VECTOR SEMANTICS AND EMBEDDINGS

Spring 2023

CS6431 Natural Language Processing

Credits

B1: *Speech and Language Processing (Third Edition draft – Jan2022)*

Daniel Jurafsky, James H. Martin

Assignment

Read:

B1: Chapter 6

Problems:

Context

- Words in similar context tend to have similar meaning
  - Synonyms (Oculist and eye-doctor)
  - Same environment (eye and examined)
- Words => Embeddings (number vectors)
  - Preserve contextual relation
  - Representation learning

Lexical Semantics

Word Representation

- Indexing based representation is insufficient
- Word representation should tell us that
  - Some words have similar meanings (e.g., cat and dog)
  - Some have positive connotation (e.g., happy and great)
  - Some have negative connotation (e.g., sad and painful)
  - Related to the same event (e.g., buy, sell, and pay are related to purchase)
  - Have same gender (e.g., woman and queen)
- It should allow us to draw inferences to address meaning-related tasks like question-answering or dialogue.

Lemmas and Senses

- Same word, multiple meaning

```
mouse (N)
1.   any of numerous small rodents...
2.   a hand-operated device that controls a cursor...
```

  - mouse: lemma/citation form; mice: word form (inflicted from mouse)
  - Mouse has two ***word senses*** (i.e., polysemous and requires word-sense disambiguation)

- Synonymy – different word, similar meaning
  - couch/sofa, vomit/throw up, car/automobile
  - Can be mutually substituted without changing the 'truth condition' of a sentence
- Principle of contrast – "*A difference in linguistic form is always associated with some difference in meaning*"
  - H_2O vs. Water – mutually substitutable but used in different context

Word Similarity

- Synonyms: Cat and feline
- Similar meaning: Cat and dog (both are pets, animal, similar size etc.)
- A formula to compute word similarity can lead to a formula to compute sentence/phrase similarity
- Human judged word similarity datasets
  - SimLex-999 dataset (Hill et al., 2015)

Word Relatedness

- Coffee and cup; scalpel and surgeon
  - Don't have similar meanings but co-participate in a common event
  - Belong to the same semantic field
    - Semantic fields of hospitals (surgeon, scalpel, nurse, anesthetic, hospital)
    - Semantic fields of Restaurants (waiter, menu, plate, food, chef)
    - Semantic fields of houses (door, roof, topic models kitchen, family, bed)

Semantic Frames and Roles

- A semantic frame is a set of words that denote perspectives or participants in a particular type of event.
  - E.g., the event "transaction"
    - buy, buyer (the event from the perspective of the buyer)
    - sell (from the perspective of the seller)
    - pay (focusing on the monetary aspect)
- Sam bought the book from Ling
  - Sam: buyer perspective, Ling: seller perspective
  - Can be paraphrased as "Ling sold the book to Sam"

Connotation

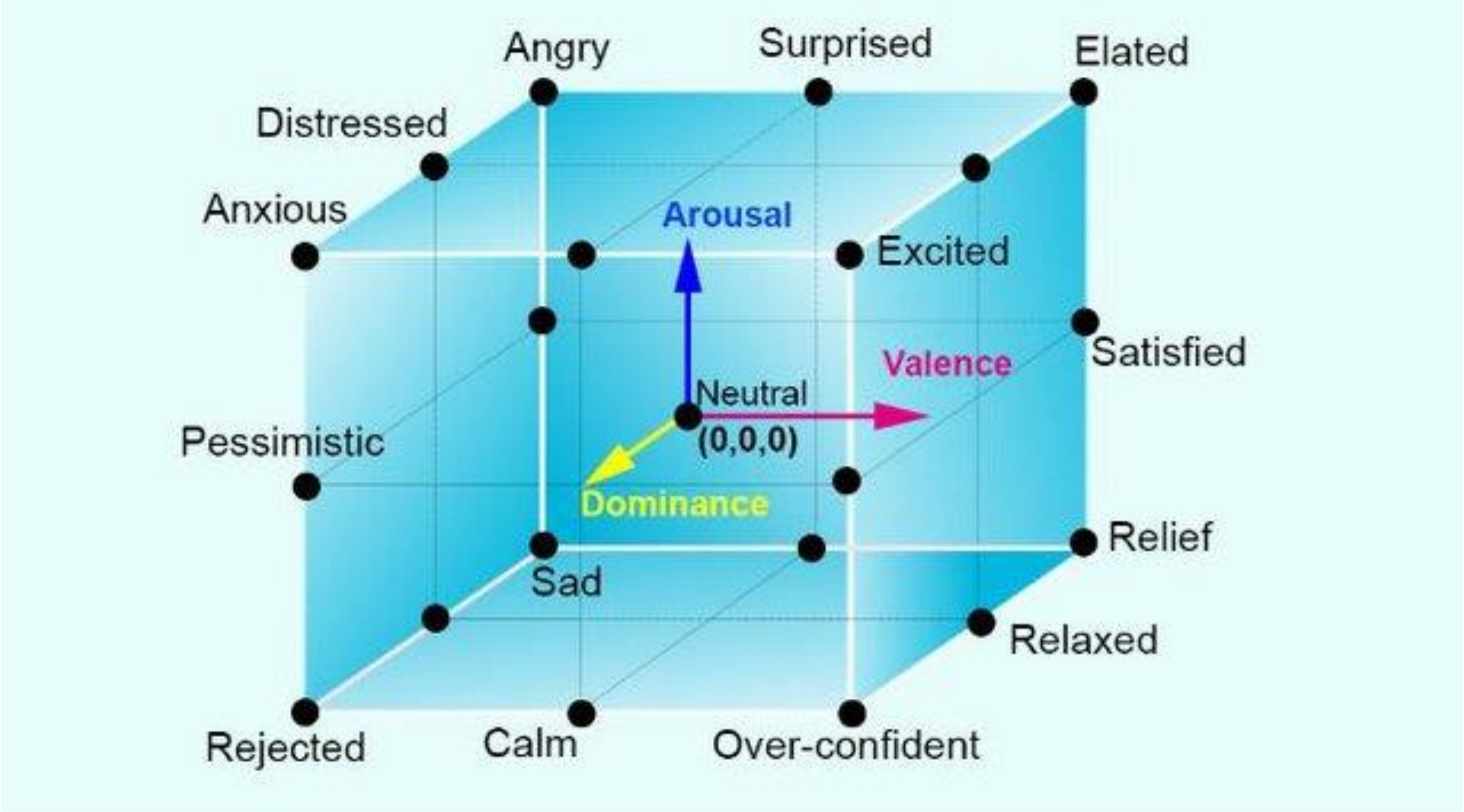
- Affective meanings (relating to moods, feelings, and attitudes)
  - fake, knockoff, forgery?
  - copy, replica, reproduction?
- Dimensions of affected meanings
  - Valence: the pleasantness of the stimulus
  - Arousal: the intensity of emotion provoked by the stimulus
  - Dominance: the degree of control exerted by the stimulus

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

Write on desk

- Valence: the pleasantness of the stimulus
- Arousal: the intensity of emotion provoked by the stimulus
- Dominance: the degree of control exerted by the stimulus

Write on desk



Vector Semantics

Chard

Collard Greens

- Suppose you don't know the meaning of Ongchoi but you have seen the following

(1) Ongchoi is delicious sauteed with garlic.

(2) Ongchoi is superb over rice.

(3) ...ongchoi leaves with salty sauces...

You know the meaning of spinach, chard and collard greens and have seen the following:

(4) ...spinach sauteed with garlic over rice...

(5) ...chard stems and leaves are delicious...

(6) ...collard greens and other salty leafy greens

Can we derive the meaning of Ongchoi? How?

Ongchoi has the same context words like spinach, chard and collard greens

- Ongchoi:

- Word embeddings
  - Represent word in an n-dimensional space (similar to Valence, Arousal, Dominance)
  - Neighbouring words are related (similar meanings/ related)

Original 60-d embeddings from Li et al. (2015) simplified to 2d

Vector Embeddings

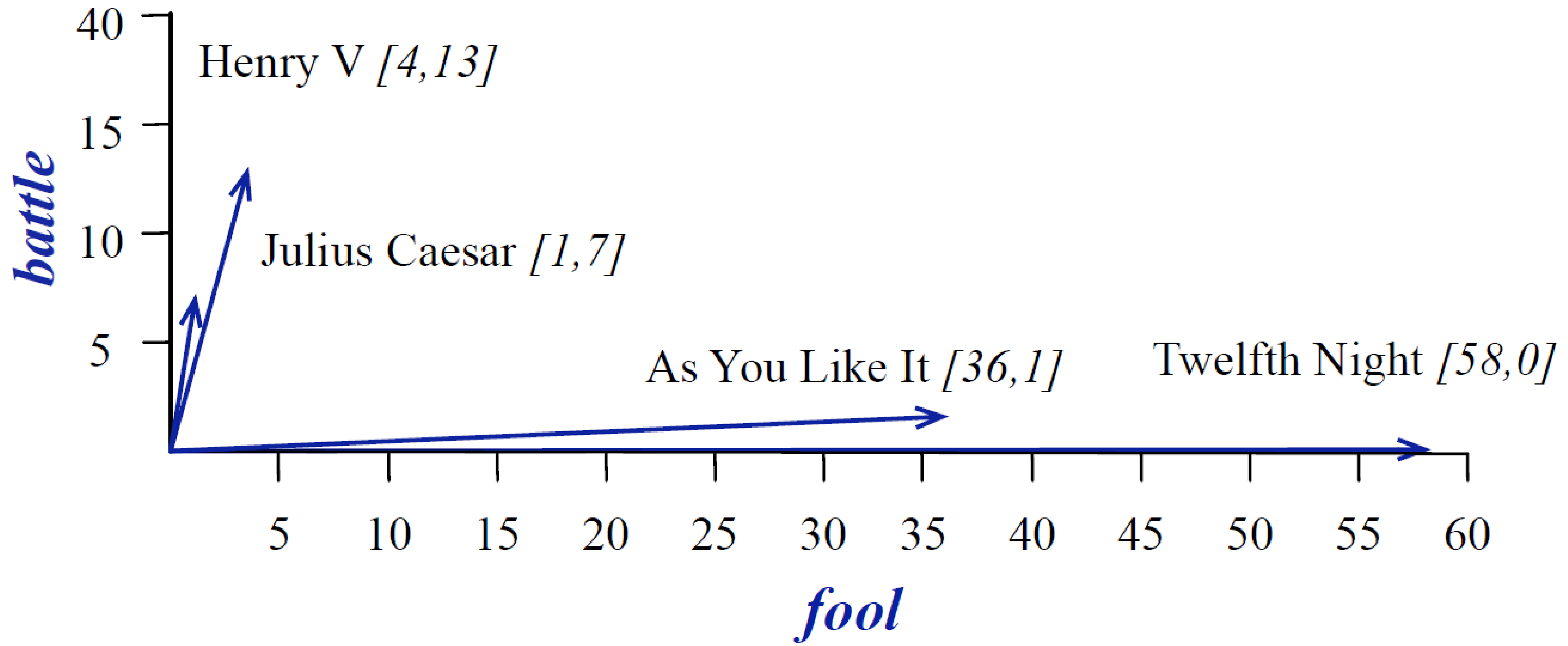
- Word vectors based co-occurrence matrix
  - Term-document matrix and the Term-term matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Term-document matrix

  - Both terms and documents can be represented as vectors
    - Similar documents have use in information retrieval

write on desk



Term-term matrix

- Captures if two terms occur in the same context
  - Context can be document/sentence/window
    - E.g., a window of 4 words on either side

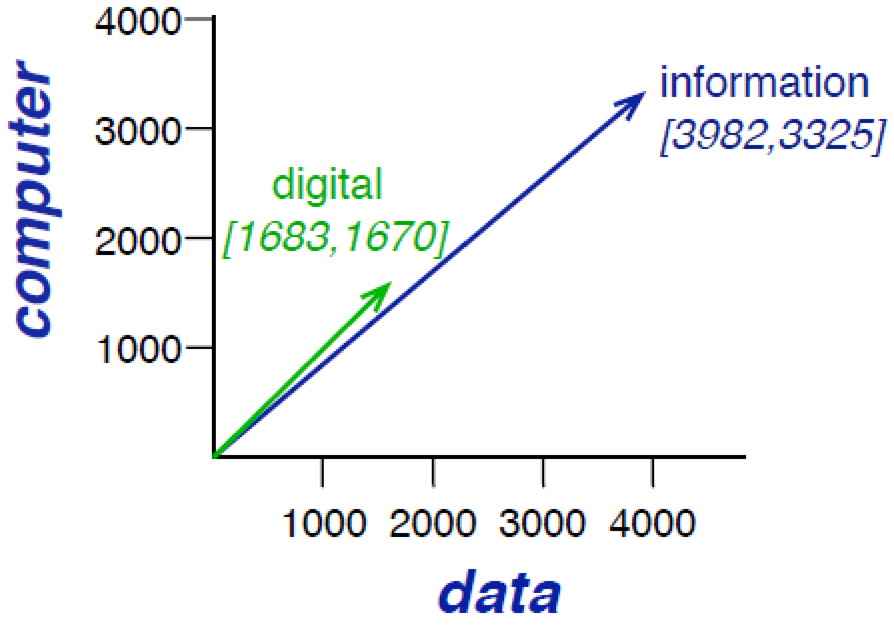
is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	aardvark	...	**computer**	**data**	**result**	**pie**	**sugar**	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

A sub co-occurrence matrix from Wikipedia corpus (Davies, 2015).

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

- Very sparse matrix
  - Limit to most (say 50k) words in a corpus



Cosine Similarity

Cosine Similarity or Dot Product

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- Problem: favors long vectors, i.e., frequent words
- Solution: normalized dot product

$$\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos\theta$$

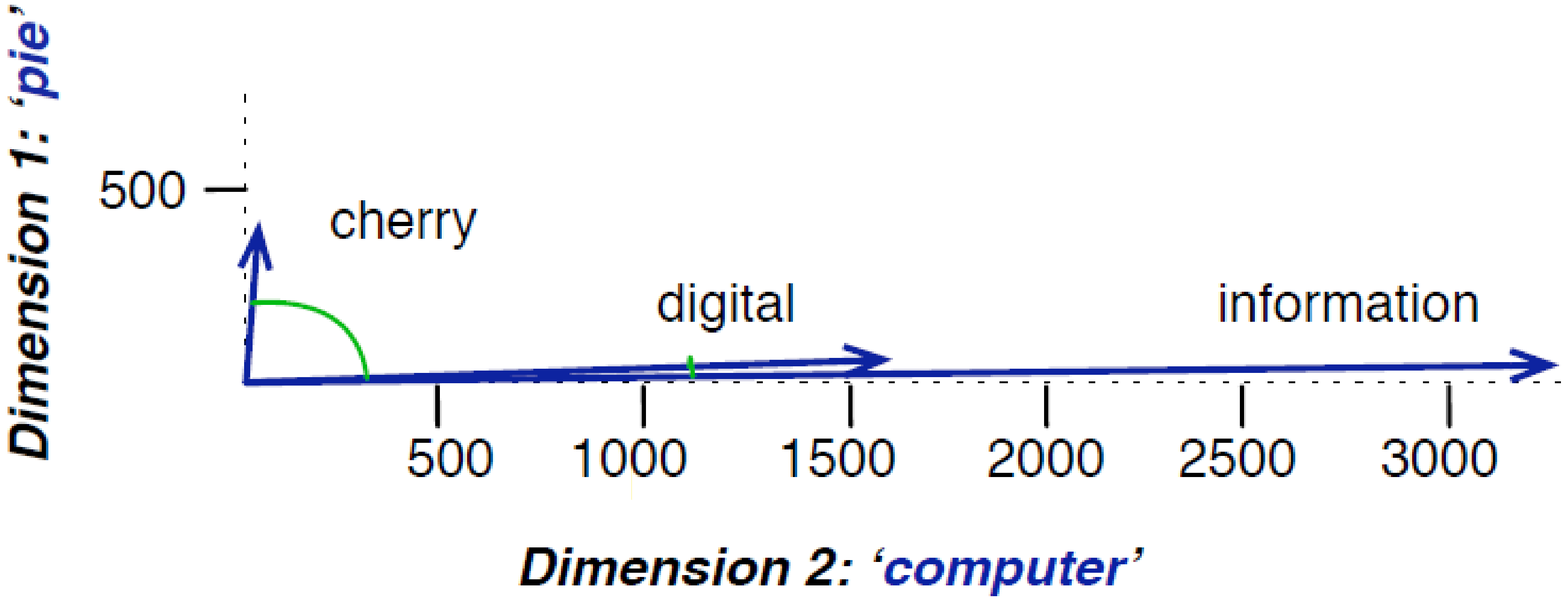
$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

write on desk

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry}, \text{information}) = \frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .018$$

$$\cos(\text{digital}, \text{information}) = \frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .996$$



TF-IDF

- Paradox: both low and high word-frequency are not good

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- We want frequent co-occurrence but not words that too frequent
- Two solutions
  - tf-idf : usually used with word to word co-occurrence matrix
  - PPMI : usually used with word to document co-occurrence matrix

Tf-idf

- tf: term frequency – favors high co-occurrence frequency
- Idf: inverse document frequency – penalizes too frequent words / favors rare words
- Net score: tf*idf

$$\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$$

- Normalize high frequency with log and add +1 to avoid log 0

$$\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1)$$

- Document frequency vs. Collection frequency

$$\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

Where:

- N: total number of documents,
- df_t: the number of documents containing term t
- Log normalizes large values

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

- Net score

write on desk

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Original co-occurrence values

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Tf-idf values

Pointwise Mutual Information (PMI)

Pointwise Mutual Information (PMI)

- Used for term-term matrix
- Intuition: how much more the two words co-occur in our corpus than we would have a priori expected them to appear by chance

$$\mathrm{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

Write on desk

  - Range of PMI values?
  - Negative values tend to be unreliable. (why?)
  - Use only positive PMI

$$\mathrm{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

- How to compute $P(w,c)$, $P(w)$, and $P(c)$?
- $F[W \times C]$ be the co-occurrence matrix
  - W: #words
  - C: #context
  - f_{ij}: #times word w_i occurs with context c_i

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\mathrm{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0)$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

- Compute PPMI(information,data)

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

Write on desk

$$\mathrm{PPMI}_{ij} = \max\left(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0\right)$$

- PMI is biased towards infrequent events (how?)
- Solution 1:

$$\mathrm{PPMI}_{\alpha}(w,c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P_{\alpha}(c)}, 0\right)$$

$$P_{\alpha}(c) = \frac{count(c)^{\alpha}}{\sum_c count(c)^{\alpha}} \quad ; 0 < \alpha \leq 1$$

- Solution 2: Laplace smoothing (how?)

Word2Vec

- Generate short dense embeddings of desired size
  - Previous embeddings were large (#documents, #nodes)
- Short dense embeddings perform better than large sparse embeddings
  - Why?
- word2vec (Mikolov et al. 2013)

"After them, you threw in slices of dried apple and apricot."

- So far: count how many times "apple" and "apricot" occur together
- Instead: train a classifier on the binary prediction task: "Is word w likely to show up near apricot?"
  - Don't care about this prediction task
  - Instead take the learned weights as embeddings
- Any running text becomes supervised training data
  - Abundantly available, no need for hand-labelled supervised signal

Skip-gram model for Word2Vec

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples.
3. Use logistic regression to train a classifier to distinguish those two cases.
4. Use the learned weights as the embeddings.

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

Window size =2

- $P(+|w,c)$: probability that c is a true context word
- $P(-|w,c) = 1 - P(+|w,c)$ probability that c is not a true context word
- If w and c are context words, their embeddings would be similar
  - Use dot product to compute embedding similarity

$$Similarity(w,c) \approx \mathbf{c} \cdot \mathbf{w}$$

- Dot product does not give a probability value: use sigmoid

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

$$\begin{aligned} P(-|w,c) &= 1 - P(+|w,c) \\ &= \sigma(-\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(\mathbf{c}\cdot\mathbf{w})} \end{aligned}$$

Write on desk

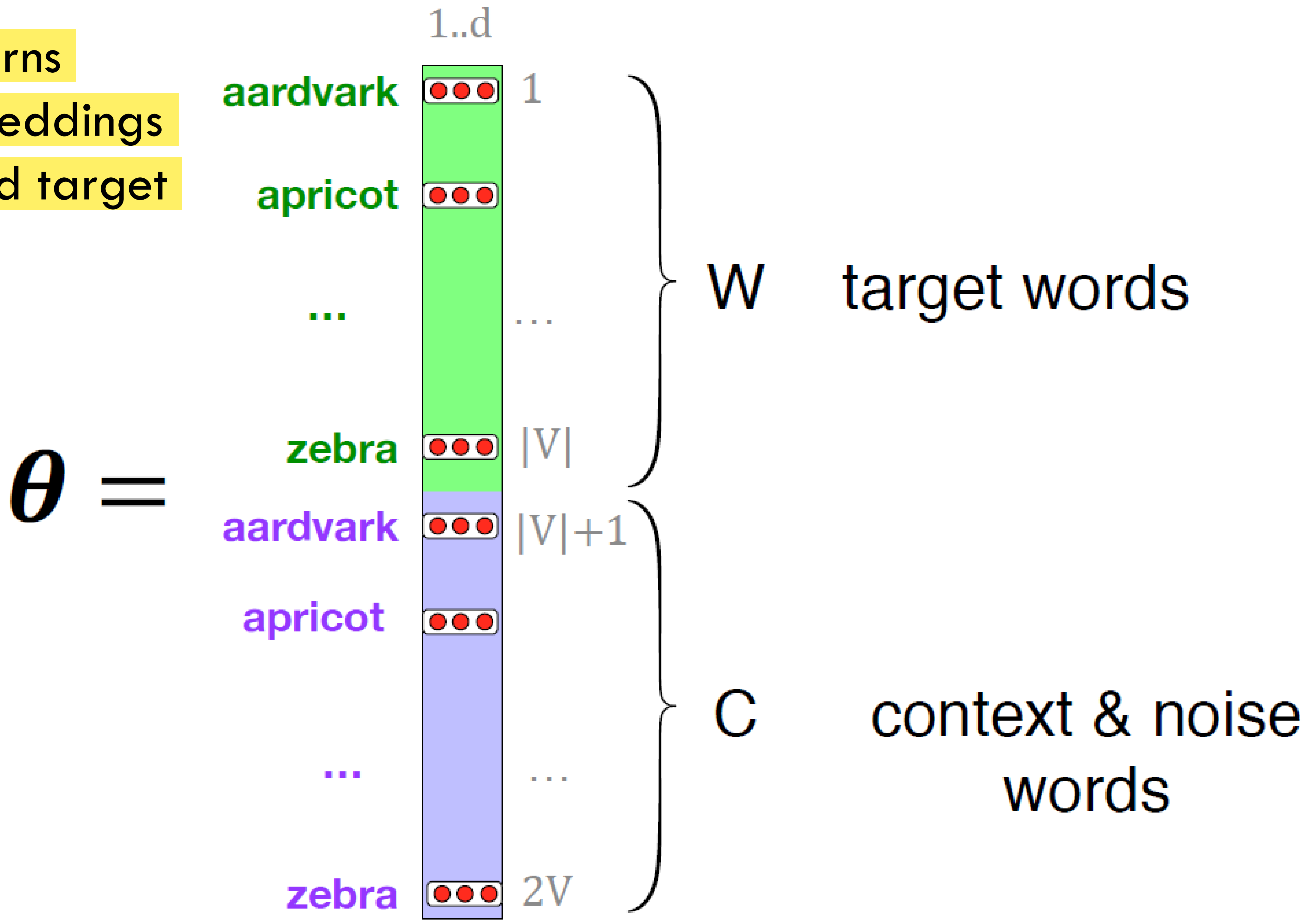
- How to compute the combined probability for all the words in a context window?
  - Assume all context words are independent

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

write on desk

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

- Skip-gram learns separate embeddings for context and target words



- Initially all embeddings are randomly assigned
- Iteratively shift the embedding of each word *w* to be more like the embeddings of words that occur nearby in texts, and less like the embeddings of words that don't occur nearby.

```
... 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 examples -

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

- For each positive sample, $k > 1$ negative samples are created
  - Noise sample: chosen randomly from the lexicon
  - As per weighted unigram frequency

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

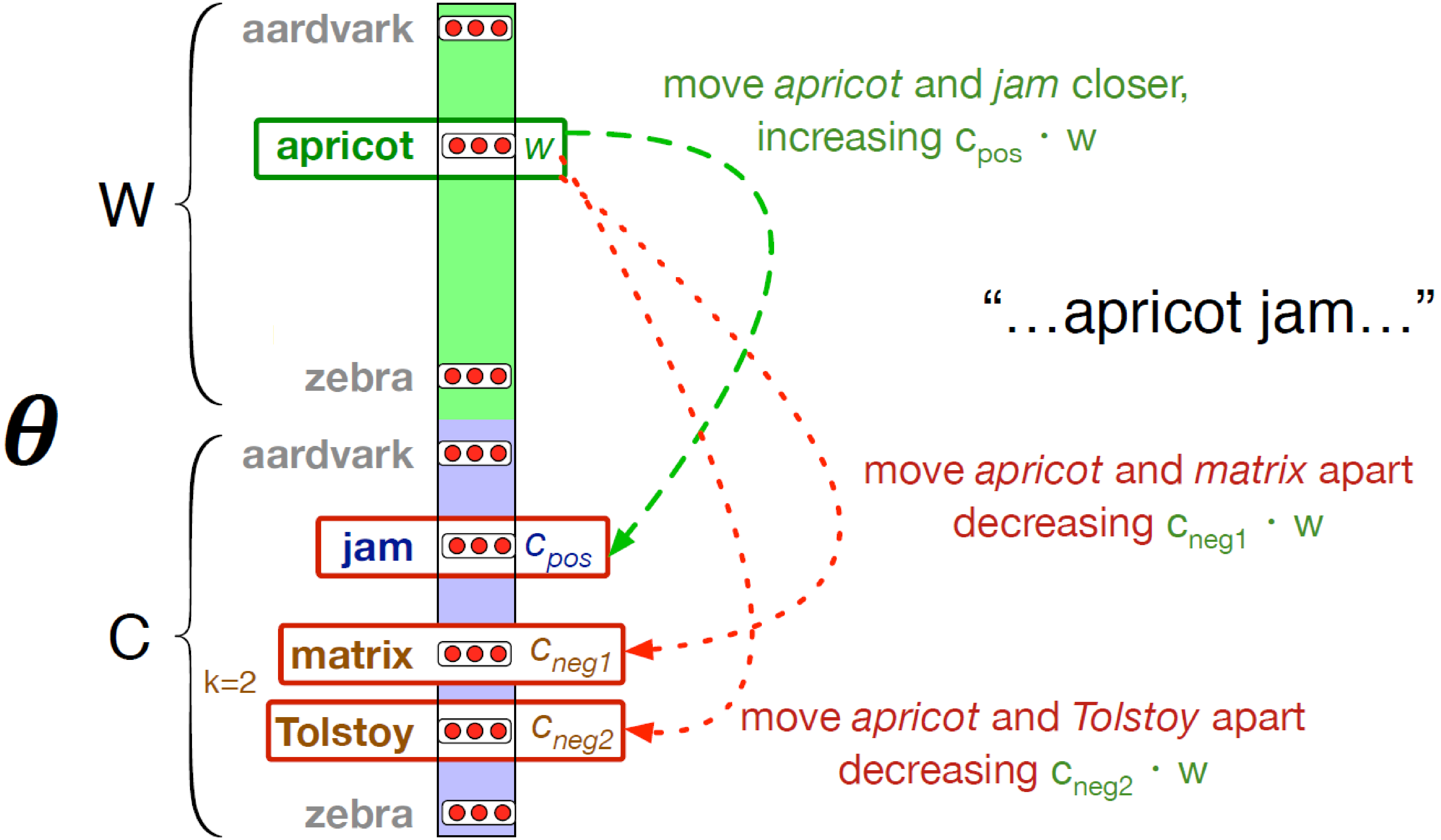
    - $\alpha = .75$ (say) gives more weightage to rare words

- Given
  - Set of +ve and –ve training instances
  - Initial random embeddings
- Goal:
  - Maximize the similarity of the target word, context word pairs (w, c_{pos}) drawn from the positive examples
  - Minimize the similarity of the (w, c_{neg}) pairs from the negative examples.

Loss Function

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log\sigma(c_{pos}\cdot w)+\sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

Write on desk



W
aardvark
apricot
w
zebra
θ
C
aardvark
jam
c pos
k=2
matrix
c neg1
Tolstoy
c neg2
zebra
move *apricot* and *jam* closer,
increasing $c_{pos} \cdot w$
"…apricot jam…"
move *apricot* and *matrix* apart
decreasing $c_{neg1} \cdot w$
move *apricot* and *Tolstoy* apart
decreasing $c_{neg2} \cdot w$

Summary of Word2vec

- Assign random embeddings to words
  - Separate embeddings for target and context words or use the same embeddings
- For every training sample set consisting of one +ve sample and k –ve samples
  - Compute loss function

$$L_{CE} = -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right]$$

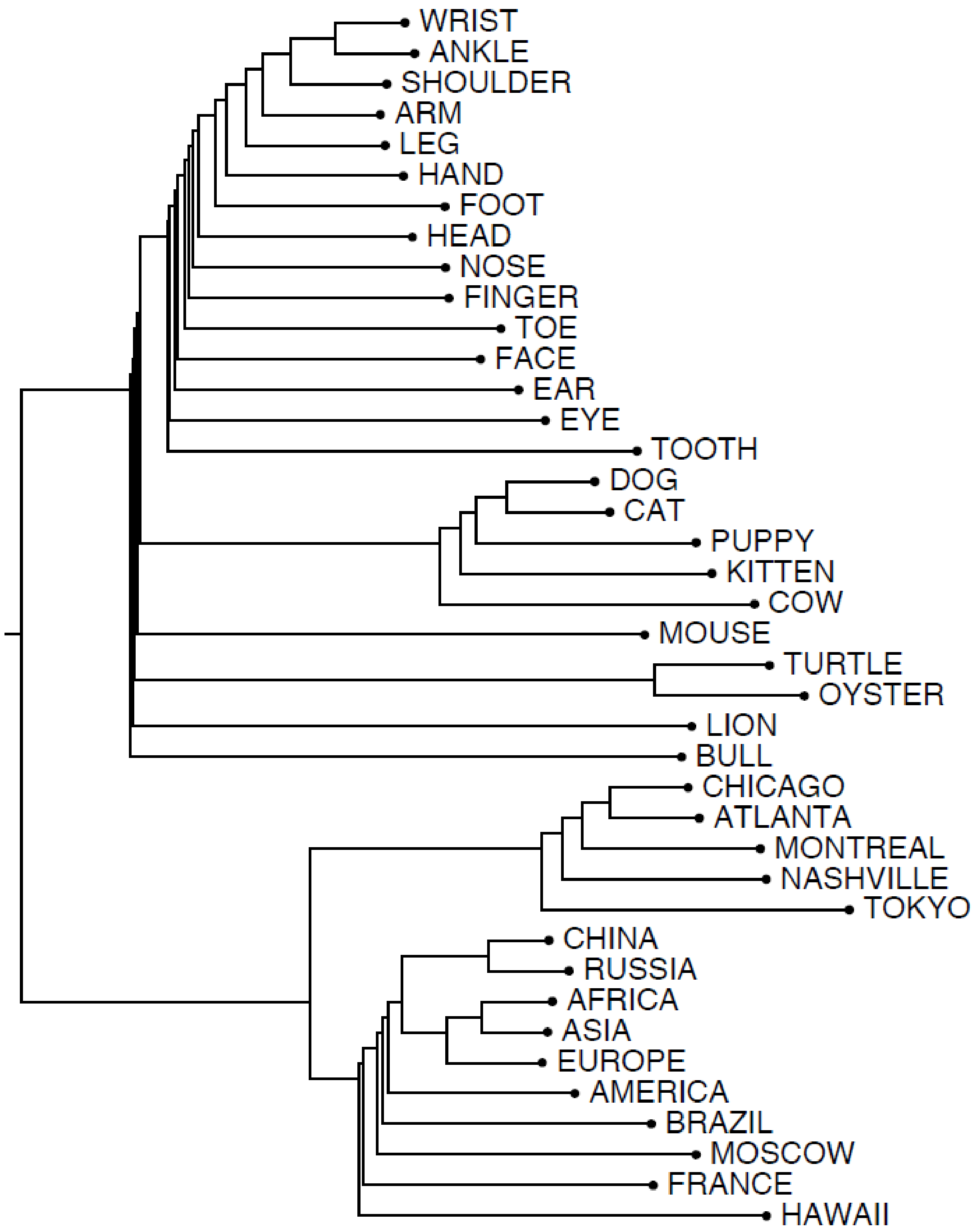
  - Update embeddings of only words in the sample as per gradient descent

Fasttext (Bojanowski et al., 2017),

- Word2vec – no good way to deal with unknown sparse words
- Solution: store each word as it is + constituent n-grams
  - E.g., “where” is stored as “where”, “<wh”, “whe”, “her”, “ere”, “re>”
    - “<“ and “>” special boundary symbols

Visualizing Embeddings

- List $k = 7$ most similar words to w (Pennington et al., 2014)
  - E.g., frog: frogs, toad, litoria, leptodactylidae, rana, lizard, and eleutherodactylus
- Use hierarchical clustering (Rohde et al., 2006).
- t-SNE visualization (van der Maaten and Hinton, 2008).
  - Project $n = 100$ dimensions into two



Semantic properties of embeddings

Size of context window

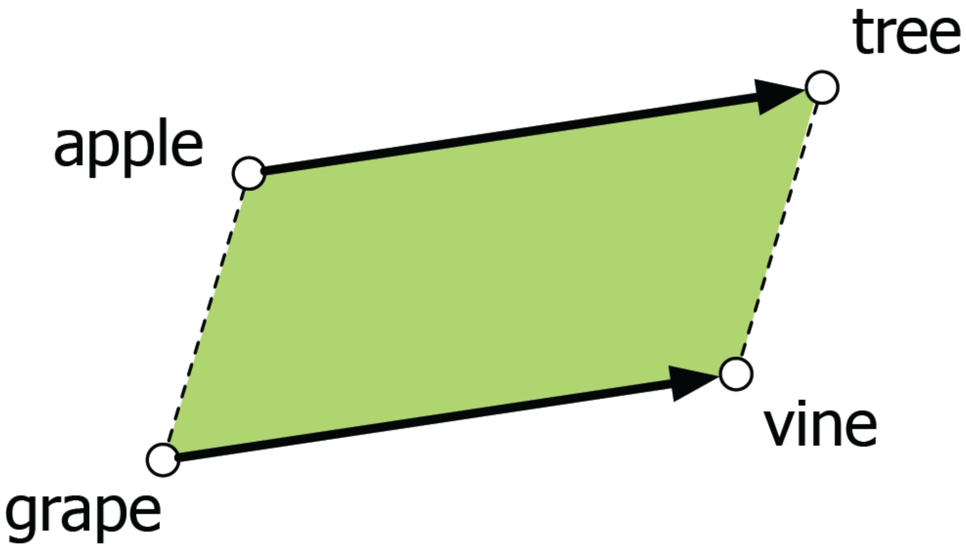
- Typically 1-10 words on either side
- Shorter context window: the most similar words (to a target word w) tend to be semantically similar words with the same parts of speech.
  - E.g., NIT Patna=> NIT Agartala, IIT Patna
- Longer context window: the most similar words tend to be topically related but not similar
  - E.g., NIT Patna=> SAC, Sankalp, Ganga ghat

Word Association/Similarity

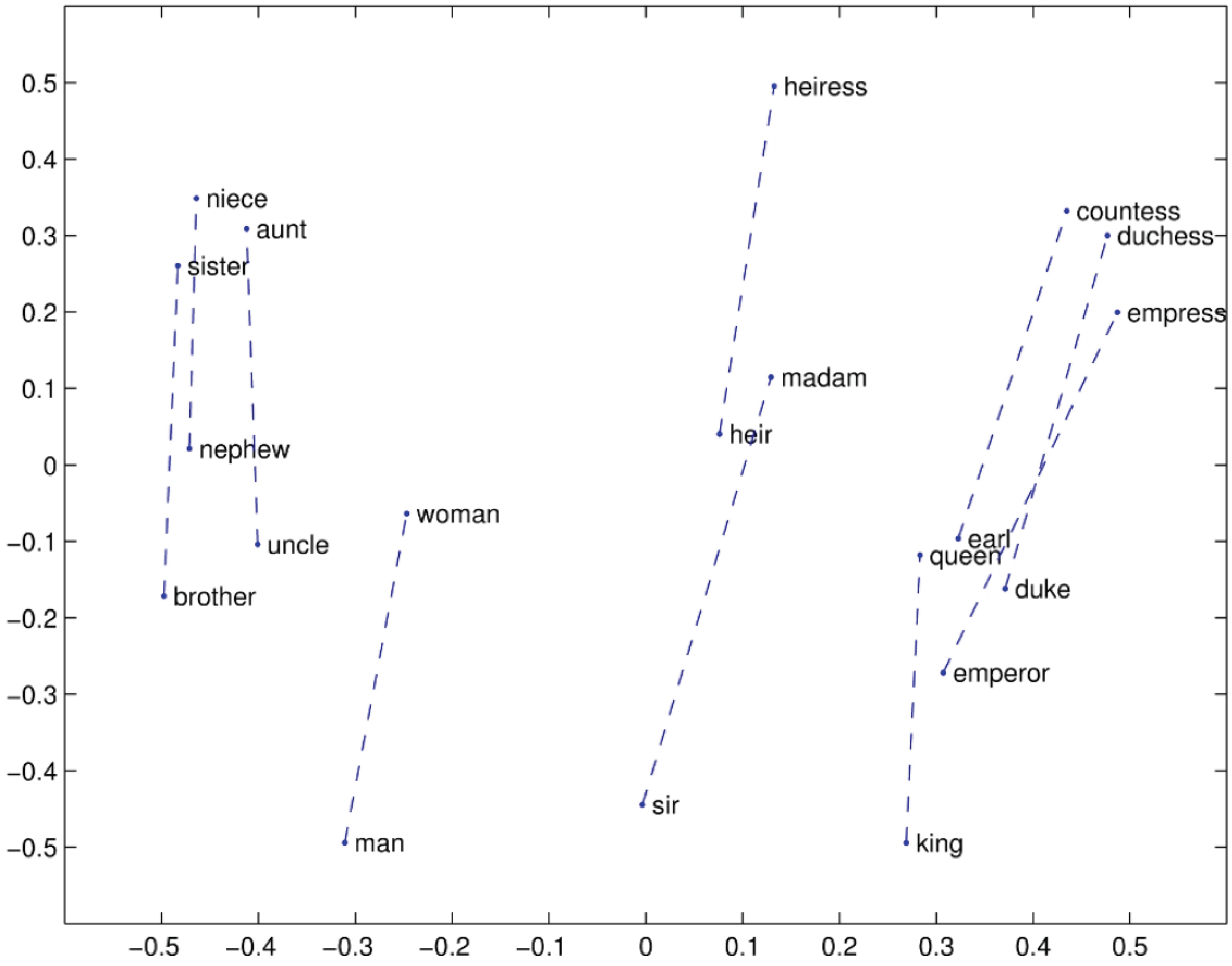
- Syntagmatic association or First order co-occurrence – if they are nearby
  - E.g., wrote is a first order associate of book or poem
- Paradigmatic association – if they have similar neighbours
  - E.g., wrote is a second-order associate of words like said or remarked

Analogy/Relational Similarity

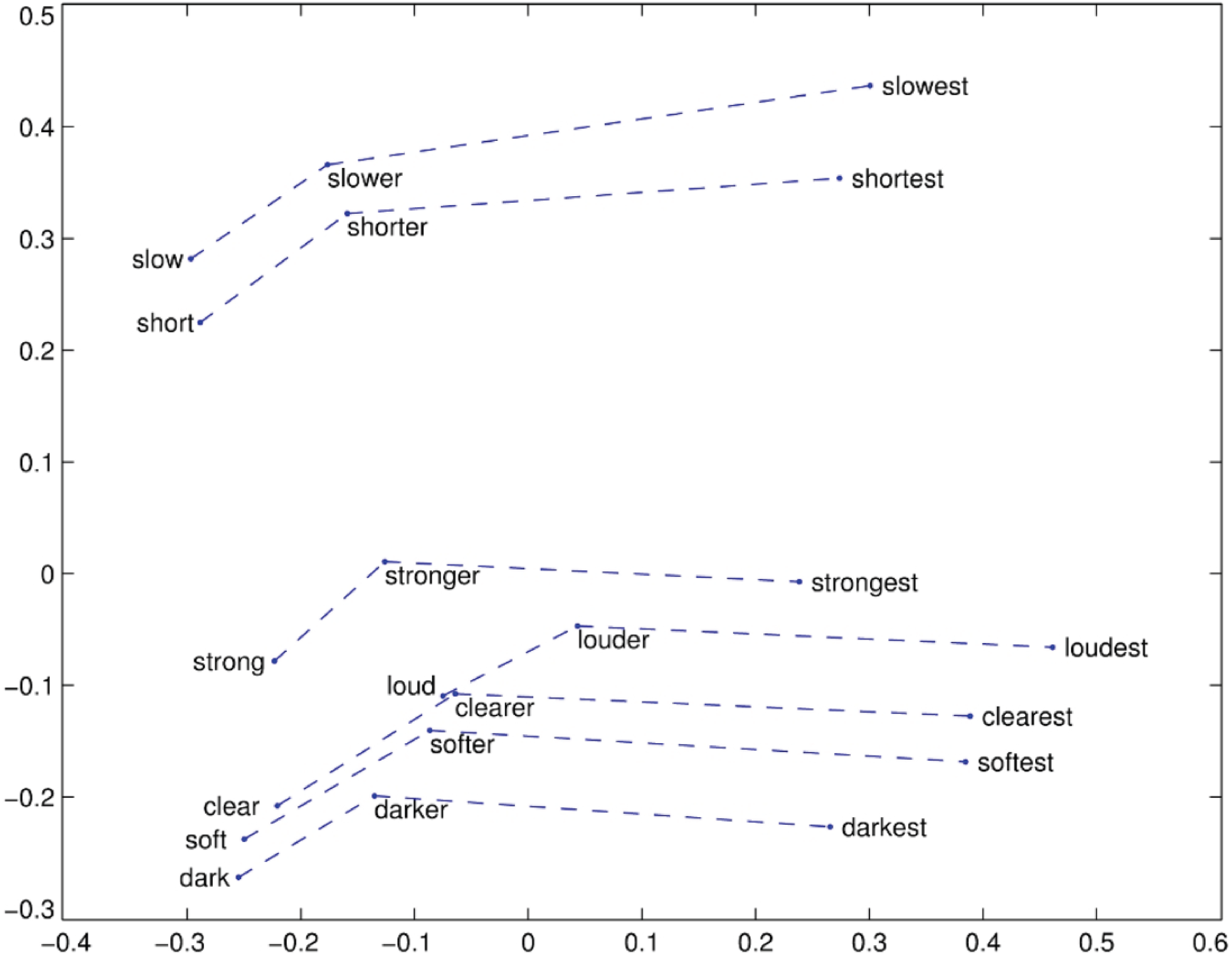
- Parallelogram model (Rumelhart and Abrahamson, 1973):- solving simple analogy problems of the form a is to b as a* is to what?
  - E.g., apple:tree::grape:? (Ans: vine)
  - $(\overrightarrow{tree} - \overrightarrow{apple}) = (\overrightarrow{vine} - \overrightarrow{grape})$



- While sparse models achieved success in solving analogy problems, it was much more with word2Vec and GloVe vectors
  - Capturing MALE-FEMALE, CAPITAL-CITY-OF, COMPARATIVE/SUPERLATIVE,



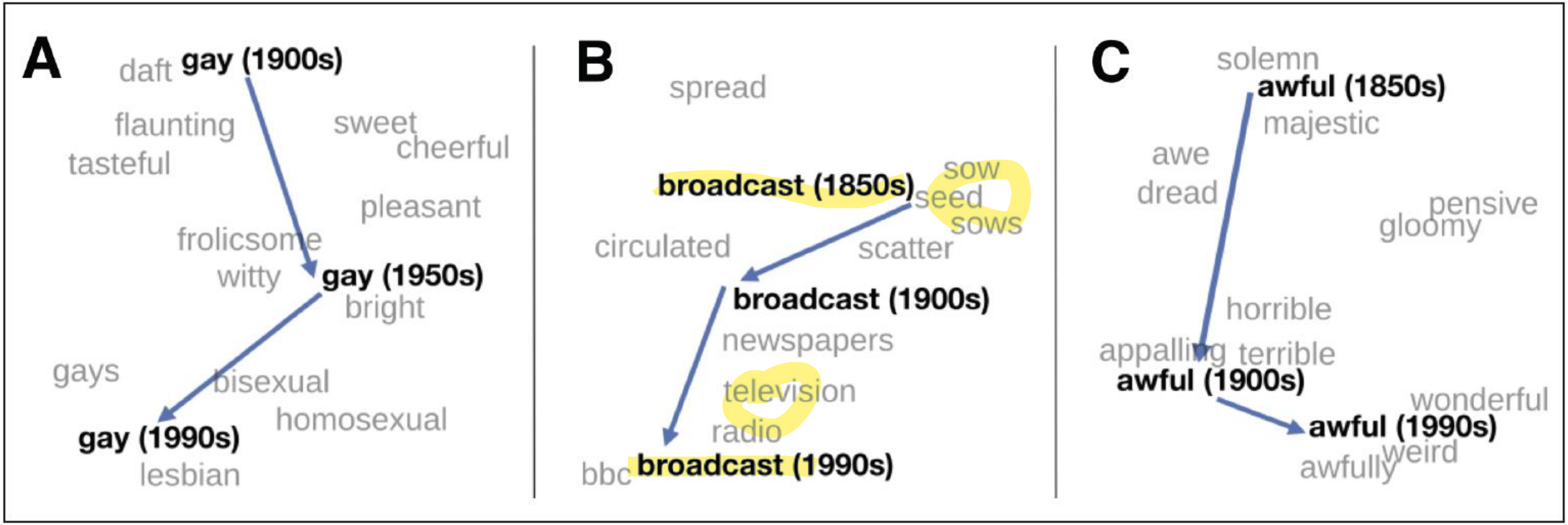
GloVe vector space capturing MALE-FEMALE analogies



GloVe vector space capturing COMPARATIVE/SUPERLATIVE analogies

Embeddings and Historical Semantics

- Tracing how word embeddings change over time



Using Word2Vec and t-SNE visualization

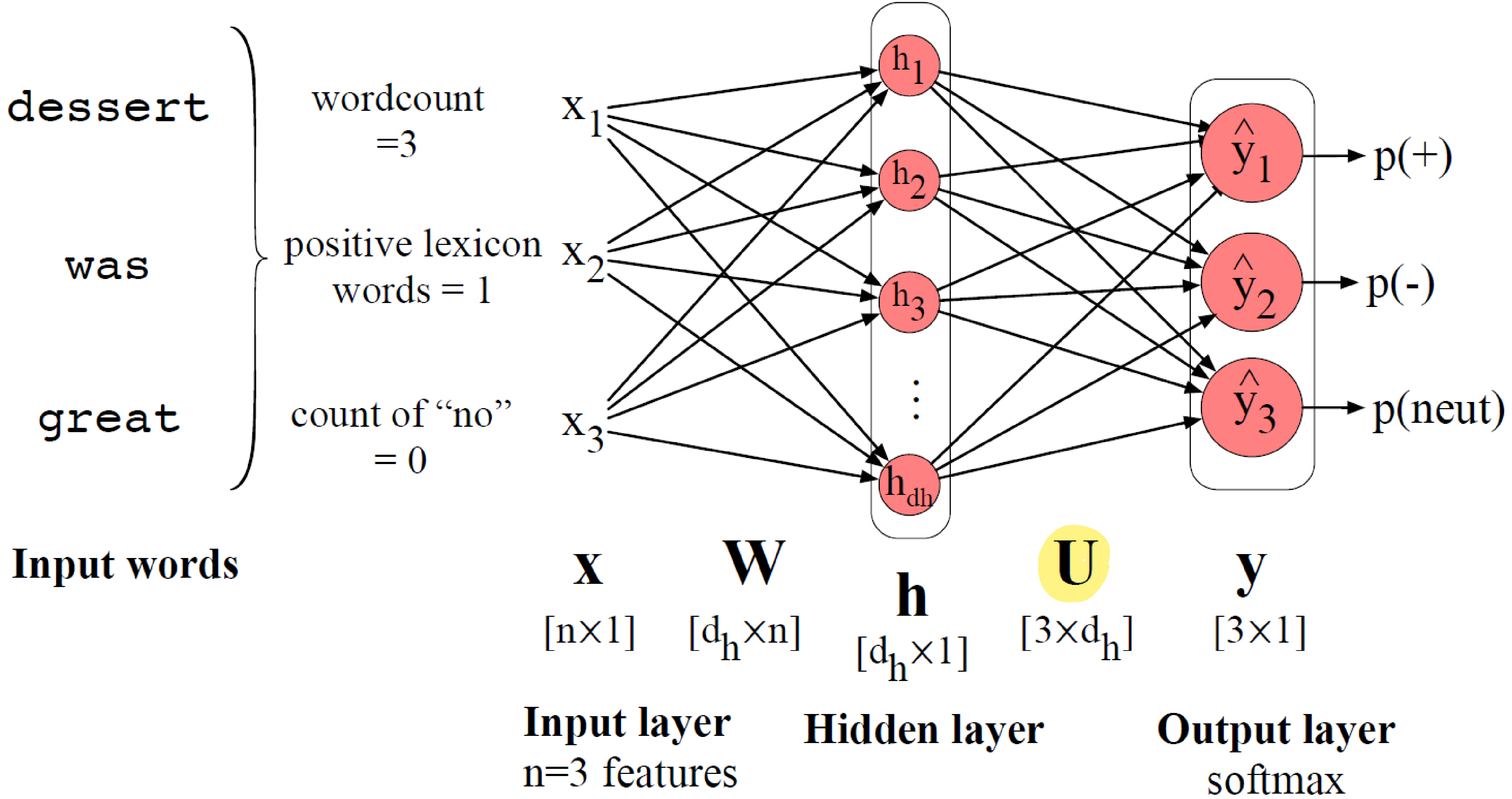
Bias and Embeddings

- Embeddings also reproduce the implicit biases and stereotypes that were latent in the text
- Gender stereotypes – Bolukbasi et al. (2016) using Word2vec
  - Closest occupation to 'man' – 'computer programmer' + 'woman' ?
    - 'homemaker'
  - 'father' is to 'doctor', 'mother' is to _?
    - 'nurse'
  - Allocational harm – Crawford (2017) and Blodgett et al. (2020)
    - When a system allocates resources (jobs or credit) unfairly to different groups

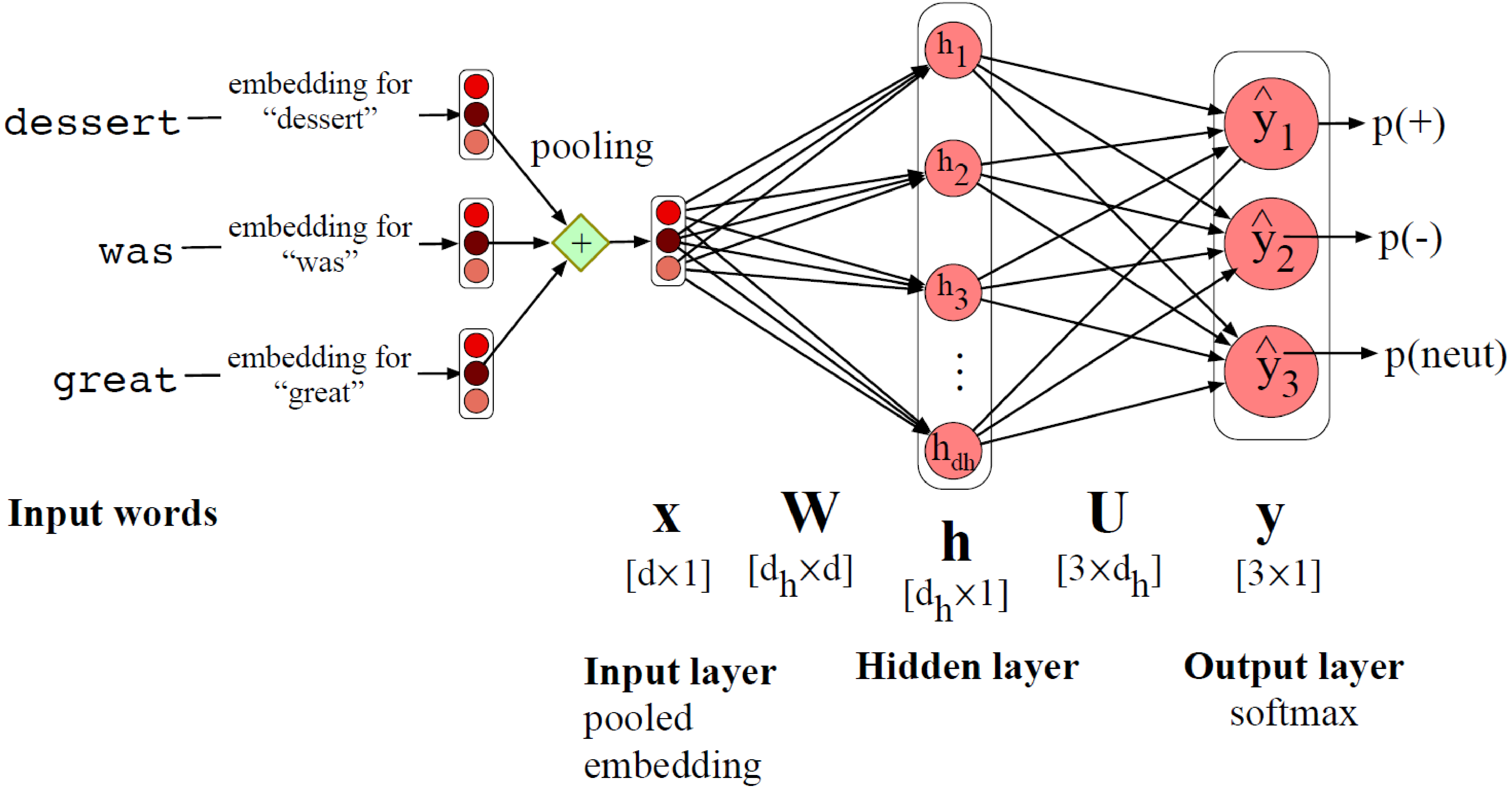
- Caliskan et al. (2017) using GloVe vectors
  - African-American names like 'Leroy' and 'Shaniqua' had a higher GloVe cosine with unpleasant words, while European-American names ('Brad', 'Greg', 'Courtney') had a higher cosine with pleasant words.
  - Male names more with mathematics and female names with the arts
  - Old people names with unpleasant words
- Representational harm (Crawford 2017, Blodgett et al. 2020)
  - Harm caused by a system demeaning or even ignoring some social groups.
- Work on embedding debiasing has been done but with limited success

Feedforward networks for NLP: Classification

Hand-crafted Features



- Word embeddings to sentence embeddings?
  - Mean/ max pooling

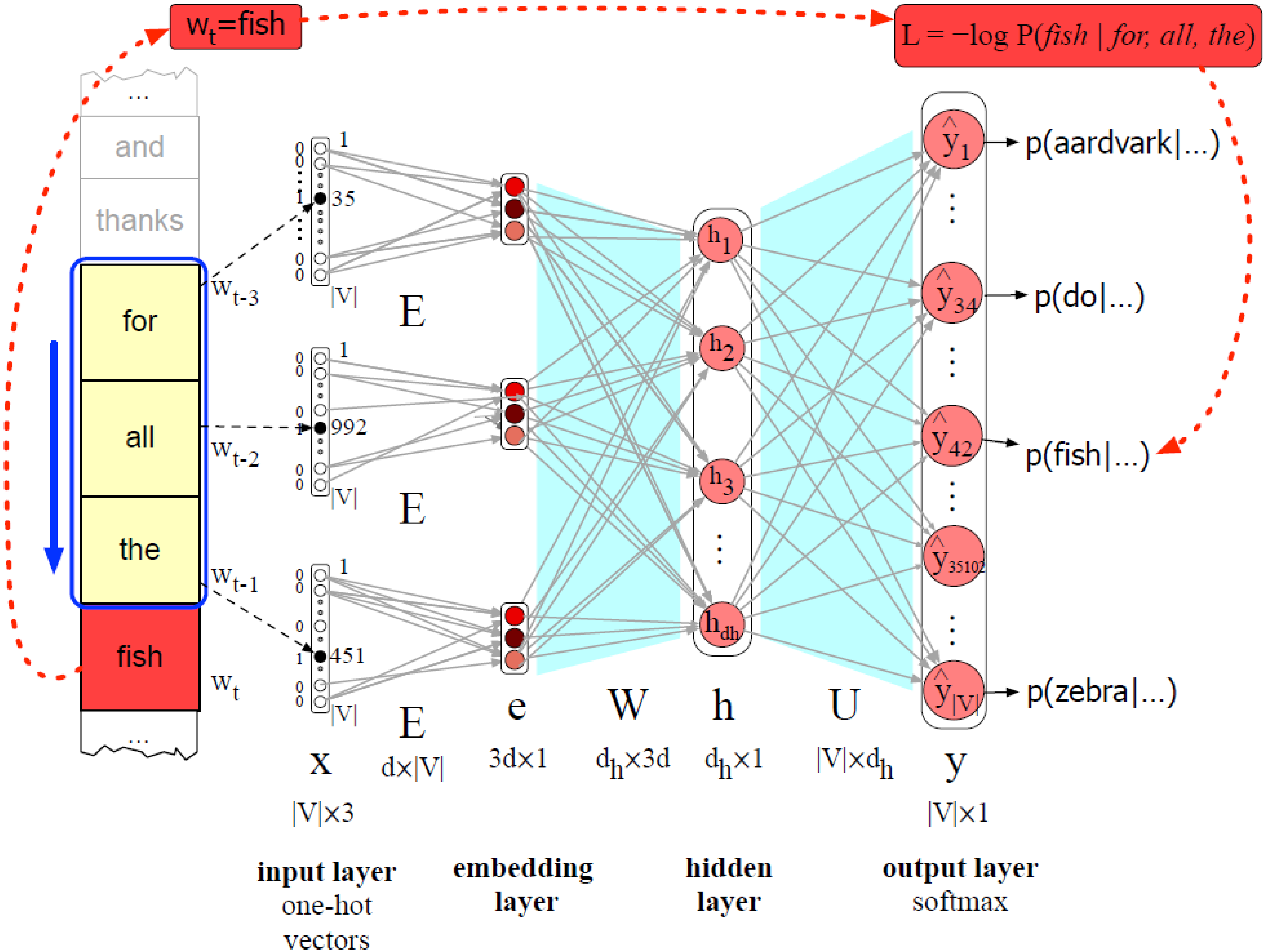


Language Modelling

Predicting upcoming words from prior word context

Neural Language Model over N-gram Model

- Pros
  - Handle much longer sentences
  - Generalize better over contexts of similar words
  - More accurate at word-prediction
- Cons
  - Computationally expensive
  - Less interpretable



w_t=fish
L = −log P(fish | for, all, the)
...
and
thanks
for
all
the
fish
...
w_{t-3}
w_{t-2}
w_{t-1}
w_t
1
35
992
451
|V|
E
e
W
h
U
h_1
h_2
h_3
h_{dh}
$\hat{y}_1$
$\hat{y}_{34}$
$\hat{y}_{42}$
$\hat{y}_{35102}$
$\hat{y}_{|V|}$
p(aardvark|...)
p(do|...)
p(fish|...)
p(zebra|...)
x
|V|×3
d×|V|
3d×1
d_h×3d
d_h×1
|V|×d_h
y
|V|×1
input layer
one-hot vectors
embedding layer
hidden layer
output layer
softmax