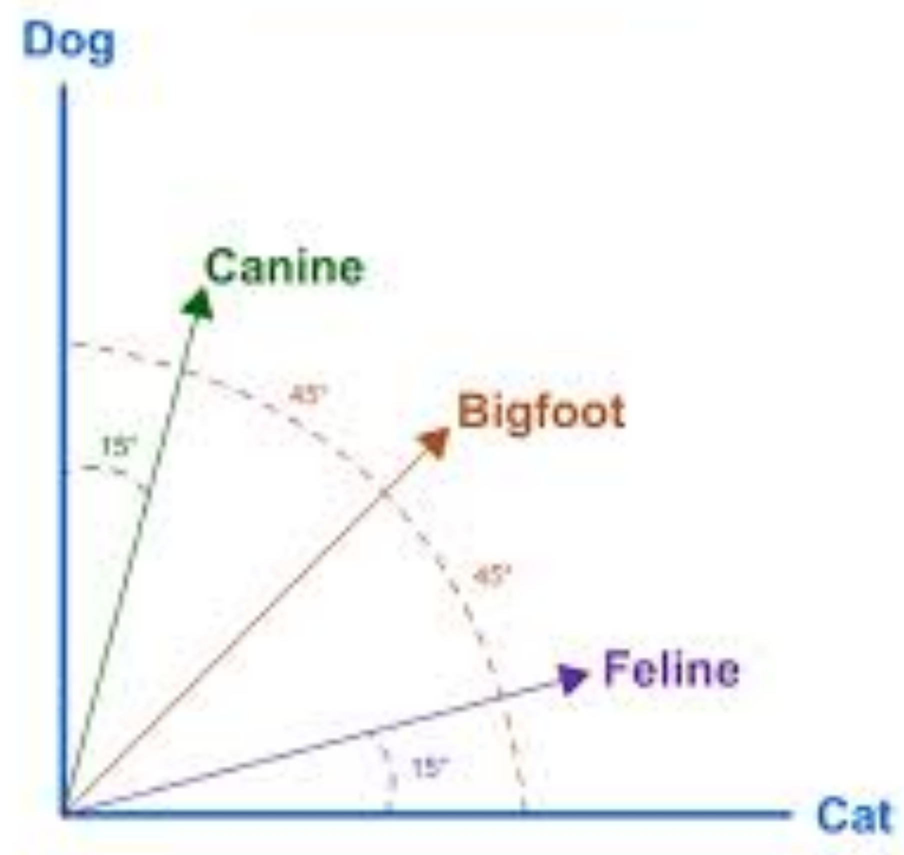


COMP90042 LECTURE 4

DISTRIBUTIONAL SEMANTICS

LEXICAL DATABASES - PROBLEMS

- Manually constructed
  - Expensive
  - Human annotation can be biased and noisy
- Language is dynamic
  - New words: slangs, 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*
- Two approaches:
  - Count-based (Vector Space Models)
  - Prediction-based

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

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

tf matrix

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

tf-idf matrix

	…	the	country	hell	…
…					
425		0	25.8	6.2	
426		0	5.2	0	
427		0	0	18.5	
…					

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

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

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

A

(term-document matrix)

$m = Rank(A)$

U

(new term matrix)

Σ

(singular values)

V^T

(new document matrix)

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

TRUNCATING – LATENT SEMANTIC ANALYSIS

- Truncating U, Σ, and V^T 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

$$U \qquad m \quad | \quad k$$

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

$$U_k \qquad k$$

WORDS AS CONTEXT

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

- 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

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

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

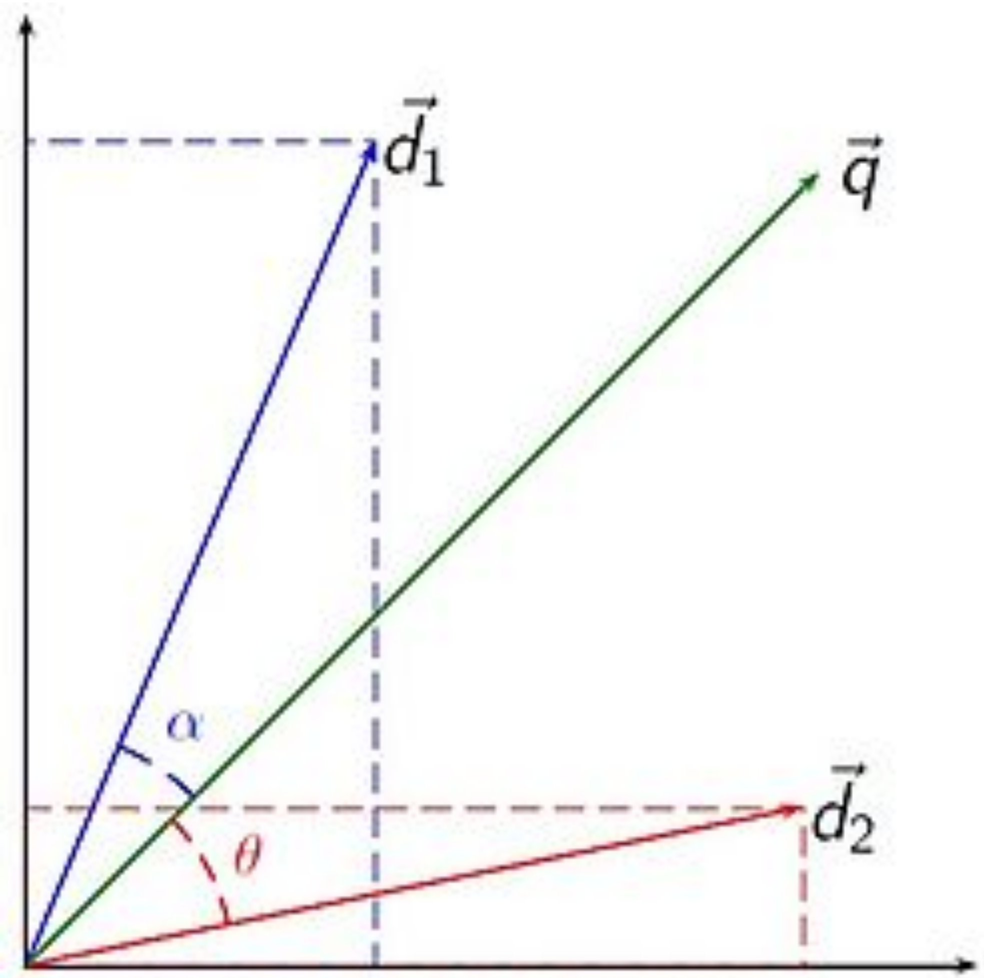
- PMI does a better job of capturing interesting semantics
  - E.g. *heaven* and *hell*
- But it is obviously biased towards rare words
- And doesn't handle zeros well

PMI TRICKS

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

SIMILARITY

- Regardless of vector representation, classic use of vector is comparison with other vector
  - Though vectors can also be used directly as features
- For IR: find documents most similar to query



COSINE SIMILARITY

The cosine of the angle between two vectors is the dot product of the two vectors divided by the product of their norms:

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

Where

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{N} a_i b_i$$

And

$$|\vec{a}| = \sqrt{\sum_{i=1}^{N} {a_i}^2}$$

COSINE SIMILARITY EXAMPLE

$$\vec{a} = [0,0,1,0,1,1,1]$$

$$\vec{b} = [0,1,1,0,1,1,0]$$

$$\vec{a} \cdot \vec{b} = 1 + 1 + 1 = 3$$

$$|\vec{a}| = \left|\vec{b}\right| = \sqrt{1 + 1 + 1 + 1} = 2$$

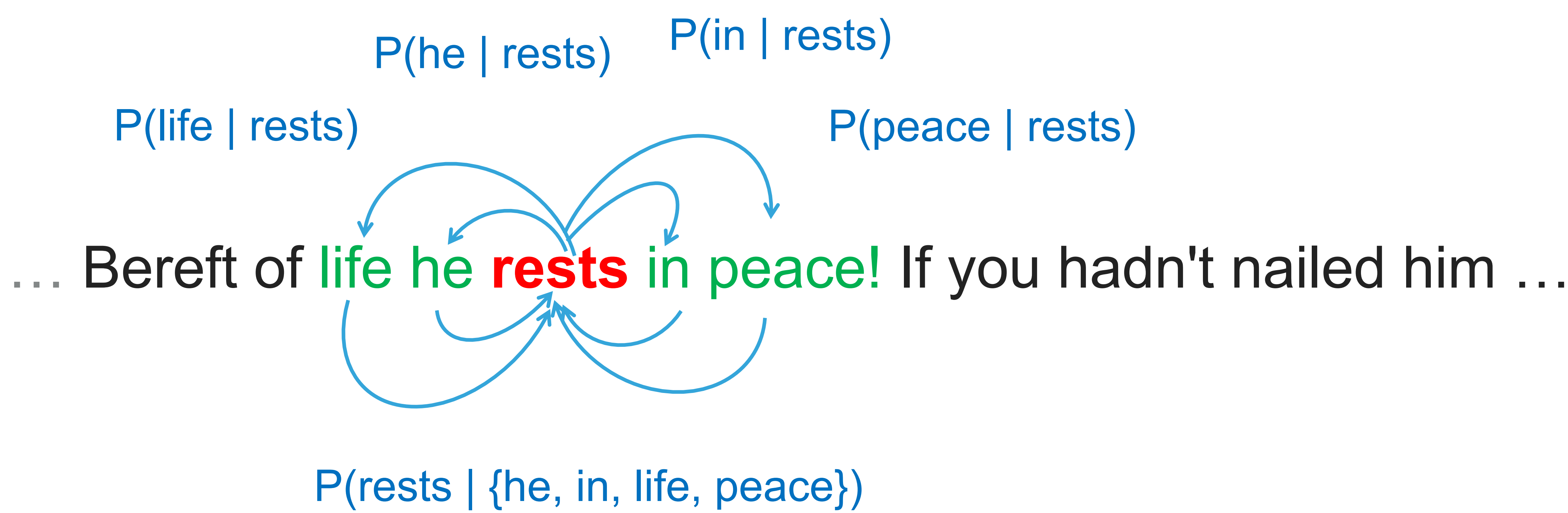
$$\cos\theta = \frac{3}{2*2} = 0.75$$

EMBEDDINGS FROM PREDICTIONS

- 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

EMBEDDINGS FROM PREDICTIONS

- Framed as learning a classifier…
  - Skip-gram: predict words in local context surrounding given word



  - CBOW: predict word in centre, given words in the local surrounding context
- Local context means words within L positions, e.g., L=2

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
- For each word, $$P(w_k|w_j) = \frac{\exp(c_{w_k} \cdot v_{w_j})}{\sum_{w' \in V} \exp(c_{w'} \cdot v_{w_j})}$$

EMBEDDING PARAMETERISATION

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

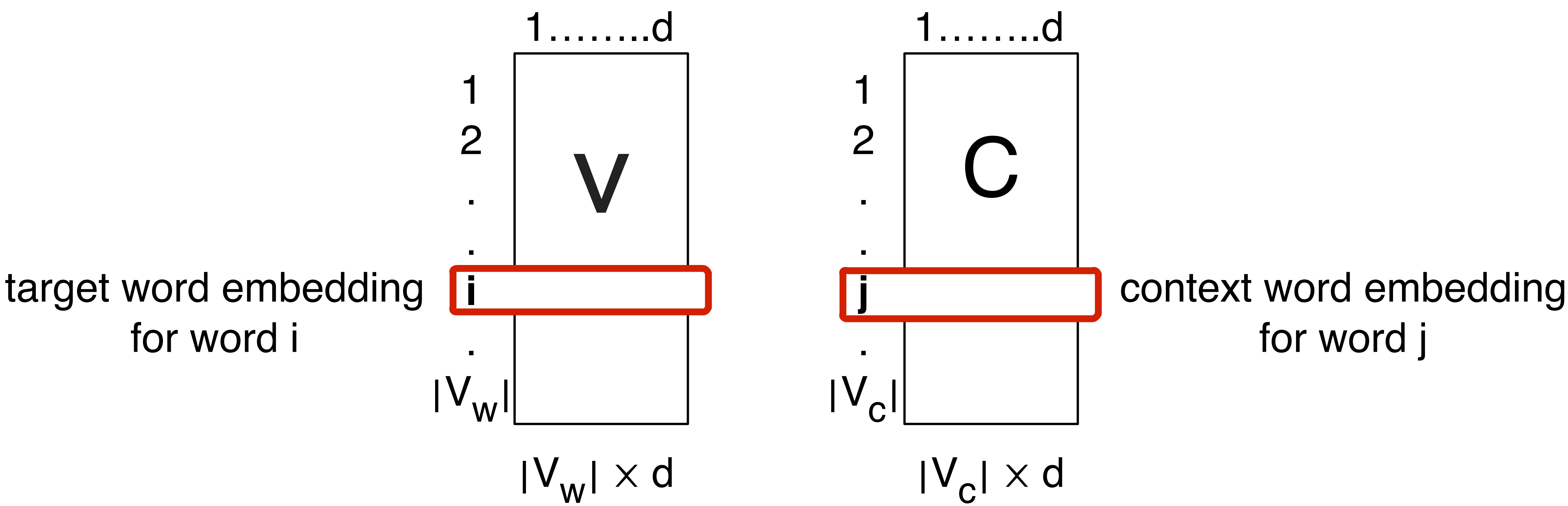


Fig 19.17, JM

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

ONE-HOT VECTORS AND EMBEDDINGS

- Words are integer numbers, e.g., "cat" = 17235^{th} word
  - The embeddings for "cat" are then:
    - V_{17235} = [1.23, 0.8, -0.15, 0.7, 1.1, -1.3, …] (d-dim. vector)
    - C_{17235} = [0.32, 0.1, 0.27, 2.5, -0.1, 0.45, …] (d-dim. vector)
  - Using a separate embedding for "cat" appearing in the centre and appearing in the context of another word
- A "one-hot vector" is all 0s, with a single 1 at index i
  - E.g., x = "cat" = [0,0,0, …, 0,1,0, …, 0]
    where index 17235 is set to 1, all other V-1 entries are 0
  - This allows us to write $V_{\text{"cat"}}$ as V x

SKIP-GRAM MODEL

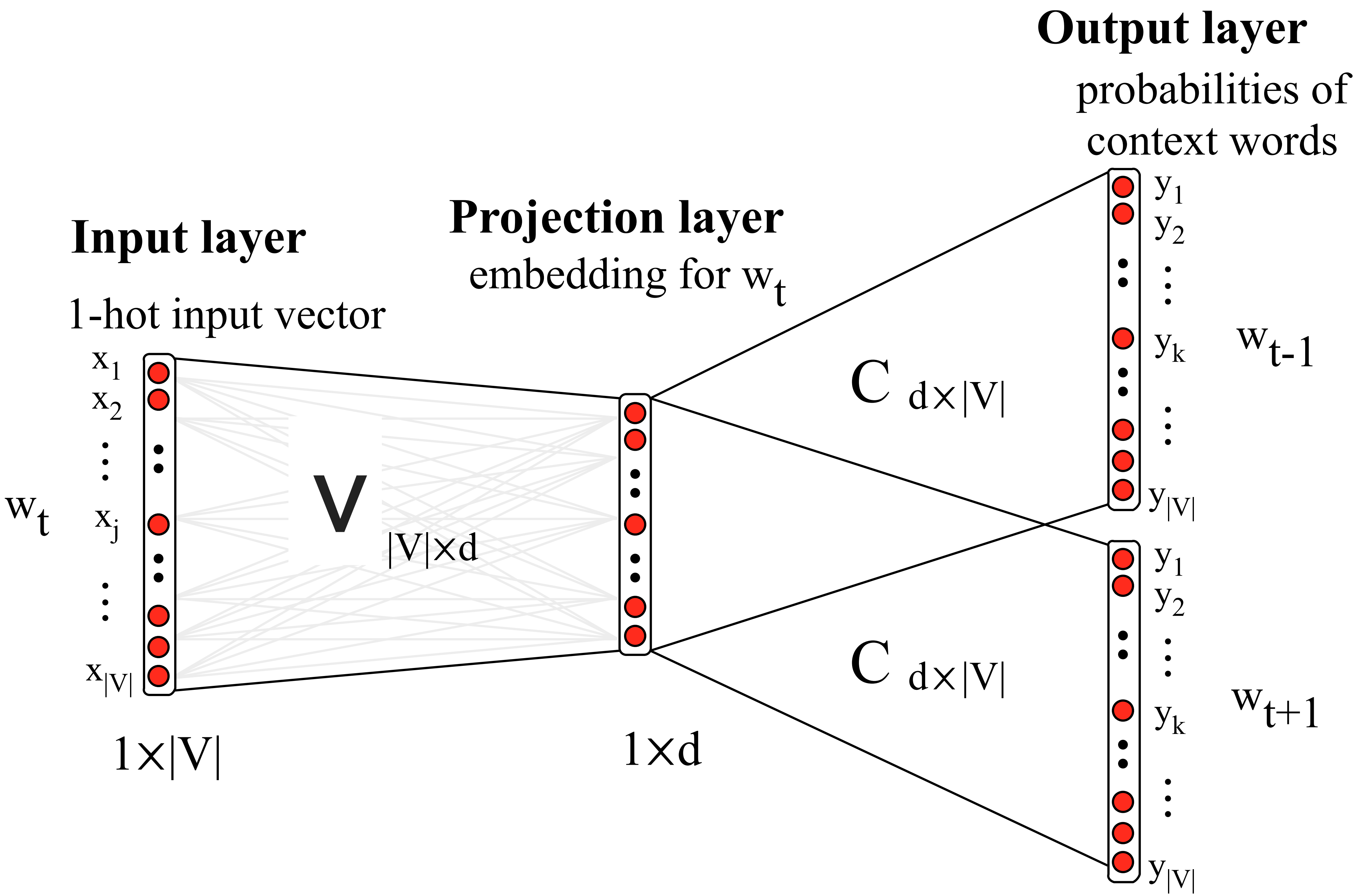


Fig 19.18, JM

SKIP-GRAM COMPONENTS

1. Lookup embeddings from W for centre word
   - $v_j = V\,x$
2. Compute the dot product with all possible context words
   - $v_j \,.\, c_k$ for all possible words in the vocabulary $k \in V$
3. Normalise output vector to ensure values are positive and sum to one
   - Softmax transformation

$$\mathbf{z} \rightarrow \left\{ \frac{\exp z_i}{\sum_i \exp z_i} \right\}_i$$

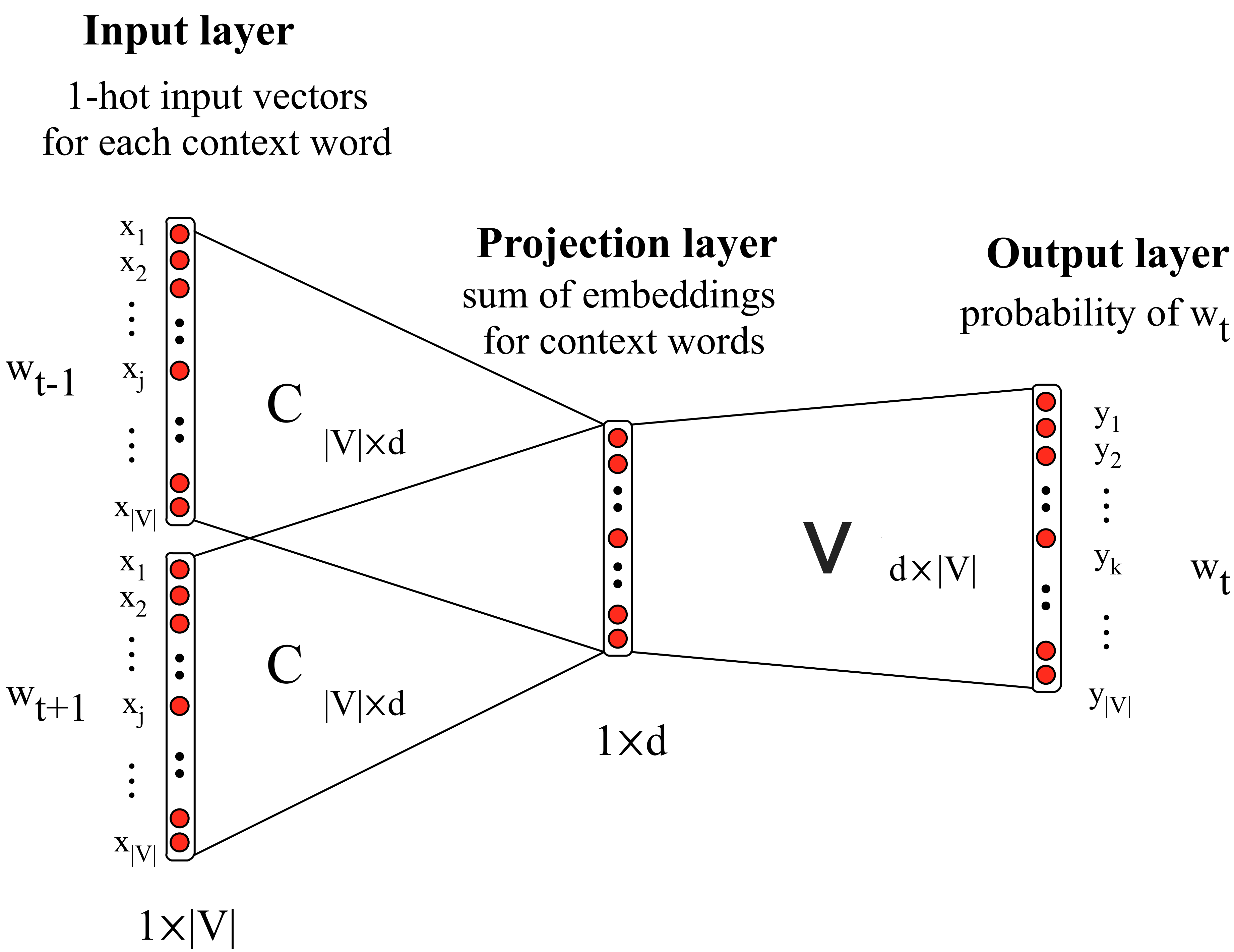
These values can now be considered probabilities; hope that

- Prob for observed context words > Prob other words.

TRAINING THE SKIP-GRAM MODEL

- Only data requirement is raw text
- Train to *maximise likelihood* of the text, using gradient descent
  - But is too slow: to obtain the probability of a word need to sum over the entire vocabulary
- Alternative: *negative sampling*. Reduced the problem to binary classification: predict the correct sample instead of a random one.
  - More efficient than MLE

CBOW ARCHITECTURE



PROPERTIES

- Skip-gram and CBOW both perform fairly well
  - No clear reason to prefer one over another, choice is task dependent
- Very fast to train using negative sampling approximation
- In fact Skip-gram with negative sampling related to LSA
  - Can be viewed as factorisation of the PMI matrix over words and their contexts
  - See Levy and Goldberg (2014) for details

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
  - …
- 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

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

- JM3, Ch 15, JM3 Ch 16.1 – 16.2
- **Optional:**
  - From Frequency to Meaning: Vector Space Models of Semantics
  - Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. NIPS 2013. *Distributed representations of words and phrases and their compositionality*.
  - J. Pennington, R. Socher, and C. D. Manning. EMNLP 2014. *GloVe: Global Vectors for Word Representation*.
  - O. Levy, and Y. Goldberg. NIPS 2014, *Neural word embedding as implicit matrix factorization*.