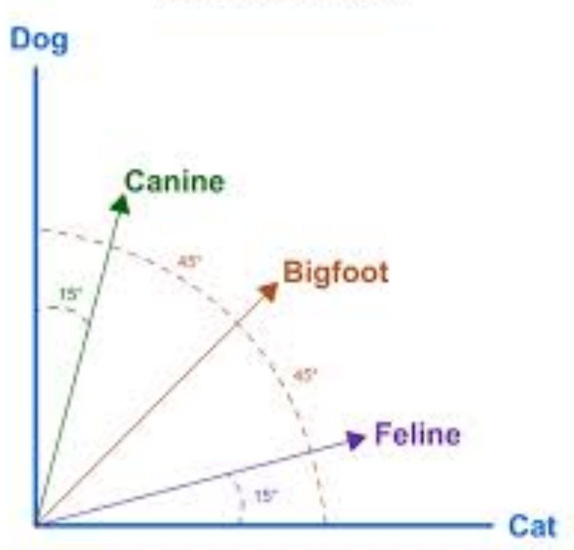
Distributional Semantics

COMP90042 Lecture 10



Lexical databases - Problems

- Manually constructed
  - Expensive
  - Human annotation can be biased and noisy
- Language is dynamic
  - New words: slang, terminology, etc.
  - New senses
- The Internet provides us with massive amounts of text. Can we use that to obtain word meanings?

Distributional semantics

- “You shall know a word by the company it keeps” (Firth)
- Document co-occurrence often indicative of topic (*document* as context)
  - E.g. *voting* and *politics*
- Local context reflects a word’s semantic class (*word window* as context)
  - E.g. *eat a pizza, eat a burger*

Guessing meaning from context

- Learn unknown word from its usage
- *E.g., tezgüino*

(14.1) A bottle of _____ is on the table.

(14.2) Everybody likes _____.

(14.3) Don't have _____ before you drive.

(14.4) We make _____ out of corn.

- Look at other words in same (or similar) contexts

	(14.1)	(14.2)	(14.3)	(14.4)	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
choices	0	1	0	0	
wine	1	1	1	0	

From E18, Ch14, originally Lin

Distributed and distributional semantics

Distributed = represented as a numerical vector (in contrast to *symbolic*)

Cover both count-based vs neural prediction-based methods

The Vector space model

- Fundamental idea: represent meaning as a vector
- Consider documents as context (ex: tweets)
- One matrix, two viewpoints
  - Documents represented by their words (web search)
  - Words represented by their documents (text analysis)

	…	state	fun	heaven	…
…					
425		0	1	0	
426		3	0	0	
427		0	0	0	
……					

Manipulating the VSM

- Weighting the values
- Creating low-dimensional dense vectors
- Comparing vectors

Tf-idf

- Standard weighting scheme for information retrieval
- Also discounts common words

$$idf_w = \log \frac{|D|}{df_w}$$

	…	the	country	hell	…
…					
425		43	5	1	
426		24	1	0	
427		37	0	3	
…					
df		500	14	7	

tf matrix

	…	the	countr y	hell	…
…					
425		0	25.8	6.2	
426		0	5.2	0	
427		0	0	18.5	
…					

tf-idf matrix

Dimensionality reduction

- Term-document matrices are very *sparse*
- Dimensionality reduction: create shorter, denser vectors
- More practical (less features)
- Remove noise (less overfitting)

Singular value Decomposition

$$A = U\Sigma V^T$$

$$\begin{array}{cc} & |\mathrm{D}| \\ |\mathrm{V}| & \begin{bmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \end{array}$$

A
(term-document matrix)

$m = Rank(A)$

U
(new term matrix)

Σ
(singular values)

V^T
(new document matrix)

$$\begin{array}{cc} & m \\ |\mathrm{V}| & \begin{bmatrix} 2.2 & 0.3 & & 8.7 \\ 5.5 & -2.8 & \cdots & 0.1 \\ -1.3 & 3.7 & & 3.5 \\ \vdots & & \ddots & \vdots \\ 2.9 & -2.1 & \cdots & -1.9 \end{bmatrix} \end{array} \quad \begin{array}{cc} & m \\ m & \begin{bmatrix} 9.1 & 0 & 0 & & 0 \\ 0 & 4.4 & 0 & \cdots & 0 \\ 0 & 0 & 2.3 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0.1 \end{bmatrix} \end{array} \quad \begin{array}{cc} & |\mathrm{D}| \\ m & \begin{bmatrix} -0.2 & 4.0 & & -1.3 \\ -4.1 & 0.6 & \cdots & -0.2 \\ 2.6 & 6.1 & & 1.4 \\ \vdots & & \ddots & \vdots \\ -1.9 & -1.8 & \cdots & 0.3 \end{bmatrix} \end{array}$$

Truncating – latent semantic analysis

- Truncating *U, Σ,* and *V* to *k* dimensions produces best possible *k* rank approximation of original matrix
- So truncated, U_k (or V_k^T) is a new low dimensional representation of the word (or document)
- Typical values for *k* are 100-5000

$$U \qquad\qquad\qquad\qquad\qquad U_k$$

$$\begin{array}{c|c} & \quad m \qquad\qquad k \\ |V| & \begin{bmatrix} 2.2 & 0.3 & & -2.4 & & 8.7 \\ 5.5 & -2.8 & \cdots & 1.1 & \cdots & 0.1 \\ -1.3 & 3.7 & & 4.7 & & 3.5 \\ \vdots & & \ddots & \vdots & \ddots & \vdots \\ 2.9 & -2.1 & \cdots & -3.3 & \cdots & -1.9 \end{bmatrix} \end{array} \Rightarrow \begin{array}{cc} & k \\ |V| & \begin{bmatrix} 2.2 & 0.3 & & -2.4 \\ 5.5 & -2.8 & \cdots & 1.1 \\ -1.3 & 3.7 & & 4.7 \\ \vdots & & \ddots & \vdots \\ 2.9 & -2.1 & \cdots & -3.3 \end{bmatrix} \end{array}$$

Words as context

- Lists how often words appear with other words
  - In some predefined context (usually a window)
- The obvious problem with raw frequency: dominated by common words

	…	the	country	hell	…
…					
state		1973	10	1	
fun		54	2	0	
heaven		55	1	3	
……					

Pointwise mutual information

For two events *x* and *y*, pointwise mutual information (PMI) comparison between the actual joint probability of the two events (as seen in the data) with the expected probability under the assumption of independence

$$PMI(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)}$$

Calculating PMI

	…	the	country	hell	…		Σ
…							
state		1973	10	1			12786
fun		54	2	0			633
heaven		55	1	3			627
…							
Σ		1047519	3617	780			15871304

$p(x,y) = \text{count}(x,y)/\Sigma$

$p(x) = \Sigma_x / \Sigma$

$p(y) = \Sigma_y / \Sigma$

x= state, y = country

$p(x,y) = 10/15871304 = 6.3 \times 10^{-7}$

$p(x) = 12786/15871304 = 8.0 \times 10^{-4}$

$p(y) = 3617/15871304 = 2.3 \times 10^{-4}$

$PMI(x,y) = \log_2(6.3 \times 10^{-7})/((8.0 \times 10^{-4})\ (2.3 \times 10^{-4}))$

$= 1.78$

PMI matrix

- PMI does a better job of capturing interesting semantics
  - E.g. *heaven* and *hell*
- But it is obviously biased towards rare words
- And doesn't handle zeros well

	…	the	country	hell	…
…					
state		1.22	1.78	0.63	
fun		0.37	3.79	-inf	
heaven		0.41	2.80	6.60	
……					

PMI tricks

- Zero all negative values (PPMI)
  - Avoid –inf and unreliable negative values
- Counter bias towards rare events
  - Smooth probabilities

Similarity

- Regardless of vector representation, classic use of vector is comparison with other vector
- For IR: find documents most similar to query
- For Text Analysis: find synonyms, based on proximity in vector space
  - automatic construction of lexical resources
  - more generally, knowledge base population
- Use vectors as features in classifier — more robust to different inputs (*movie* vs *film*)

Neural Word Embeddings

Learning **distributional** & **distributed** representations using "deep" classifiers

Skip-gram: Factored Prediction

- Neural network inspired approaches seek to learn vector representations of words and their contexts
- Key idea
  - *Word embeddings should be similar to embeddings of neighbouring words*
  - *And dissimilar to other words that don't occur nearby*
- Using vector dot product for vector 'comparison'
  - $u \cdot v = \sum_j u_j v_j$
- As part of a '*classifier*' over a word and its immediate context

Skip-gram: Factored Prediction

- Framed as learning a classifier (a weird language model)…
  - **Skip-gram**: predict words in local context surrounding given word

P(he | rests) P(in | rests)

P(life | rests) P(peace | rests)

… Bereft of life he **rests** in peace! If you hadn't nailed him …

P(rests | {he, in, life, peace})

  - **CBOW**: predict word in centre, given words in the local surrounding context

- Local context means words within L positions, L=2 above

Skip-gram model

- Generates each word in context given centre word

P(he | rests) P(in | rests)

P(life | rests) P(peace | rests)

… Bereft of life he **rests** in peace! If you hadn't nailed him …

- Total probability defined as $$\prod_{l \in -L, \ldots, -1, 1, \ldots, L} P(w_{t+l} | w_t)$$
  - Where subscript denotes position in running text
- Using a logistic regression model $$P(w_k | w_j) = \frac{\exp(c_{w_k} \cdot v_{w_j})}{\sum_{w' \in V} \exp(c_{w'} \cdot v_{w_j})}$$

Embedding parameterisation

- Two parameter matrices, with d-dimensional embedding for all words

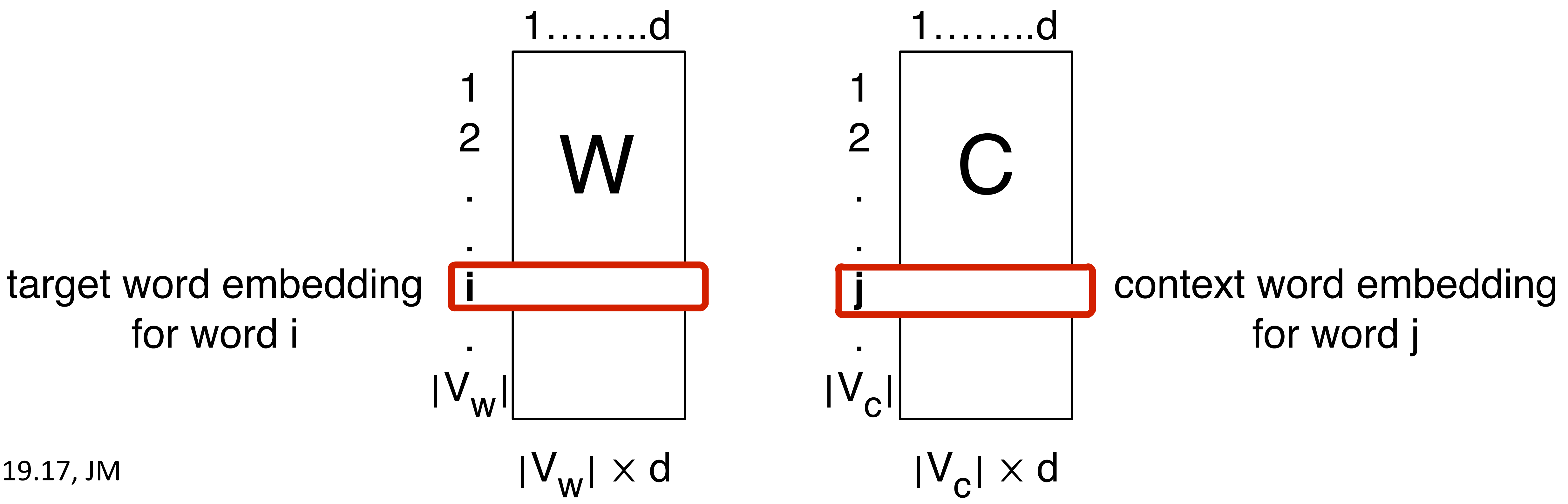


Fig 19.17, JM

- Words are numbered, e.g., by sorting vocabulary and using word location as its index

Skip-gram model

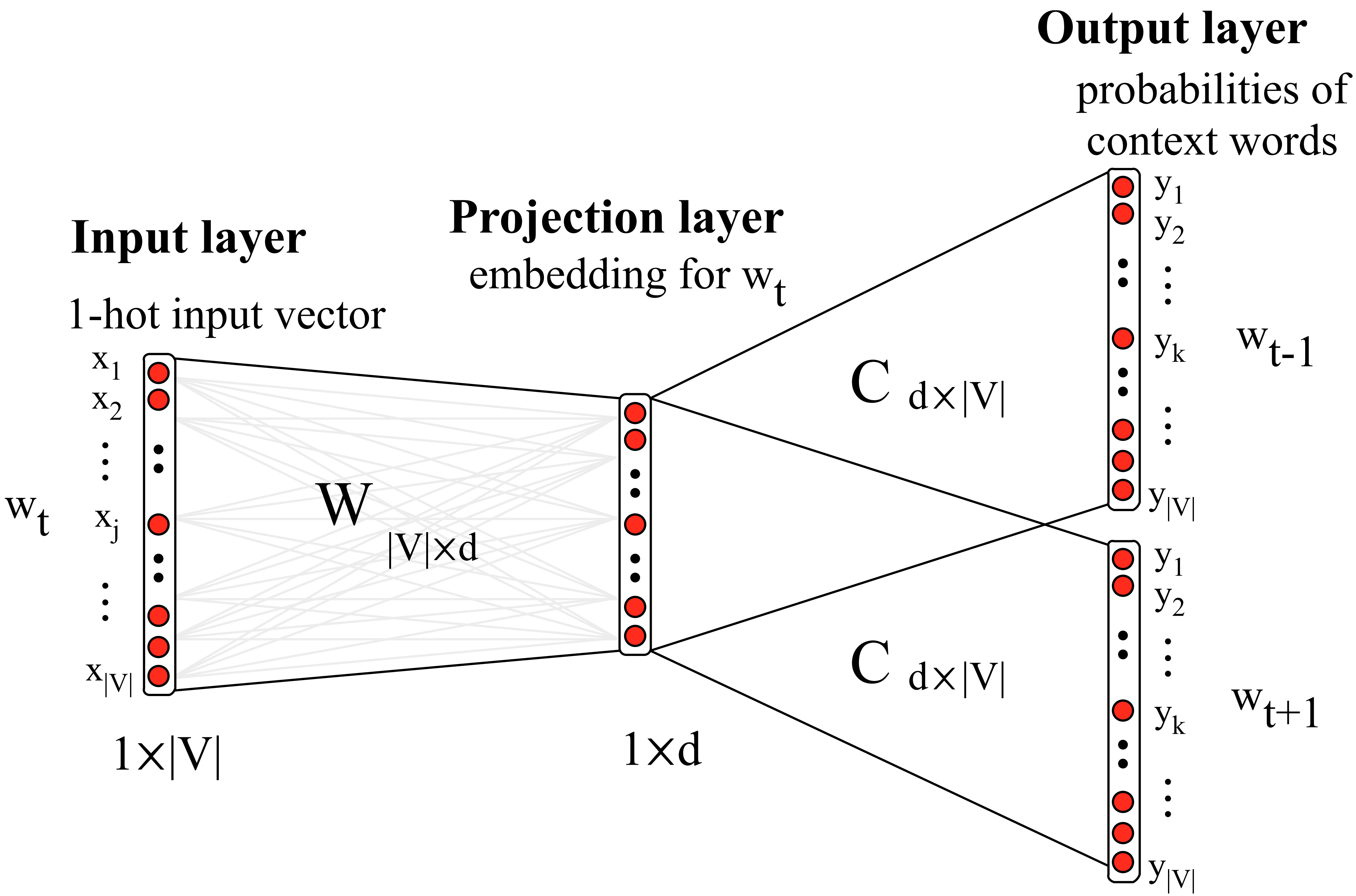


Fig 19.18, JM

Training the skip-gram model

- Train to *maximise likelihood* of **raw text**
- Too slow in practice, due to normalisation over |V|
- Reduce problem to binary classification, distinguish real context words from "negative samples"
  - words drawn randomly from V

Source: JM3 Ch 6

Negative Sampling

$$P(+|w_k, w_j) = \frac{1}{1 + \exp(-c_k \cdot v_j)}$$

$$P(-|w_k, w_j) = 1 - \frac{1}{1 + \exp(-c_k \cdot v_j)}$$

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

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

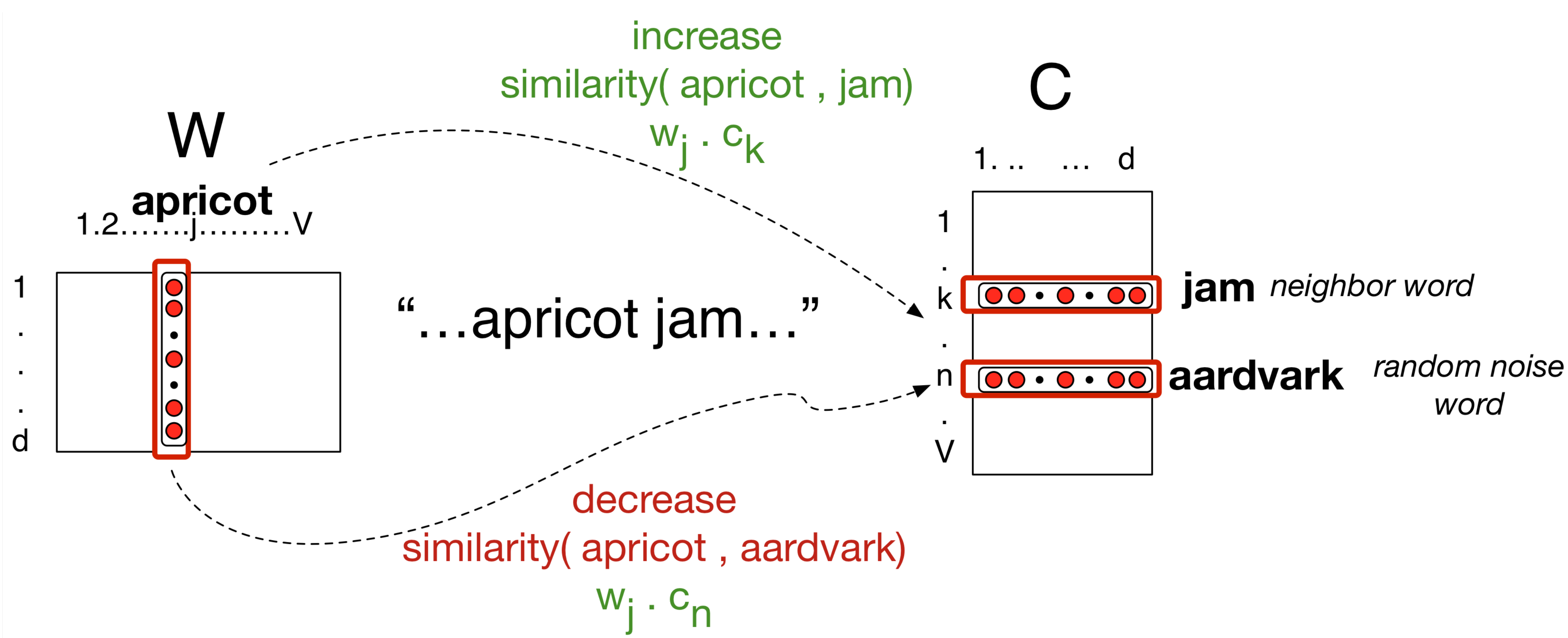
negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Source: JM3 Ch 6; note mistake from reference, corrected above in red

Training illustration

- Iterative process (stochastic gradient descent)
  - each step moves embeddings closer for context words
  - and moves embeddings apart for noise samples

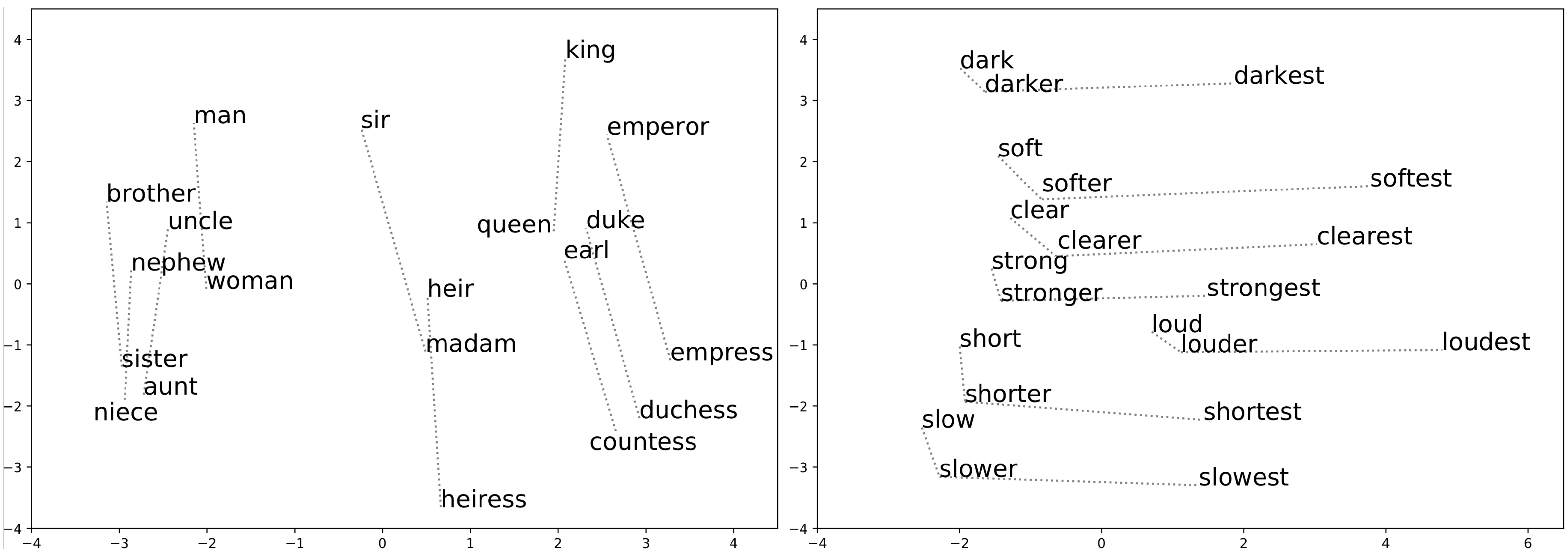


Source: JM3 Ch 6

Evaluating word vectors

- Lexicon style tasks
  - *WordSim-353* are pairs of nouns with judged relatedness
  - *SimLex-999* also covers verbs and adjectives
  - *TOEFL* asks for closest synonym as multiple choice
  - …
- Test compatibility of word pairs using cosine similarity in vector space

Embeddings exhibit meaningful geometry



- Word analogy task
  - Man is to King as Woman is to ???
  - France is to Paris as Italy is to ???
  - Evaluate where in the ranked predictions the correct answer is, given tables of known relations

Evaluating word vectors

- Best evaluation is in other downstream tasks
  - Use bag-of-word embeddings as a feature representation in a classifier (e.g., sentiment, QA, tagging etc.)
  - First layer of most deep learning models is to embed input text; use pre-trained word vectors as embeddings, possibly with further training ("fine-tuning") for specific task
- Recently "contextual word vectors" shown to work even better, ELMO (AI^2), BERT (Google AI), …

Pointers to software

- Word2Vec
  - C implementation of Skip-gram and CBOW
    https://code.google.com/archive/p/word2vec/
- GenSim
  - Python library with many methods include LSI, topic models and Skipgram/CBOW
    https://radimrehurek.com/gensim/index.html
- GLOVE
  - http://nlp.stanford.edu/projects/glove/

Further reading

- Either one of:
  - E18, 14-14.6 (skipping 14.4)
  - JM3, Ch 6