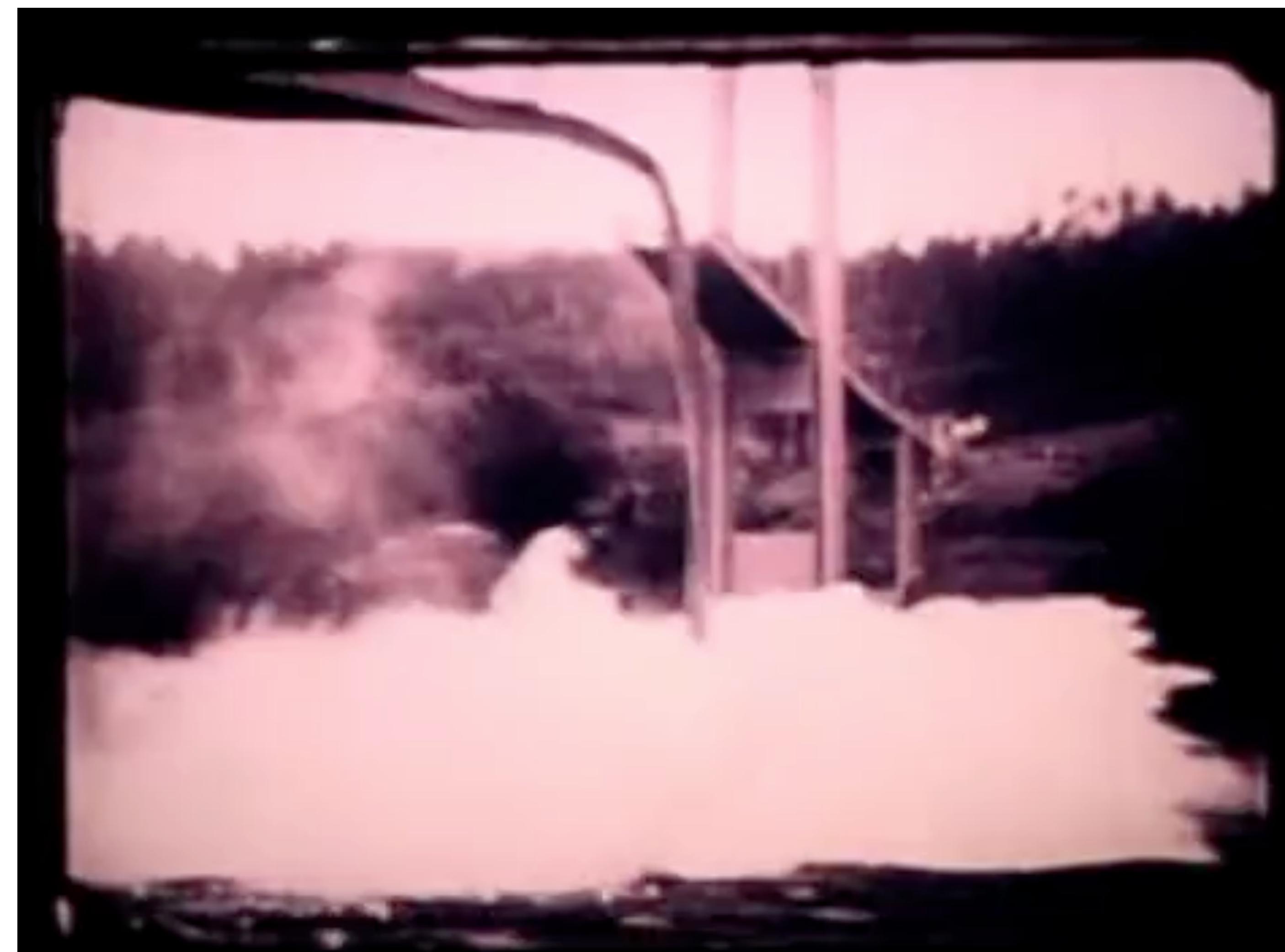
- Factor  $X$  into three new matrices:
  - $W \rightarrow$  one row per word, but columns are now arbitrary  $m$  dimensions
  - $\Sigma \rightarrow$  Diagonal matrix, where every (1,1) (2,2) etc... is the *rank* for  $m$
  - $C^T \rightarrow$  arbitrary  $m$  dimensions, as spread across c documents

$$\begin{matrix} \text{word-word} \\ \text{PPMI matrix} \\ X \end{matrix} = \begin{matrix} W \\ m \times m \end{matrix} \begin{matrix} \Sigma \\ m \times m \end{matrix} \begin{matrix} C \\ m \times c \end{matrix}$$

# SVD Animation

[youtu.be/R9UoFyqJca8](https://youtu.be/R9UoFyqJca8)

Enjoy some 3D Graphics from 1976!



# SVD Animation

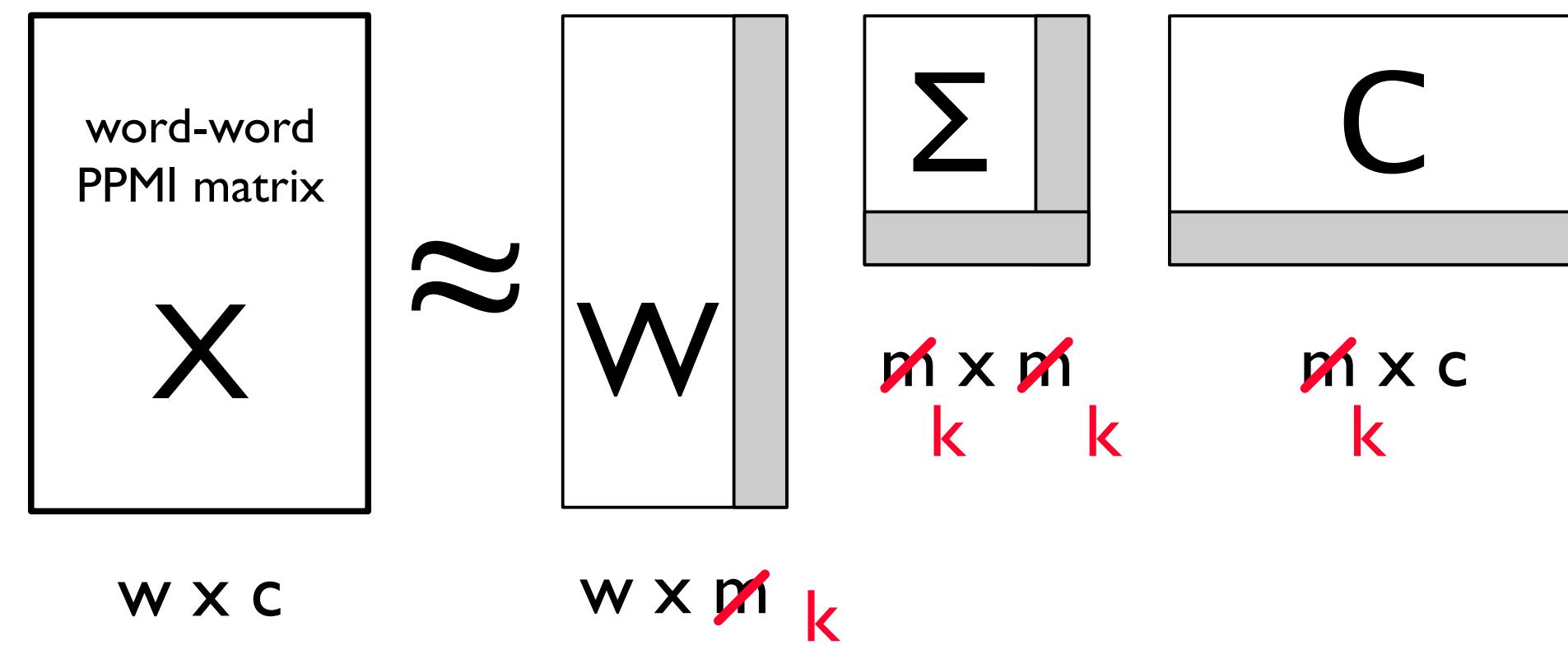
[youtu.be/R9UoFyqJca8](https://youtu.be/R9UoFyqJca8)

Enjoy some 3D Graphics from 1976!



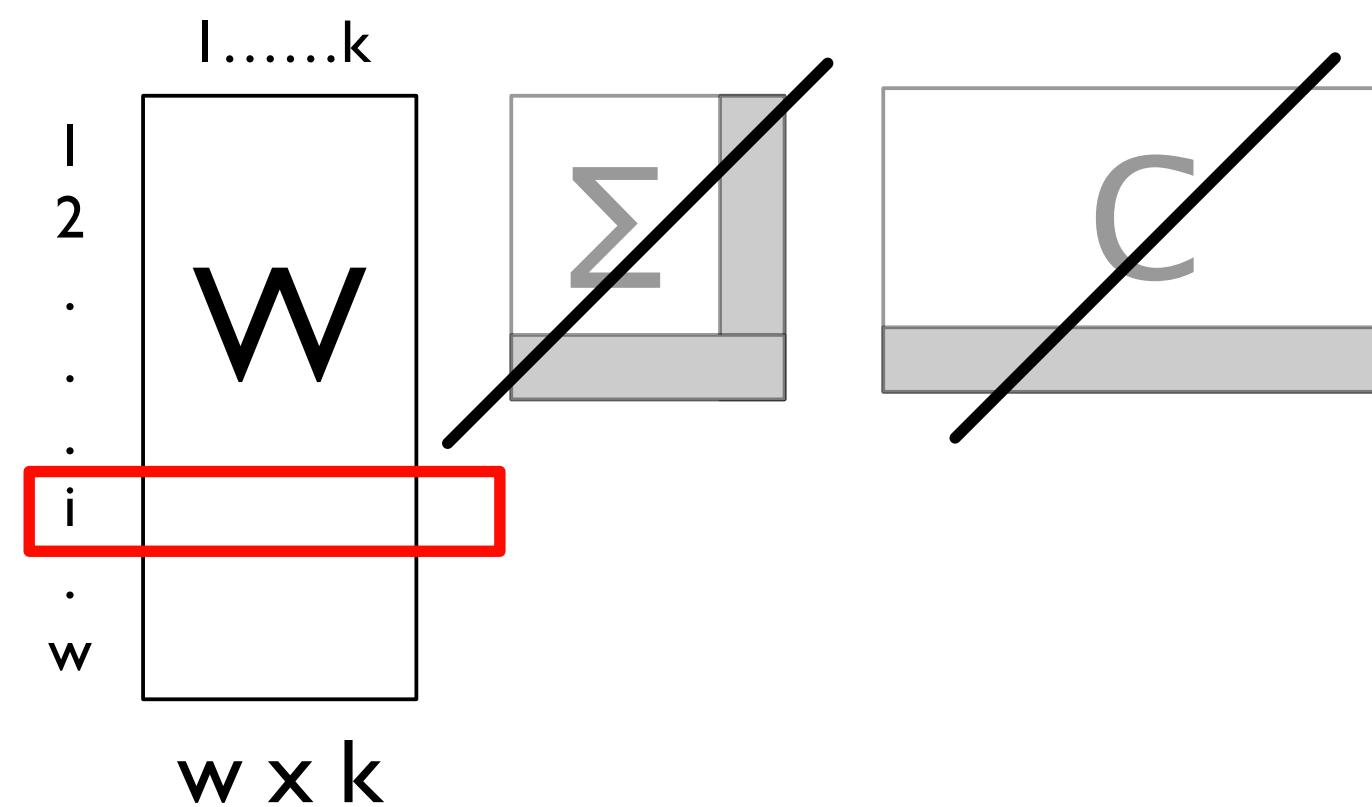
# Latent Semantic Analysis (LSA)

- LSA implementations typically:
- **truncate** initial  $m$  dimensions to top  $k$



# Latent Semantic Analysis (LSA)

- LSA implementations typically:
  - **truncate** initial  $m$  dimensions to top  $k$
  - then **discard**  $\Sigma$  and  $C$  matrices
    - Leaving matrix  $W$
    - Each row is now an “embedded” representation of each  $w$  across  $k$  dimensions



# Singular Value Decomposition (SVD)

Original Matrix  $X$  (zeroes blank)

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
User1	1	1	1		
User2	3	3	3		
User3	4	4	4		
User4	5	5	5		
User5		2		4	4
User6				5	5
User7		1		2	2

# Singular Value Decomposition (SVD)

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

	m1	m2	m3
$\Sigma$ (m×m)	12.4		
m1			
m2			9.5
m3			1.3

	Avengers	Star Wars	Iron Man	Titanic	The Notebook	
C (m×c)	m1	0.56	0.59	0.56	0.09	0.09
	m2	0.12	-0.02	0.12	-0.69	-0.69
	m3	0.40	-0.80	0.40	0.09	0.09

# Singular Value Decomposition (SVD)

$W$  ( $w \times m$ )

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

$\Sigma$  ( $m \times m$ )

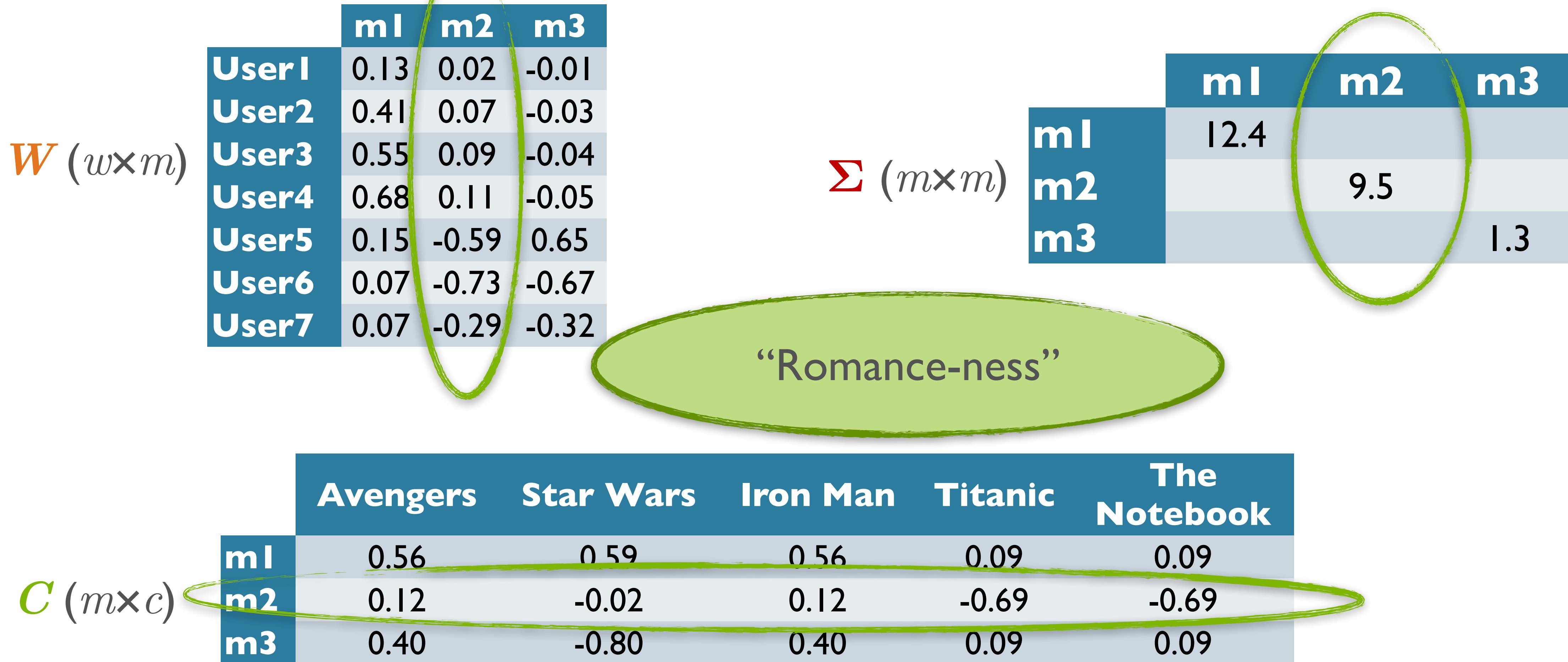
m1	m2	m3
m1	12.4	
m2		9.5
m3		1.3

“Sci-fi-ness”

$C$  ( $m \times c$ )

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
m1	0.56	0.59	0.56	0.09	0.09
m2	0.12	-0.02	0.12	-0.69	-0.69
m3	0.40	-0.80	0.40	0.09	0.09

# Singular Value Decomposition (SVD)



# Singular Value Decomposition (SVD)

$W$  ( $w \times m$ )

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

$\Sigma$  ( $m \times m$ )

m1	12.4
m2	9.5
m3	1.3

Catchall (noise)

$C$  ( $m \times c$ )

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
m1	0.56	0.59	0.56	0.09	0.09
m2	0.12	-0.02	0.12	-0.69	-0.69
m3	0.40	-0.80	0.40	0.09	0.09

# LSA Document Contexts

- Deerwester et al, 1990: "*Indexing by Latent Semantic Analysis*"
- Titles of scientific articles

- c1 **Human** machine **interface** for ABC **computer** applications
  - c2 A **survey** of **user** opinion of **computer system response time**
  - c3 The **EPS user interface** management system
  - c4 **System** and **human system** engineering testing of **EPS**
  - c5 Relation of **user** perceived **response time** to error measurement
- 
- m1 The generation of random, binary, ordered **trees**
  - m2 The intersection **graph** of paths in **trees**
  - m3 **Graph minors** IV:Widths of **trees** and well-quasi-ordering
  - m4 **Graph minors**:A **survey**

# Document Context Representation

- Term x document:
- $\text{corr}(\text{human}, \text{user}) = -0.38$ ;     $\text{corr}(\text{human}, \text{minors}) = -0.29$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

# Improved Representation

- Reduced dimension projection:
- $\text{corr}(\text{human}, \text{user}) = 0.98$ ;  $\text{corr}(\text{human}, \text{minors})=-0.83$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.05	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.33	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

# Python Tutorial for LSA

- For those interested in seeing how LSA works in practice:
- [technowiki.wordpress.com/2011/08/27/latent-semantic-analysis-lsa-tutorial/](http://technowiki.wordpress.com/2011/08/27/latent-semantic-analysis-lsa-tutorial/)

# Dimensionality Reduction for Visualization

- “I see well in many dimensions as long as the dimensions are around two.”
  - —Martin Shubek
- Even with ‘dense’ embeddings, techniques like PCA are useful for visualization
- Another popular one: t-SNE
- Useful for exploratory analysis

# Prediction-Based Models

# Prediction-based Embeddings

- LSA models: good, but expensive to compute

# Prediction-based Embeddings

- LSA models: good, but expensive to compute
- *Skip-gram* and *Continuous Bag of Words* (CBOW) models

# Prediction-based Embeddings

- LSA models: good, but expensive to compute
- *Skip-gram* and *Continuous Bag of Words* (CBOW) models
- Intuition:
  - Words with similar meanings share similar contexts
  - Train language models to learn to predict context words
  - Models train embeddings that make current word more like nearby words and less like distance words
  - Provably related to PPMI models under SVD

# Embeddings: Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW):

- $P(\text{word} \mid \text{context})$
- Input:  $(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \dots)$
- Output:  $p(w_t)$

# Embeddings: Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW):

- $P(\text{word} \mid \text{context})$
- Input:  $(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \dots)$
- Output:  $p(w_t)$

- Skip-gram:

- $P(\text{context} \mid \text{word})$
- Input:  $w_t$
- Output:  $p(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \dots)$

# Embeddings: Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW):

- $P(\text{word} \mid \text{context})$
- Input:  $(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \dots)$

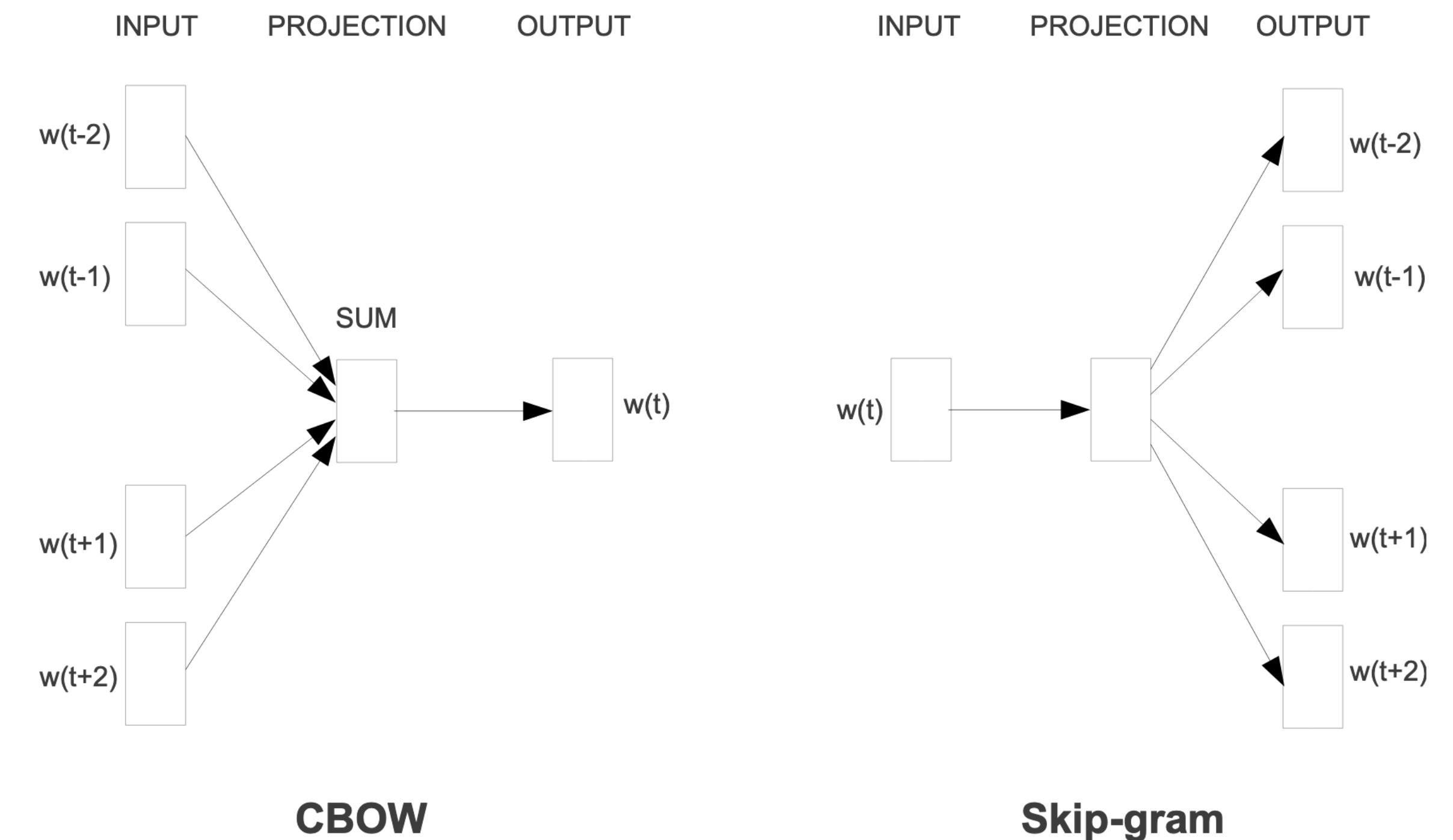
- Output:  $p(w_t)$

- Skip-gram:

- $P(\text{context} \mid \text{word})$

- Input:  $w_t$

- Output:  $p(w_{t-1}, w_{t-2}, w_{t+1}, w_{t+2} \dots)$



Mikolov et al 2013a (the OG word2vec paper)

# Skip-Gram Model

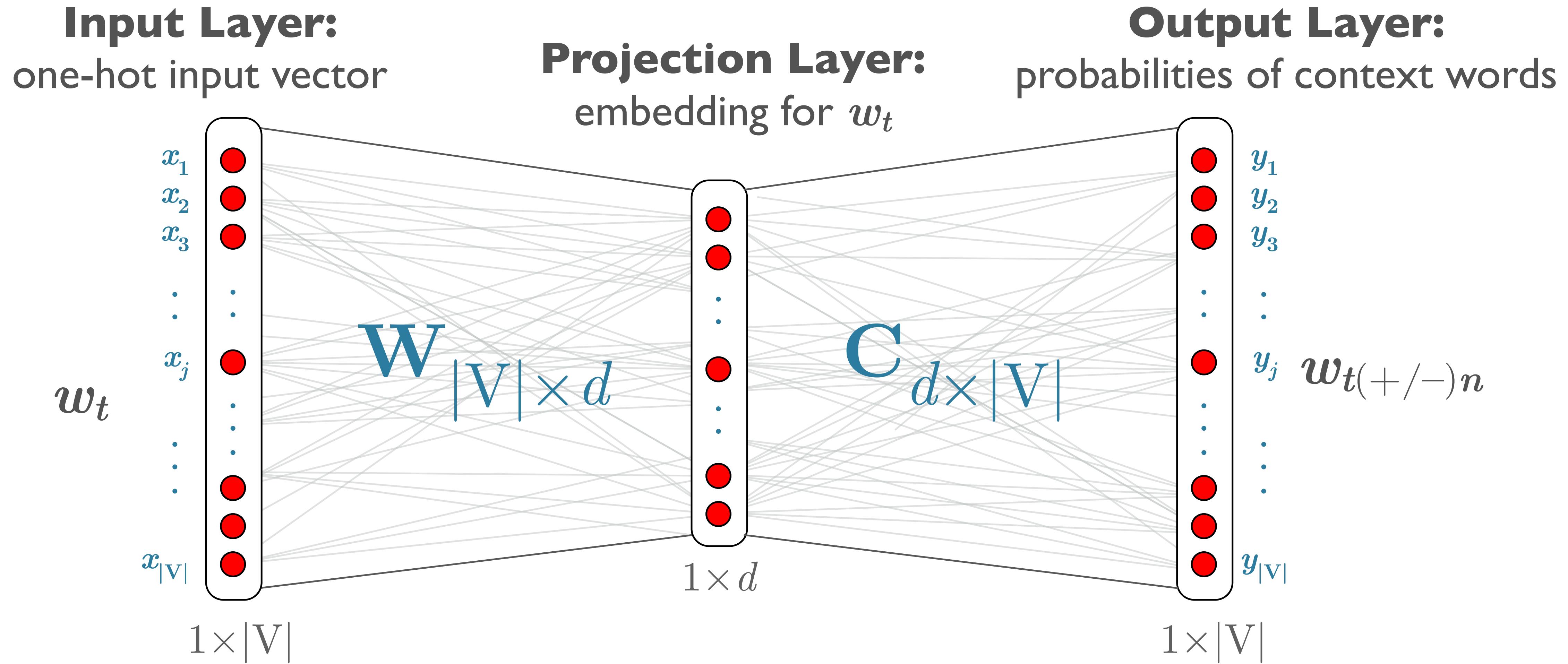
- Learns two embeddings
  - $W$ : word
  - $C$ : context of some fixed dimension

# Skip-Gram Model

- Learns two embeddings
  - $W$ : word
  - $C$ : context of some fixed dimension
- Prediction task:
  - Given a word, predict each neighbor word in window
  - Compute  $p(w_k|w_j)$  represented as  $c_k \cdot v_j$ 
    - For each context position
  - Convert to probability via softmax

$$p(w_k|w_j) = \frac{\exp(c_k \cdot v_j)}{\sum_{i \in |V|} \exp(c_i \cdot v_j)}$$

# Skip-Gram Network Visualization



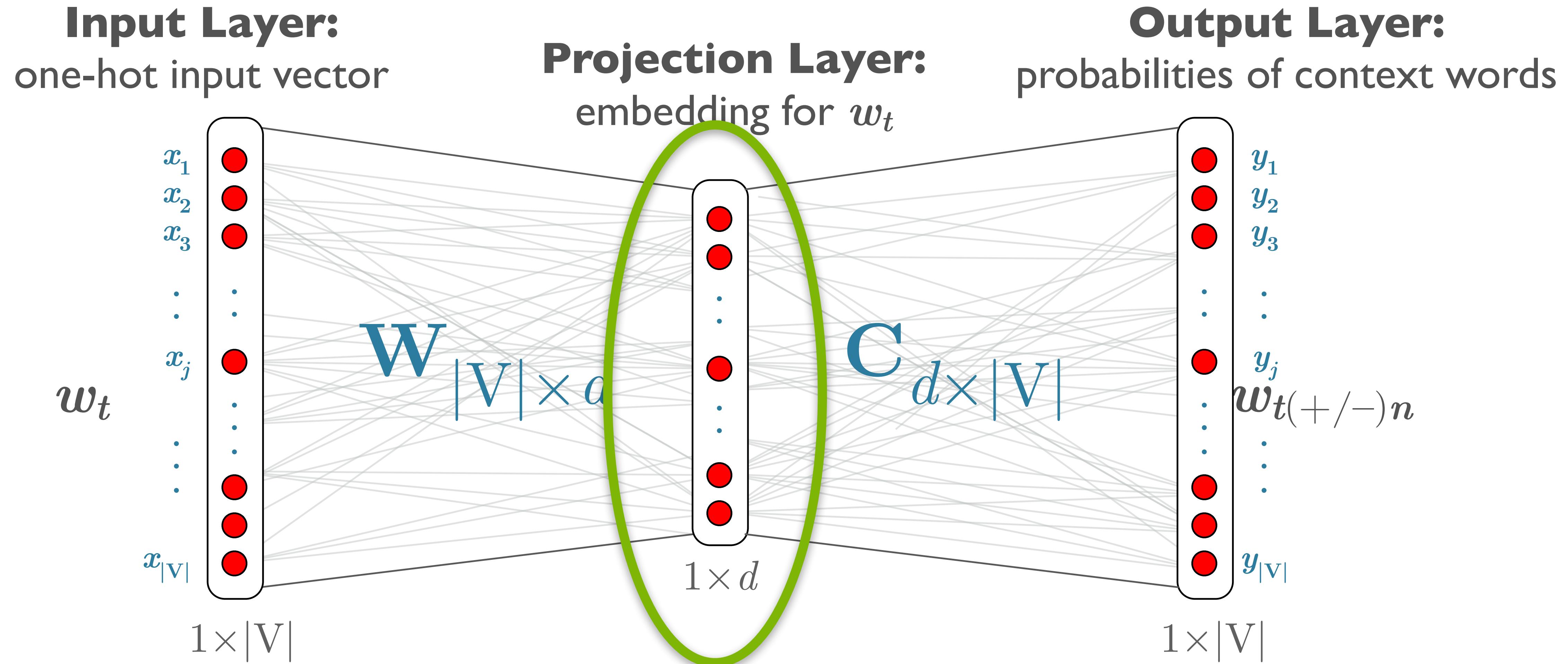
# Training The Model

- Issue:
  - Denominator computation is very expensive
- Strategy:
  - Approximate by negative sampling (efficient approximation to Noise Contrastive Estimation):
    - + example: true context word
    - – example:  $k$  other words, sampled

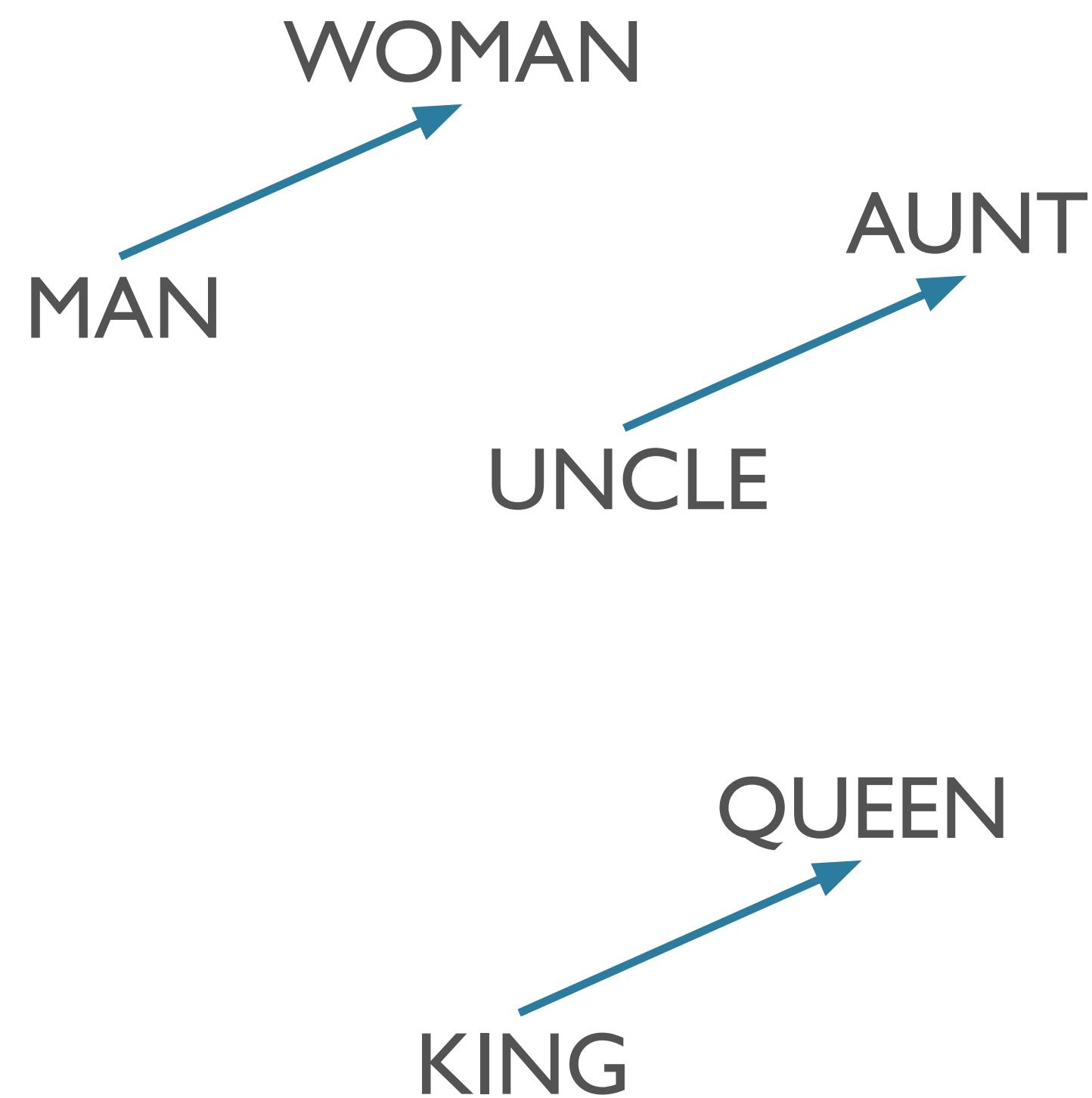
# Training The Model

- Approach:
  - Randomly initialize  $W, C$
  - Iterate over corpus, update w/ stochastic gradient descent
  - Update embeddings to improve loss function
- Use trained embeddings directly as word representations

# Skip-Gram Network Visualization

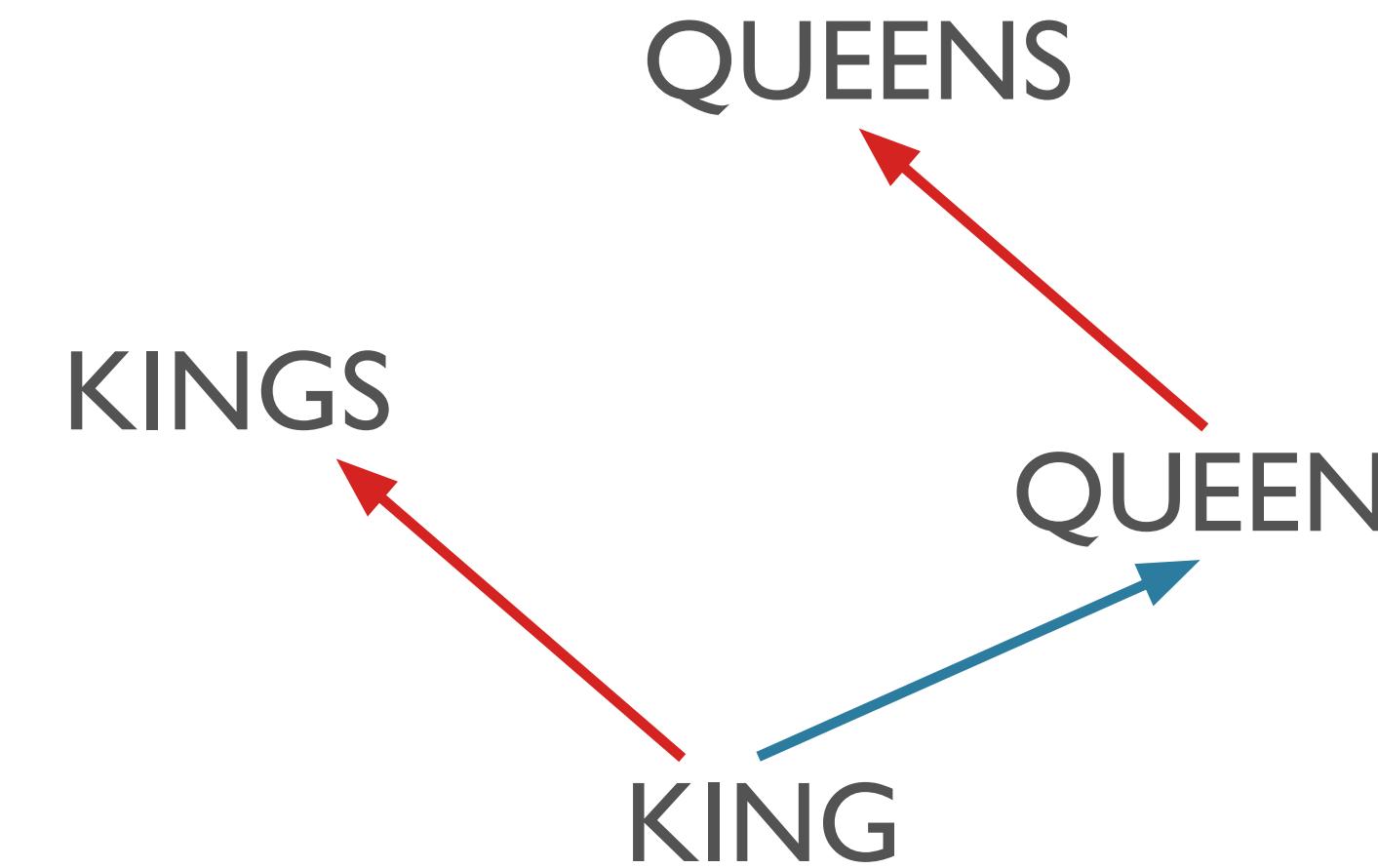
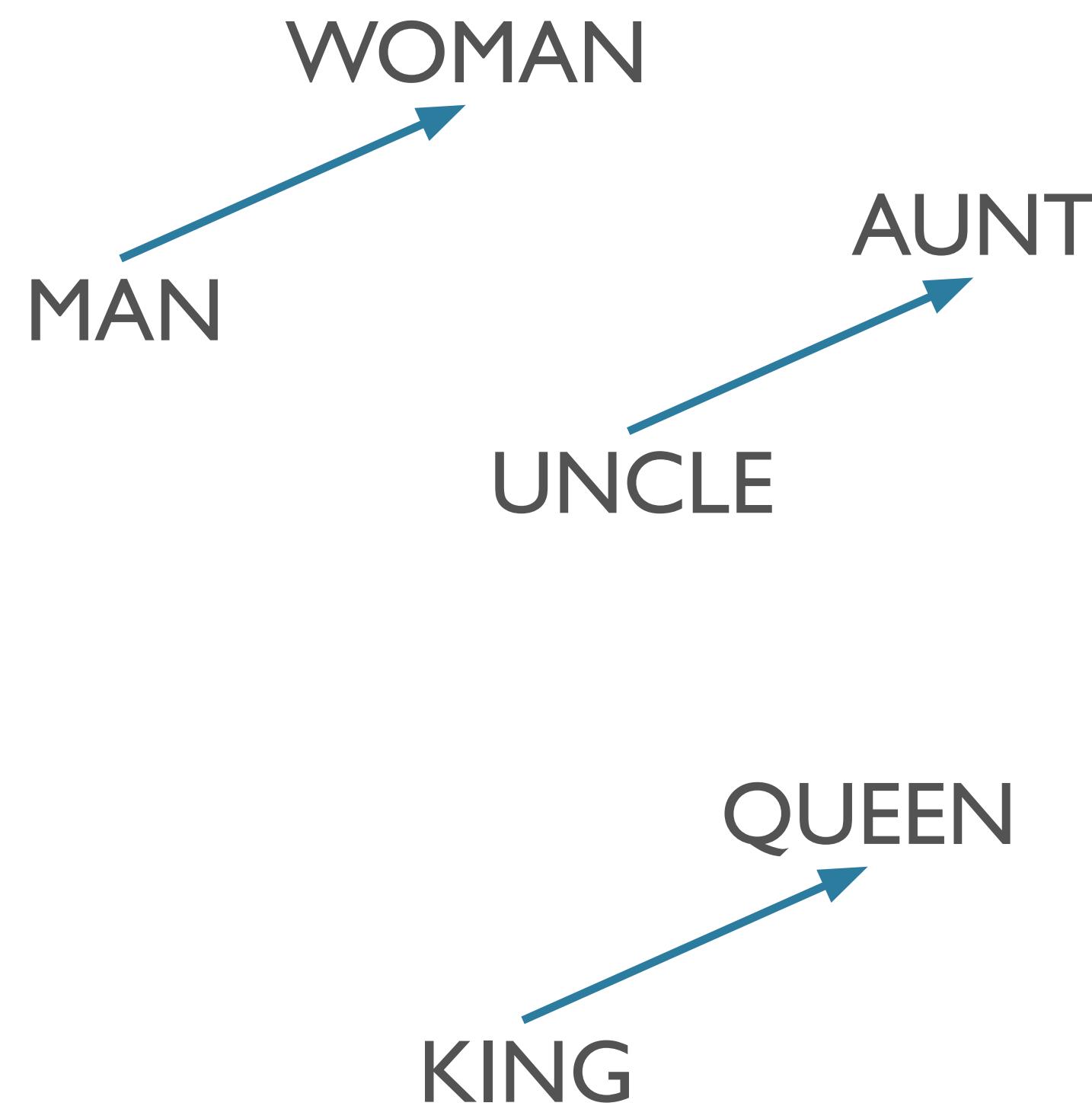


# Relationships via Offsets



Mikolov et al 2013b

# Relationships via Offsets



Mikolov et al 2013b

# One More Example

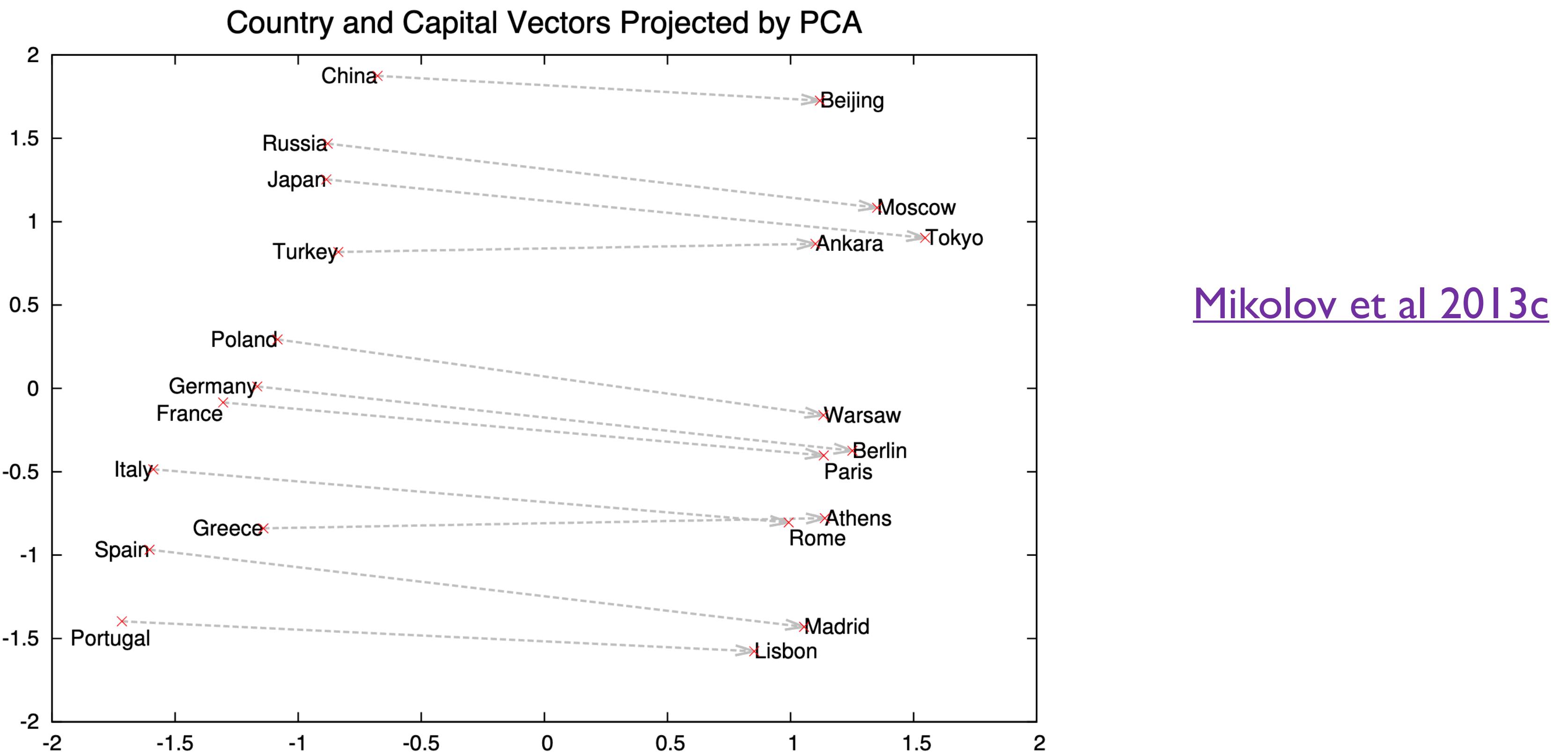
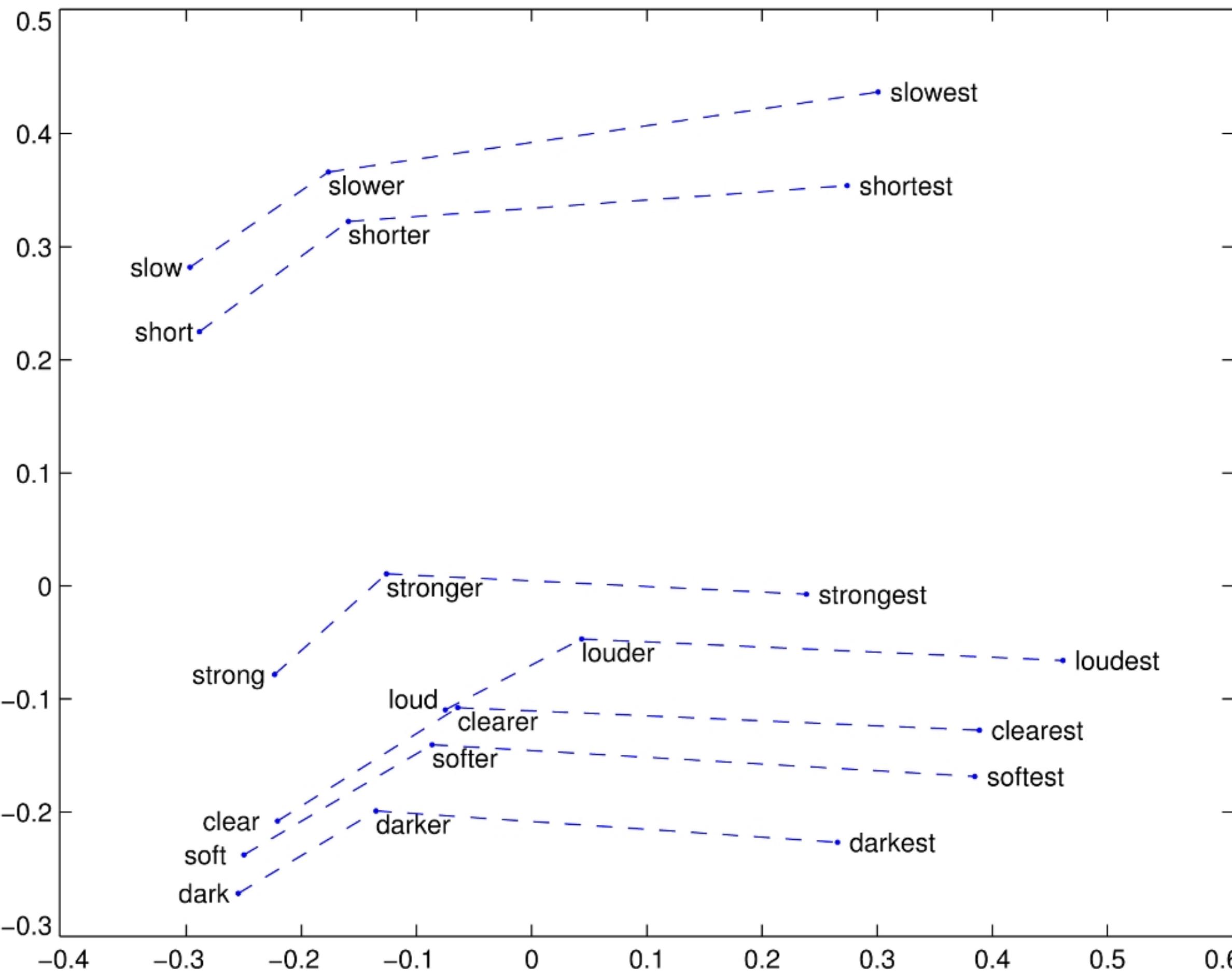


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

# One More Example



# Caveat Emptor

## Issues in evaluating semantic spaces using word analogies

Tal Linzen

LSCP & IJN

École Normale Supérieure

PSL Research University

tal.linzen@ens.fr

### Abstract

The offset method for solving word analogies has become a standard evaluation tool for vector-space semantic models: it is considered desirable for a space to represent semantic relations as consistent vector offsets. We show that the method's reliance on cosine similarity conflates offset consistency with largely irrelevant neighborhood structure, and propose simple baselines that should be used to improve the utility of the method in vector space evaluation.

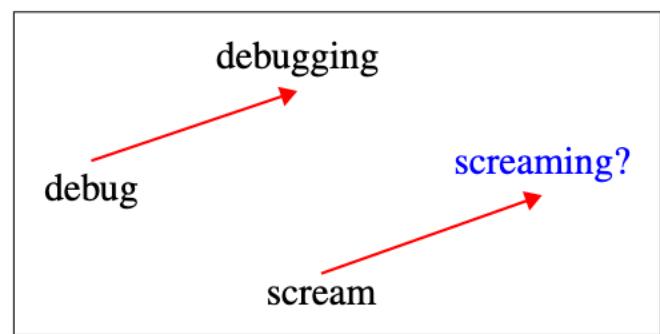


Figure 1: Using the vector offset method to solve the analogy task (Mikolov et al., 2013c).

cosine similarity to the landing point. Formally, if the analogy is given by

$$a : a^* :: b : \underline{\quad} \quad (1)$$

Linzen 2016, a.o.

# Diverse Applications

- Unsupervised POS tagging
- Word Sense Disambiguation
- Essay Scoring
- Document Retrieval
- Unsupervised Thesaurus Induction
- Ontology/Taxonomy Expansion
- Analogy Tests, Word Tests
- Topic Segmentation

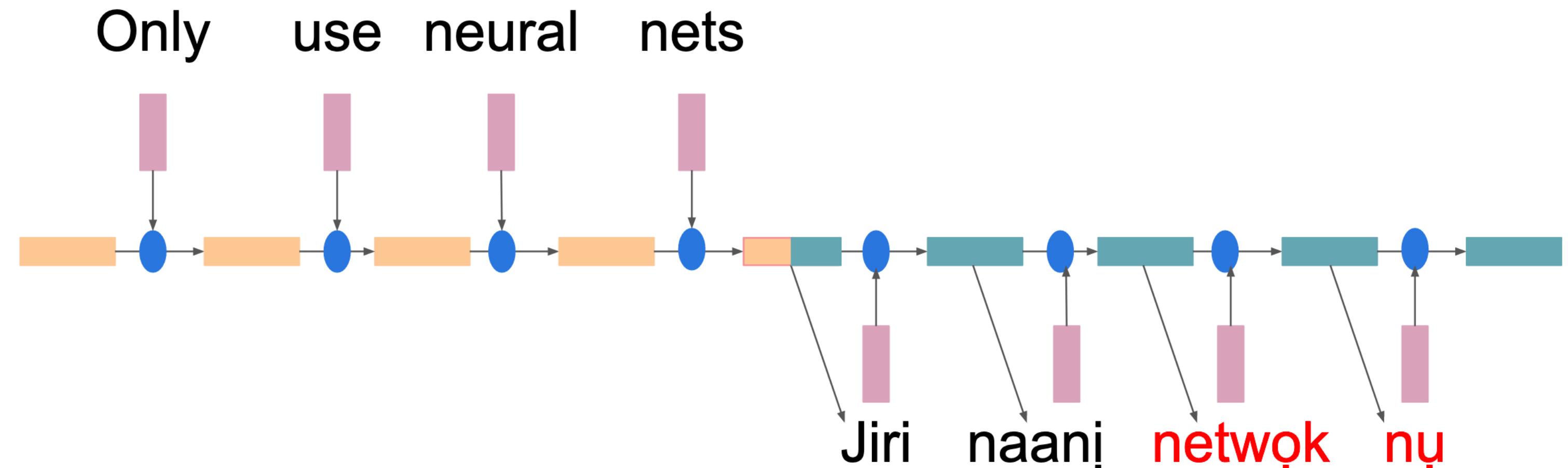
# General Recipe

# General Recipe

- Embedding layer (~300-dimensions):
  - download pre-trained embeddings
  - Use as look-up table for every word
  - Then feed those vectors into model of choice

# General Recipe

- Embedding layer (~300-dimensions):
  - download pre-trained embeddings
  - Use as look-up table for every word
  - Then feed those vectors into model of choice

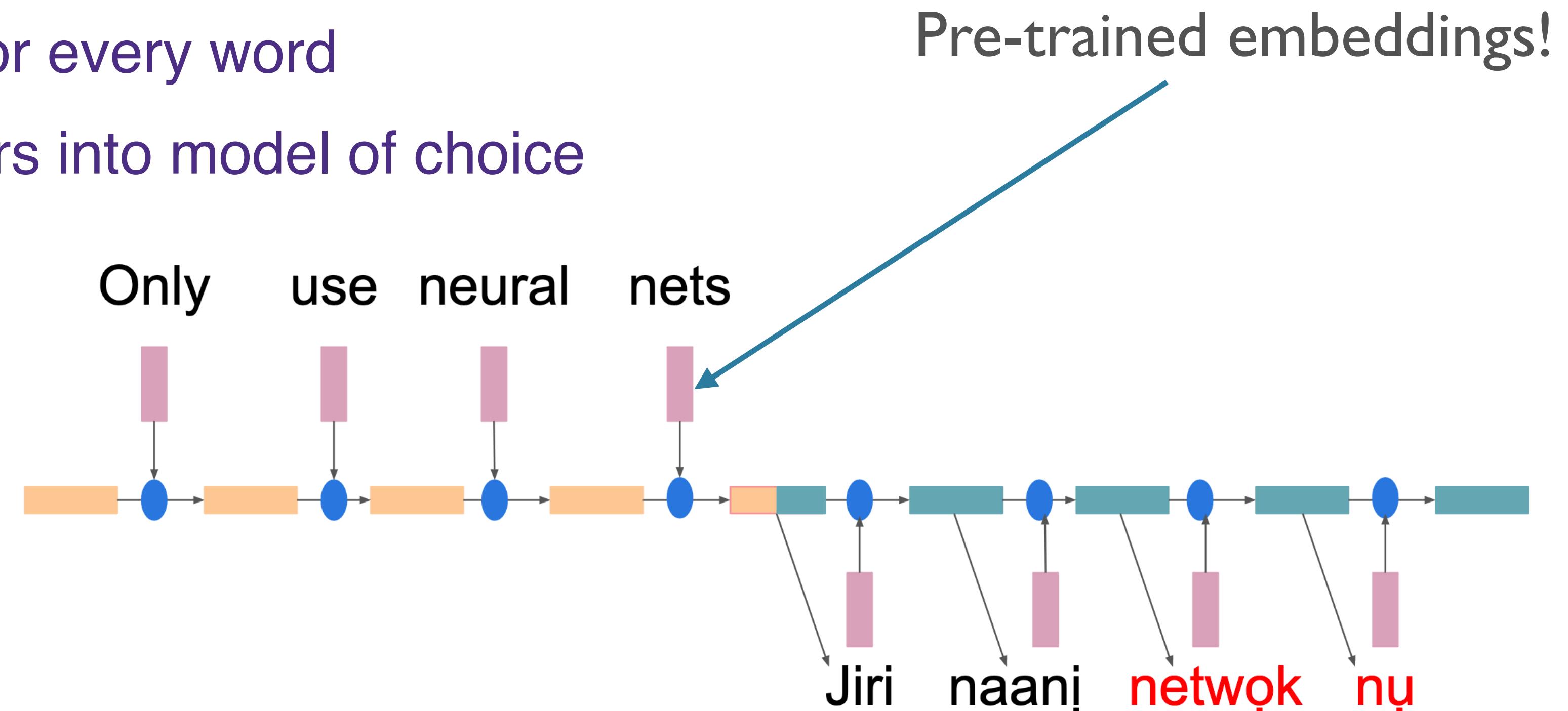


Depiction of seq2seq NMT architecture

c/o Hewitt & Kriz

# General Recipe

- Embedding layer (~300-dimensions):
  - download pre-trained embeddings
  - Use as look-up table for every word
  - Then feed those vectors into model of choice

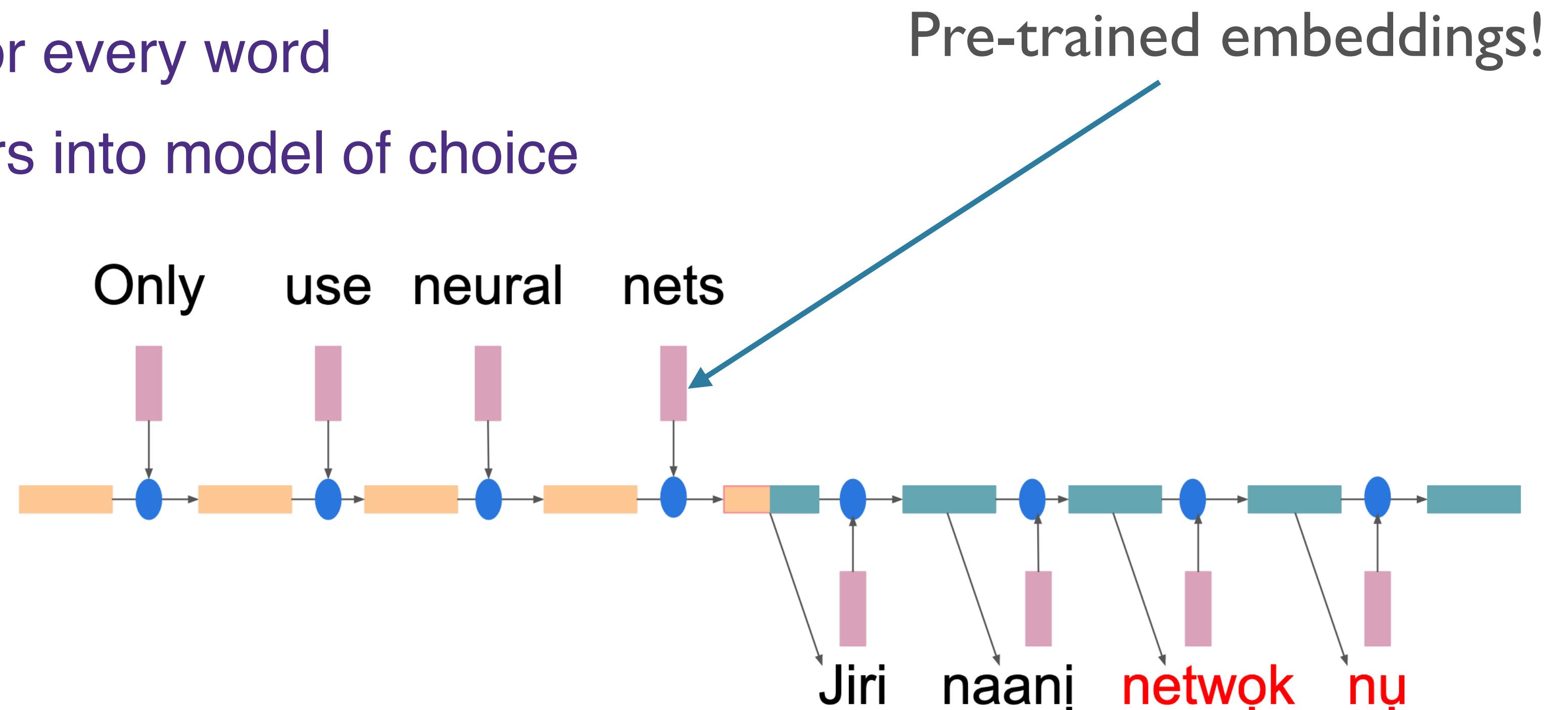


Depiction of seq2seq NMT architecture

c/o Hewitt & Kriz

# General Recipe

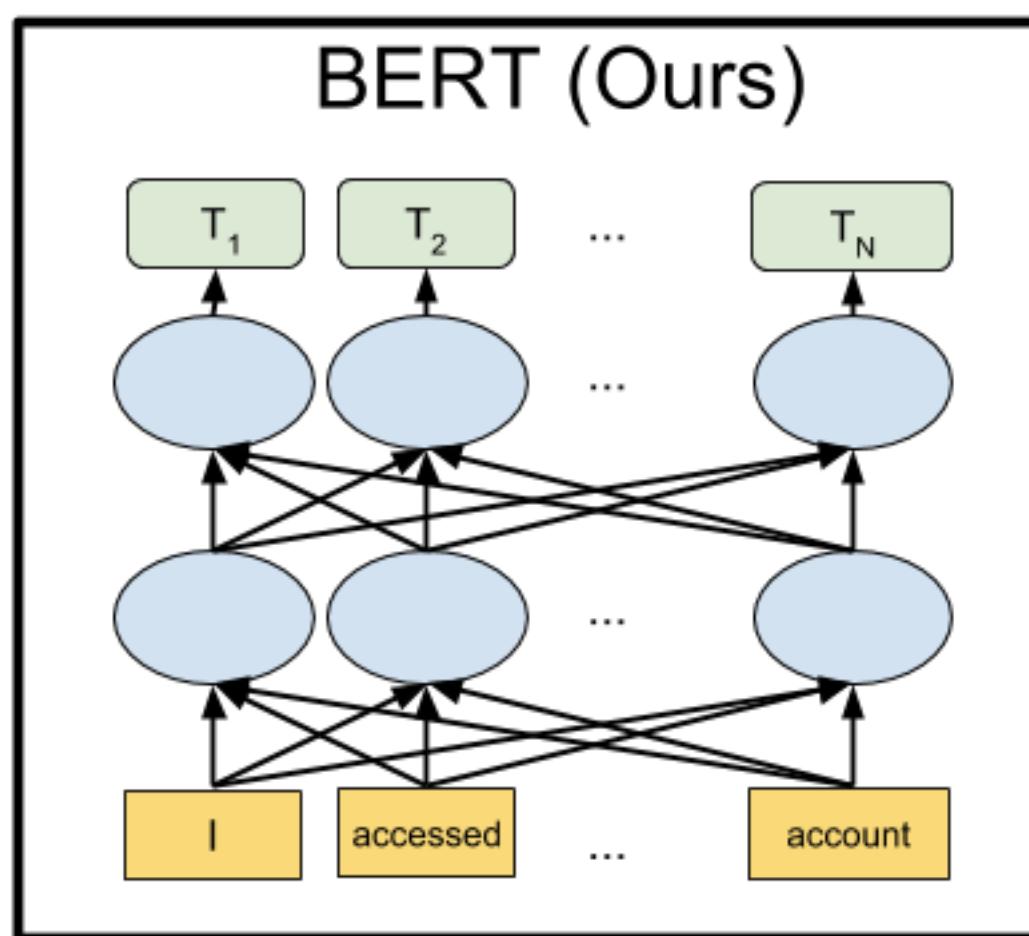
- Embedding layer (~300-dimensions):
  - download pre-trained embeddings
  - Use as look-up table for every word
  - Then feed those vectors into model of choice
- Newer embeddings:
  - [fastText](#)
  - [GloVe](#)



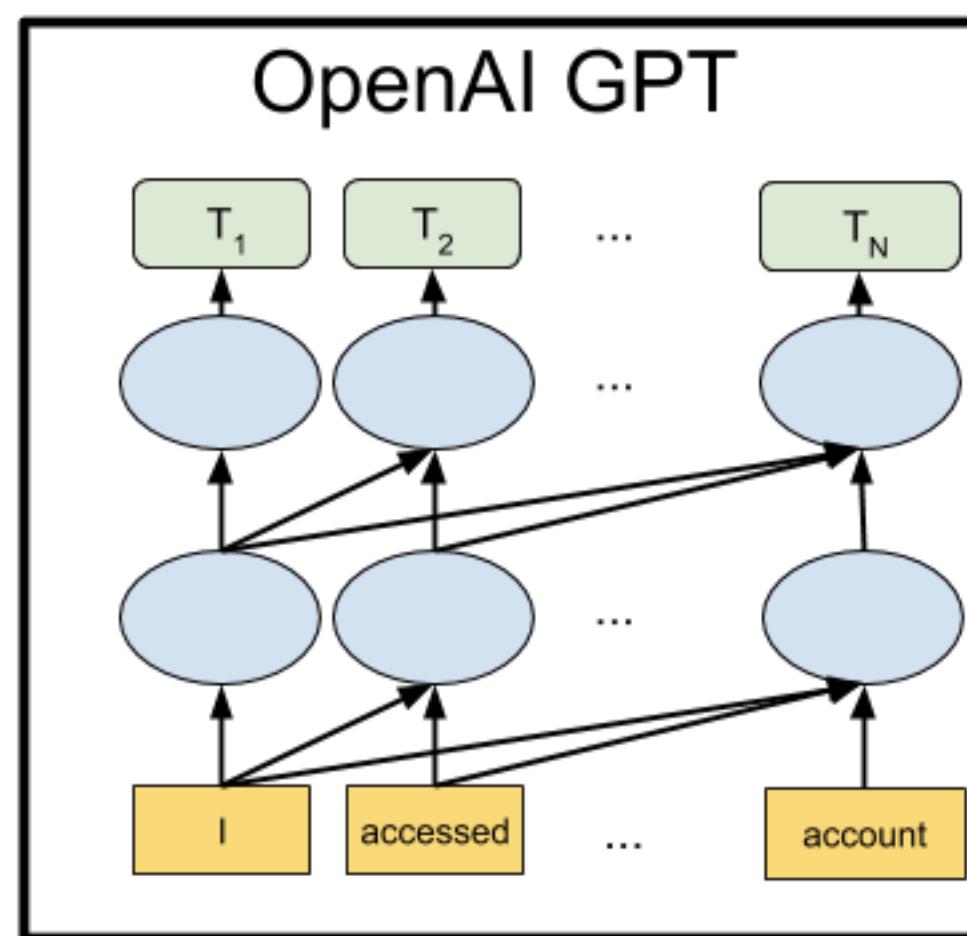
# Contextual Word Representations

- Global embeddings: single fixed word-vector look-up table
- Contextual embeddings:
  - Get a different vector for every occurrence of every word
- A recent revolution in NLP
- Here's a nice "contextual introduction"

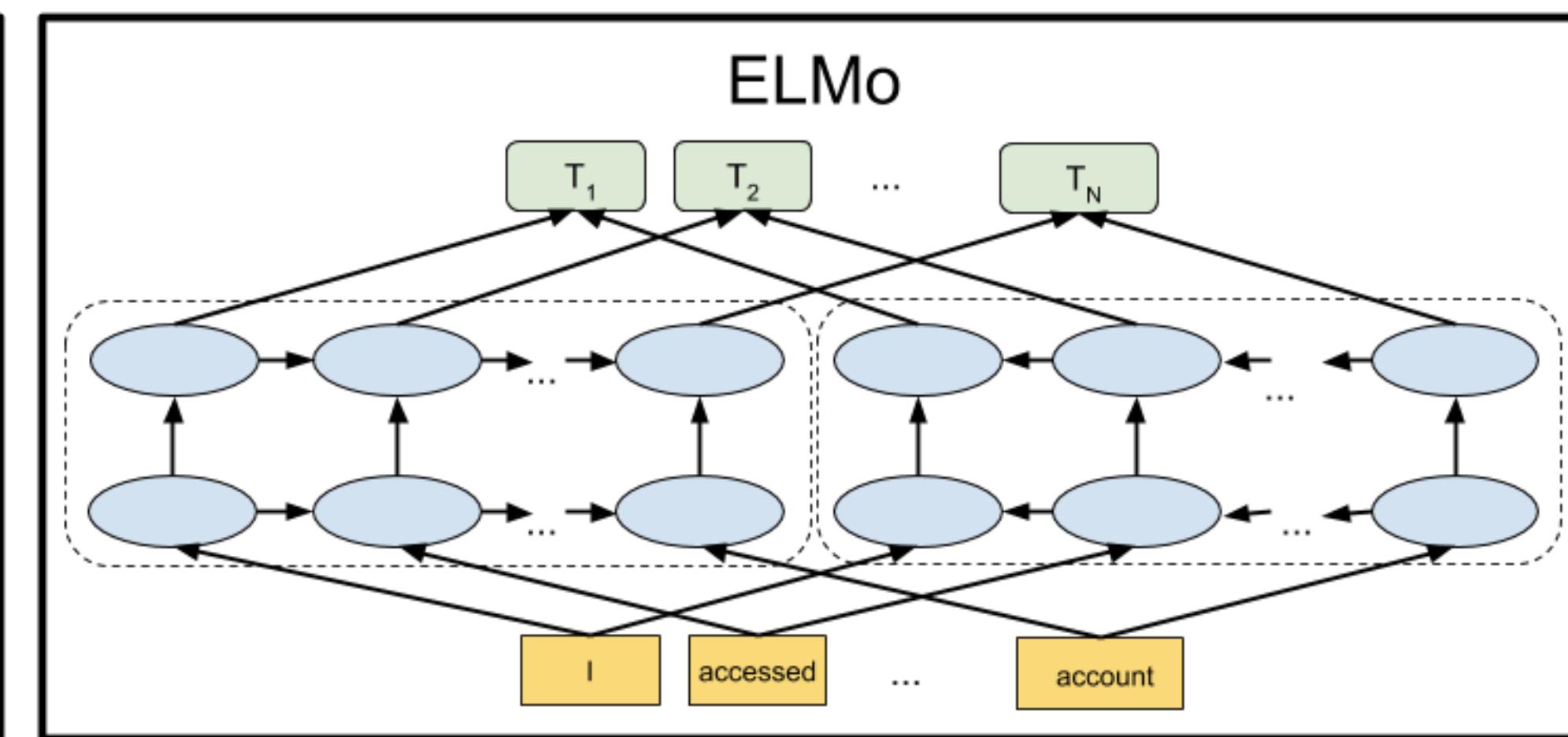
# Contextual Word Representations



Devlin et al 2018



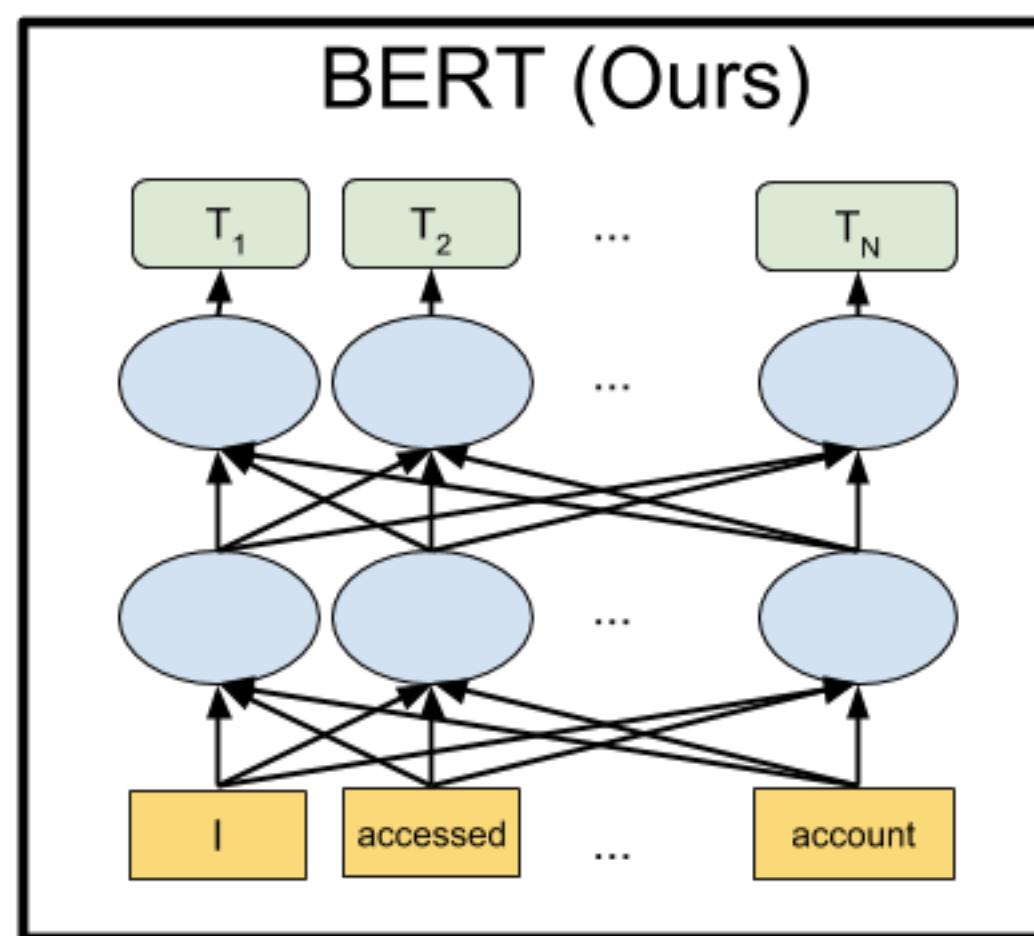
Radford et al 2019



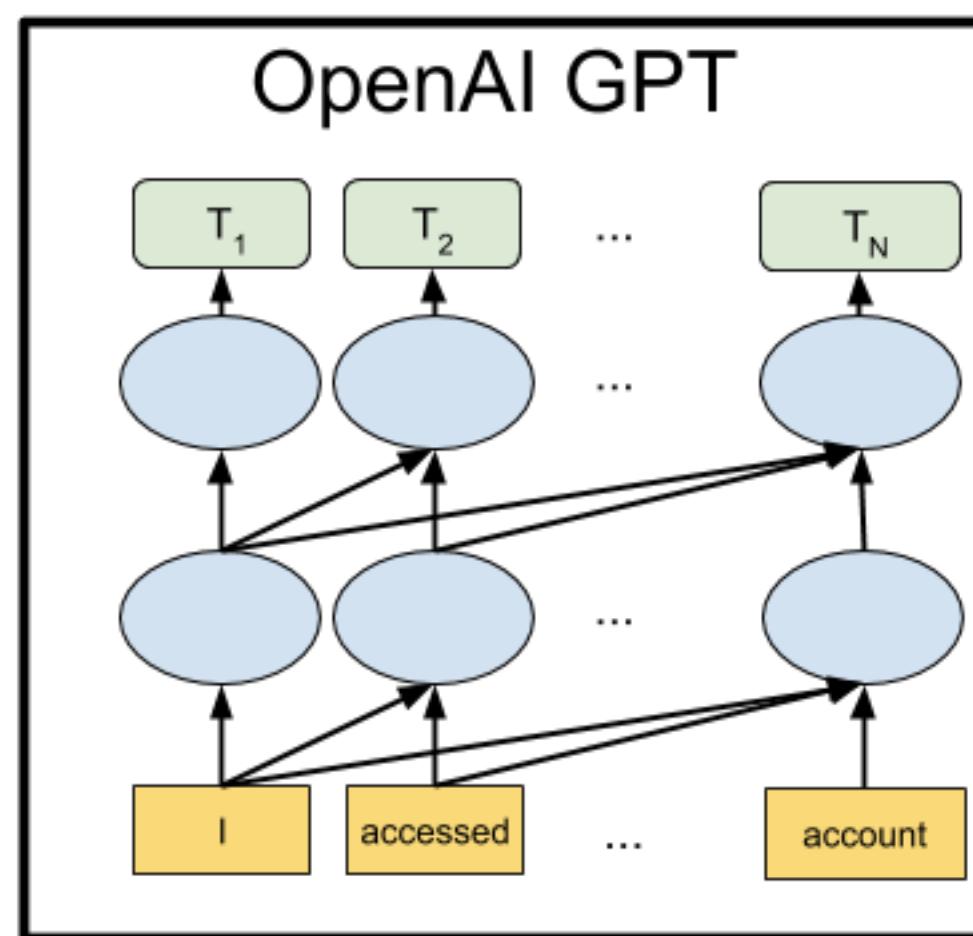
Peters et al 2018

# Contextual Word Representations

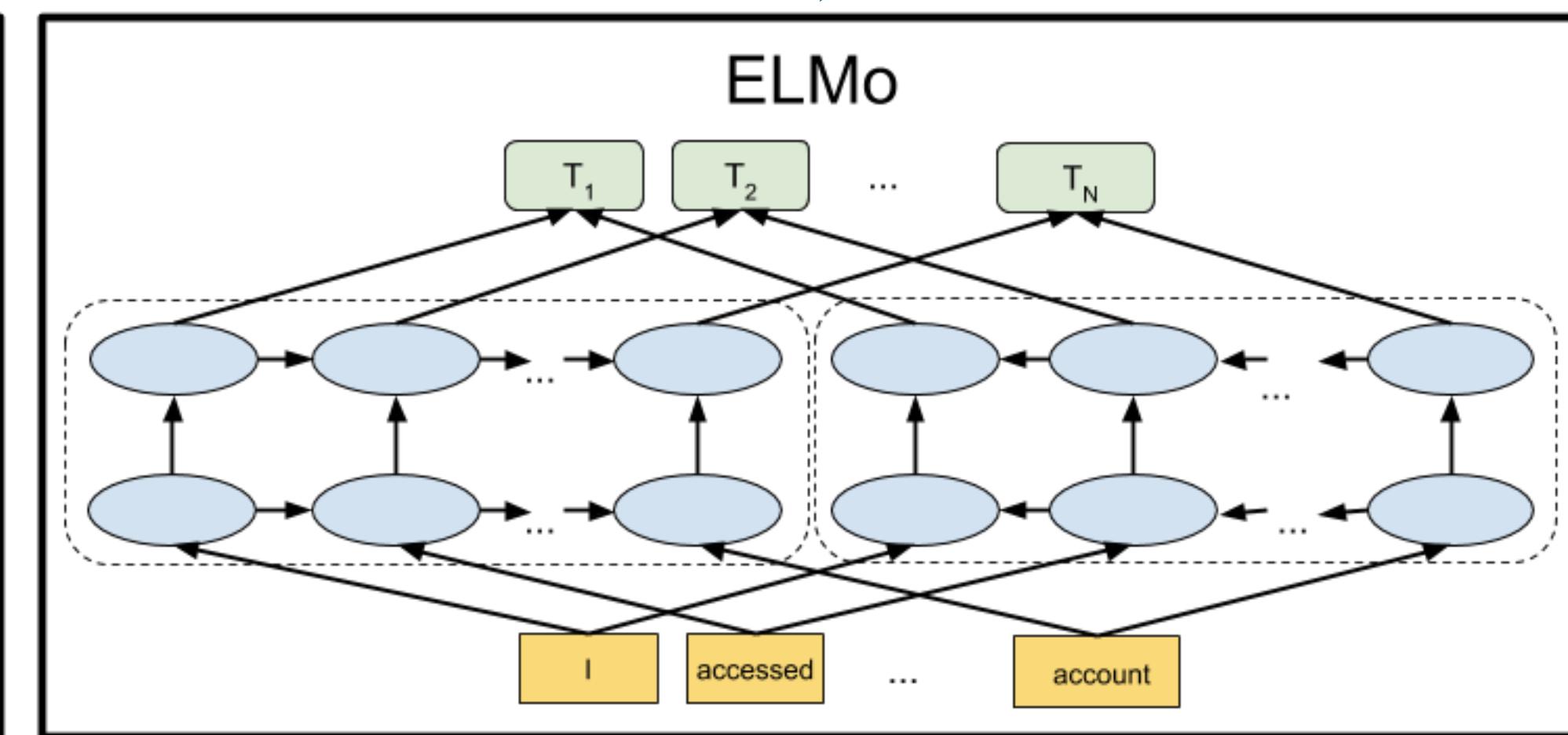
“Embeddings from Language Models”



Devlin et al 2018

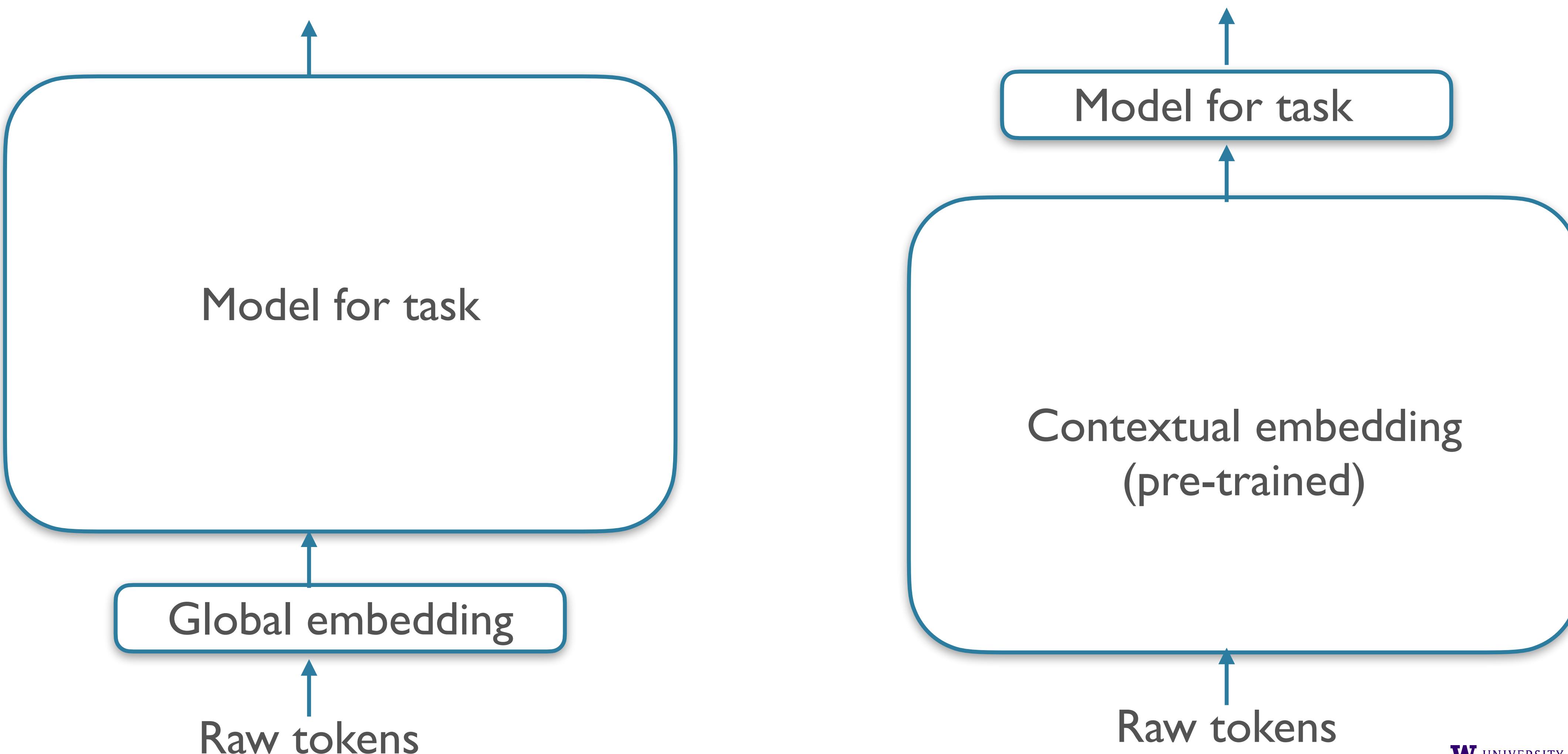


Radford et al 2019



Peters et al 2018

# Global vs Contextual Representations



# Ethical Issues Around Embeddings

- Models that learn representations from reading human-produced raw text also learn our biases

---

## Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

---

Tolga Bolukbasi<sup>1</sup>, Kai-Wei Chang<sup>2</sup>, James Zou<sup>2</sup>, Venkatesh Saligrama<sup>1,2</sup>, Adam Kalai<sup>2</sup>

<sup>1</sup>Boston University, 8 Saint Mary's Street, Boston, MA

<sup>2</sup>Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

### Abstract

The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and natural language processing tasks. We show that even word embeddings trained on Google News articles exhibit female/male gender stereotypes to a disturbing extent. This raises concerns because their widespread use, as we describe, often tends to amplify these biases. Geometrically, gender bias is first shown to be captured by a direction in the word embedding. Second, gender neutral words are shown to be linearly separable from gender definition words in the word embedding. Using these properties, we provide a methodology for modifying an embedding to remove gender stereotypes, such as the association between the words *receptionist* and *female*, while maintaining desired associations such as between the words *queen* and *female*. Using crowd-worker evaluation as well as standard benchmarks, we

Boukbasi et al 2016

# Ethical Issues Around Contextual Embeddings

- Gebru, Bender, and others' "[On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?](#) 

  - Environmental + financial costs
  - Research opportunity costs
  - Datasets so large they are impossible to audit

- Media coverage, including of Google's response (e.g. firing of Gebru and Mitchell): <https://faculty.washington.edu/ebender/stochasticparrots.html>
- More on this during the last week of class

## On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

Emily M. Bender\*  
ebender@uw.edu  
University of Washington  
Seattle, WA, USA

Angelina McMillan-Major  
aymm@uw.edu  
University of Washington  
Seattle, WA, USA

Timnit Gebru\*  
timnit@blackinai.org  
Black in AI  
Palo Alto, CA, USA

Shmargaret Shmitchell  
shmargaret.shmitchell@gmail.com  
The Aether

### ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

# Distributional Similarity for Word Sense Disambiguation

There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.

### ***Biological Example***

The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We're engineering, manufacturing and commissioning worldwide ready-to-run **plants** packed with our comprehensive know-how. Our Product Range includes pneumatic conveying systems for carbon, carbide, sand, lime and many others. We use reagent injection in molten metal for the...

### ***Industrial Example***

Label the First Use of “Plant”

# Word Representation

- 2<sup>nd</sup> Order Representation:
  - Identify words in context of  $w$
  - For each  $x$  in context of  $w$ :
    - Compute  $x$  vector representation
    - Compute centroid of these  $\vec{x}$  vector representations

# Computing Word Senses

- Compute context vector for each occurrence of word in corpus

# Computing Word Senses

- Compute context vector for each occurrence of word in corpus
- Cluster these context vectors
  - # of clusters = # of senses

# Computing Word Senses

- Compute context vector for each occurrence of word in corpus
- Cluster these context vectors
  - # of clusters = # of senses
- Cluster centroid represents word sense

# Computing Word Senses

- Compute context vector for each occurrence of word in corpus
- Cluster these context vectors
  - # of clusters = # of senses
- Cluster centroid represents word sense
- Link to specific sense?
  - Pure unsupervised: no sense tag, just  $i^{\text{th}}$  sense
  - Some supervision: hand label clusters, or tag training

# Disambiguating Instances

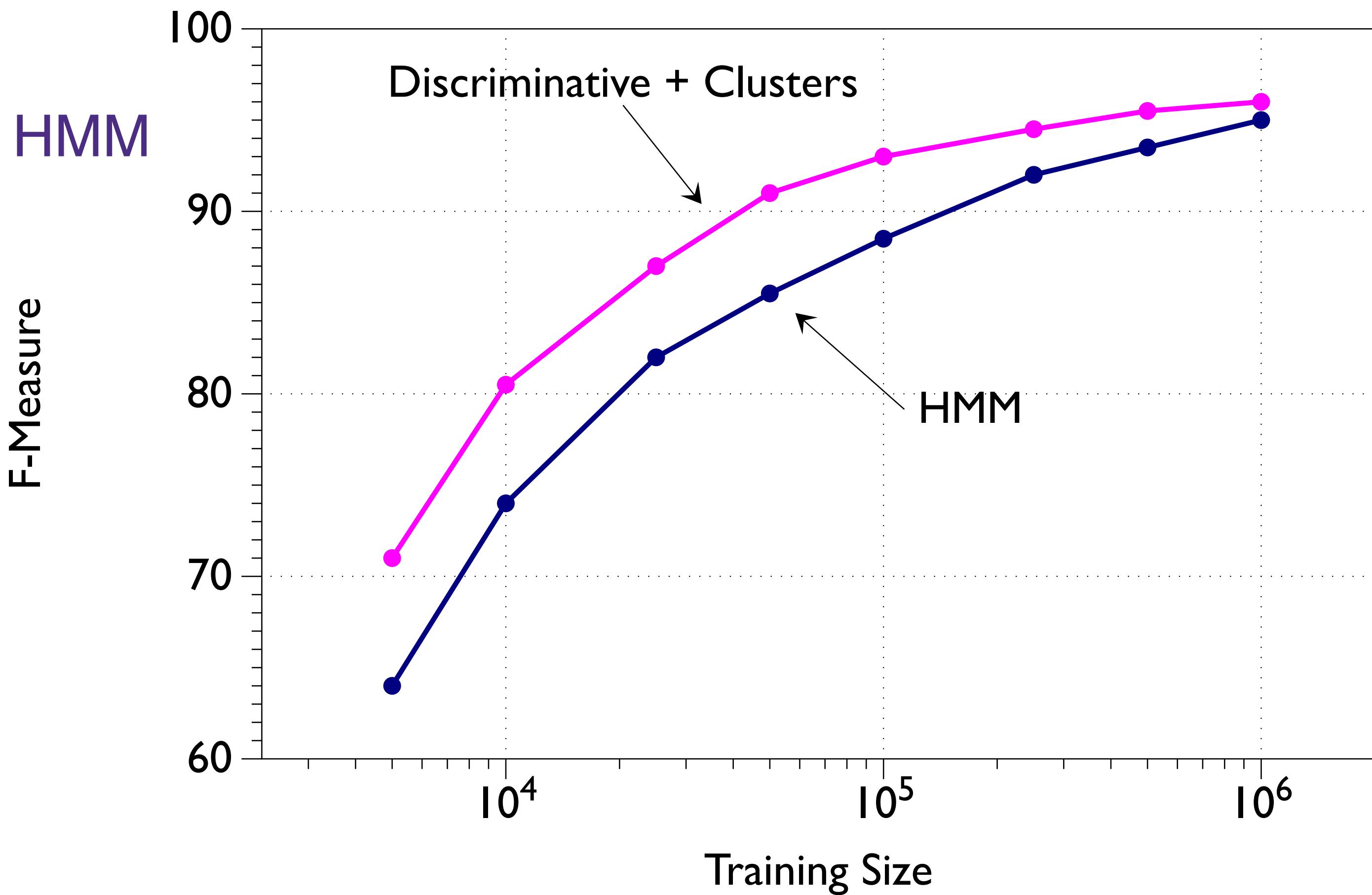
- To disambiguate an instance  $t$  of  $w$ :
  - Compute context vector for instance
  - Retrieve all senses of  $w$
  - Assign  $w$  sense with closest centroid to  $t$

# Local Context Clustering

- “Brown” (aka IBM) clustering (1992)
  - Generative model over adjacent words
  - Each  $w_i$  has class  $c_i$ 
    - $\log P(W) = \sum_i \log P(w_i|c_i) + \log P(c_i|c_{i-1})$
  - Greedy clustering
    - Start with each word in own cluster
    - Merge clusters based on log prob of text under model
      - Merge those which maximize  $P(W)$

# Clustering Impact

- Improves downstream tasks
- Named Entity Recognition vs. HMM
- Miller et al '04



# Distributional Models: Summary

- Upsurge in distributional compositional
  - Embeddings:
    - Discriminatively trained, “low”-dimensional representations
    - e.g. word2vec
      - skipgrams, etc. over large corpora
  - Composition?
    - Methods for combining word vector models
    - Capture phrasal, sentential meanings

# HW #6

# Goals

- Semantics
  - Gain better understanding of semantic representations
  - Develop experience with lambda calculus and FOL
  - Create semantic attachments
  - Understand semantic composition

# Compositional Semantics

- **Part 1:**

- *Manually* create target semantic representations
- Use Neo-Davidsonian event representation
  - e.g. verb representation with event variable, argument conjuncts
- Can use as test cases for part 2

# Compositional Semantics

- **Part 1:**

- *Manually* create target semantic representations
- Use Neo-Davidsonian event representation
  - e.g. verb representation with event variable, argument conjuncts
- Can use as test cases for part 2

- **Part 2:**

- Create semantic attachments to reproduce (NLTK)
- Add to grammatical rules to derive sentence representations

# Compositional Semantics

- **Part 1:**

- *Manually* create target semantic representations
- Use Neo-Davidsonian event representation
  - e.g. verb representation with event variable, argument conjuncts
- Can use as test cases for part 2

- **Part 2:**

- Create semantic attachments to reproduce (NLTK)
- Add to grammatical rules to derive sentence representations
- Note: Lots of ambiguities (scope, etc)
  - Only need to produce one

# Semantics in NLTK

- Grammar files:
  - .fcfg extension
  - Example format in [NLTK Book Chapter 10](#)
  - /corpora/nltk/nltk-data/grammars/book\_grammars/simple-sem.fcfg
  - Note: Not “event-style”
- Parsing:
  - Use `nltk.parse.FeatureChartParser` (or similar)

# Semantics in NLTK

- Printing semantic representations:

```
item.label()['SEM'].simplify()
    all x.(dog(x) -> exists e.(barking(e) & barker(e,x)))
```

- Also `nltk.sem.util.root_semrep(item)`

# Semantic attachments in NLTK:

## Syntax

(The programming kind)

- $a, b, e, x$

- lowercase variables can be arguments:

- $\lambda x. \text{dog}(x)$

- $P, Q, X$

- uppercase lambda variables are functors

- $\lambda P. P(\text{john})$

$\lambda$	=	\
$\exists$	=	exists
$\forall$	=	all
$\wedge$	=	&
$\vee$	=	
$\Rightarrow$	=	->
$\neg$	=	-

# More NLTK Logic Format

- Added to typical CFG rules
  - Basic approach similar to HW #5
  - Composing semantics:
    - $S [ SEM=<?np (?vp)> ] \rightarrow NP [ SEM=?np ] VP [ SEM=?vp ]$

# More NLTK Logic Format

- Added to typical CFG rules
  - Basic approach similar to HW #5
  - Composing semantics:
    - $S [ SEM=<?np (?vp)> ] \rightarrow NP [ SEM=?np ] VP [ SEM=?vp ]$
- Creating lambdas:
  - $IV[ SEM=<\lambda x. \text{exists } e. (\text{barking}(e) \ \& \ \text{barker}(e, x))> ] \rightarrow 'barks'$

# More NLTK Logic Format

- Added to typical CFG rules
  - Basic approach similar to HW #5
  - Composing semantics:
    - $S [ SEM=<?np (?vp)> ] \rightarrow NP [ SEM=?np ] VP [ SEM=?vp ]$
- Creating lambdas:
  - $IV [ SEM=<\lambda x. \exists e. (\text{barking}(e) \ \& \ \text{barker}(e, x))> ] \rightarrow 'barks'$
- Nested lambdas:
  - $\lambda x. \lambda y. \text{Etc} \rightarrow \lambda x y.$   
Can remove '.' between sequences of lambda elements  
Keep '.' between sections: lambdas, quantifiers, body

