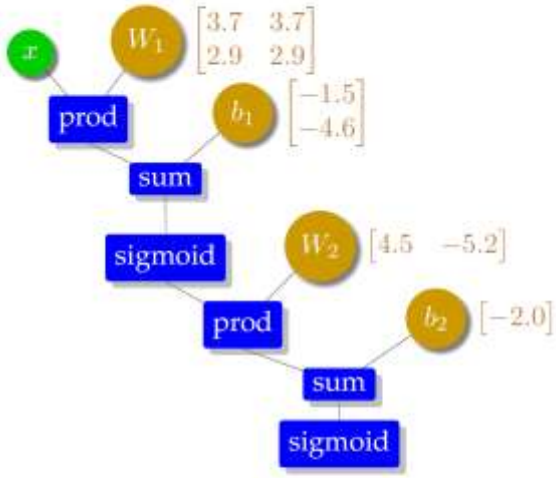
Neural Language Models

Marine Carpuat

Based on slides by Philipp Koehn (JHU)

Recap: Computation Graph



- To build a system, we only need to:
  - Define network structure
  - Define error/loss function
  - Provide data
  - (and set a few more hyperparameters to control training)

- Given network structure
  - Prediction is done by forward pass through graph (forward propagation)
  - Training is done by backward pass through graph (back propagation)
  - Based on simple matrix vector operations

- Forms the basis of neural network libraries
  - Tensorflow, Pytorch, mxnet, etc.

Language Modeling

- Goal: compute the probability of a sentence or sequence of words

  $P(E) = P(e_1,e_2,e_3,e_4,e_5 \ldots e_n)$

- Related task: probability of an upcoming word

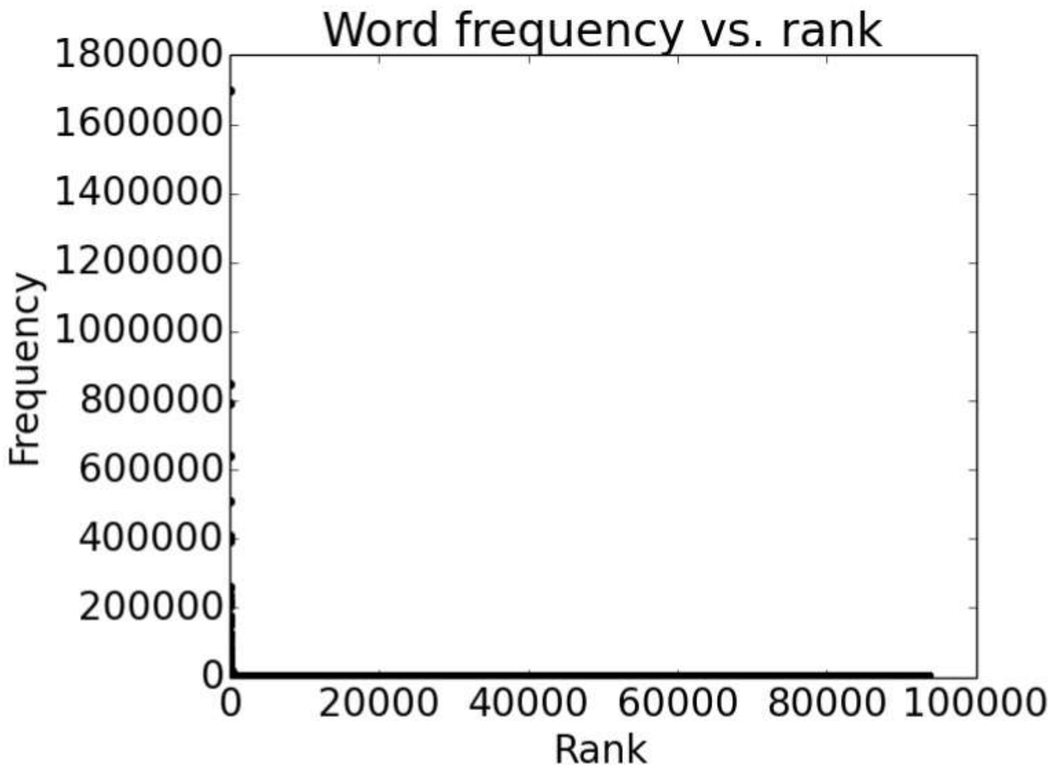
  $P(e_5 | e_1,e_2,e_3,e_4)$

- A model that computes either of these:

  $P(E)$ or $P(e_n | e_1,e_2 \ldots e_{n-1})$

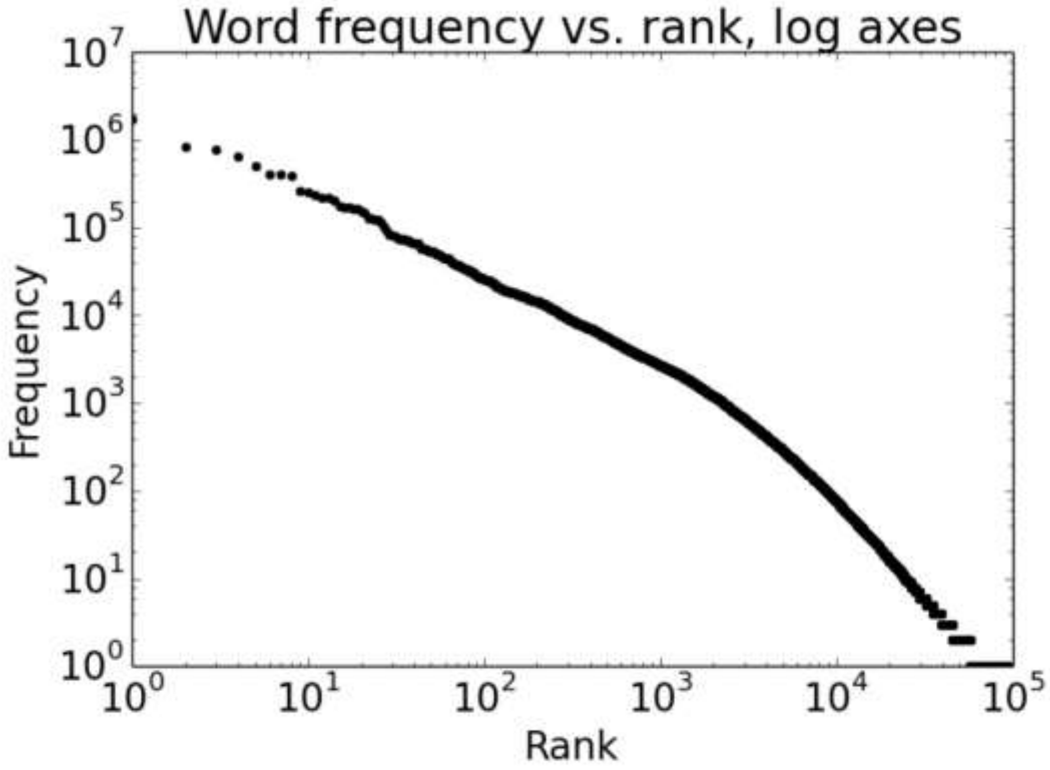
  is called a **language model**.

Zipf’s Law



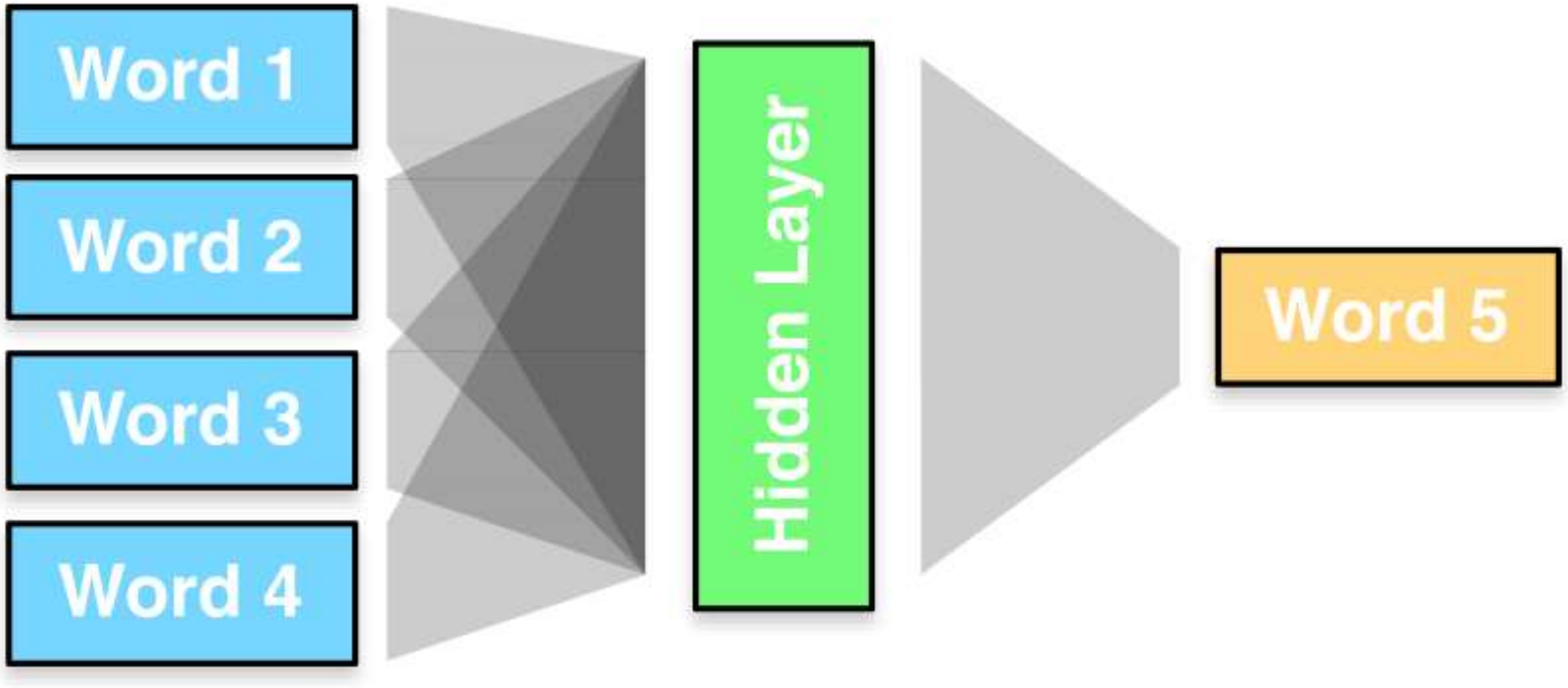
Word frequency vs. rank
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
Frequency
0
20000
40000
60000
80000
100000
Rank

Zipf's Law



- Even in a very large corpus, there will be a lot of infrequent words
- The same holds for many other levels of linguistic structure
- NLP/MT challenge: we need to be able to make predictions for things we have rarely or never seen

Toward a Neural Language Model



Figures by Philipp Koehn (JHU)

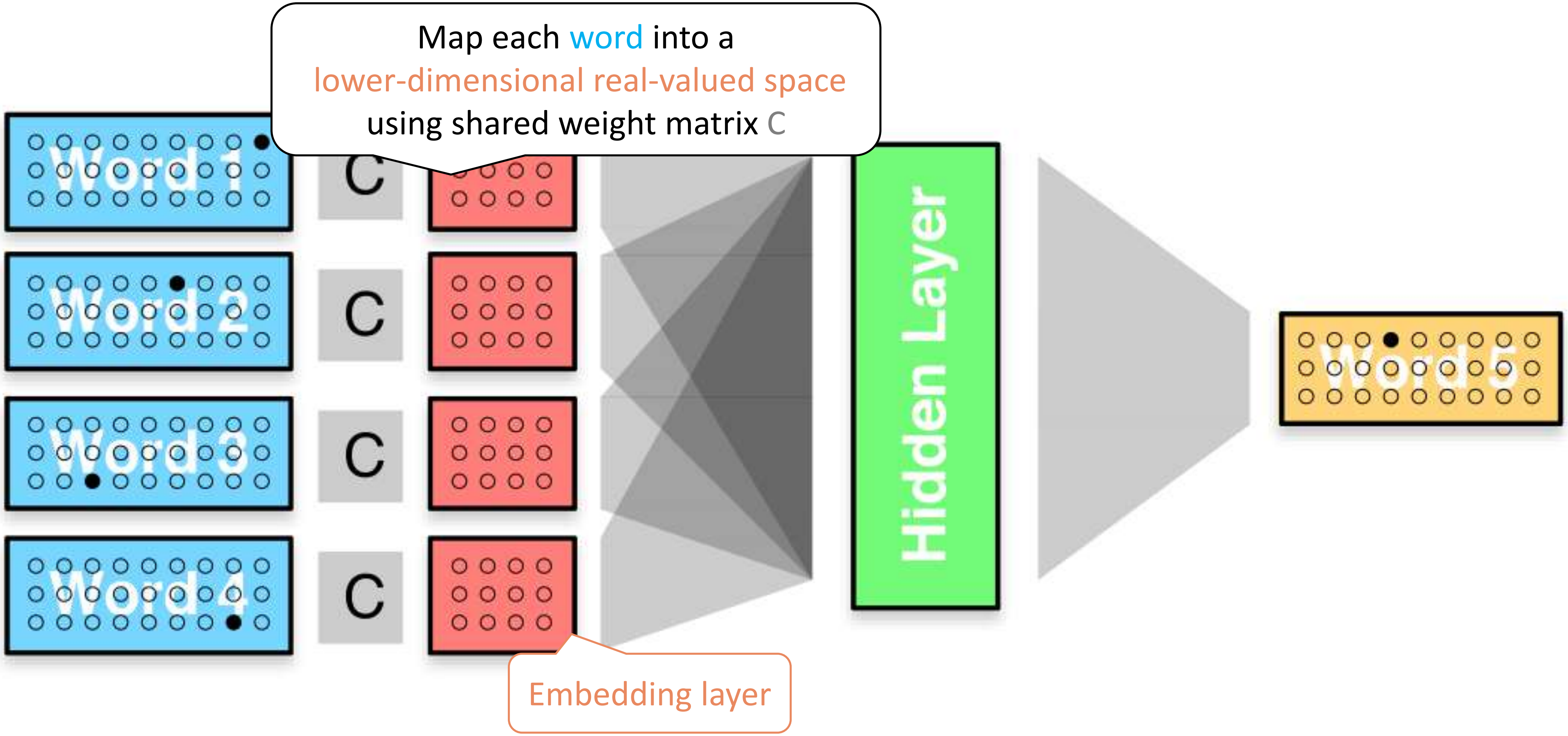
Representing Words

- "one hot vector"

```
dog = [ 0, 0, 0, 0, 1, 0, 0, 0 …]
cat = [ 0, 0, 0, 0, 0, 0, 1, 0 …]
eat = [ 0, 1, 0, 0, 0, 0, 0, 0 …]
```

- That's a large vector! practical solutions:
  - limit to most frequent words (e.g., top 20000)
  - cluster words into classes
  - break up rare words into subword units

Language Modeling with Feedforward Neural Networks



Bengio et al. 2003

An Output Layer to Predict Words

- Network will output a probability for each word in the vocabulary V
- Step 1: compute a score for each word in V $\quad s = Wx + b$

$s \in \mathbb{R}^{|V|}$ $\quad$ $W \in \mathbb{R}^{|V| \times N}$ $\quad$ $b \in \mathbb{R}^{|V|}$

- Step 2: turn scores into probabilities using softmax function

$$p = \mathrm{softmax}(s)$$

Where the probability of the j-th word in V is $p_j = \dfrac{e^{s_j}}{\sum_{\tilde{j}} e^{s_{\tilde{j}}}}$

Estimating Model Parameters

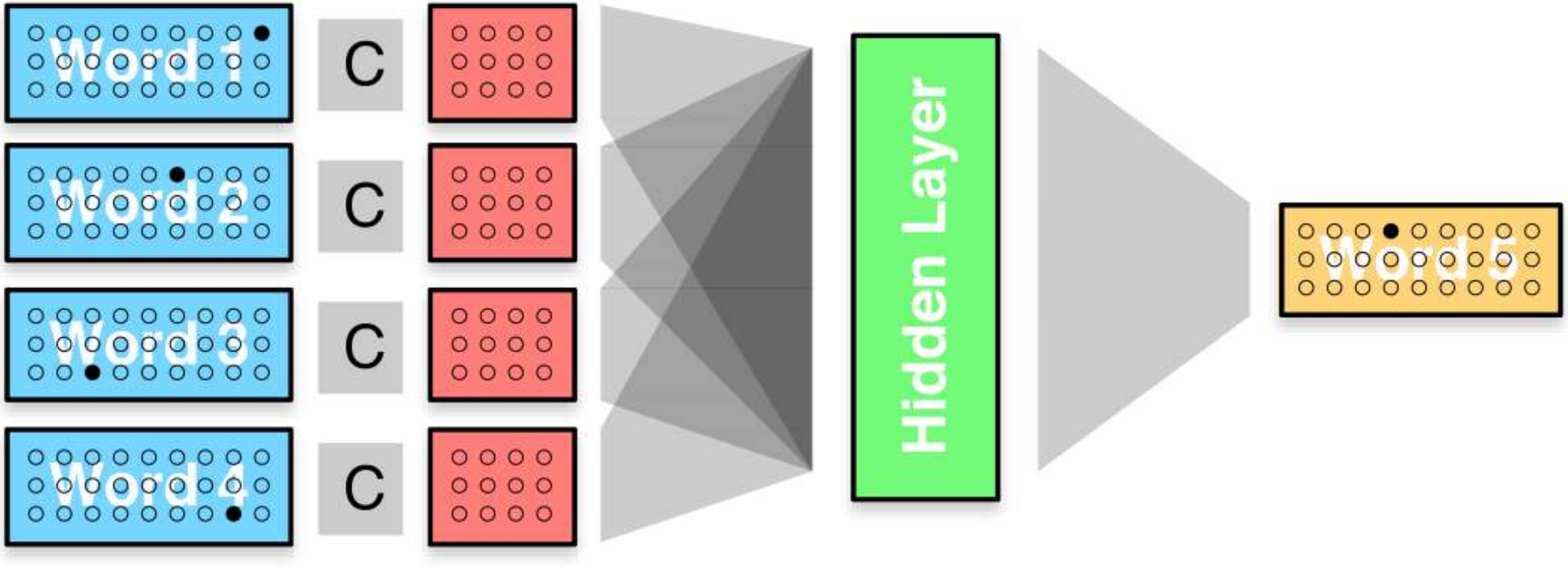
- Intuition: a model is good if it gives high probability to existing word sequences

- Training examples:
    - sequences of words in the language of interest

- Error/loss: negative log likelihood
    - At the corpus level $\text{error}(\lambda) = -\sum_{E \text{ in corpus}} \log P_\lambda(E)$

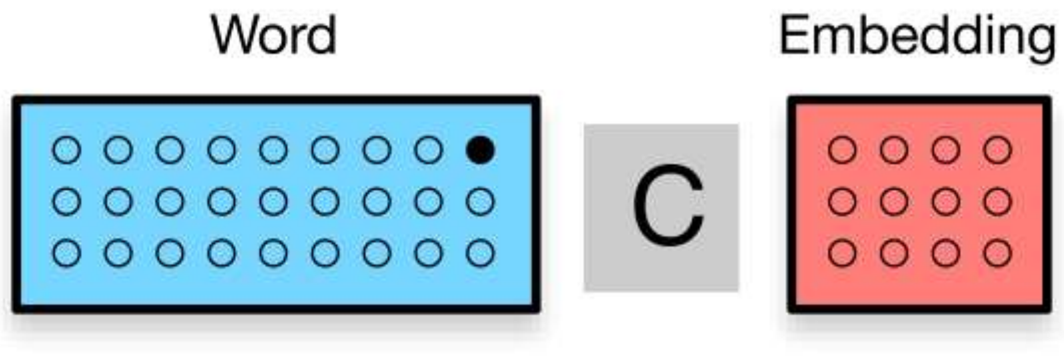
    - At the word level $\text{error}(\lambda) = -\log P_\lambda(e_t | e_1 \dots e_{t-1})$

Language Modeling with Feedforward Neural Networks



Bengio et al. 2003

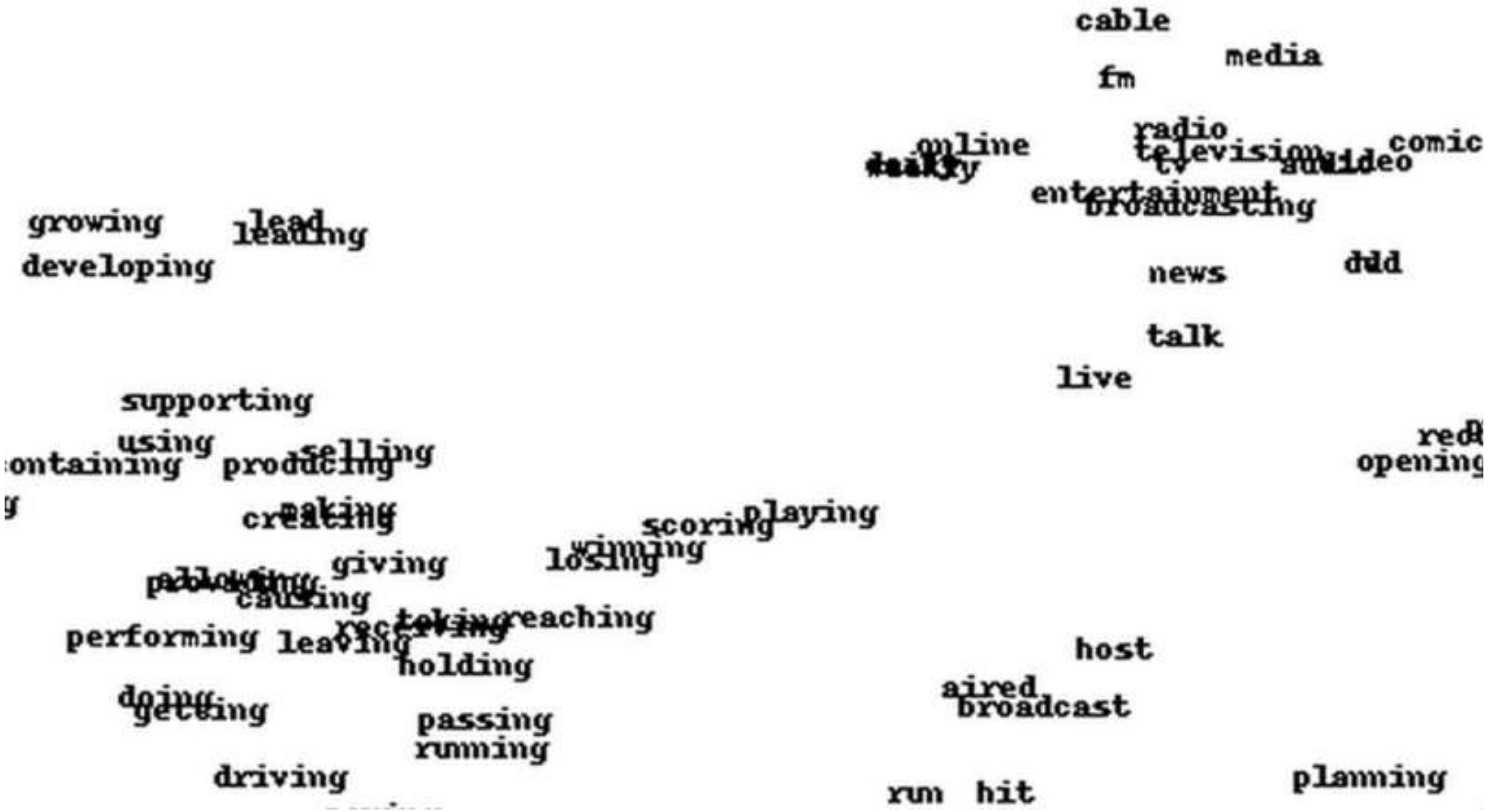
Word Embeddings: a useful by-product of neural LMs



- Words that occurs in similar contexts tend to have similar embeddings
- Embeddings capture many usage regularities
- Useful features for many NLP tasks

Word Embeddings

Word Embeddings



Word Embeddings Capture Useful Regularities

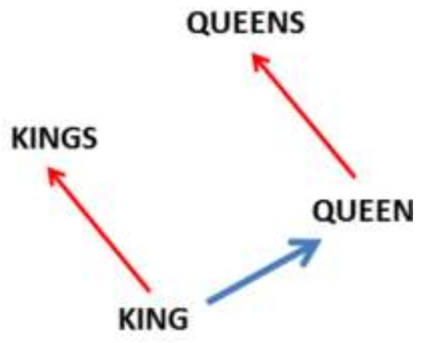
Morpho-Syntactic

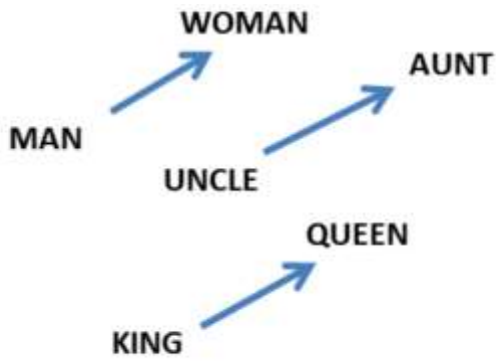
- Adjectives: base form vs. comparative
- Nouns: singular vs. plural
- Verbs: present tense vs. past tense

[Mikolov et al. 2013]

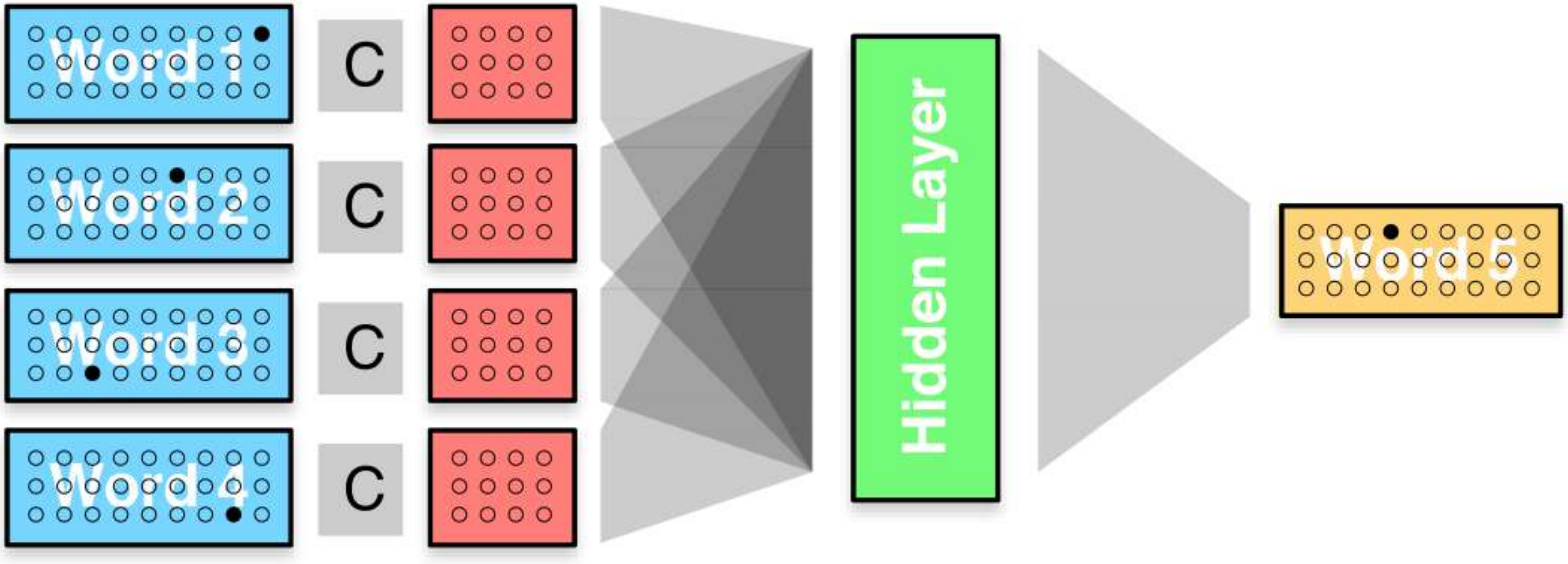
Semantic

- Word similarity/relatedness
- Semantic relations
- But tends to fail at distinguishing
  - Synonyms vs. antonyms
  - Multiple senses of a word





Language Modeling with Feedforward Neural Networks



Bengio et al. 2003

Language Modeling with Recurrent Neural Networks

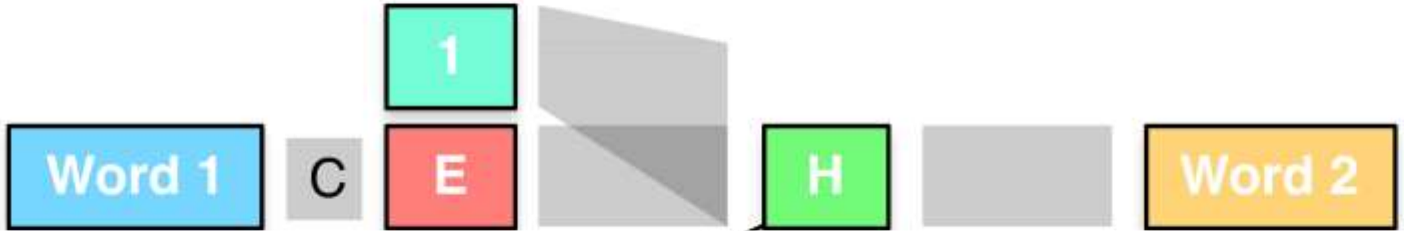
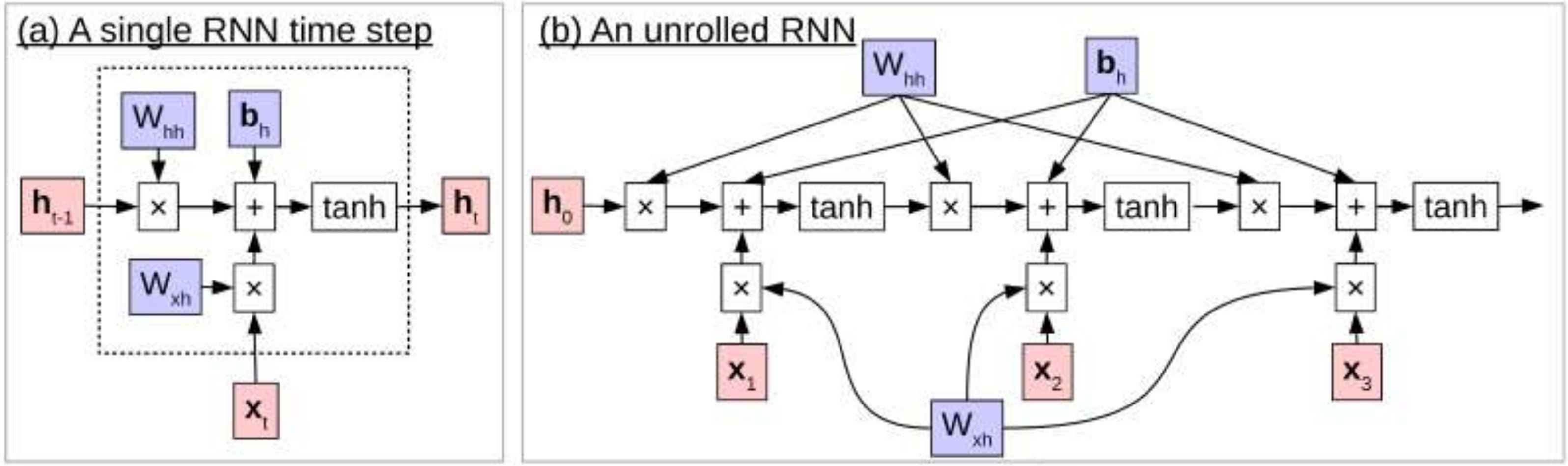


Figure by Philipp Koehn

Formalizing our Recurrent Language Model



$$\boldsymbol{m}_t = M_{\cdot, e_{t-1}}$$

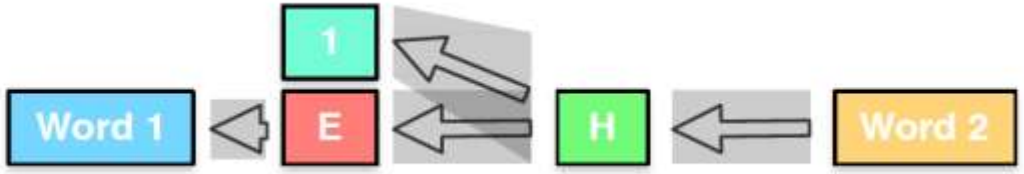
$$\boldsymbol{h}_t = \begin{cases} \tanh(W_{mh}\boldsymbol{m}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) & t \geq 1, \\ \boldsymbol{0} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t = \text{softmax}(W_{hs}\boldsymbol{h}_t + b_s).$$

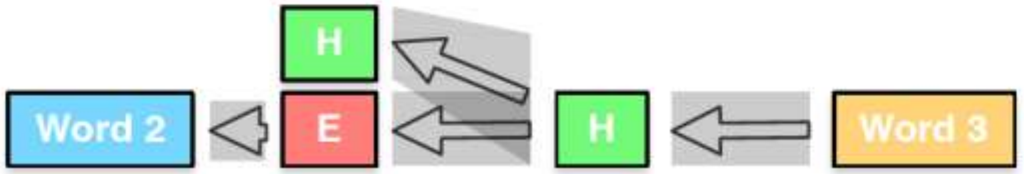
Figure by Graham Neubig

Training

- Process 1st example
- Update weights with backprop



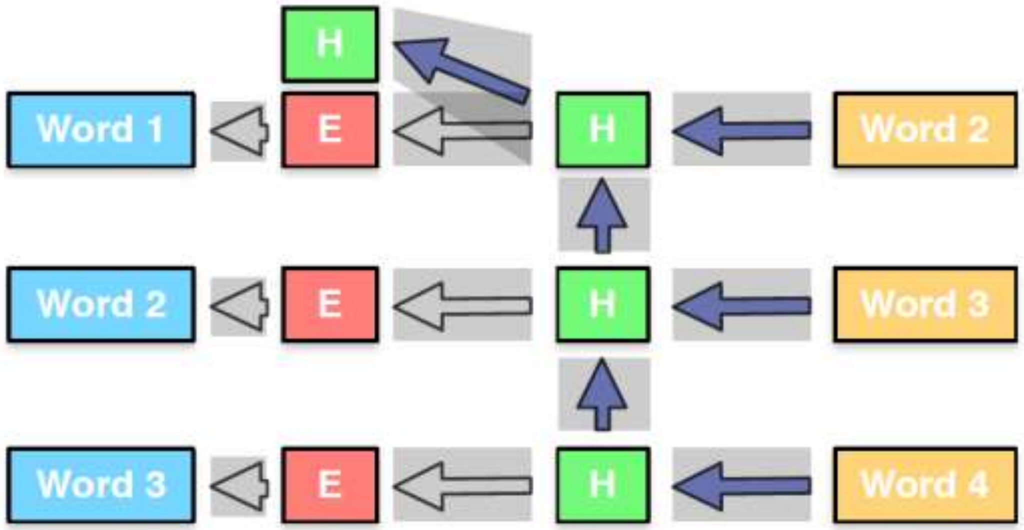
- Process 2nd example
- Update weights with backprop



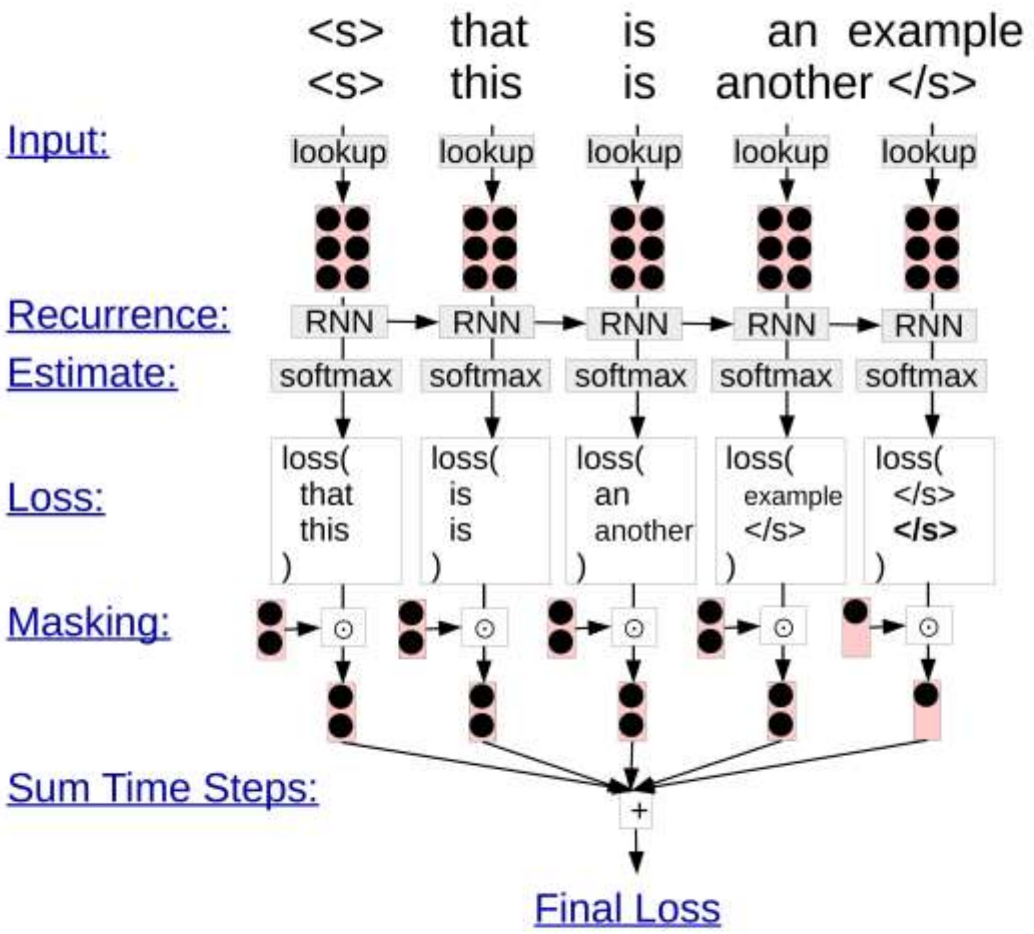
- No feedback to previous history!

Training: Backpropagation Through Time

- Process a few examples
- Backpropagate through unfolded neural network



Practical Training Issues



- Compute parameter updates based on a "minibatch" of examples
  - instead of using one example at a time
- More efficient
  - matrix-matrix operations as opposed to multiple matrix-vector operations
- Can lead to better model parameters
  - middle ground between online and batch training

Figure by Graham Neubig

Practical Training Issues: vanishing/exploding gradients

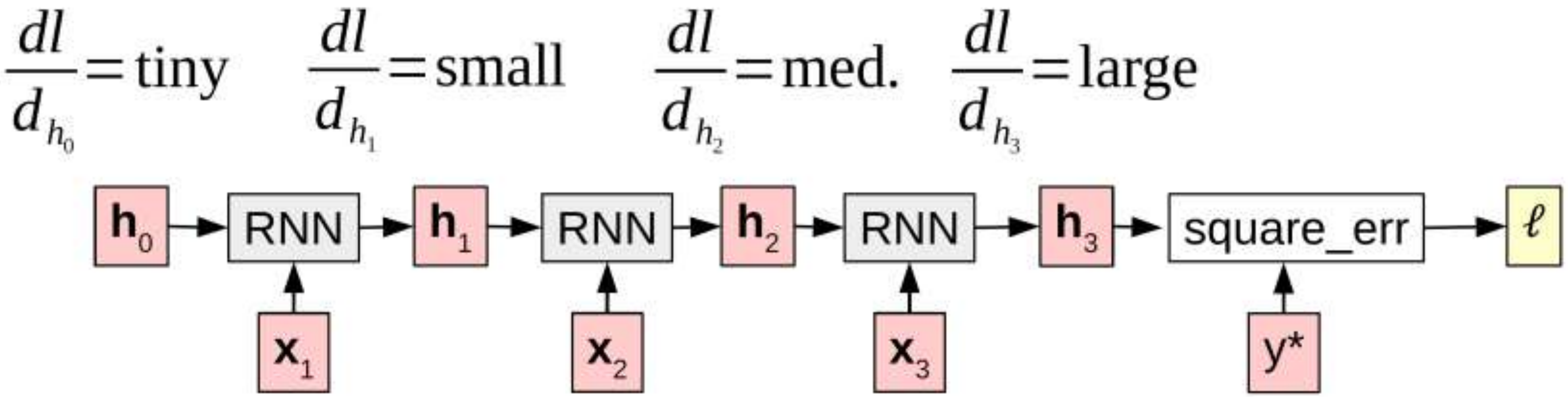


Figure 16: An example of the vanishing gradient problem.

Figure by Graham Neubig

Multiple ways to work around this problem:

- ReLU activations help

- Dedicated RNN architecture (Long Short Term Memory Networks)

Aside: Long Short Term Memory Networks

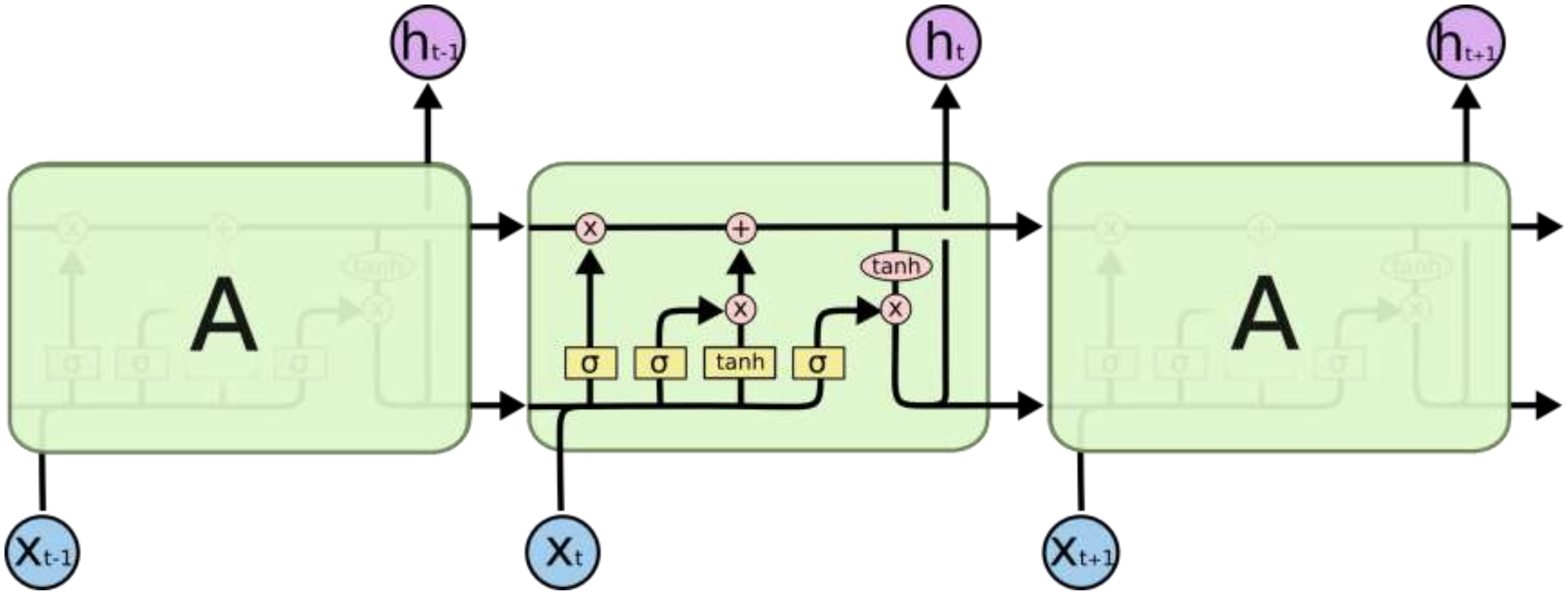


Figure by Christopher Olah

What do Recurrent Language Models Learn?

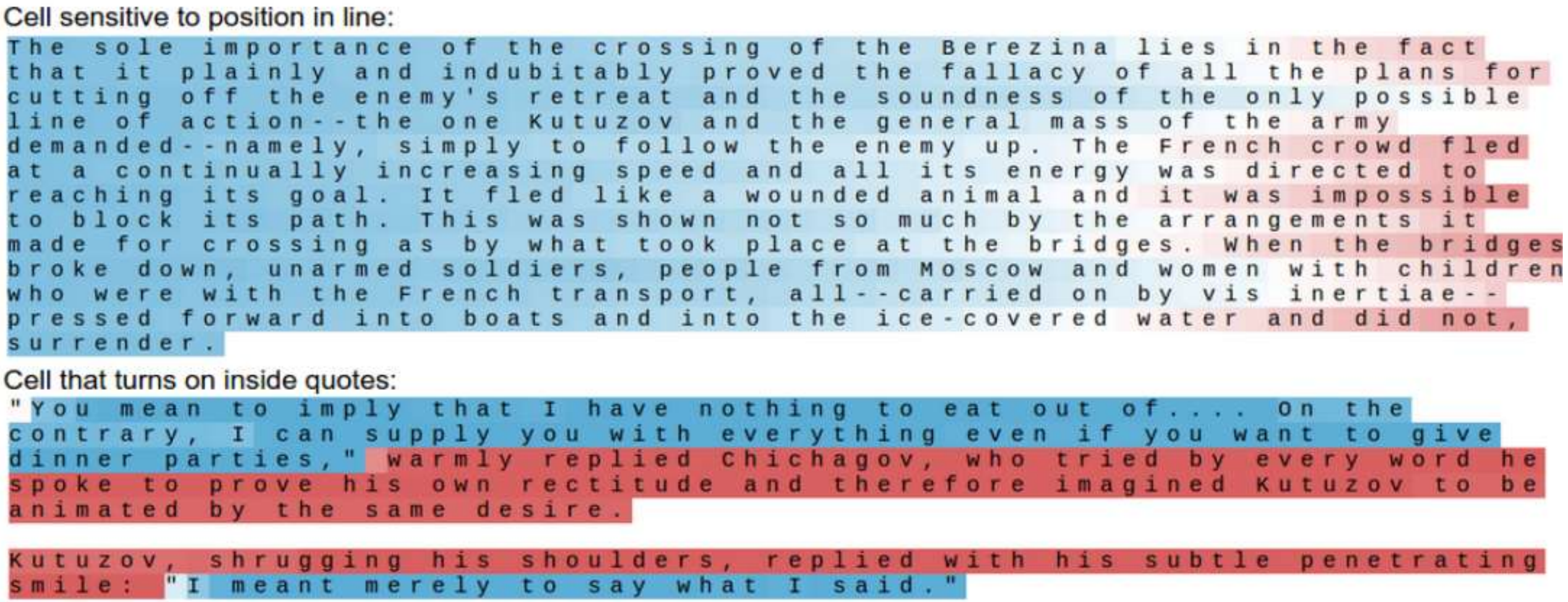
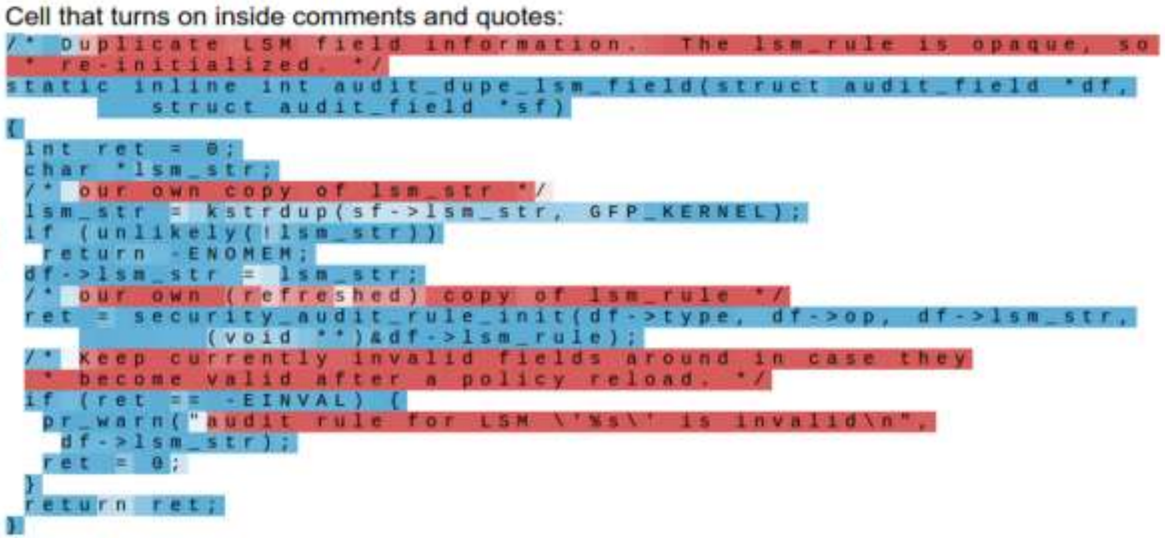


Figure from Karpathy 2015

What do Recurrent Language Models Learn?



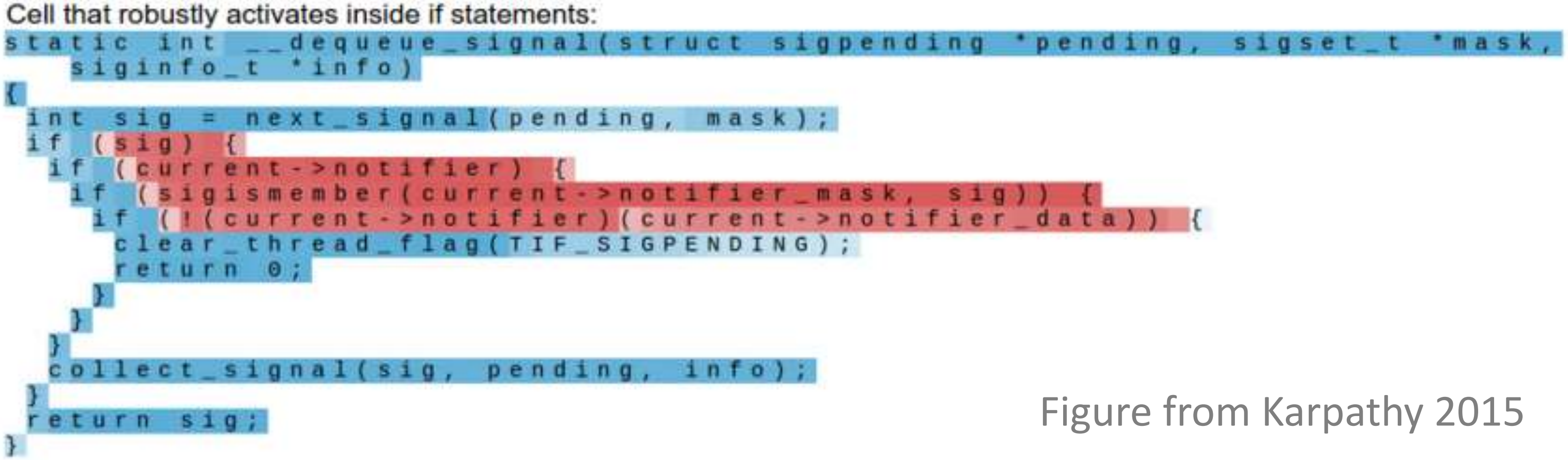


Figure from Karpathy 2015

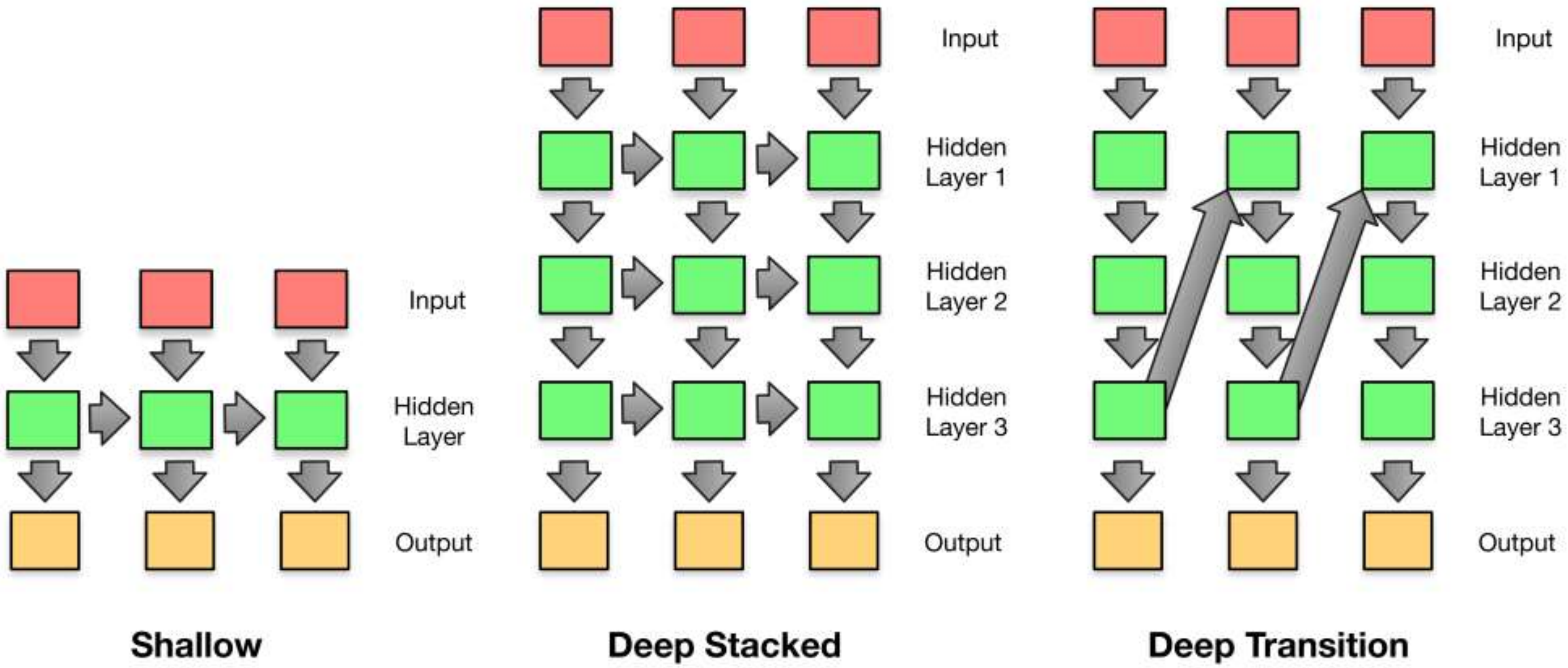
What do Recurrent Language Models Learn?

- Can capture (some) long-distance dependencies

```
After much economic progress over the years, the country has…

The country, which has made much economic progress over the years, still has…
```

Deeper Models



Neural Language Models Summary

- A powerful tool for modeling language
  - Captures generalizations over words via embeddings
  - Captures some long-distance dependencies

- Build on computation graphs
  - some tricks required to train and predict efficiently

- Not just a building block of Neural MT systems
  - Have proved useful in statistical machine translation (Devlin et al. 2014)