

# Distributional Semantics

COMP90042

Natural Language Processing

Lecture 10

Semester 1 2022 Week 5  
Jey Han Lau



THE UNIVERSITY OF  
MELBOURNE

# 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 Hypothesis

- *You shall know a word by the company it keeps*  
— Firth, 1957
- Document co-occurrence indicative of topic  
(**document** as context)
  - E.g. *voting* and *politics*
- Local context reflects its meaning  
(**word window** as context)
  - E.g. *eat a pizza* vs. *eat a burger*

# Guessing Meaning from Context

- Learn unknown word from its usage

- 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.

- Another way: look at words that share similar contexts!

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

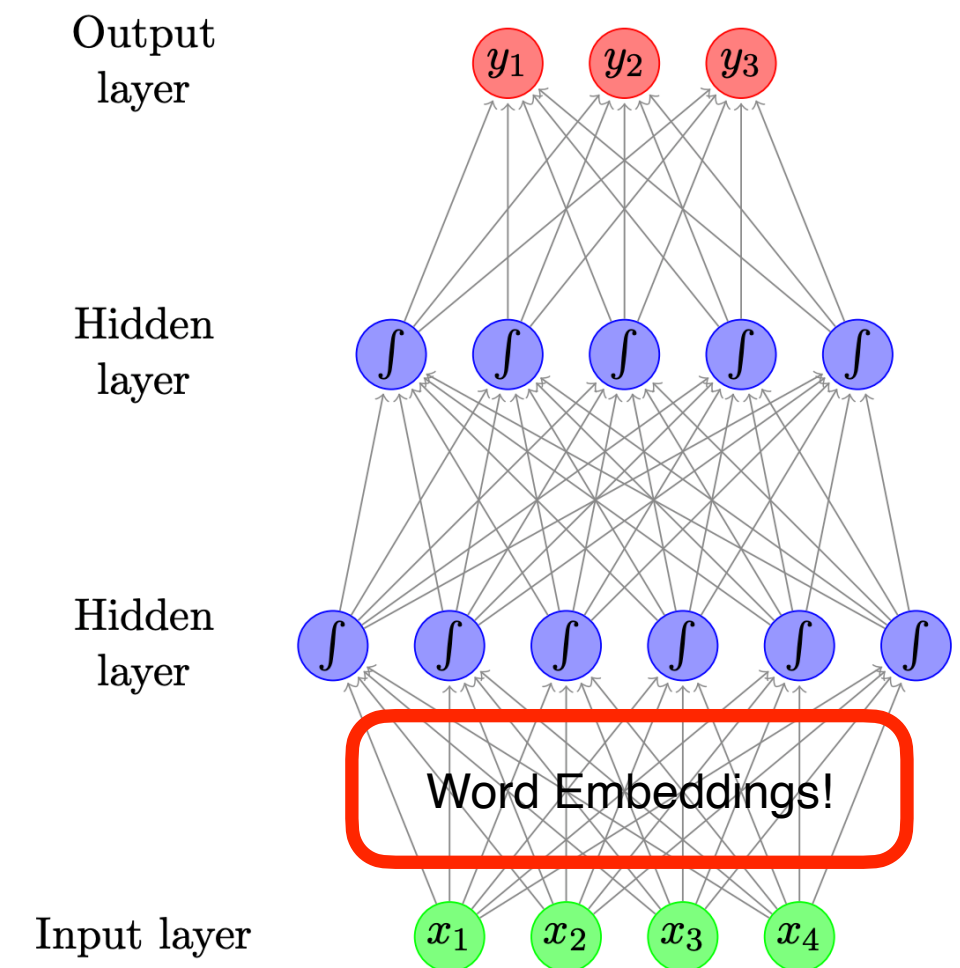
# Word Vectors

	(14.1)	(14.2)	(14.3)	(14.4)	...
<i>tezgiino</i>	1	1	1	1	
<i>loud</i>	0	0	0	0	
<i>motor oil</i>	1	0	0	1	
<i>tortillas</i>	0	1	0	1	
<i>choices</i>	0	1	0	0	
<i>wine</i>	1	1	1	0	

- Each row can be thought of a **word vector**
- It describes the distributional properties of a word
  - i.e. encodes information about its context words
- Capture all sorts of semantic relationships (synonymy, analogy, etc)

# Word Embeddings?

- We've seen word vectors before: word embeddings!
- Here we will learn other ways to produce word vectors
  - Count-based methods
  - More efficient neural methods designed just for learning word vectors



# Outline

- Count-based methods
- Neural methods
- Evaluation

# Count-Based Methods



# Learning Count-Based Word Vectors

- Generally two flavours
  - Use document as context
  - Use neighbouring words as context

# Document as Context: The Vector Space Model

- Core idea: represent word meaning as a vector
- Consider documents as context
- One matrix, two viewpoints
  - Documents represented by their words
  - Words represented by their documents

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

# Manipulating the VSM

- Weighting the values (beyond frequency)
- Creating low-dimensional dense vectors

# Tf-idf

- Standard weighting scheme for information retrieval

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

$|D|$  ← total #docs  
 $df_w$  ← #docs that has w

- Discounts common words!

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

*tf* matrix

	...	the	country	hell	...
...					
425		0	26.0	6.2	
426		0	5.2	0	
427		0	0	18.6	
...					

*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

|D|

|V|

$$\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

$A$   
(term-document matrix)

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



$U$   
(new term matrix)

$m$

|V|

$$\begin{bmatrix} 2.2 & 0.3 & \dots & 8.7 \\ 5.5 & -2.8 & \dots & 0.1 \\ -1.3 & 3.7 & \dots & 3.5 \\ \vdots & & \ddots & \vdots \\ 2.9 & -2.1 & \dots & -1.9 \end{bmatrix}$$

$\Sigma$   
(singular values)

$m$

$$\begin{bmatrix} 9.1 & 0 & 0 & \dots & 0 \\ 0 & 4.4 & 0 & \dots & 0 \\ 0 & 0 & 2.3 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0.1 \end{bmatrix}$$

$V^T$   
(new document matrix)

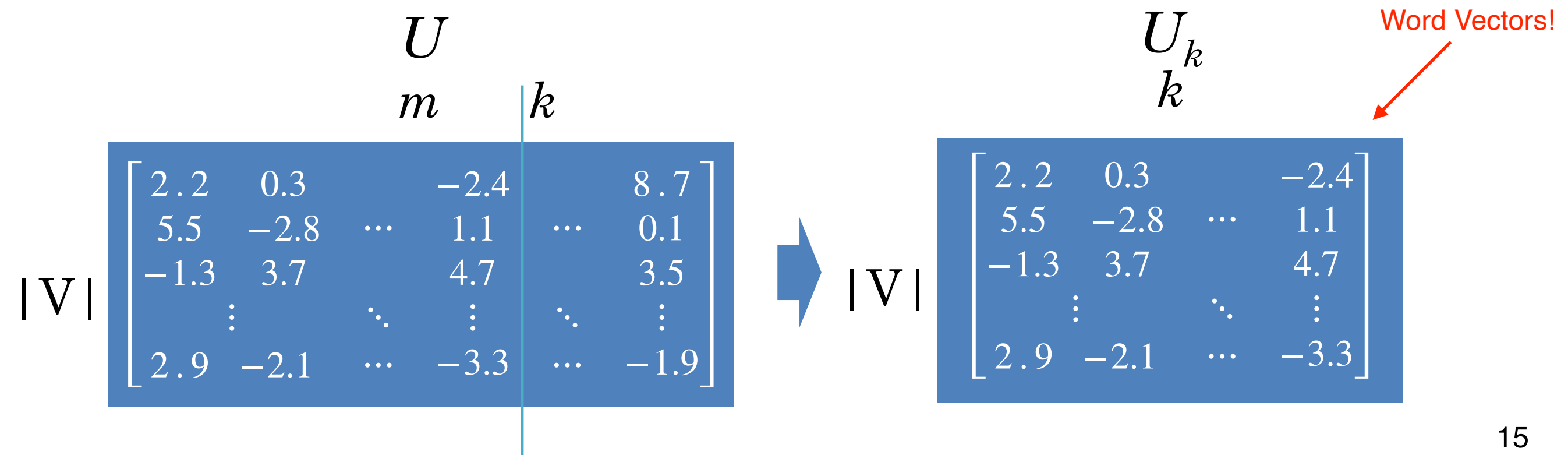
|D|

$$\begin{bmatrix} -0.2 & 4.0 & \dots & -1.3 \\ -4.1 & 0.6 & \dots & -0.2 \\ 2.6 & 6.1 & \dots & 1.4 \\ \vdots & & \ddots & \vdots \\ -1.9 & -1.8 & \dots & 0.3 \end{bmatrix}$$

$$m = \text{Rank}(A)$$

# Truncating – Latent Semantic Analysis

- Truncating  $U$ ,  $\Sigma$ , and  $V$  to  $k$  dimensions produces best possible  $k$  rank approximation of original matrix
- $U_k$  is a new low dimensional representation of words
- Typical values for  $k$  are 100-5000



# Words as Context

- Lists how often words appear with other words
  - In some predefined context (e.g. a window of N words)
- The obvious problem with raw frequency:  
dominated by common words
  - But we can't use tf-idf!

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



# Pointwise Mutual Information

- For two events  $x$  and  $y$ , PMI computes the discrepancy between:
  - Their joint distribution =  $P(x, y)$
  - Their individual distributions (assuming independence) =  $P(x)P(y)$

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

# Calculating PMI

	...	the	country	hell	...		$\Sigma$
...							
state		1973	10	1			12786
fun		54	2	0			633
heaven		55	1	3			627
...							
$\Sigma$		1047519	3617	780			15871304

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

$$P(x, y) = \frac{\text{count}(x, y)}{\Sigma}$$

$$P(x) = \frac{\Sigma_x}{\Sigma}$$

$$P(y) = \frac{\Sigma_y}{\Sigma}$$

$x = \text{state}, y = \text{country}$

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

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

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

$$\text{PMI}(x, y) = \log_2 \frac{6.3 \times 10^{-7}}{(8.0 \times 10^{-4})(2.3 \times 10^{-4})} = 1.78$$

# PMI(heaven, hell)?

	...	the	country	hell	...		$\Sigma$
...							
state		1973	10	1			12786
fun		54	2	0			633
heaven		55	1	3			627
...							
$\Sigma$		1047519	3617	780			15871304

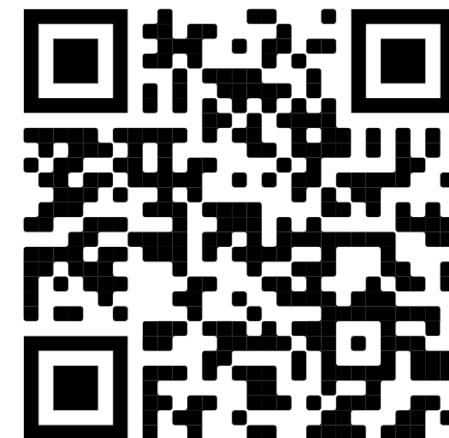
$$\text{PMI}(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

$$P(x, y) = \frac{\text{count}(x, y)}{\Sigma}$$

$$P(x) = \frac{\Sigma_x}{\Sigma}$$

$$P(y) = \frac{\Sigma_y}{\Sigma}$$

[PollEv.com/jeyhanlau569](http://PollEv.com/jeyhanlau569)



# PMI Matrix

- PMI does a better job of capturing semantics
  - E.g. *heaven* and *hell*
- But very biased towards rare word pairs
- 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	6.61	
.....					

# PMI Tricks

- Zero all negative values (Positive PMI)
  - Avoid  $-\infty$  and unreliable negative values
- Counter bias towards rare events

$$\text{Normalised PMI} \left( \frac{\text{PMI}(x, y)}{-\log P(x, y)} \right)$$

$$\text{SVD } (A = U\Sigma V^T)$$

	...	the	country	hell	...
...					
425		0	26.0	6.2	
426		0	5.2	0	
427		0	0	18.6	
...					

*tf-idf* matrix

	...	the	country	hell	...
...					
state		1.22	1.78	0.63	
fun		0.37	3.79	0	
heaven		0.41	2.80	6.60	
.....					

PPMI matrix

- Regardless of whether we use document or word as context, SVD can be applied to create dense vectors

# Neural Methods

# Word Embeddings

- We've seen word embeddings used in neural networks (lecture 7 and 8)
- But these models are designed for other tasks:
  - Classification
  - Language modelling
- Word embeddings are just a **by-product**



# Neural Models for Embeddings

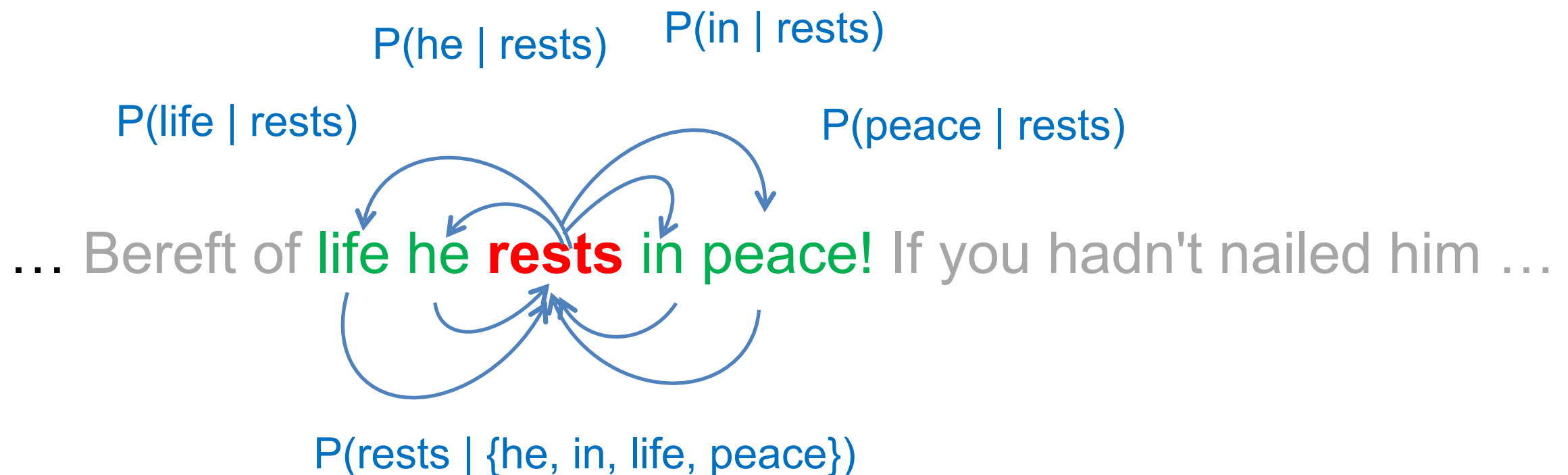
- Can we design neural networks whose goal is to purely learn word embeddings?
- Desiderata:
  - Unsupervised
  - Efficient

# Word2Vec

- Core idea
  - *You shall know a word by the company it keeps*
  - Predict a word using context words

# Word2Vec

- Framed as learning a classifier
  - **Skip-gram**: predict surrounding words of target word



- **CBOW**: predict target word using surrounding words
- Use surrounding words within L positions, L=2 above

# Skip-gram Model

- Predicts each neighbouring word given target word

$P(\text{he} \mid \text{rests})$     $P(\text{in} \mid \text{rests})$   
 $P(\text{life} \mid \text{rests})$     $P(\text{peace} \mid \text{rests})$

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

- Total probability defined as

$$\prod_{l \in -L, \dots, -1, 1, \dots, L} P(w_{t+l} \mid w_t)$$

- Using a logistic regression model

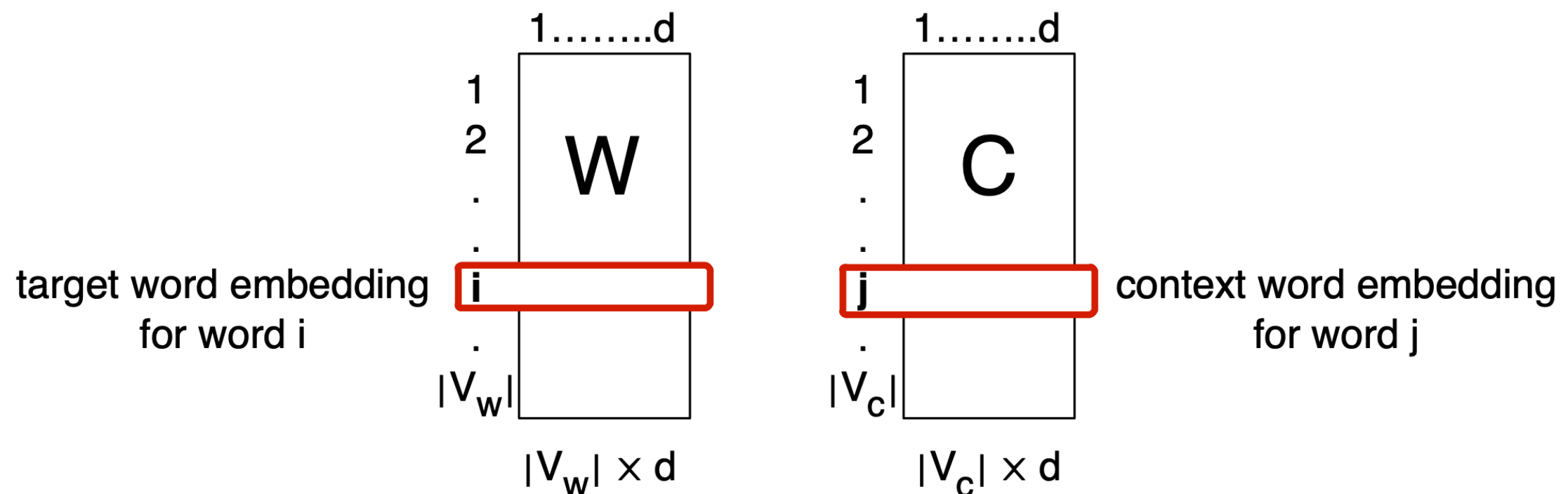
$$P(\text{life} \mid \text{rests}) = \frac{\exp(W_{\text{rests}} \cdot C_{\text{life}})}{\sum_{u \in V} \exp(W_{\text{rests}} \cdot C_u)}$$

word embedding of *rests*  
 word embedding of *life*  
 dot product

# Embedding parameterisation

$$P(\text{life} \mid \text{rests}) = \frac{\exp(W_{\text{rests}} \cdot C_{\text{life}})}{\sum_{u \in V} \exp(W_{\text{rests}} \cdot C_u)}$$

- Two word embedding matrices ( $W$  and  $C$ )!



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

# Skip-gram model

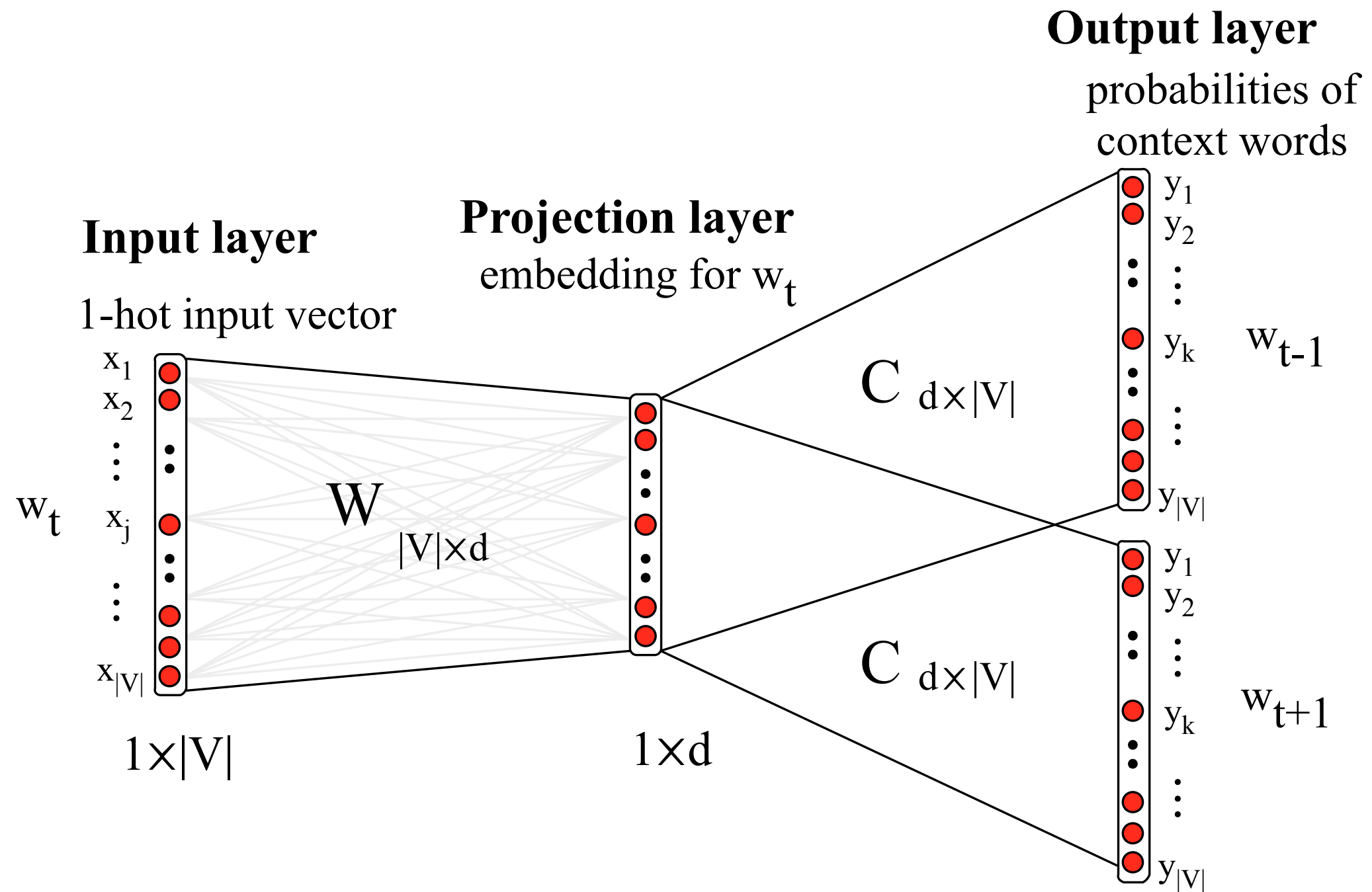


Fig 19.18, JM3

# Training the skip-gram model

- Train to maximise likelihood of **raw text**
- Too slow in practice, due to normalisation over  $|V|$

$$P(\text{life} \mid \text{rests}) = \frac{\exp(W_{\text{rests}} \cdot C_{\text{life}})}{\sum_{u \in V} \exp(W_{\text{rests}} \cdot C_u)}$$

- Reduce problem to binary classification
  - (*life*, *rests*) → real context word
  - (*aardvark*, *rests*) → non-context word
  - How to draw non-context word or **negative samples**?
  - Randomly from  $V$





# Skip-gram Loss

$$L(\theta) = \sum_{(t,c) \in +} \log P(+|t, c) + \sum_{(t,c) \in -} \log P(-|t, c)$$

- In practice, use  $k$  negative examples

$$L(\theta) = \log P(+|t, c) + \sum_{i=1}^k \log P(-|t, n_i)$$

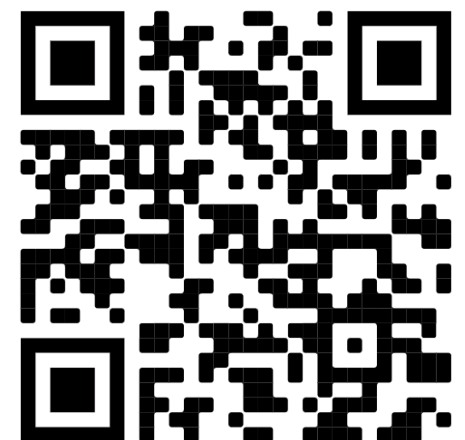
# Desiderata

- Unsupervised
  - Unlabelled corpus
- Efficient
  - Negative sampling (avoid softmax over full vocabulary)
  - Scales to very very large corpus

# Problems with word vectors/embeddings (count and neural methods)?

- Difficult to quantify the quality of word vectors
- Don't capture polysemous words
- Data sparsity (not enough data to learn them)
- Difficult to incorporate them in downstream tasks

[PollEv.com/jeyhanlau569](https://PollEv.com/jeyhanlau569)



# Evaluation

# Word Similarity

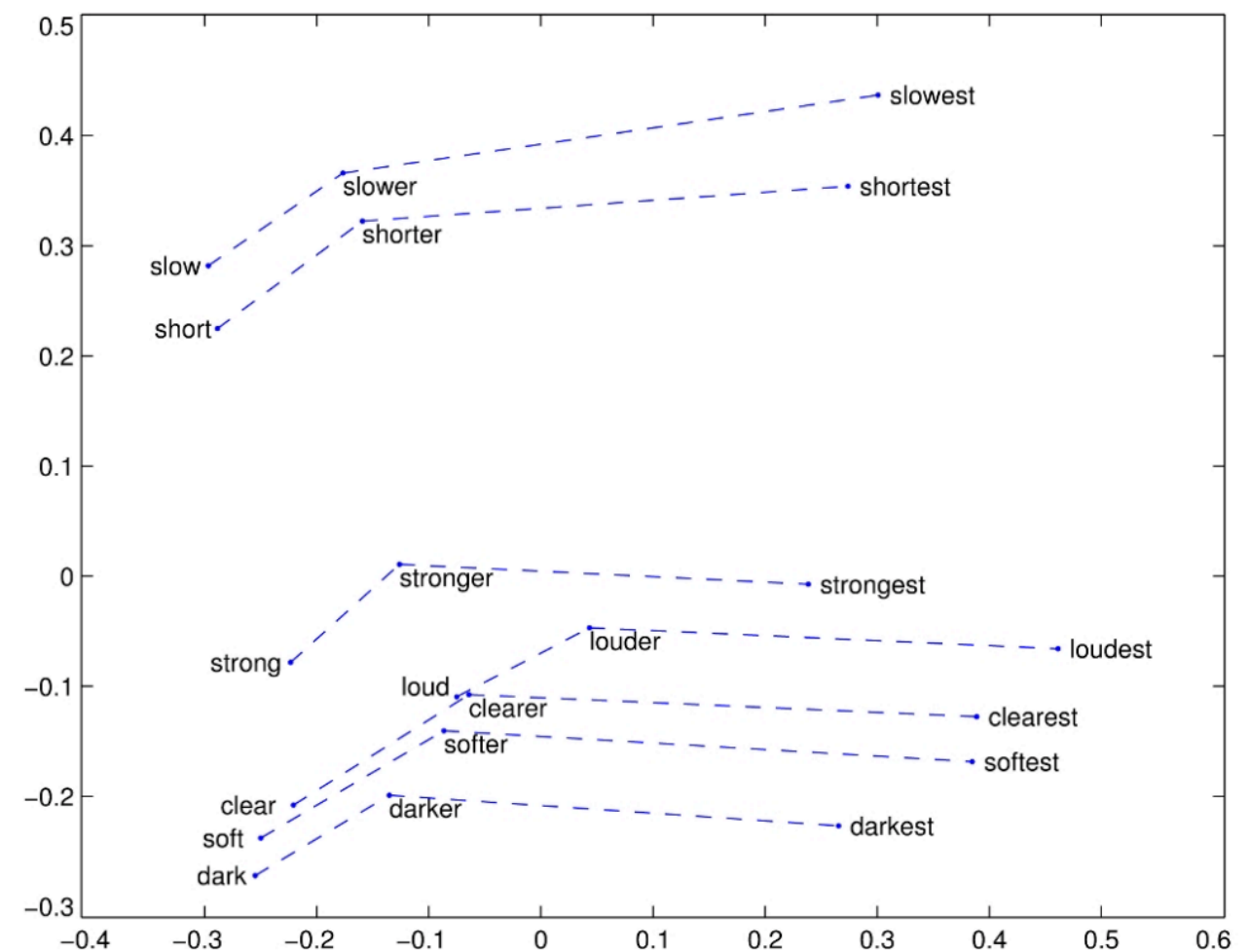
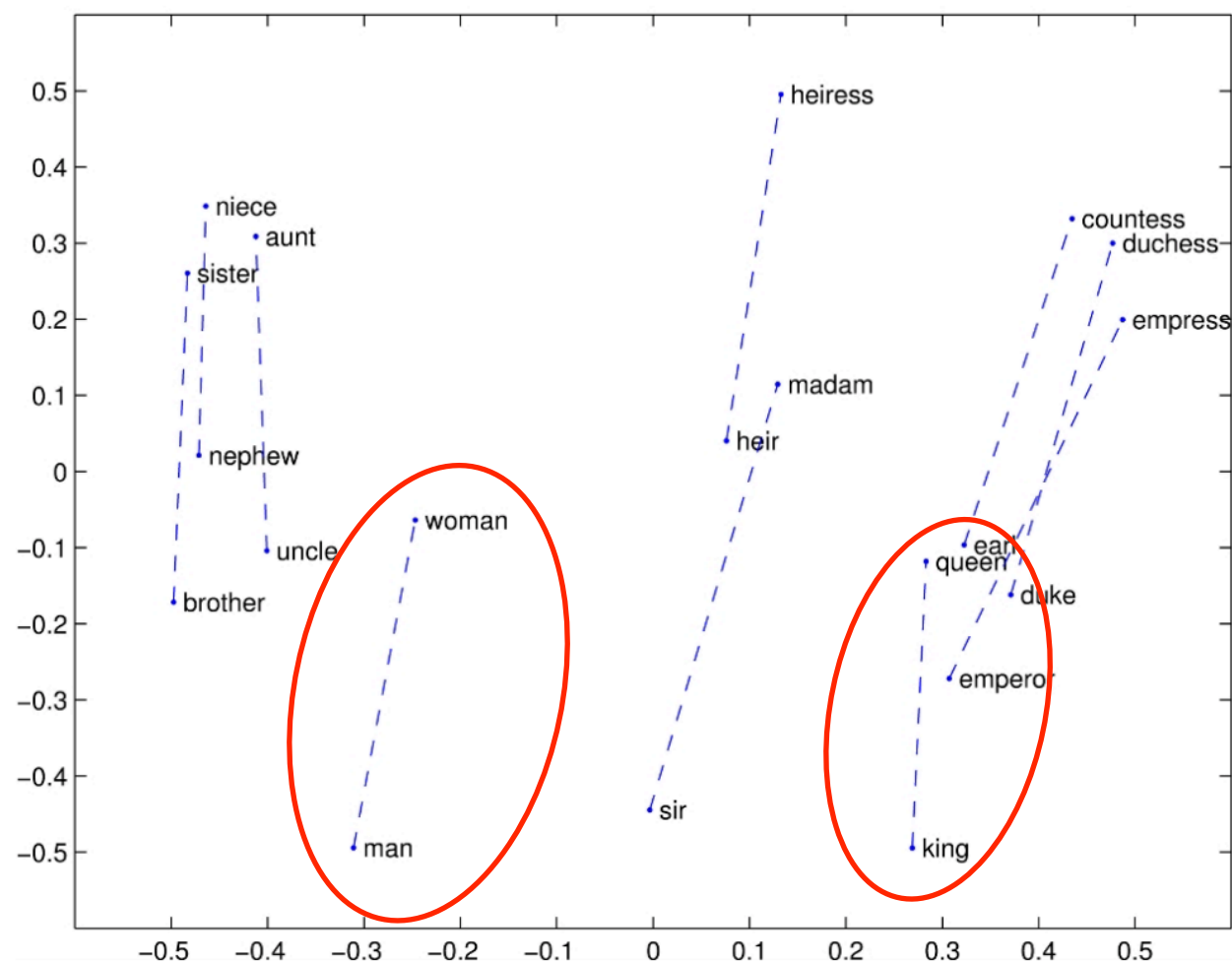
- Measure similarity of two words using cosine similarity
- Compare predicted similarity with human intuition
- Datasets
  - *WordSim-353* are pairs of nouns with judged relatedness
  - *SimLex-999* also covers verbs and adjectives

# Word Analogy

- Man is to King as Woman is to ???
- $v(\text{Man}) - v(\text{King}) = v(\text{Woman}) - v(\text{???})$
- $v(\text{???}) = v(\text{Woman}) - v(\text{Man}) + v(\text{King})$
- Find word whose embedding is closest to  $v(\text{Woman}) - v(\text{Man}) + v(\text{King})$

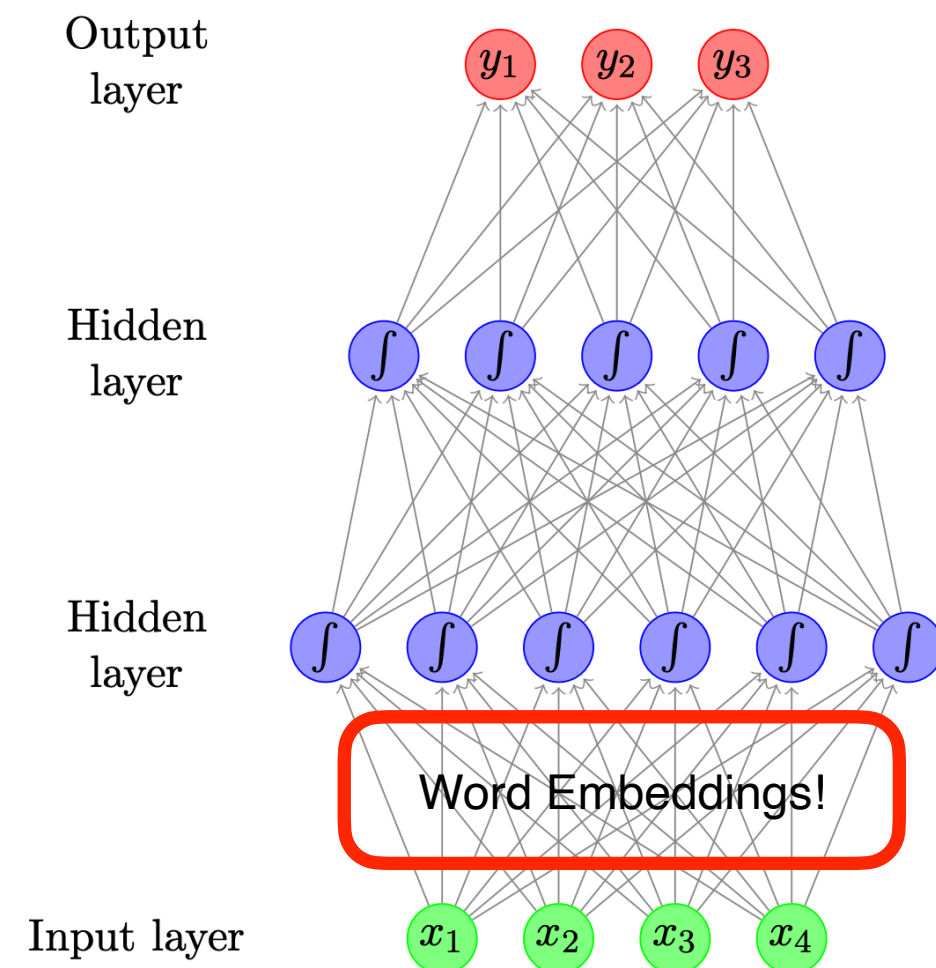
# Embedding Space

- Word2Vec embeddings show interesting geometry
- Explains why they are good in word analogy task



# Downstream Tasks

- Best evaluation is in other downstream tasks
  - Use bag-of-word embeddings as a feature representation in a classifier
  - First layer of most deep learning models is to embed input text
  - Initialise them with pretrained word vectors!





# General Findings

- neural  $>$  count
- **Contextual word representation** is shown to work even better
- Dynamic word vectors that change depending on context!
- ELMO & BERT (next lecture!)

# 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 Skip-gram/CBOW  
<https://radimrehurek.com/gensim/index.html>
- GLOVE
  - <http://nlp.stanford.edu/projects/glove/>

# Further Reading

- JM3, Ch 6