Distributional Semantics, Pt. II

LING 571 — Deep Processing for NLP

Shane Steinert-Threlkeld

Announcements

- HW6: be detailed in readme!
- A note on ‘or’ and polymorphism (Partee and Rooth 1983)
  - They ate rice or they drank milk.
  - They ate rice or beans.
  - Walking or talking is their favorite thing.
  - …
- ‘or’_sentence: \p:<s,t> . \q:<s,t> . \w:s . p(w) = 1 or q(w) = 1
- ‘or’_IV: \v1:<e, t> . \v2:<e, t> . \x:e . or_sentence(v1(x) , v2(x))
- Generally: reduce all others systematically to boolean ‘or’, systematically

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"…

	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

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

	$<taste>$	tree
pear	1	0
apple	1	1
watermelon	1	0
paw_paw	1	0
family	0	1

Curse of Dimensionality

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

Curse of Dimensionality

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

Curse of Dimensionality

- Vector representations are sparse, very high dimensional
  - # of words in vocabulary
  - # of relations × # words, etc
- Google 1T 5-gram corpus:
  - In bigram 1M × 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
  - χ^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

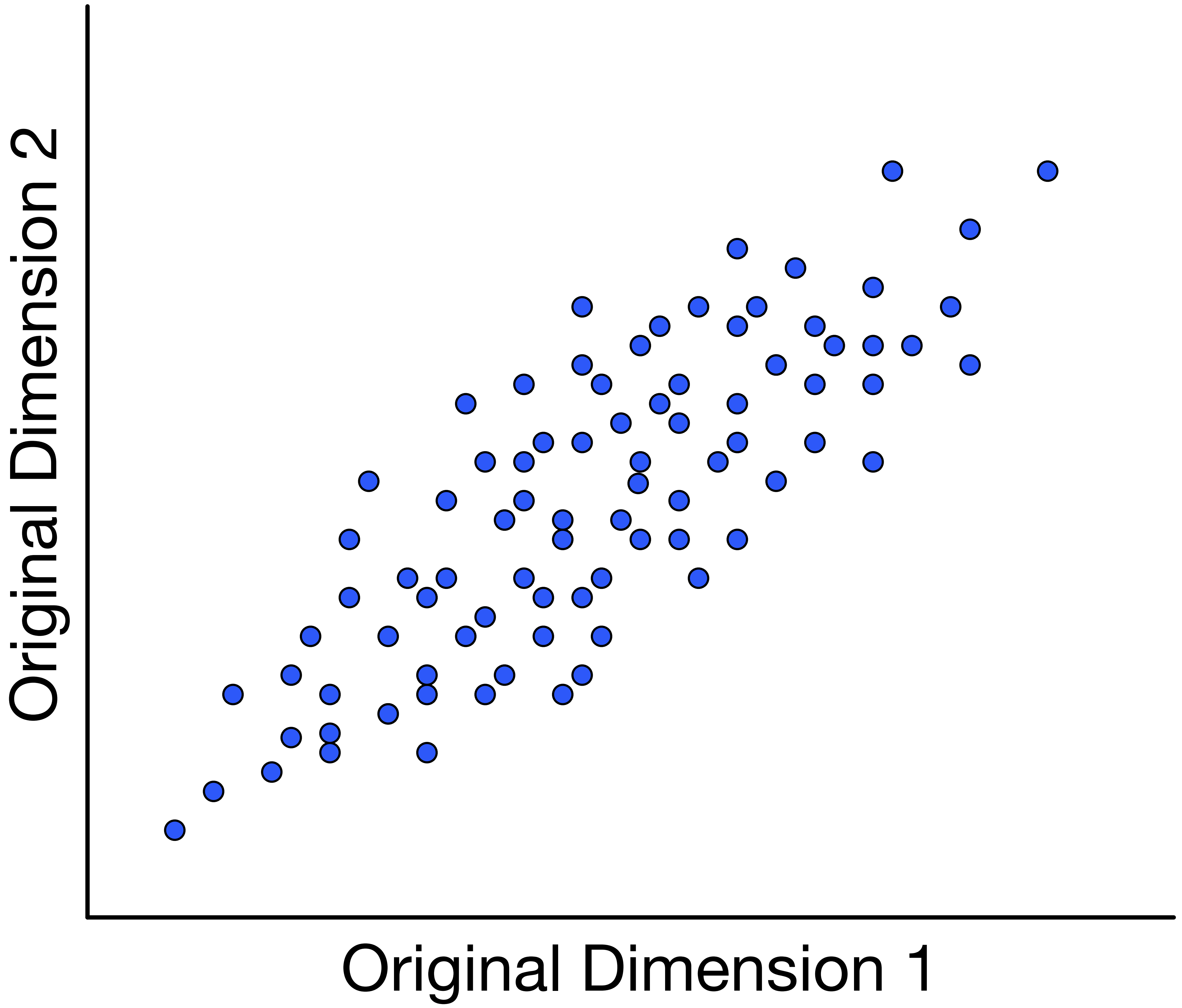
Reducing Dimensionality

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

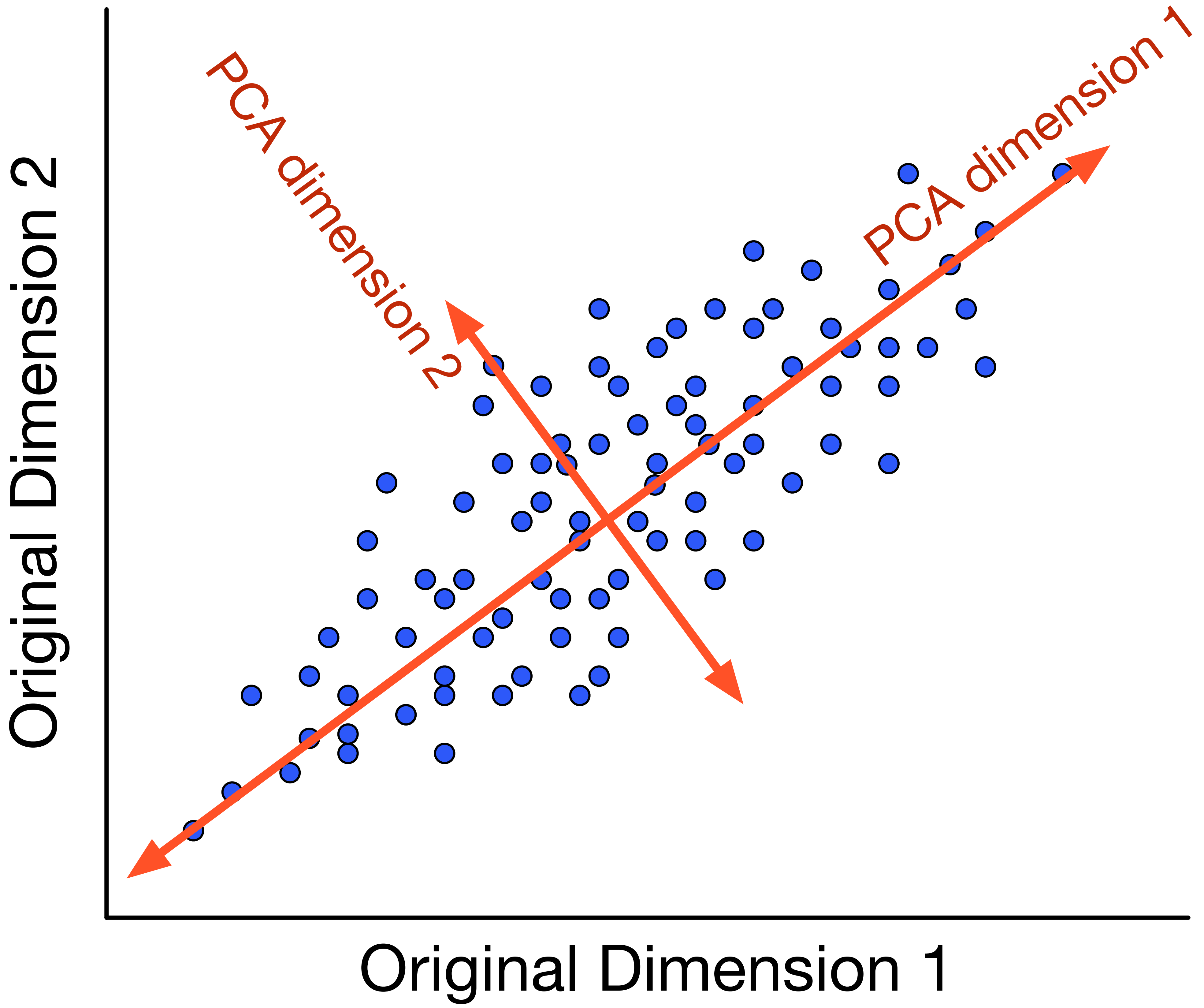
Reducing Dimensionality

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

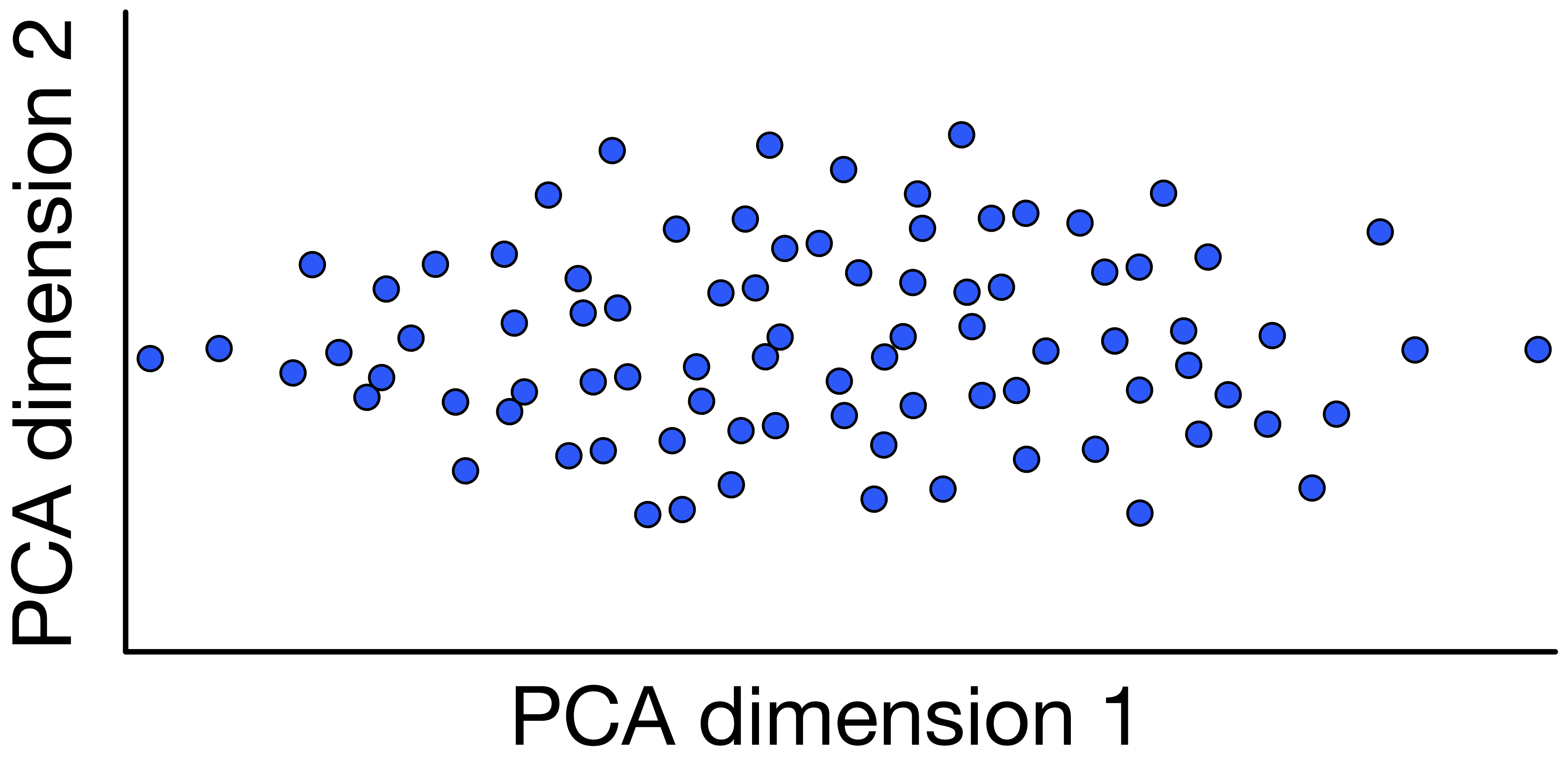
Principal Component Analysis (PCA)

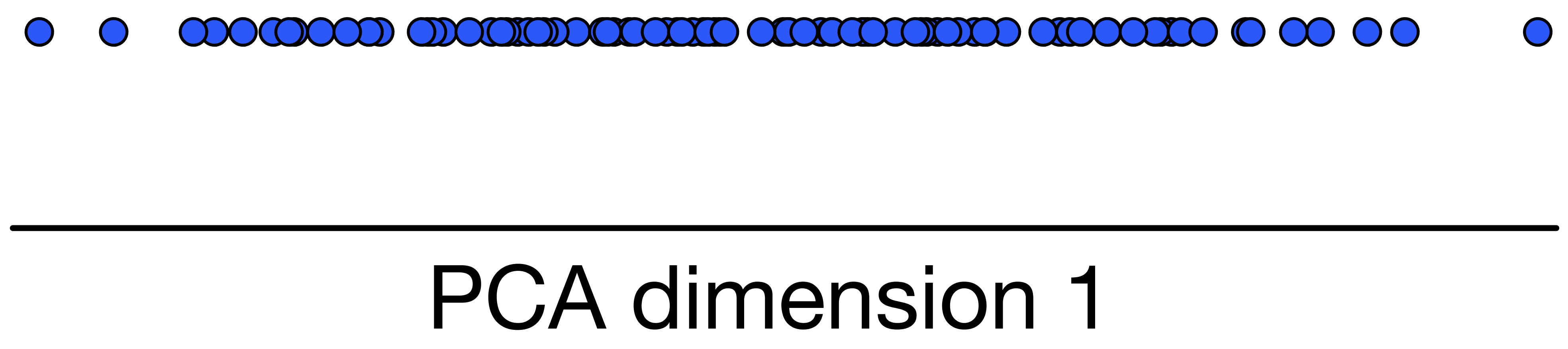


Principal Component Analysis (PCA)

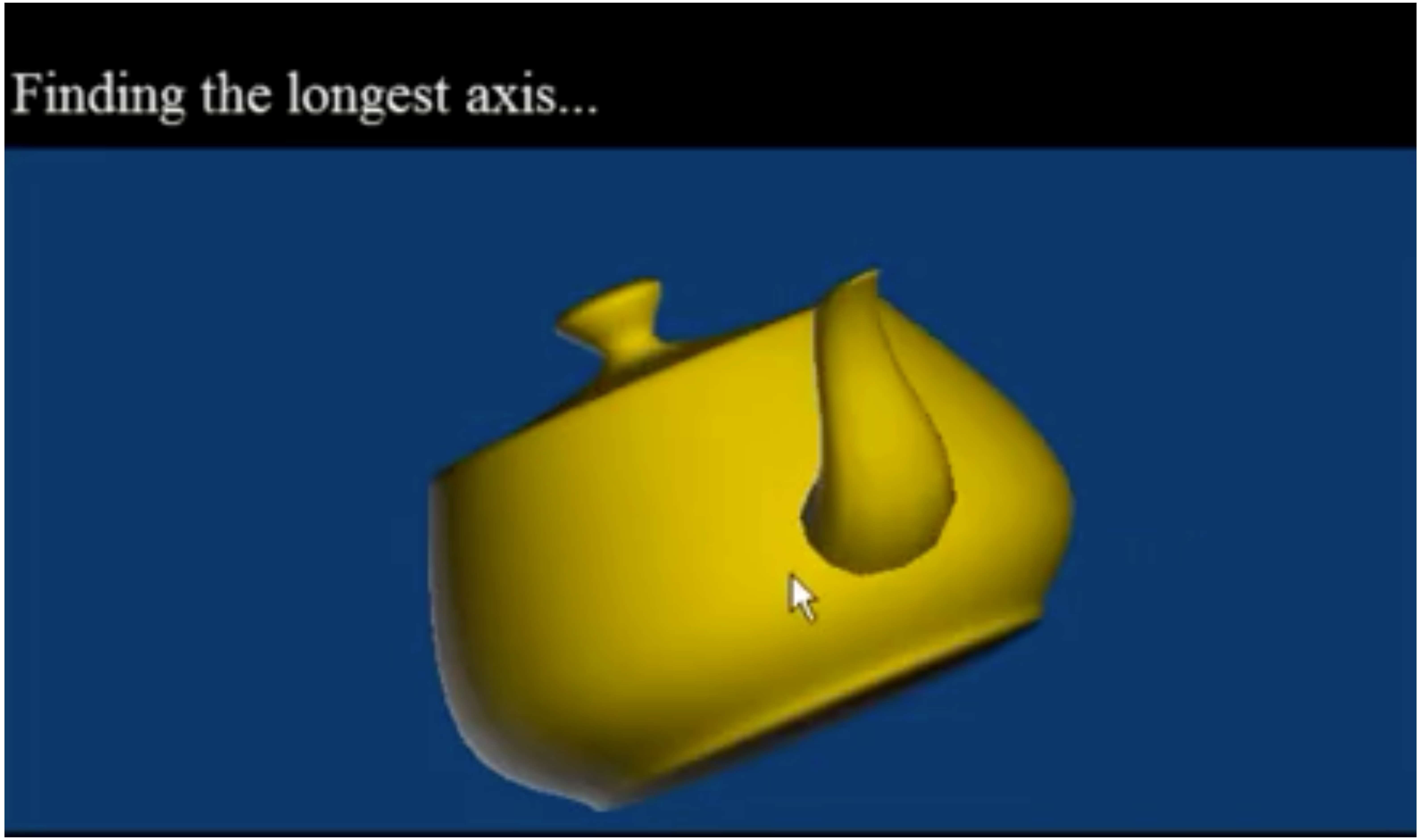


Principal Component Analysis (PCA)



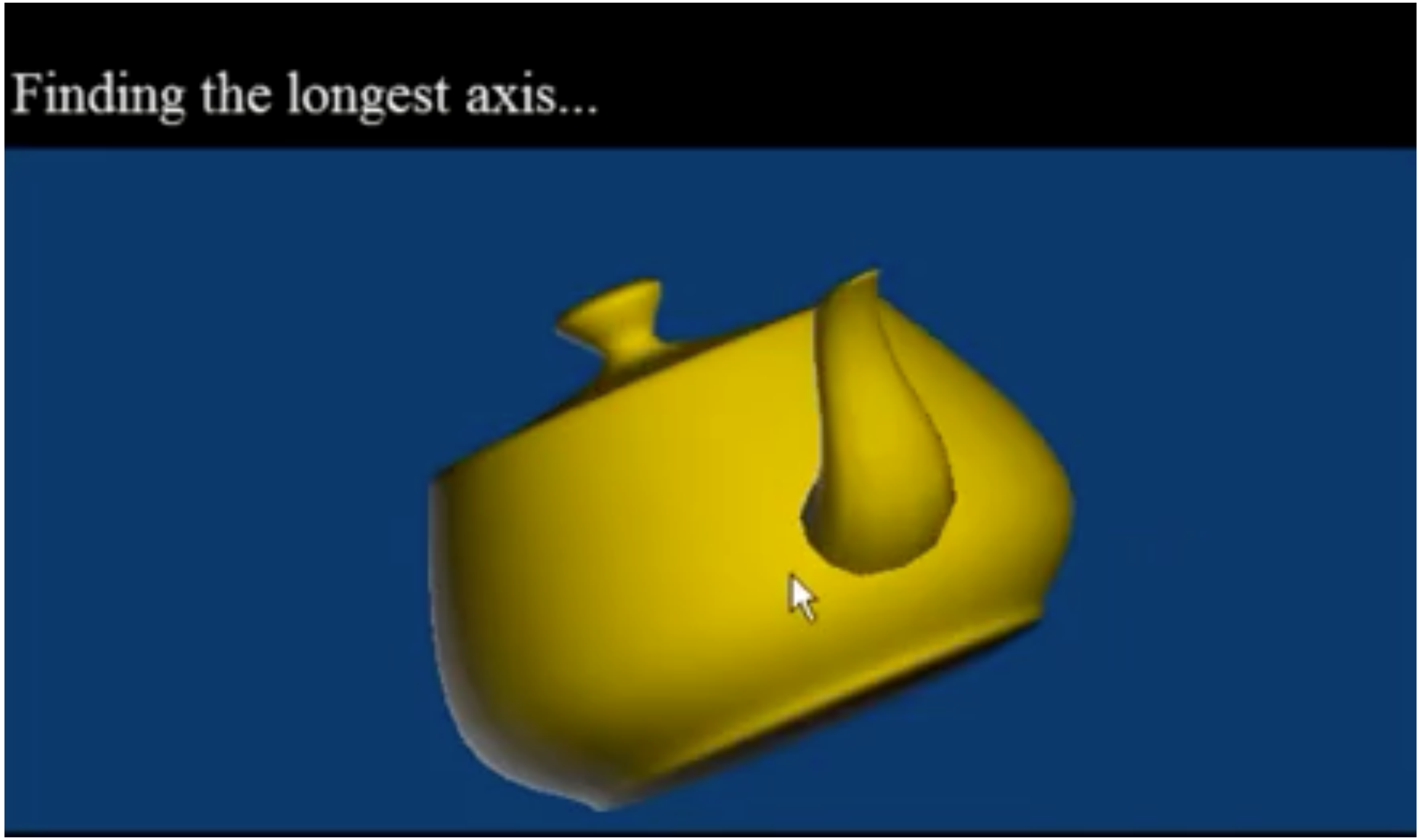


Principal Component Analysis (PCA)



via [A layman's introduction to PCA]

Principal Component Analysis (PCA)



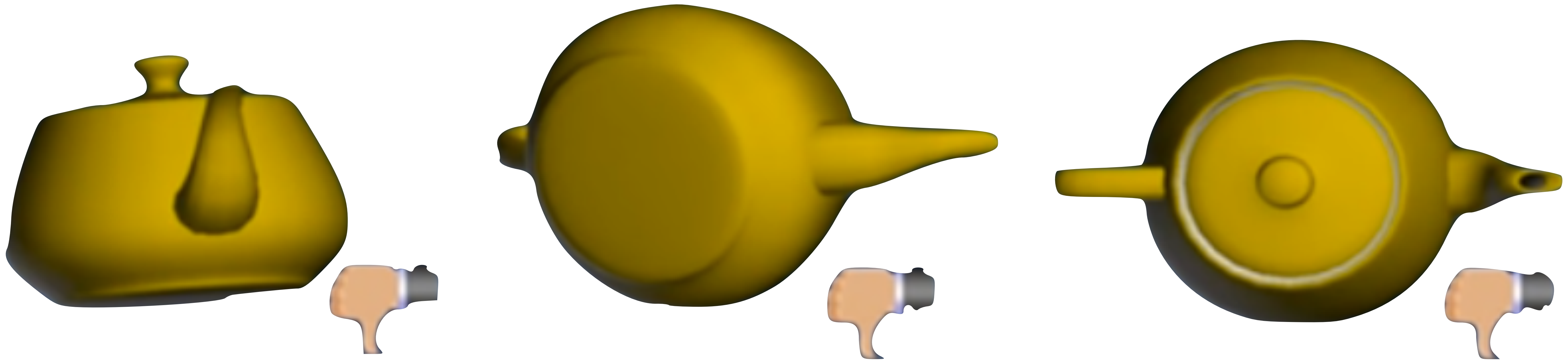
via [A layman's introduction to PCA]

Principal Component Analysis (PCA)

This →

Preserves more information than

These →



via [A layman's introduction to PCA]

PCA for Word Vectors

- Take |V| x N matrix of word-vectors
- Apply PCA to get new |V| x N matrix
  - Truncate to |V| x m matrix, for some choice of m < N
- Even with other methods discussed later, very useful for 2/3-D visualization

SVD and LSA

Singular Value Decomposition (SVD)

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

Singular Value Decomposition (SVD)

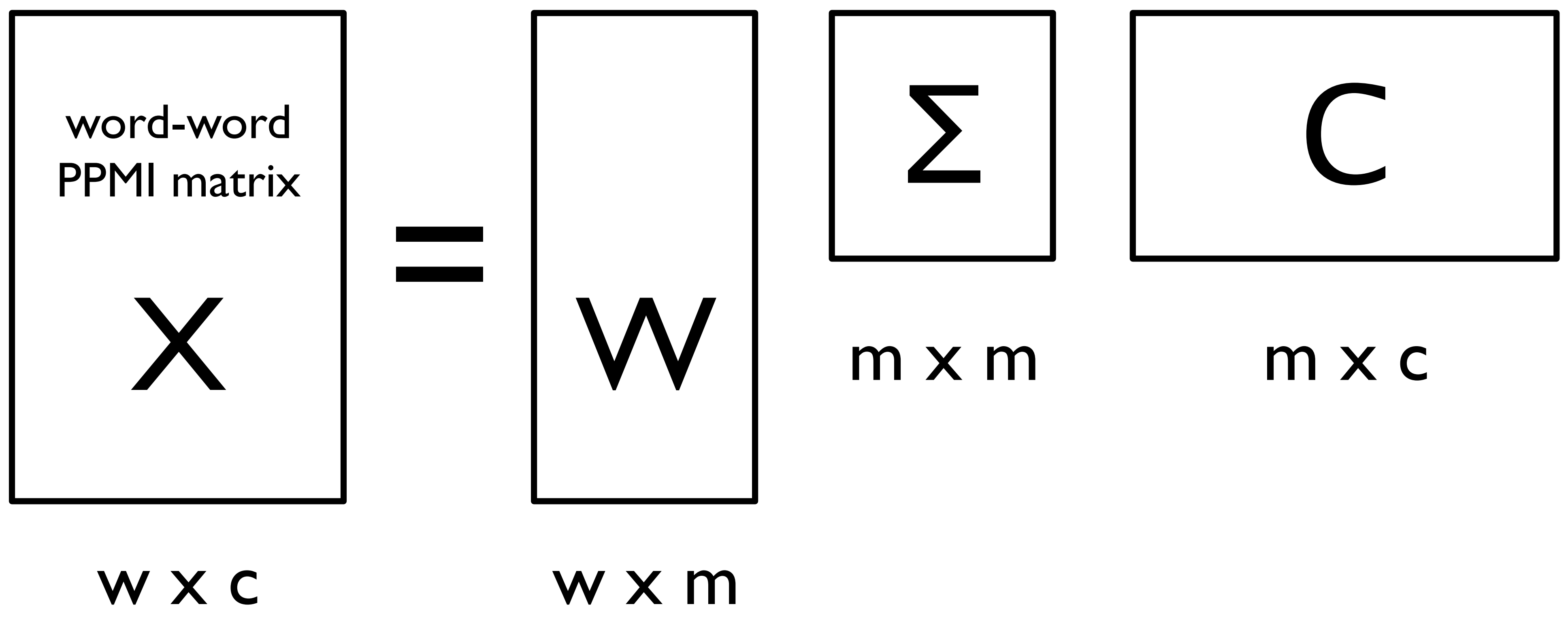
- Original matrix: high dimensional, sparse
  - Similarities missed due to word choice, etc
- Create new, projected space
  - More compact, better captures important variation
- Landauer et al (1998) argue identifies underlying “concepts”
  - Across words with related meanings

Latent Semantic Analysis (LSA)

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

Latent Semantic Analysis (LSA)

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



SVD
Animation

youtu.be/R9UoFyqJca8

Enjoy some 3D Graphics from 1976!

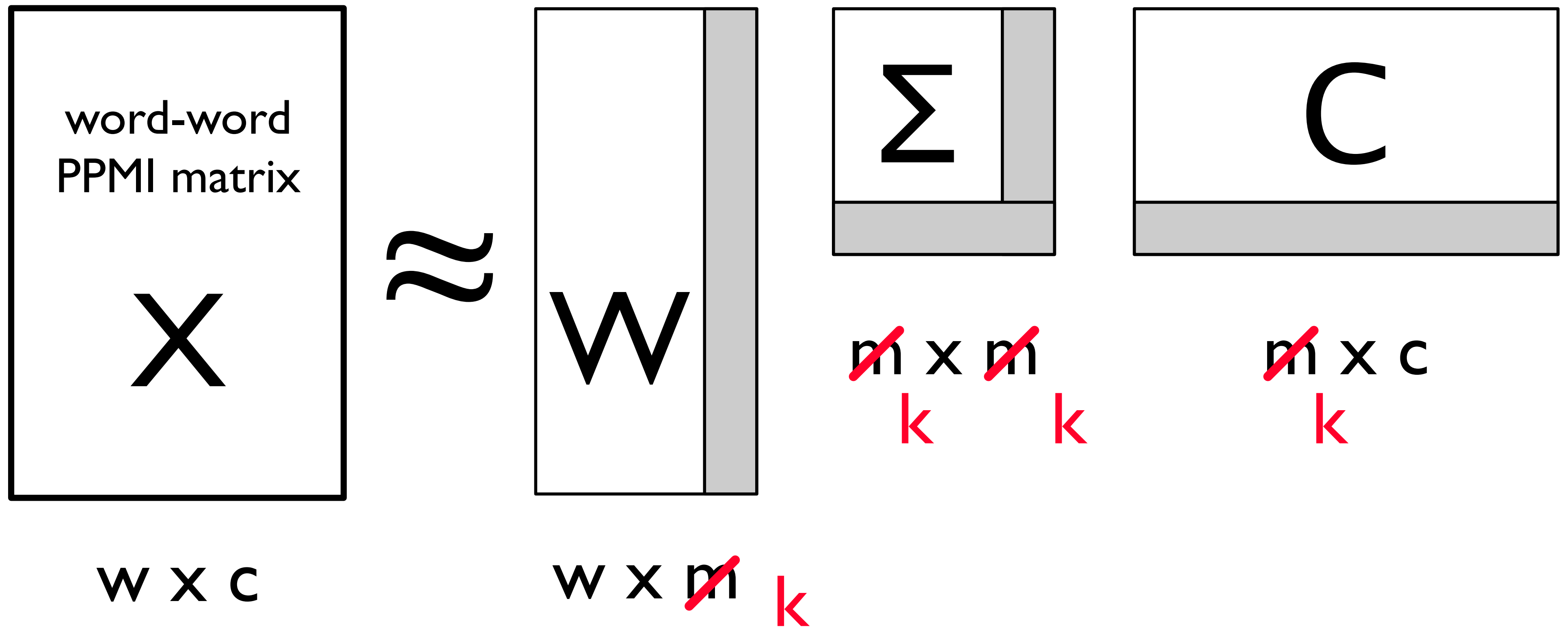
SVD
Animation

youtu.be/R9UoFyqJca8

Enjoy some 3D Graphics from 1976!

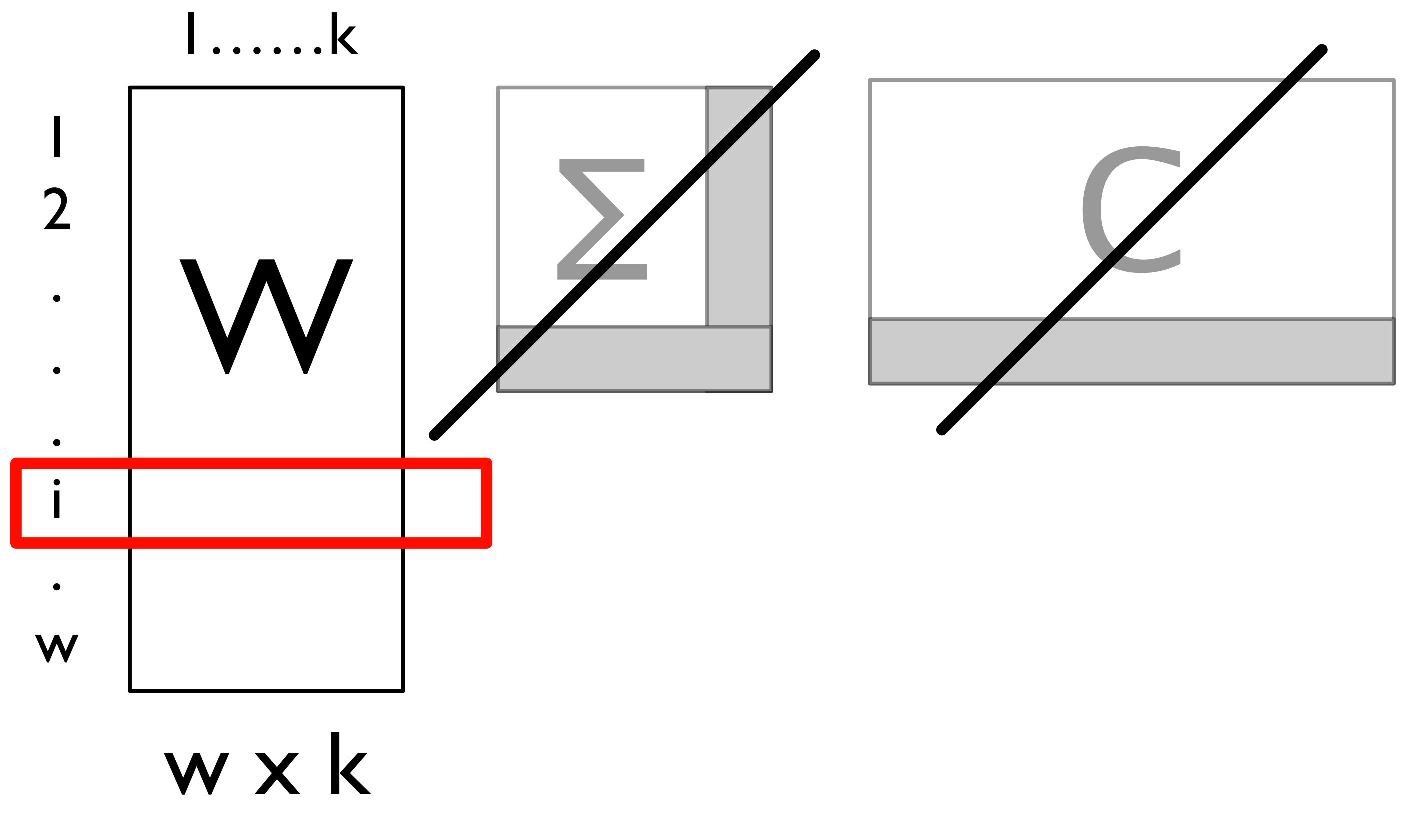
Latent Semantic Analysis (LSA)

- 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*** Σ 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)

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

Σ $(m \times m)$

	m1	m2	m3
m1	12.4		
m2		9.5	
m3			1.3

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)

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

Σ $(m{\times}m)$

	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)

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

Σ $(m \times m)$

	m1	m2	m3
m1	12.4		
m2		9.5	
m3			1.3

"Romance-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)

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

Σ $(m \times m)$

	m1	m2	m3
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:
  - $corr$(human, user) $=$ -0.38; $corr$(human, 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:
  - $corr(\text{human, user}) = 0.98; \quad corr(\text{human, minors}) = -0.83$

	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/

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 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

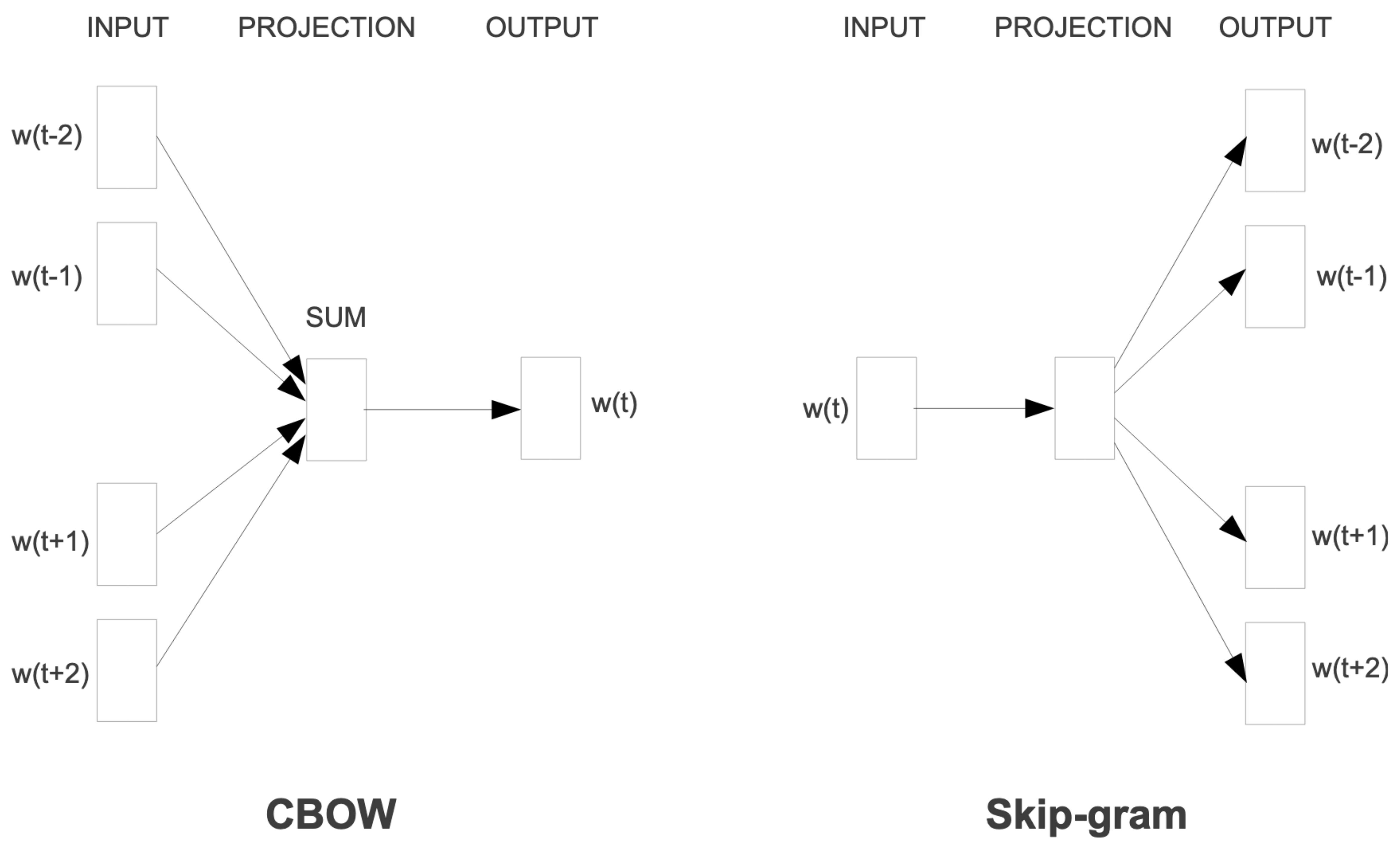
- Continuous Bag of Words (CBOW):
  - $P(\boldsymbol{word} \,|\, \boldsymbol{context})$
  - Input: $(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ wt_{+2}\ \ldots)$
  - Output: $p(\boldsymbol{w_t})$

Embeddings:
Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW):
  - $P(\boldsymbol{word} \mid \boldsymbol{context})$
  - Input: $(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ wt_{+2}\ \ldots)$
  - Output: $p(\boldsymbol{w_t})$
- Skip-gram:
  - $P(\boldsymbol{context} \mid \boldsymbol{word})$
  - Input: $\boldsymbol{w_t}$
  - Output: $p(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ w_{t+2}\ \ldots)$

Embeddings:
Skip-Gram vs. Continuous Bag of Words

- Continuous Bag of Words (CBOW):
  - $P(word \mid context)$
  - Input: $(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ wt_{+2}\ \ldots)$
  - Output: $p(w_t)$
- Skip-gram:
  - $P(context \mid word)$
  - Input: w_t
  - Output: $p(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ w_{t+2}\ \ldots)$



Mikolov et al 2013a (the OG `word2vec` paper)

Skip-Gram Model

- Learns two embeddings
  - W : word, matrix of shape [vocab_size, embedding_dimension]
  - C : context embedding, matrix of same shape

$$p(w_k \mid w_j) = \frac{e^{\mathbf{C}_k \cdot \mathbf{W}_j}}{\sum_i e^{\mathbf{C}_i \cdot \mathbf{W}_j}}$$

Skip-Gram Model

- Learns two embeddings
  - W : word, matrix of shape [vocab_size, embedding_dimension]
  - C : context embedding, matrix of same shape
- Prediction task:
  - Given a word, predict each neighbor word in window
  - Compute $p(w_k|w_j)$ as proportional to $\boldsymbol{c_k} \cdot \boldsymbol{w_j}$
    - For each context position
  - Convert to probability via softmax

$$p(w_k \mid w_j) = \frac{e^{\mathbf{C}_k \cdot \mathbf{W}_j}}{\sum_i e^{\mathbf{C}_i \cdot \mathbf{W}_j}}$$

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

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

$$p(w_k \mid w_j) = \frac{\mathbf{C}_k \cdot \mathbf{W}_j}{\sum_i \mathbf{C}_i \cdot \mathbf{W}_j}$$

Negative Sampling, Idea

- Skip-Gram:
  - $P(w_k \mid w_j)$: what is the probability that w_k occurred in the context of w_j
  - Classifier with |V| classes
- Negative sampling:
  - $P(+ \mid w_k, w_j)$: what is the probability that (w_k, w_j) was a true co-occurrence?
  - $P(- \mid w_k, w_j) = 1 - P(+ \mid w_k, w_j)$
    - Probability that (w_k, w_j) was *not* a true co-occurrence
    - Examples of “fake” co-occurrences = *negative samples*
  - Binary classifier

Generating Positive Examples

```
... lemon,  a [tablespoon of apricot jam,     a] pinch ...
               c1         c2     w      c3      c4
```

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Generating Positive Examples

- Iterate through the corpus. For each word: add all words within a *window_size* of the current word as a positive pair.

```
... lemon,  a [tablespoon of apricot jam,     a] pinch ...
               c1          c2    w       c3      c4
```

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Generating Positive Examples

- Iterate through the corpus. For each word: add all words within a *window_size* of the current word as a positive pair.
  - NB: *window_size* is a hyper-parameter

```
... lemon,  a [tablespoon of apricot jam,     a] pinch ...
               c1          c2     w     c3        c4
```

positive examples +	
w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Negative Samples

- For each positive (w, c) sample, generate *num_negatives* samples
  - (w, c’), where c’ is different from c
  - NB: *num_negatives* is a hyper-parameter

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Negative Samples, up-weighting

- It's also common to "upsample" less frequent words
- Instead of sampling from raw frequencies from the corpus, raise them to a power to "flatten" the distribution

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

The Data, Summary

- X = pairs of words
- Y = {0, 1}
  - 1 = + (positive example), 0 = - (negative example)
- Example (x, y) pairs:
  - ((“apricot”, “tablespoon”), 1)
  - ((“apricot”, “jam”), 1)
  - ((“apricot”, “aardvark”), 0)
  - ((“apricot”, “my”), 0)

The Model

- So what is $P(1 \mid w, c)$ (more specifically, $P(1 \mid w, c; \theta)$)?
- As before, learns two embeddings
  - W : word, matrix of shape [vocab_size, embedding_dimension]
    - W_w: embedding for word *w* [row of the matrix]
  - C : context embedding, matrix of same shape

The Model

$$P(1 \mid w, c) = \sigma\left(W_w \cdot C_c\right)$$

The Model

$$P(1 \mid w, c) = \sigma\left(W_w \cdot C_c\right)$$

Target word
embedding

The Model

$$P(1 \mid w, c) = \sigma\left(W_w \cdot C_c\right)$$

Target word embedding

Context word embedding

The Model

$$P(1 \mid w, c) = \sigma\left(W_w \cdot C_c\right)$$

Target word embedding

Similarity (dot-product)

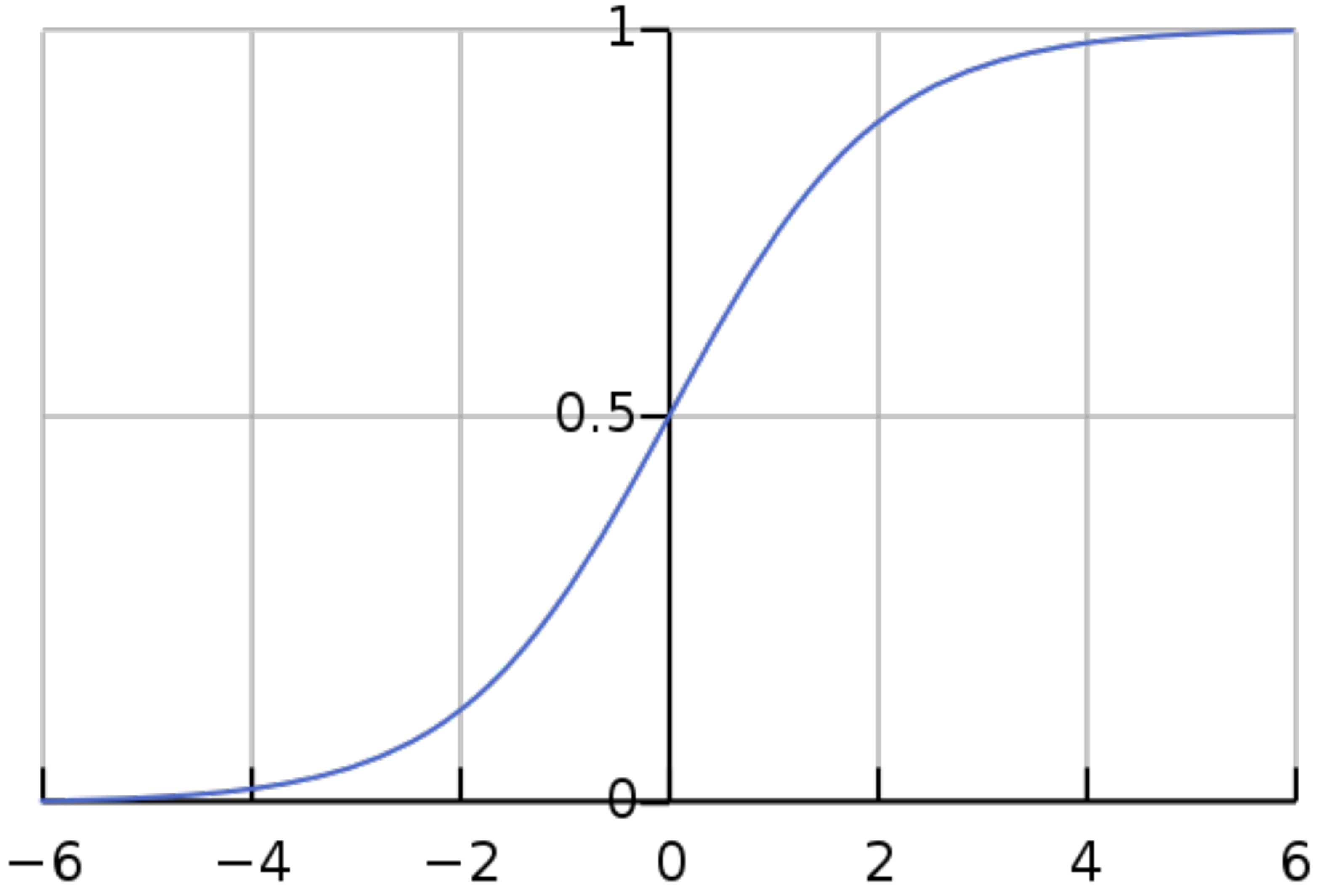
Context word embedding

The Model

$$P(1 \mid w, c) = \sigma\left(W_w \cdot C_c\right)$$

sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Target word embedding

Similarity (dot-product)

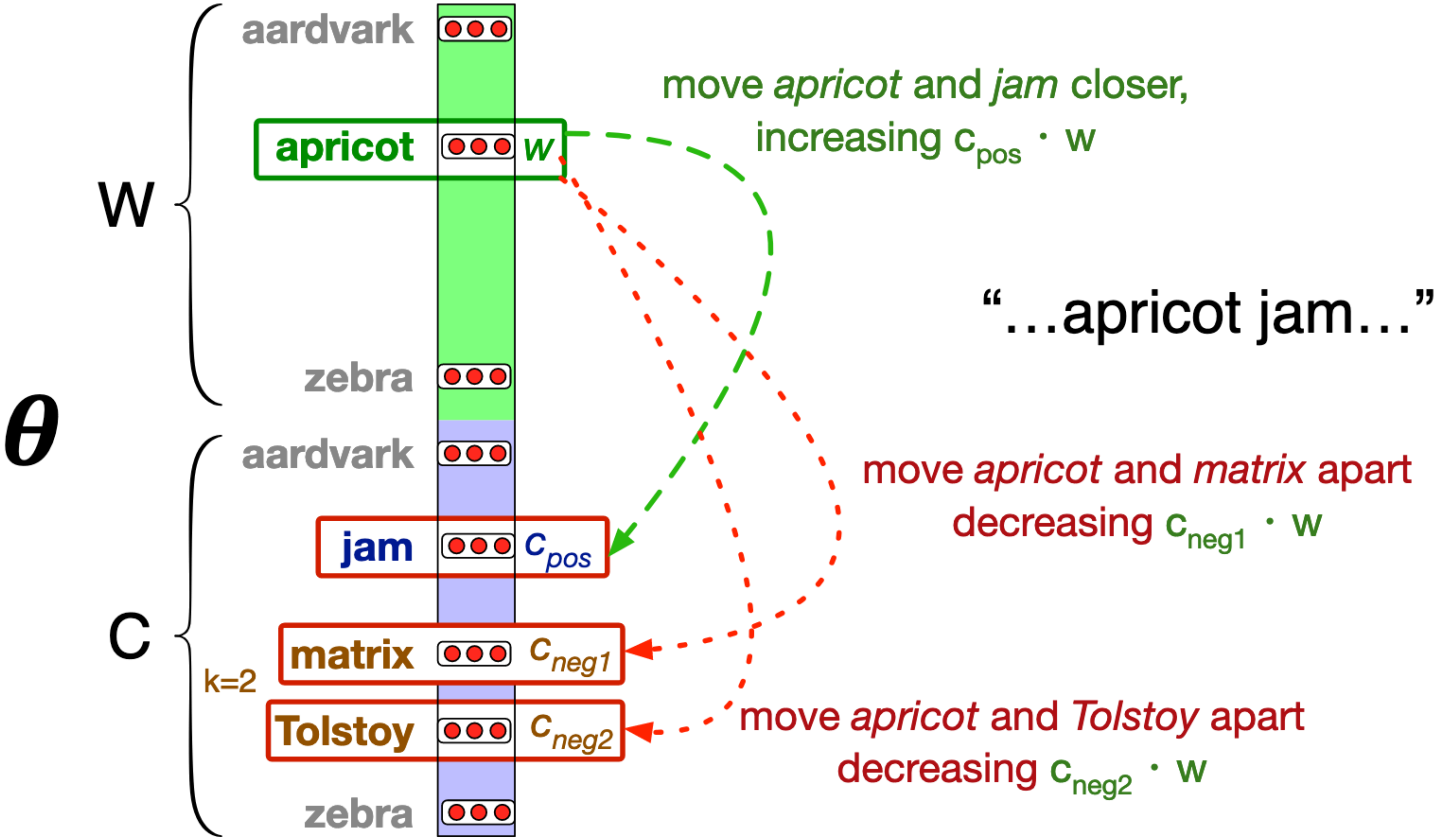
Context word embedding

The Model

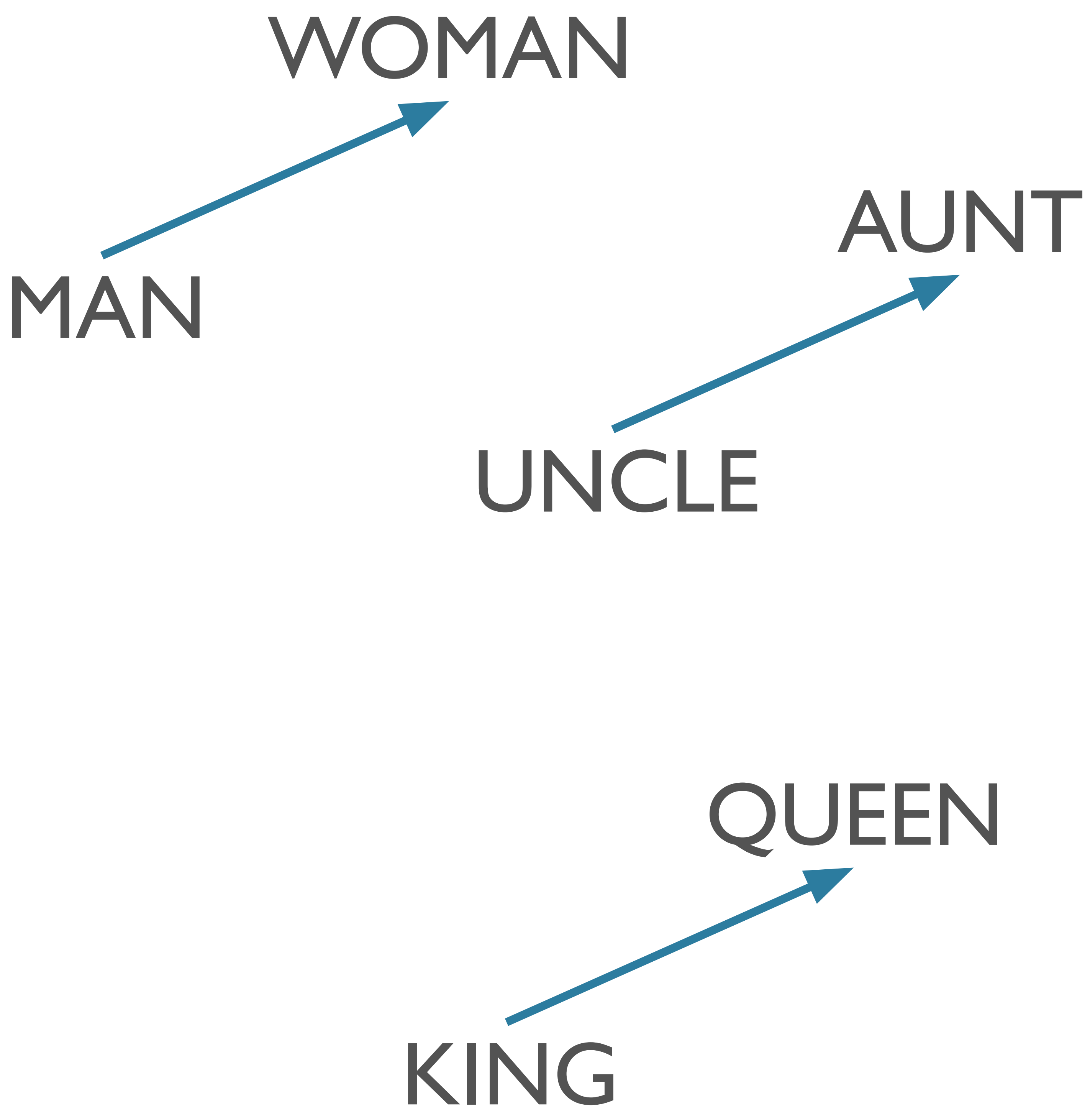
$$P(1 \mid w, c) = \sigma\left(W_w \cdot C_c\right)$$

- Target and context words that are *more similar* to each other (have more similar embeddings) have a *higher probability* of being a positive example.

Learning: Intuitively



Relationships via Offsets



Mikolov et al 2013b

Relationships via Offsets

Mikolov et al 2013b

One More Example

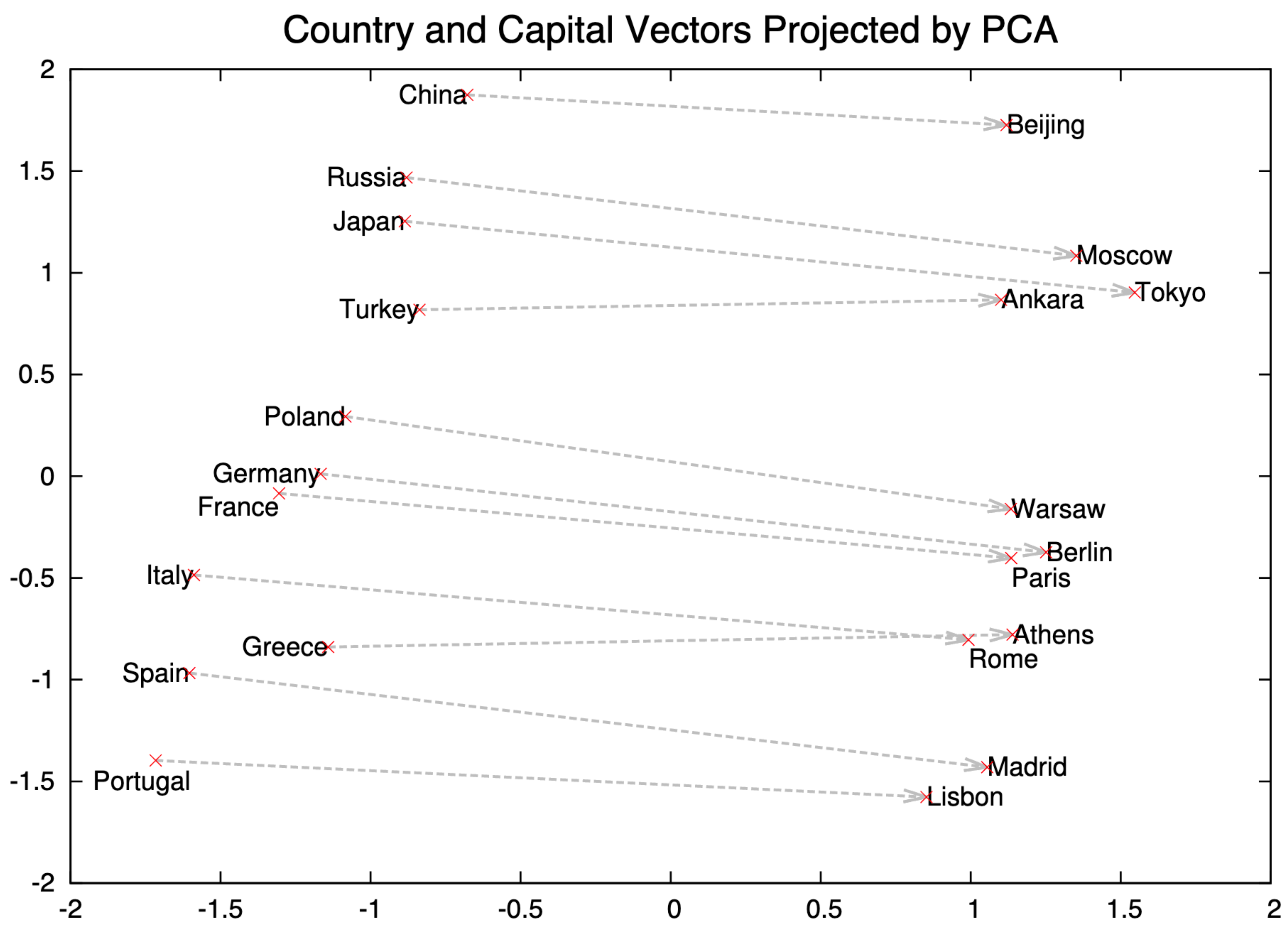
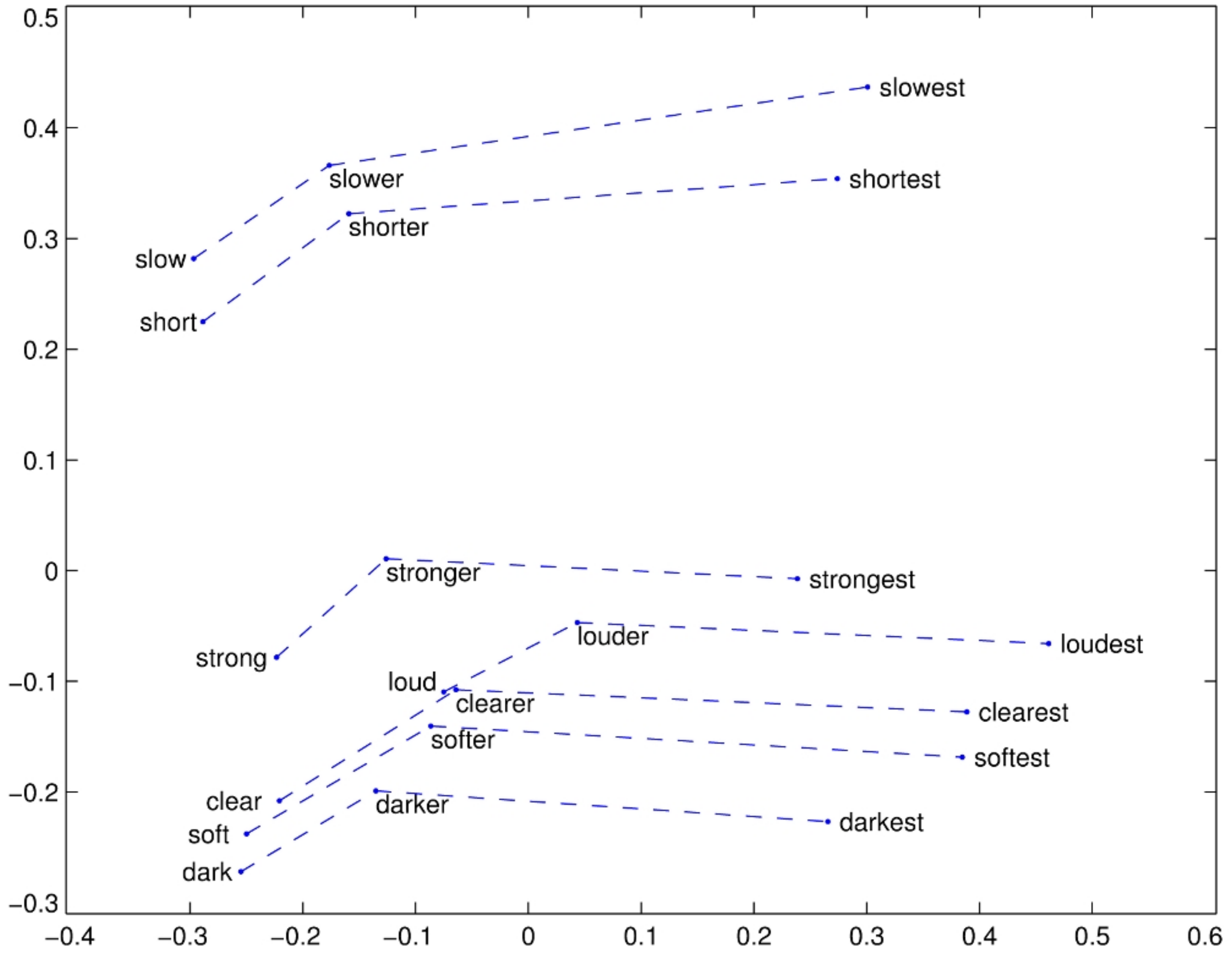


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised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

[Mikolov et al 2013c]()

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.

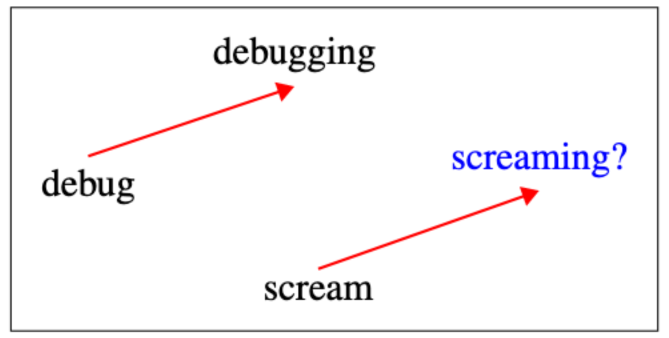


Figure 1: Using the vector offset method to solve the analogy task (Mikolov et al., 2013c).

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

$$a : a^* :: b : \_\_\_ \qquad (1)$$

Linzen 2016, a.o.

Power of Prediction-based Embeddings

- Count-based embeddings:
  - Very high-dimensional (|V|)
  - Sparse
  - Pro: features are interpretable [“occurred with word W N times in corpus”]
- Prediction-based embeddings:
  - “Low”-dimensional (typically ~300-1200)
  - Dense
  - Con: features are not immediately interpretable
    - i.e. what does “dimension 36 has value -9.63” mean?

Diverse Applications

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

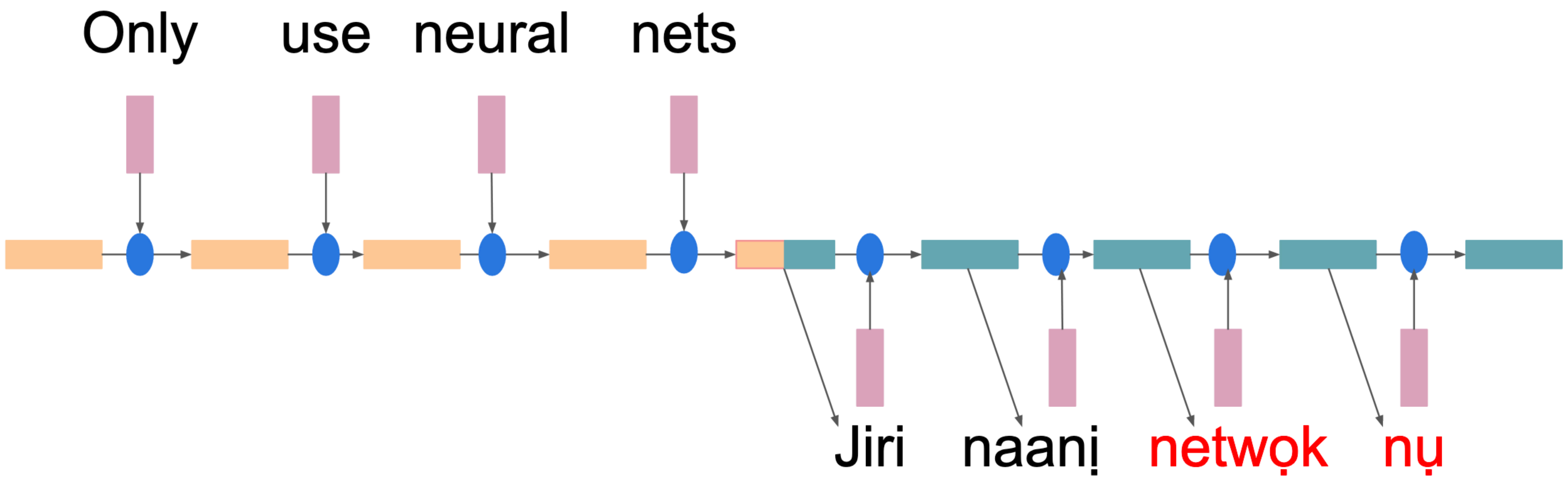
General Recipe

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

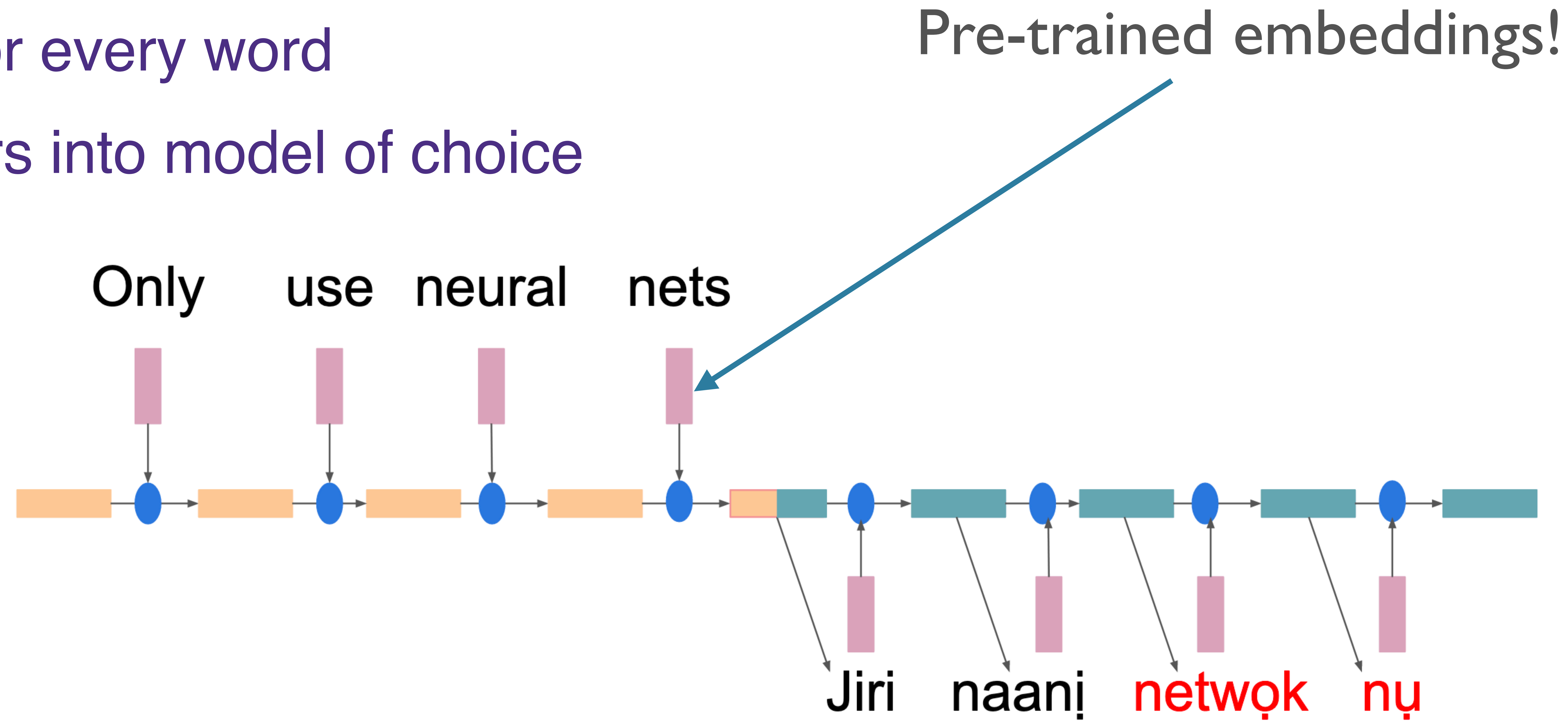
- Embedding layer (~300-dimensions):
  - download pre-trained embeddings
  - Use as look-up table for every word
  - Then feed those vectors into model of choice



Depiction of seq2seq NMT architecture
c/o [Hewitt & Kriz]

General Recipe

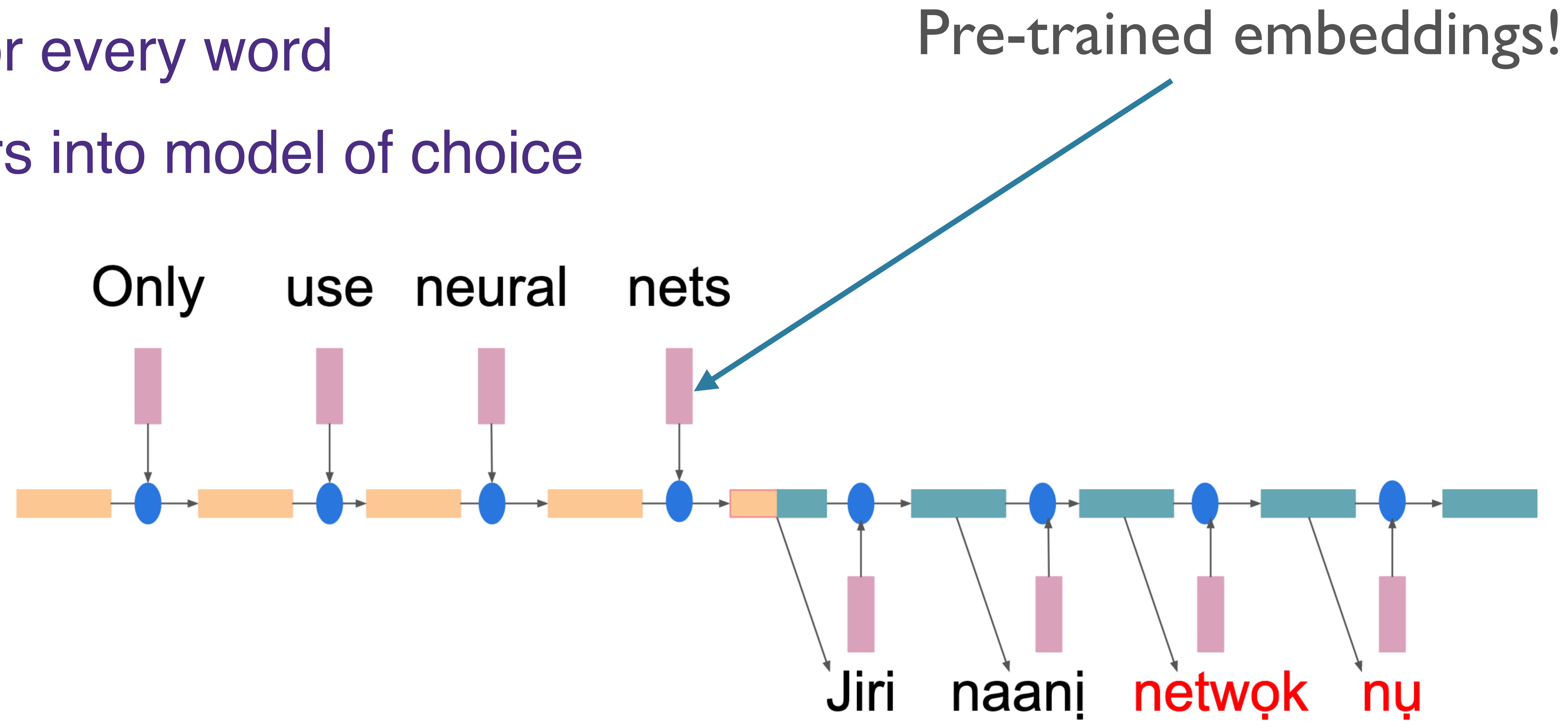
- Embedding layer (~300-dimensions):
  - download pre-trained embeddings
  - Use as look-up table for every word
  - Then feed those vectors into model of choice



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

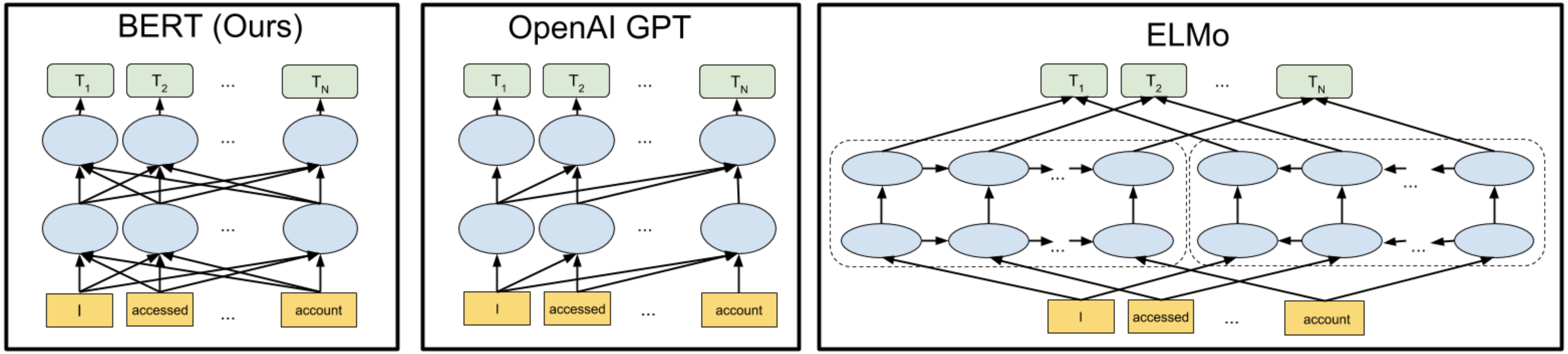


Depiction of seq2seq NMT architecture
c/o Hewitt & Kriz

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 (via pre-trained large language models)
- Here's a nice "contextual introduction"

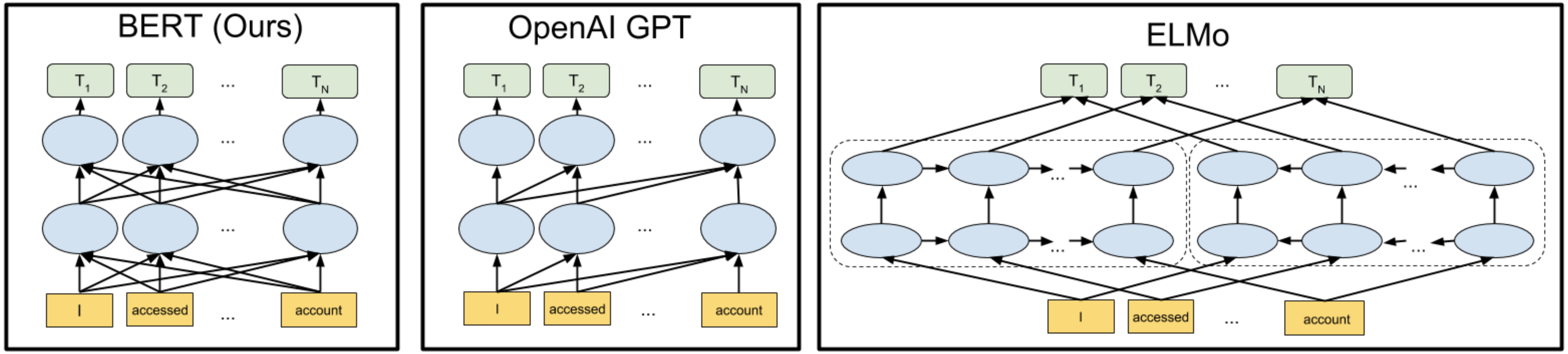
Contextual Word Representations



Devlin et al 2018 Radford et al 2019 Peters et al 2018

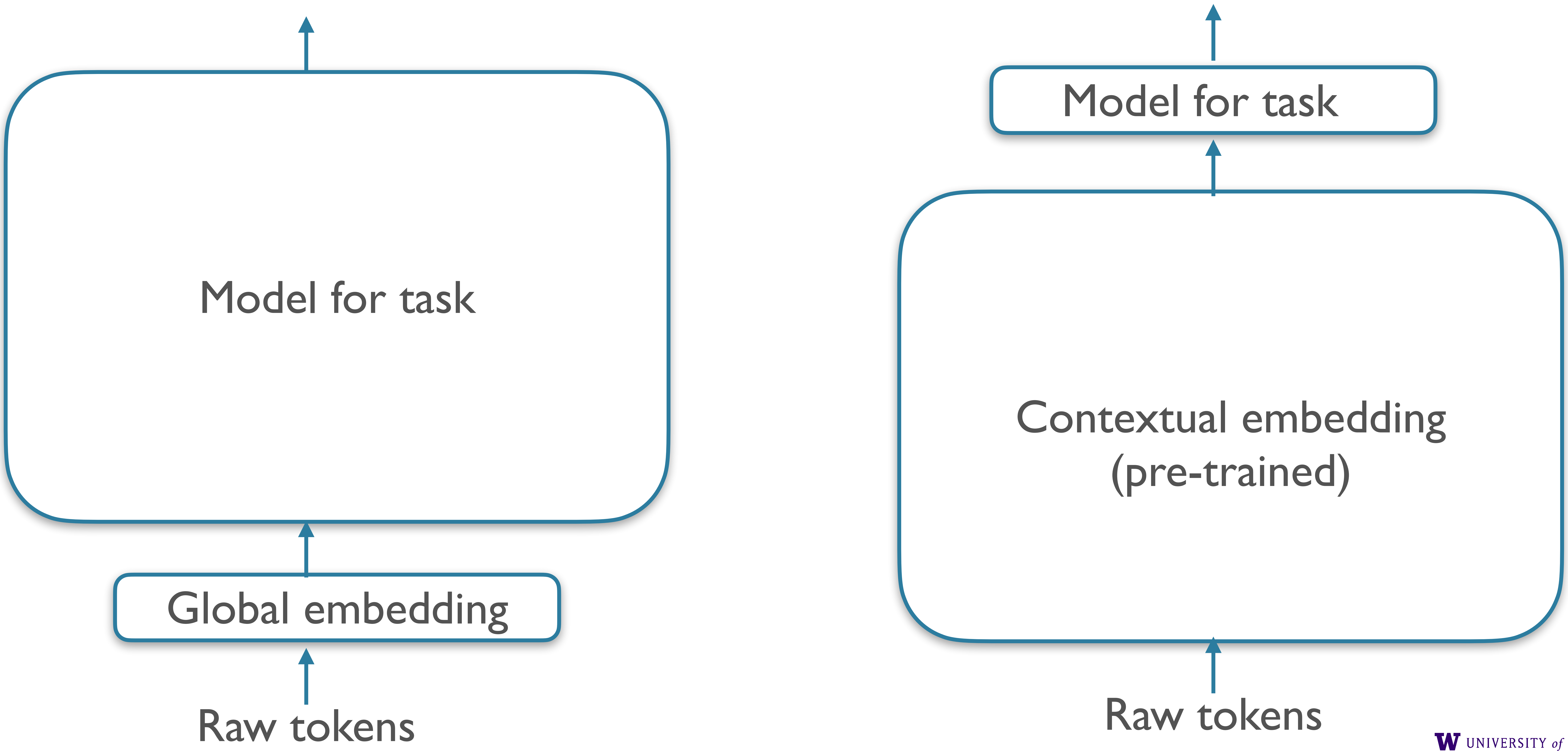
Contextual Word Representations

"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[1], **Kai-Wei Chang**[2], **James Zou**[2], **Venkatesh Saligrama**[1,2], **Adam Kalai**[2]
[1]Boston University, 8 Saint Mary's Street, Boston, MA
[2]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

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

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