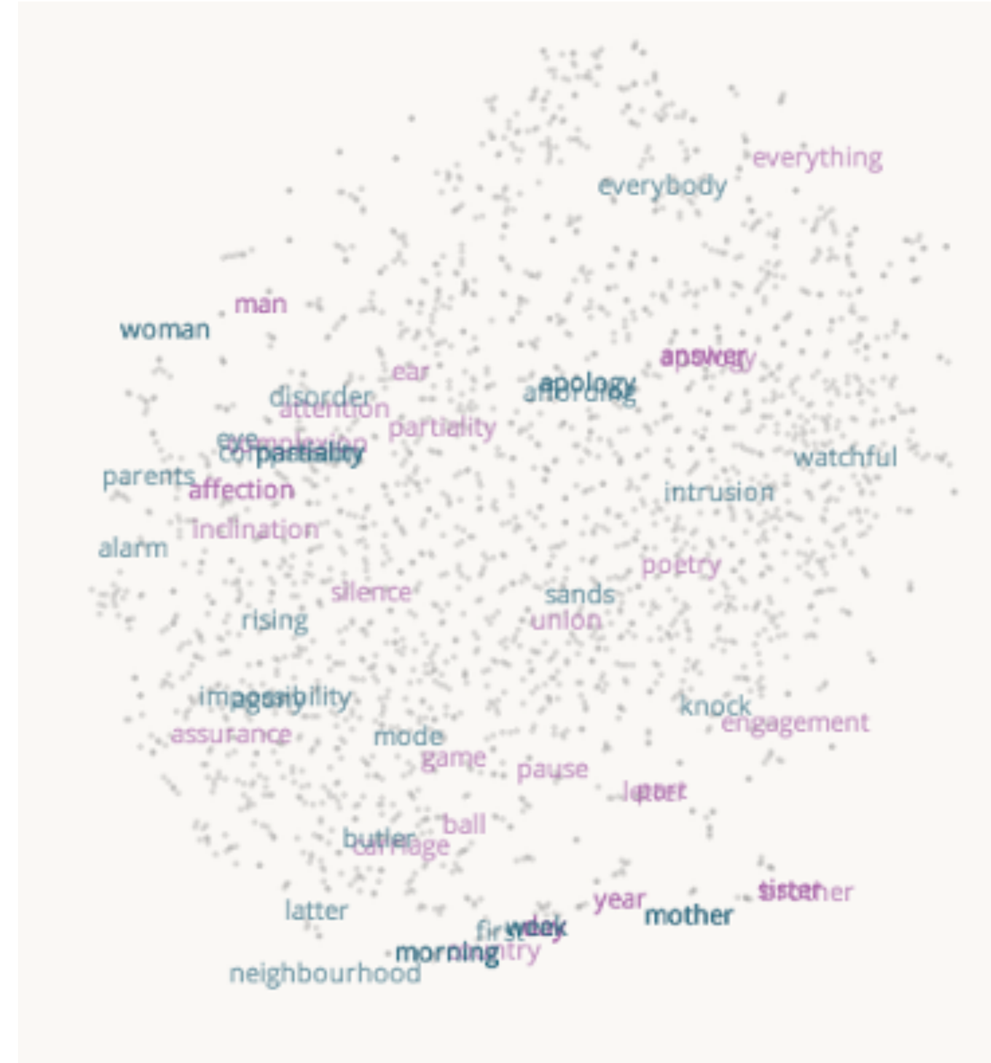


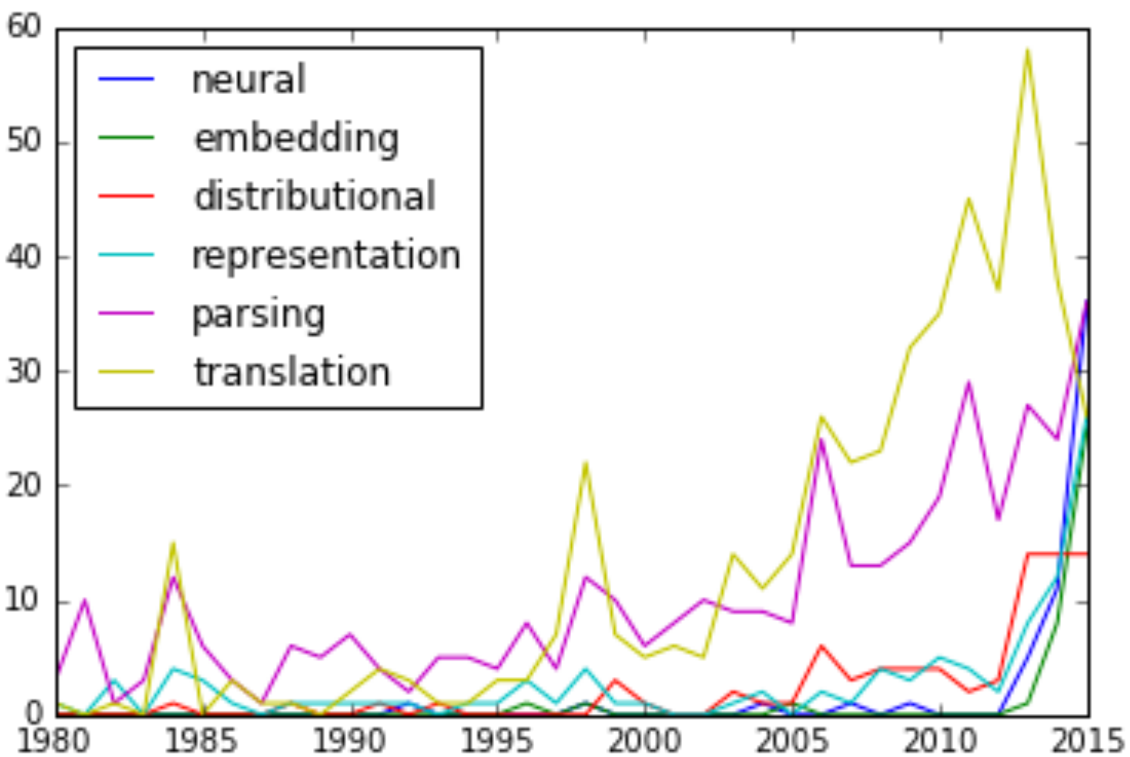
COMP90042 LECTURE 10

WORD VECTOR LEARNING

Image credits: http://blogger.ghostweather.com/2014/11/visualizing-word-embeddings-in-pride.html

THE FALL AND RISE OF NEURAL NETS

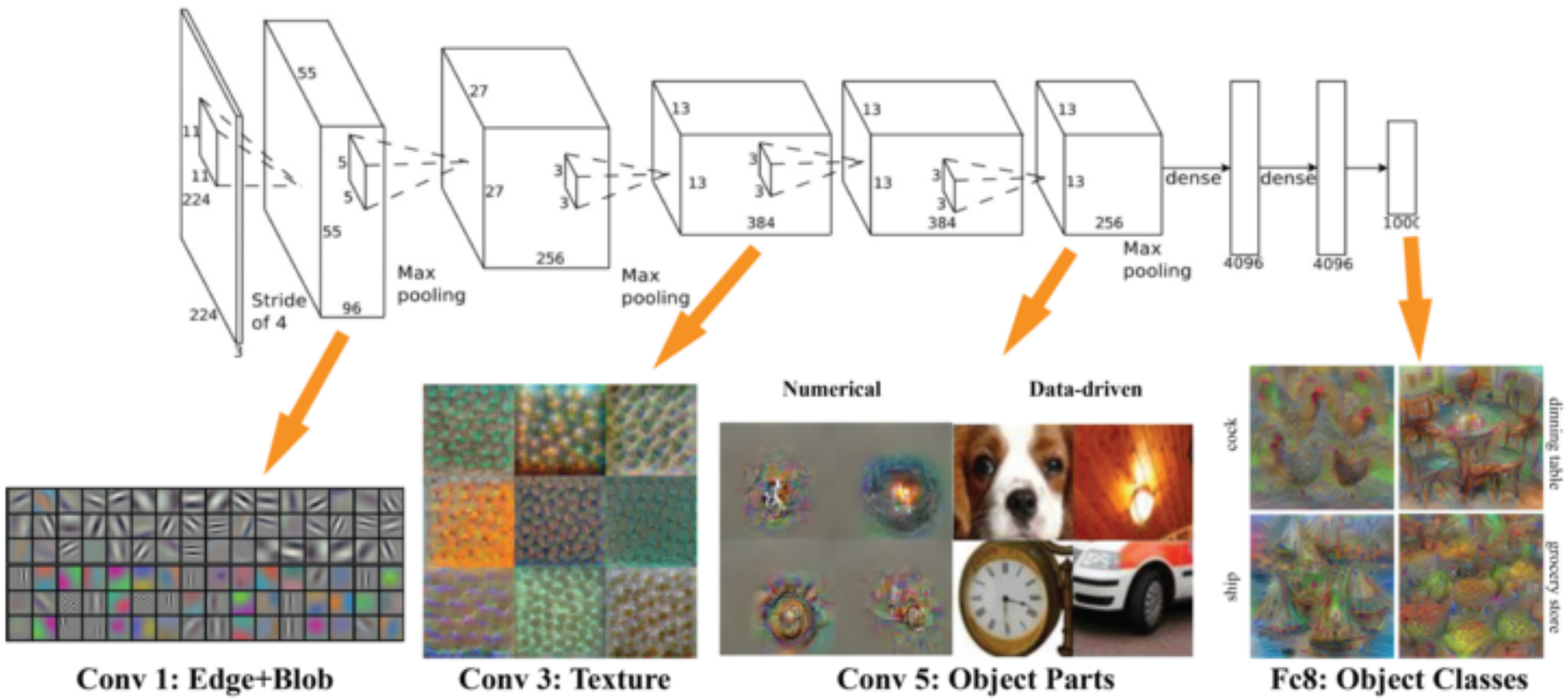
- Neural networks major focus in 1980s & 1990s
  - fell out of usage, losing to SVMs etc
  - now back in the spotlight, with outstanding results



  - Unigram counts over *CL conferences (from paper titles in ACL anthology)

WHY NEURAL?

- Biggest benefit from learning **representation** at the same time as learning a **classifier**

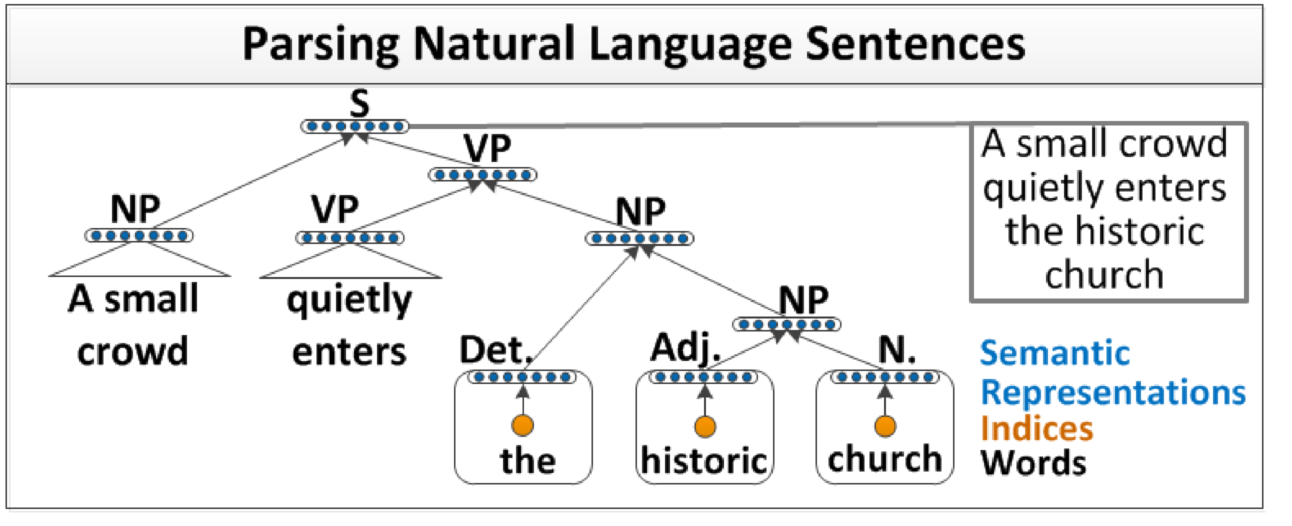


AlexNet
http://vision03.csail.mit.edu/cnn_art/index.html

- Learns *representation* of input through stacked layers…. to support classifier in final layer (e.g., logistic regression)
  - much more effective than pixel based features, or hand-coded patterns

NEURAL MODELS OF LANGUAGE?

- Intuitive application to vision, over real-valued images
- Less obviously useful for language
  - words are *discrete*, and vocabularies are enormous; how to learn to process?
  - words occur in sequences, ordering is often important
- **Key question:** can we learn dense word representations that inform other tasks?



Socher et al, ICML 2011

OUTLINE

- Neural network inspired methods for vector learning
- "Skip-gram" and "Contextual Bag of Words (CBOW)"
- Similarity to LSA type approaches
- Neural network classification approaches

WORD VECTOR LEARNING RECAP

- Matrix input, X
  - Document collection
    - Expressed as a word type x document matrix
  - Words in context
    - Expressed as a word type x word type matrix
- Output
  - Factorisation of X into matrices W, Σ, C
  - Entries of W become '*word representations*'
  - Truncate to top *k* most important dimensions
- More robust 'dense' representation of lexicon

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

UTILITY OF LEARNED VECTORS (TEASER)

- What words have similar vectors?

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	graffiti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Figure 19.21 Examples of the closest tokens to some target words using a phrase-based extension of the skip-gram algorithm (Mikolov et al., 2013a).

JM3 Ch 19

- Do vector differences follow consistent patterns?

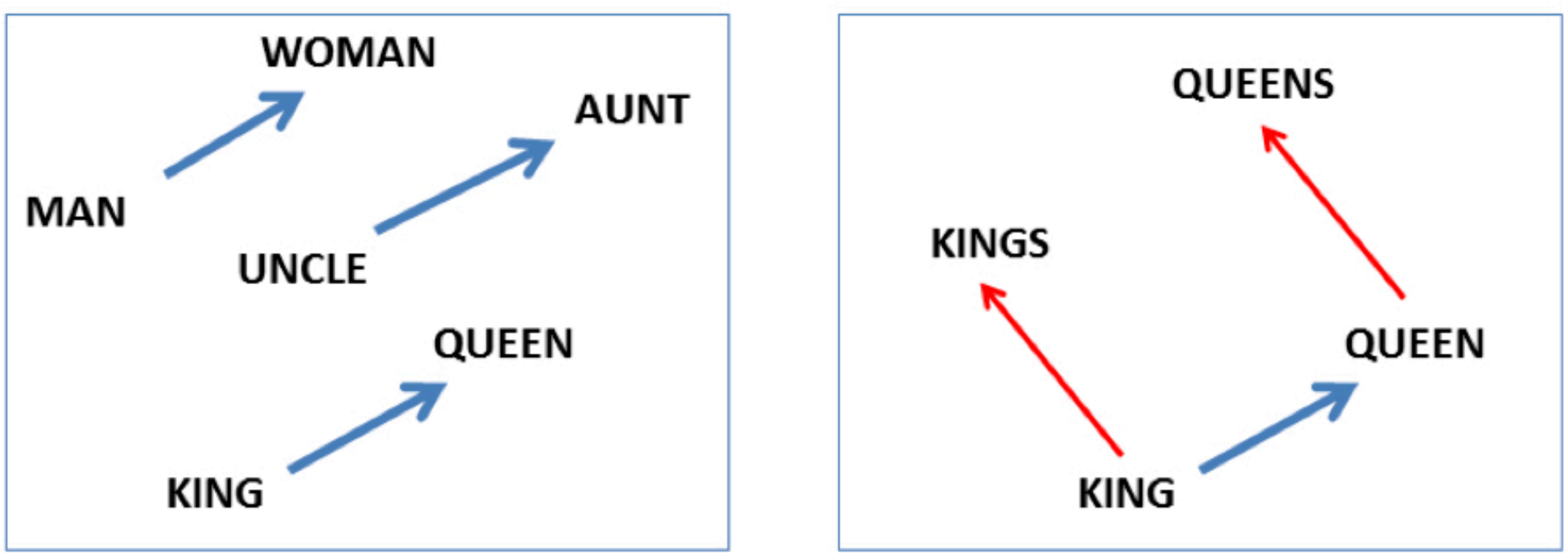
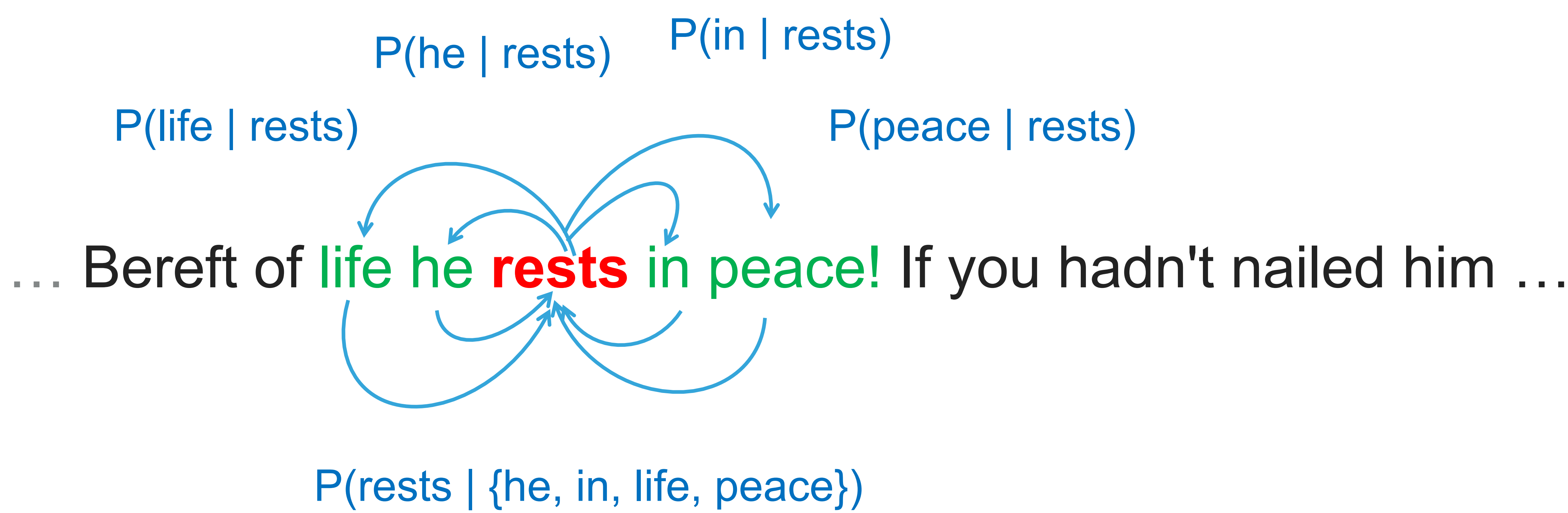


Fig from Mikolov et al, NIPS 2013

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

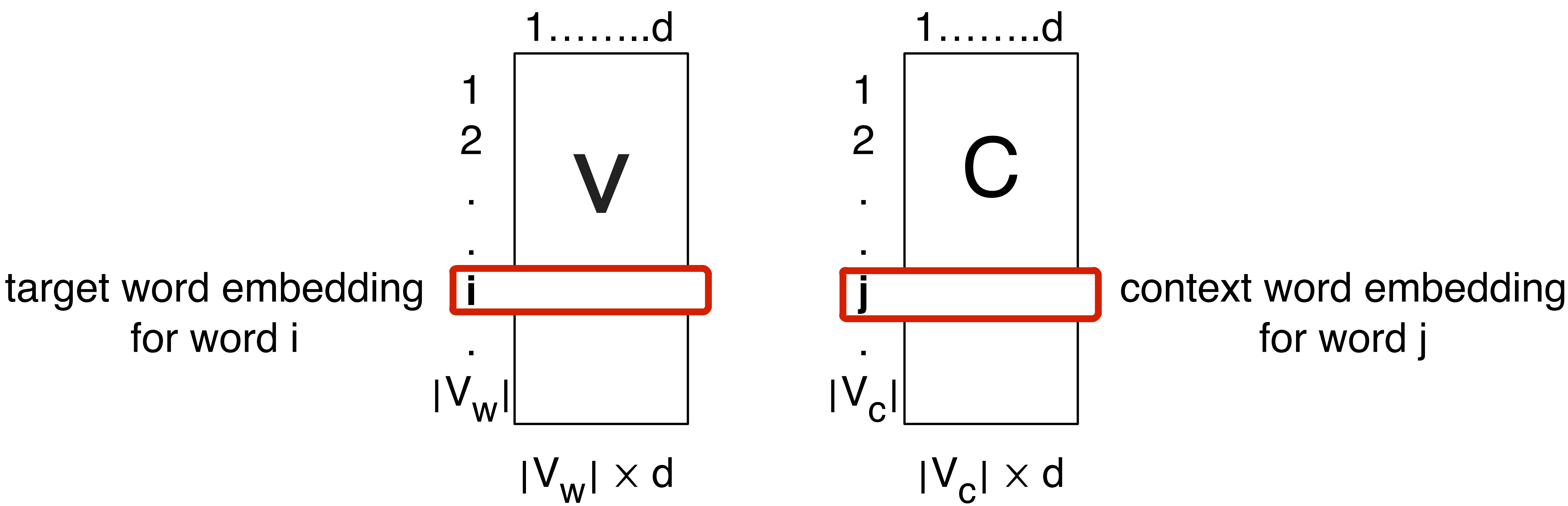


Fig 19.17, JM

- 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" = 17235th 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

VERSUS LOGISTIC REGRESSION (JFF)

- A specific parameterisation of the *logistic regression* classifier, using
  - bigrams as features
  - and factorising the parameters into d dimensions

$$P(w_k|w_j) \propto \exp(\lambda_{w_k,w_j})$$

$$\Lambda \approx CV$$

- JFF = Just for fun! I.e., getting a bit difficult for the subject, and not examinable.

SKIP-GRAM MODEL

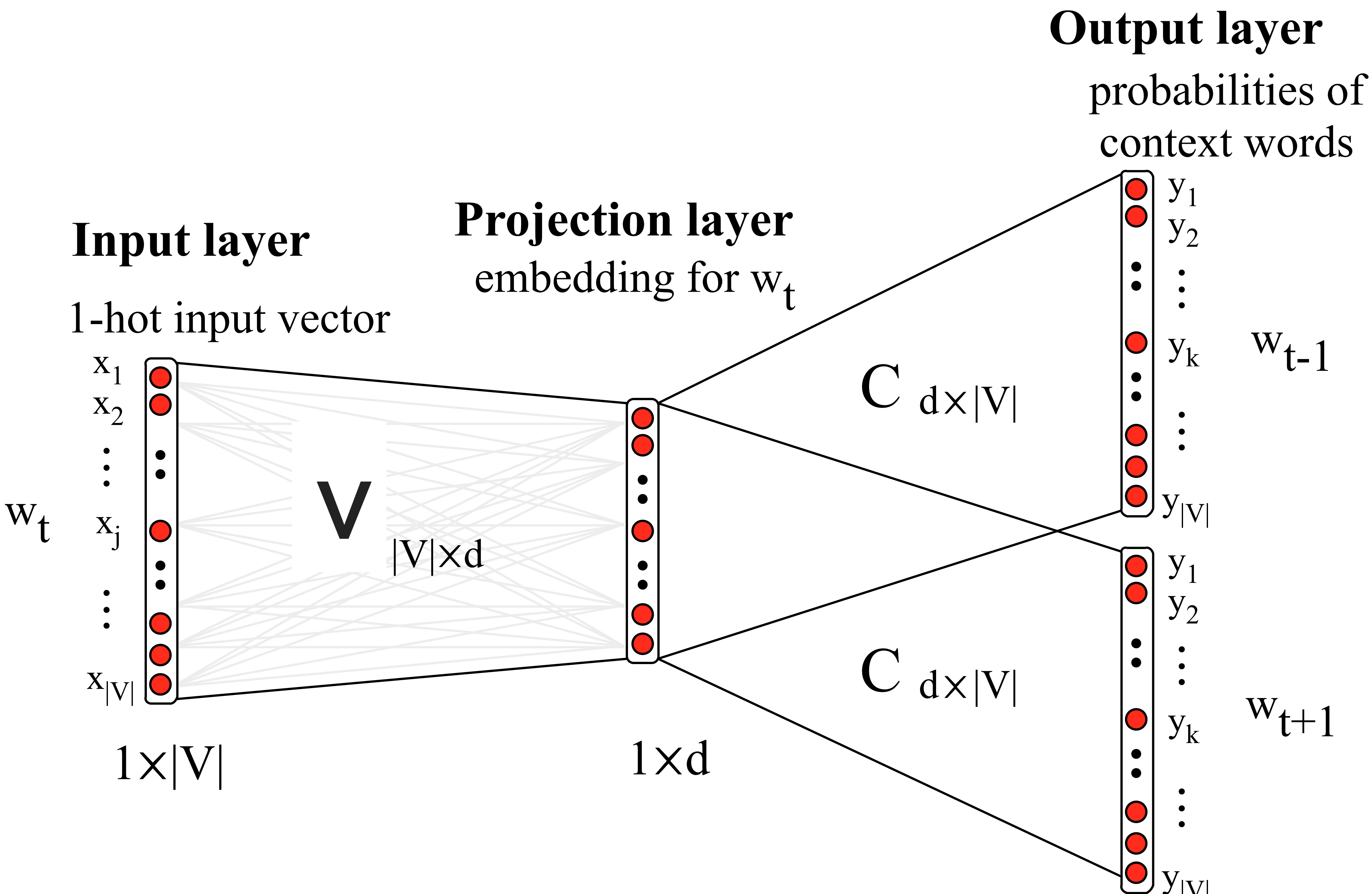


Fig 19.18, JM

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
   - $v_j \cdot c_k$ 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

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

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, due to the sum over the vocabulary…

LEARNING BY NEGATIVE SAMPLING (JFF)

- Instead reduce the problem to binary classification to distinguish
  - True context words (+ class), c, from
  - Randomly sampled words (- class), denoted n_i
- Objective for each position becomes

$$\log \sigma(c \cdot v) - \frac{1}{k} \sum_{i=1}^{k} \log \sigma(-n_i . v)$$

  - where we draw k random context words, n_i
  - v = centre word embedding

EXAMPLE (JFF)

- Given word ‘apricot’ in context, and L=2

```
lemon,  a [tablespoon of apricot preserves or] jam
           c1         c2    w       c3       c4
```

- Draw k=2 noise words for each context word

```
[cement metaphysical dear coaxial   apricot attendant whence forever puddle]
 n1     n2           n3   n4                n5        n6     n7       n8
```

- And optimise objective such that embeddings for c1, …, c4 have much higher dot product with w’s embedding (v) than for n1, …, n8

- Cheaper, independent of vocabulary size

Example 19.6.1 JM

CBOW: CONTEXTUAL BAG-OF-WORDS

- Condition on context, and generate centre word

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

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

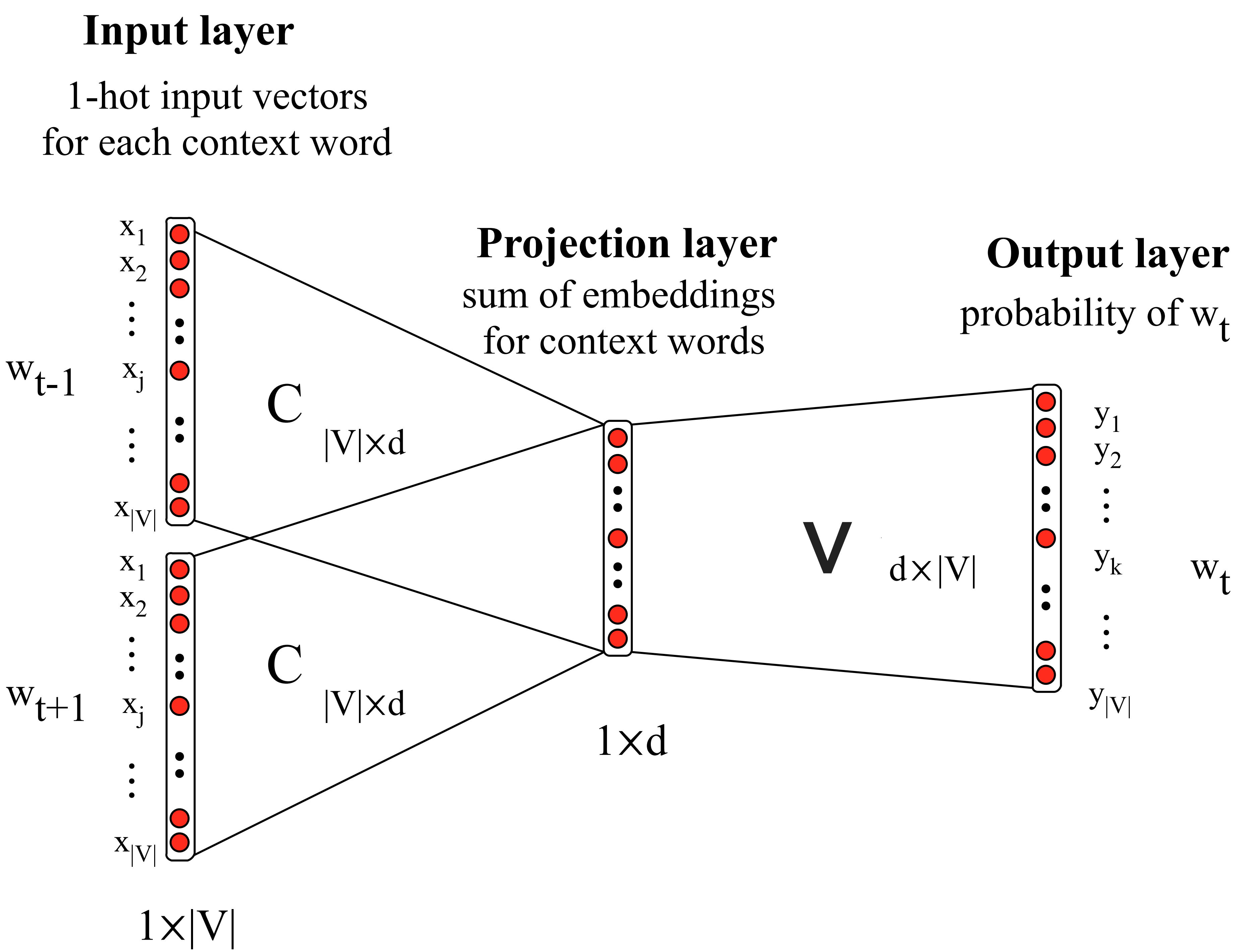
- Mirror of skip-gram

$$P(w_j|c) = \frac{\exp(v_{w_j} \cdot h(c))}{\sum_{w' \in V} \exp(v_{w'} \cdot h(c))}$$

$$h(c) = \frac{1}{2L} \sum_{-L \leq k \leq L, k \neq 0} c_{w_{j+k}}$$

  - where *h(c)* is the *average* embedding of context words

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

VECTOR SIMILARITIES

- What words have similar vectors?

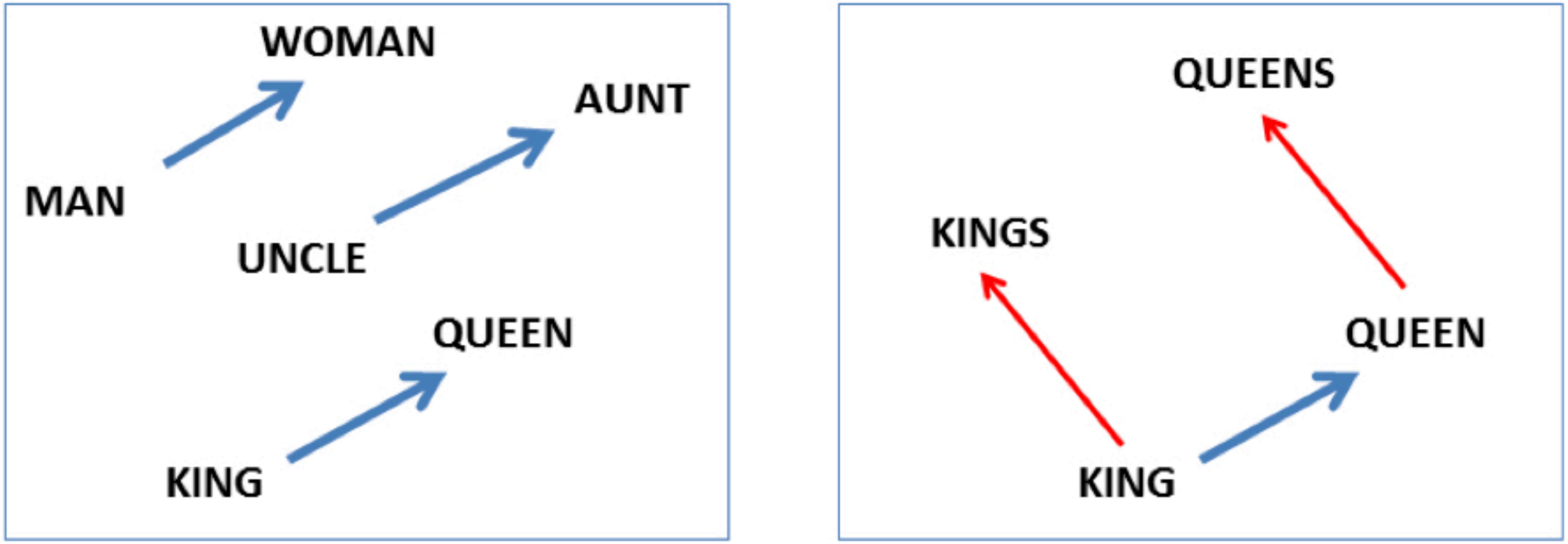
target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	graffiti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Figure 19.21 Examples of the closest tokens to some target words using a phrase-based extension of the skip-gram algorithm (Mikolov et al., 2013a).

- Automatically recovers similarity over
  - Syntactic variants of same word (e.g., verb tense, number)
  - Synonymous words
  - Related concepts

VECTOR DIFFERENCES

- What about vector differences?
  - Often follow systematic patterns



  - Fig from Mikolov et al, 2013
  - E.g., gender, number, country - capital city, country – food, country – currency etc.

WHY VECTOR DIFFERENCES?

- Consider which of w_k or w_l are more likely in a context of a single word, w_j, framed as the log-odds:

$$\log \frac{P(w_k|w_j)}{P(w_l|w_j)} = \log \frac{\exp(c_k \cdot v_j)}{Z} - \log \frac{\exp(c_l \cdot v_j)}{Z}$$
$$= c_k \cdot v_j - c_l \cdot v_j$$
$$= (c_k - c_l) \cdot v_j$$

  - End up with a vector difference between the two words!

- Can consider the difference encoding the contexts in which w_k occurs but w_l does not
  - Consider king – queen and the pronouns he/his vs she/her

WHY VECTOR DIFFERENCES?

- Even from raw counts, contexts can be very informative:

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Source: Pennington et al, 2014

  - *ratio* of probabilities illustrative of the amount of information in the context word

- It turns out that vector differences encode probability ratios in word2vec models! I.e.,

$$\frac{P(w_k|w_j)}{P(w_l|w_j)} = \frac{\exp(c_k \cdot v_j)/Z}{\exp(c_l \cdot v_j)/Z}$$

$$= \exp(c_k \cdot v_j - c_l \cdot v_j) = \exp([c_k - c_l] \cdot v_j)$$

EVALUATING WORD VECTORS

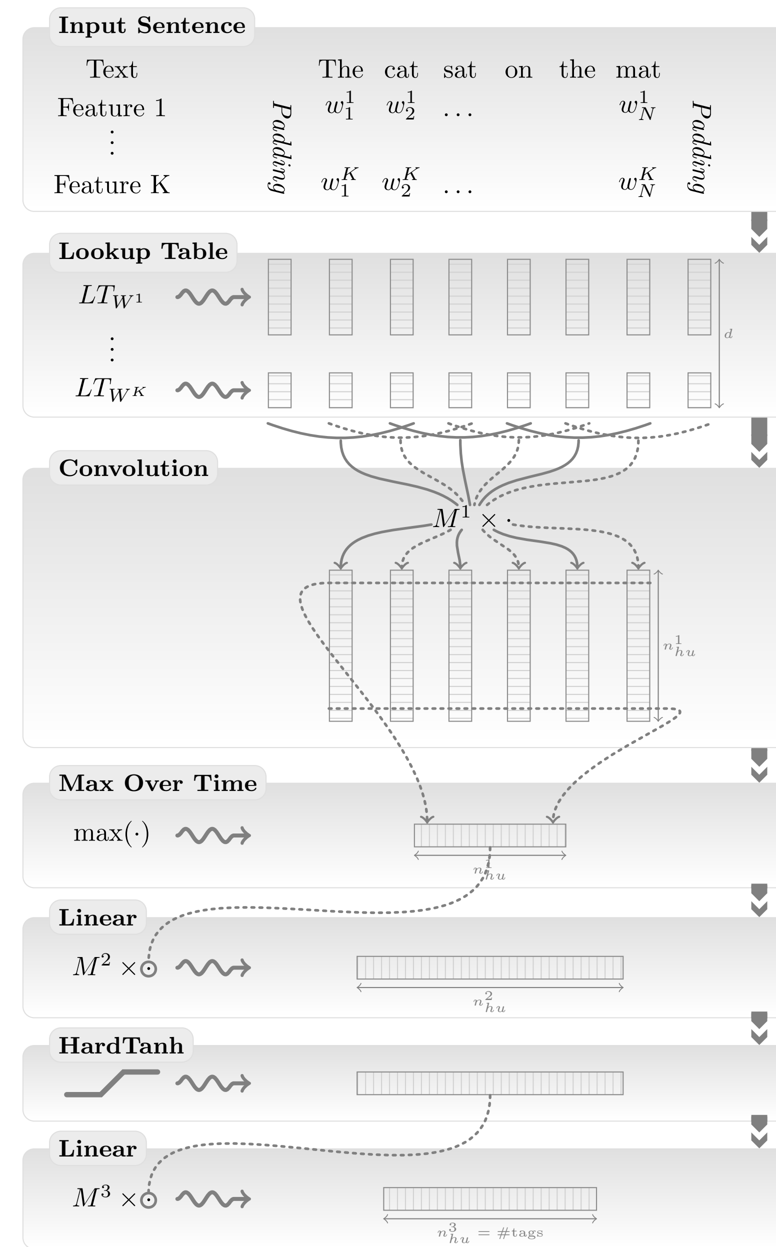
- Thesaurus style tasks
  - *WordSim-353* are pairs of nouns with judged relatedness
  - *SimLex-999* also covers verbs and adjectives
  - *TOEFL* asks for closest synonym as multiple choice
  - …
- Word analogy task
  - Man is to King as Woman is to ???
  - France is to Paris as Italy is to ???
  - Evaluate where in the ranked predictions the correct answer is, given tables of known relations

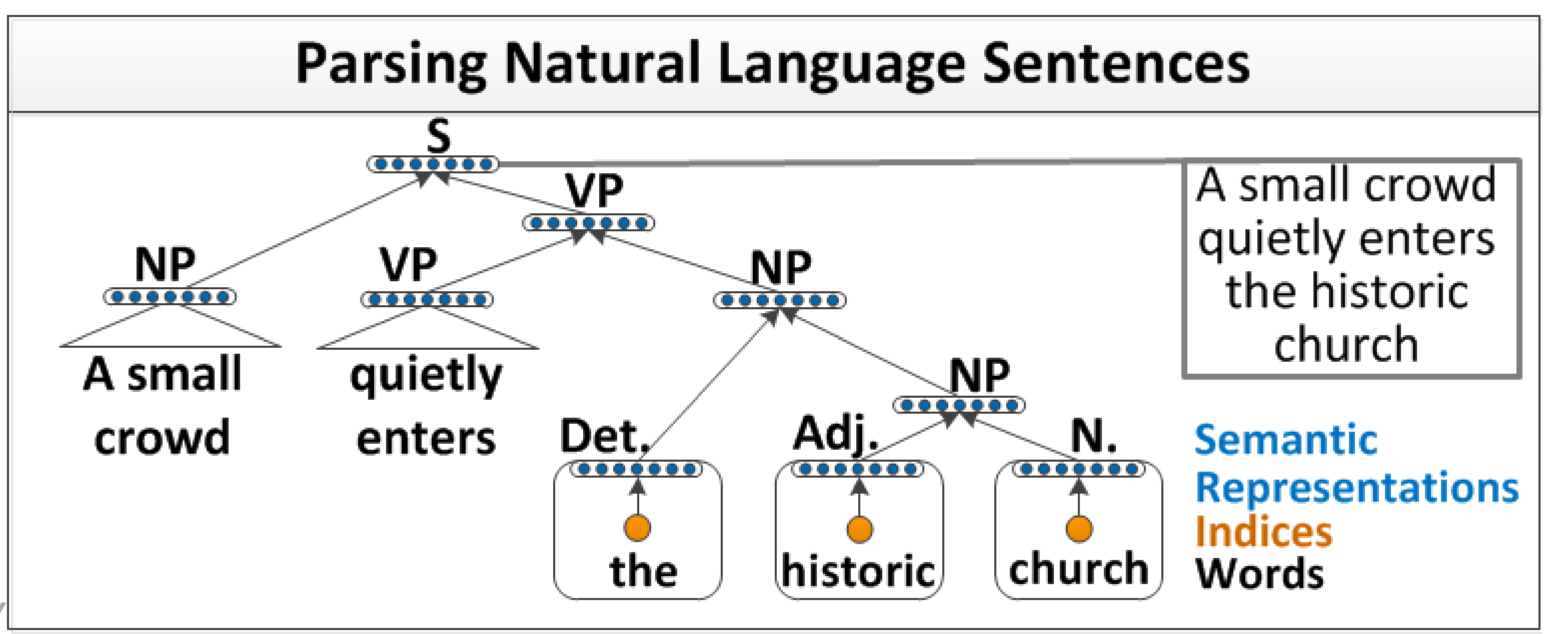
USES FOR WORD VECTORS

- Word vectors as features as part of a model (classifier / regression / CRF …)
  - Provides a dense representation of the lexicon with vastly fewer parameters, e.g., d=300 rather than |V|=100k+
  - *Transfer learning*, providing insights from large external datasets (e.g., *word2vec* trained on billions of words)
- As pre-training
  - Neural models often include word embedding step, however training can be very slow and becomes trapped in bad optima
  - Use '*pre-trained*' word vectors to start, avoiding these issues

EMBEDDINGS IN NEURAL MODELS

- SENNA: joint language model, part of speech, semantic roles (Collobert et al, JMLR 2011)
- Socher et al's recursive neural parser (Socher et al, ICML 2011)
- … and many more! E.g., translation





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

- J&M3 Ch. 19.6 onwards
- **Just for fun:**
  - 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*.