Lecture 5.2: Language model

Language models

- Probability of a sentence to be English (or French… or Korean)
- In English: P("sleepy the house in cat") < P("sleepy cat in the house")
- Useful in machine translation: useful for scoring all possible sentences

How do we build language models

- Machine learning
- On monolingual data

…The tale will appeal my children, especially those with a sleepy cat in the house. Those who can read small texts and enjoy illustrations…

- Counting words, bigrams, … ngrams and infer probabilities
- … Or using Neural Network language models

LM using probabilities, how?

- Can we compute *P*("this story will appeal my children, especially as they have a sleepy cat in the house") ? ➔ too costly

$$P(w_1 w_2 \dots w_n) = \prod_{i=1}^{n} P(w_i | w_1 w_2 \dots w_{i-1})$$

Markov assumption (limit the history to *few* words):

e.g. 4:

$$P(w_1 w_2 \dots w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-4} w_{i-3} w_{i-2} w_{i-1})$$

Estimating N-Gram Probabilities

- Maximum likelihood estimation

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

- Collect counts over a large text corpus
- Millions to billions of words are easy to get
(trillions of English words available on the web)

Courtesy of Philipp Koehn, Professor
at The Johns Hopkins University

Example: 3-grams

- Counts for trigrams and estimated word probabilities

the green (total: 1748)

word	c.	prob.
paper	801	0.458
group	640	0.367
light	110	0.063
party	27	0.015
ecu	21	0.012

the red (total: 225)

word	c.	prob.
cross	123	0.547
tape	31	0.138
army	9	0.040
card	7	0.031
,	5	0.022

the blue (total: 54)

word	c.	prob.
box	16	0.296
.	6	0.111
flag	6	0.111
,	3	0.056
angel	3	0.056

- 225 trigrams in the Europarl corpus start with the red
- 123 of them end with cross

$\rightarrow$ maximum likelihood probability is $\frac{123}{225} = 0.547$.

Credits: P. Koehn John Hopkins Univ

- Previously, we approximated

$$p(W) = p(w_1, w_2, ..., w_n)$$

- ... by applying the chain rule

$$p(W) = \sum_i p(w_i | w_1, ..., w_{i-1})$$

- ... and limiting the history (Markov order)

$$p(w_i | w_1, ..., w_{i-1}) \simeq p(w_i | w_{i-4}, w_{i-3}, w_{i-2}, w_{i-1})$$

- Each $p(w_i | w_{i-4}, w_{i-3}, w_{i-2}, w_{i-1})$ may not have enough statistics to estimate

→ we back off to $p(w_i | w_{i-3}, w_{i-2}, w_{i-1})$, $p(w_i | w_{i-2}, w_{i-1})$, etc., all the way to $p(w_i)$

– exact details of backing off get complicated — "interpolated Kneser-Ney"

Credits: P. Koehn John Hopkins Univ

Refinements of back-off

- A whole family of back-off schemes
- Skip-n gram models that may back off to $p(w_i|w_{i-2})$
- Class-based models $p(C(w_i)|C(w_{i-4}), C(w_{i-3}), C(w_{i-2}), C(w_{i-1}))$

$\Rightarrow$ We are wrestling here with

- using as much relevant evidence as possible
- pooling evidence between words

Credits: P. Koehn John Hopkins Univ

How good is the LM?

- A good model assigns a text of real English W a high probability
- This can be also measured with cross entropy:

$$H(W) = \frac{1}{n} \log p(W_1^n)$$

- Or, **perplexity**

$$\text{perplexity}(W) = 2^{H(W)}$$

Credits: P. Koehn John Hopkins Univ

Example: 3 gram

prediction	p_{LM}	$-\log_2 p_{\mathsf{LM}}$
$p_{\mathsf{LM}}(\text{i} \mid </s><s>)$	0.109	3.197
$p_{\mathsf{LM}}(\text{would} \mid <s>\text{i})$	0.144	2.791
$p_{\mathsf{LM}}(\text{like} \mid \text{i would})$	0.489	1.031
$p_{\mathsf{LM}}(\text{to} \mid \text{would like})$	0.905	0.144
$p_{\mathsf{LM}}(\text{commend} \mid \text{like to})$	0.002	8.794
$p_{\mathsf{LM}}(\text{the} \mid \text{to commend})$	0.472	1.084
$p_{\mathsf{LM}}(\text{rapporteur} \mid \text{commend the})$	0.147	2.763
$p_{\mathsf{LM}}(\text{on} \mid \text{the rapporteur})$	0.056	4.150
$p_{\mathsf{LM}}(\text{his} \mid \text{rapporteur on})$	0.194	2.367
$p_{\mathsf{LM}}(\text{work} \mid \text{on his})$	0.089	3.498
$p_{\mathsf{LM}}(. \mid \text{his work})$	0.290	1.785
$p_{\mathsf{LM}}(</s> \mid \text{work .})$	0.99999	0.000014
	average	2.634

Credits: P. Koehn John Hopkins Univ

Count smoothing

- We have seen i like to in our corpus
- We have never seen i like to smooth in our corpus

$\rightarrow p(\text{smooth}|\text{i like to}) = 0$

- Any sentence that includes i like to smooth will be assigned probability 0

Credits: P. Koehn John Hopkins Univ

Count smoothing

- Many methods
  - Add one smoothing
  - **Add-α smoothing**
  - **Deleted estimation**
  - **Good-Turing smoothing**
  - **Katz smoothing**
  - **…**

  - **See http://www.cs.cornell.edu/courses/cs674/2005sp/Handouts/DShultz-chen-goodman-smoothing.pdf**

Credits: P. Koehn John Hopkins Univ

Back-off

- In given corpus, we may never observe
  - Scottish beer drinkers
  - Scottish beer eaters
- Both have count 0

  → our smoothing methods will assign them same probability
- Better: backoff to bigrams:
  - beer drinkers
  - beer eaters

Credits: P. Koehn John Hopkins Univ

Interpolation

- Higher and lower order n-gram models have different strengths and weaknesses
  - high-order n-grams are sensitive to more context, but have sparse counts
  - low-order n-grams consider only very limited context, but have robust counts

- Combine them

$$\begin{aligned} p_I(w_3|w_1,w_2) = \; & \lambda_1 \; p_1(w_3) \\ & + \lambda_2 \; p_2(w_3|w_2) \\ & + \lambda_3 \; p_3(w_3|w_1,w_2) \end{aligned}$$

Interpolation and backoff techniques:

- Good Turing
- Witten-Bell
- Kneser-Ney

Credits: P. Koehn John Hopkins Univ

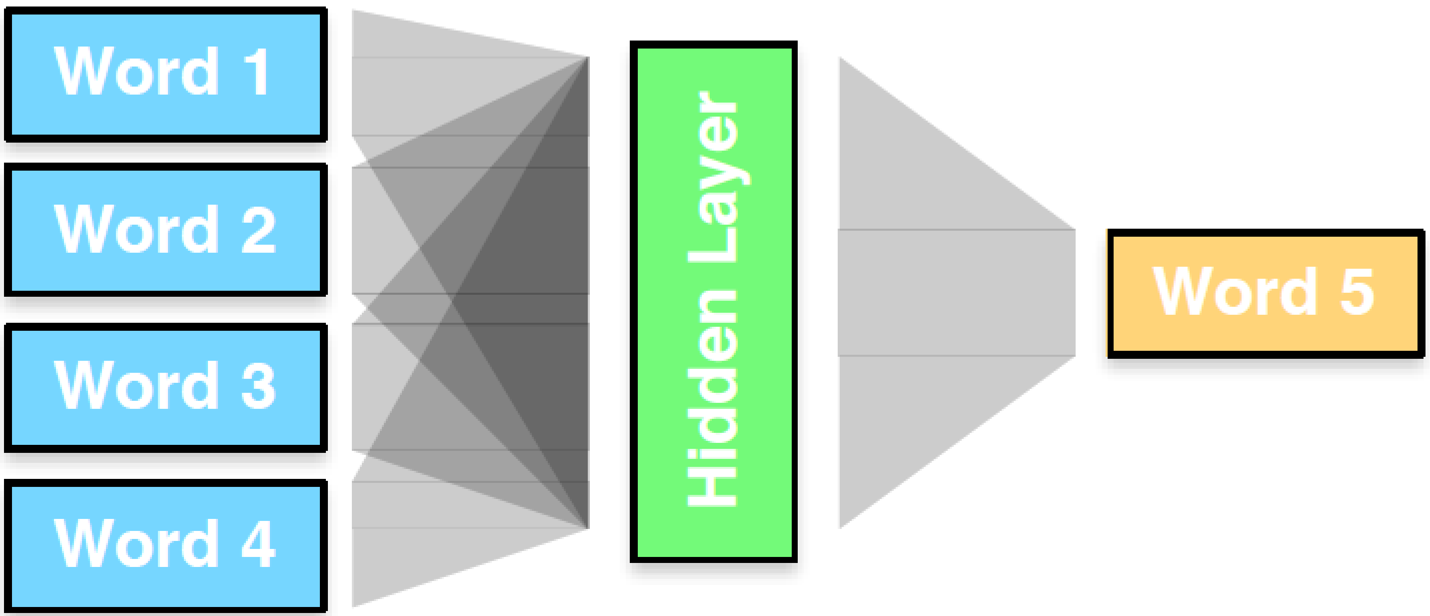
Size of the model

- Millions of (English) words are easy to get
- But building 7-gram models on big corpus needs huge amount of RAM
- Solutions:
  - Reduce *n*
  - Store on disk (e.g. using kenlm software)

    ➔In Wipo a 5-gram English model would require 23Gb or RAM, with kenlm, it now needs 4Gb
  - …

Neural language models

- Idea: generate the 5th word knowing the preceding 4



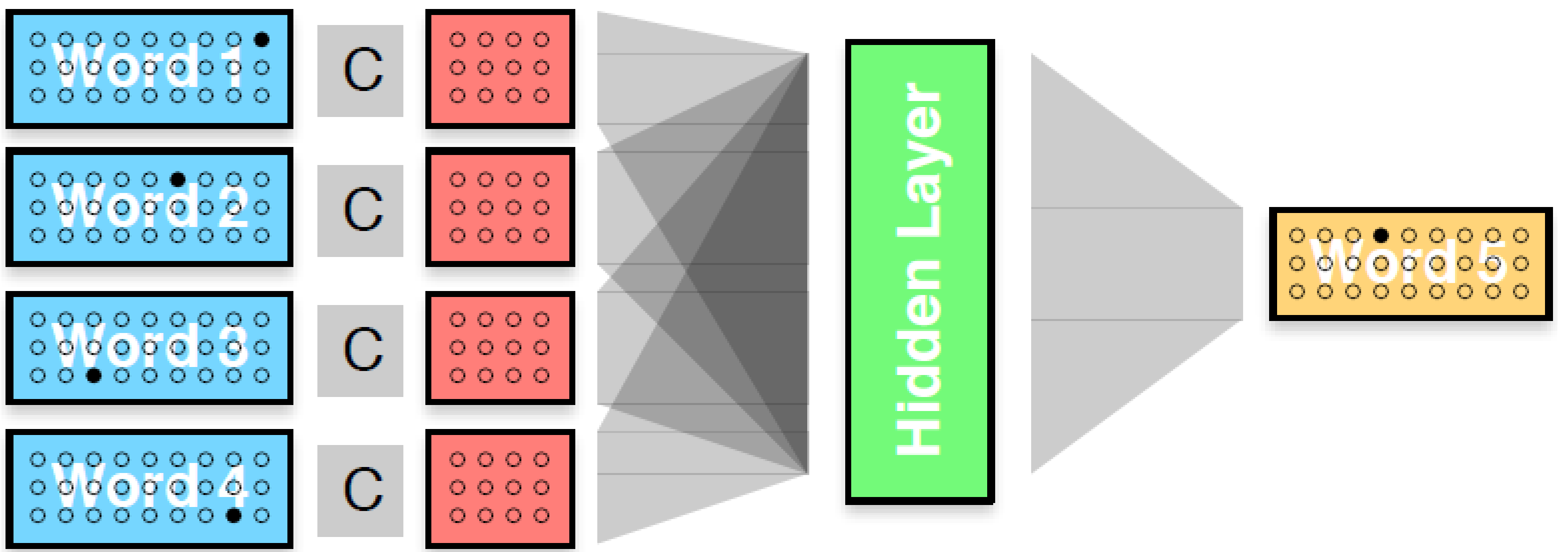
Credits: P. Koehn John Hopkins Univ

- Neural networks work with vectors…
  - Words are represented with a one-hot vector, e.g.,
    - dog = (0,0,0,0,1,0,0,0,0,....)
    - cat = (0,0,0,0,0,0,0,1,0,....)
    - eat = (0,1,0,0,0,0,0,0,0,....)
  - That's a large vector!
  - Remedies
    - limit to, say, 20,000 most frequent words, rest are OTHER
    - place words in $\sqrt{n}$ classes, so each word is represented by
      - 1 class label
      - 1 word in class label

Credits: P. Koehn John Hopkins Univ

Word embeddings

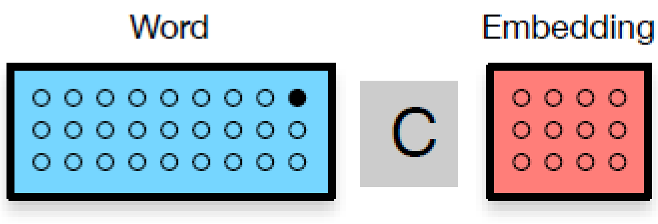
Add a new hidden layer, contains smaller vector representative of each word, learned by NN



- Map each word first into a lower-dimensional real-valued space
- Shared weight matrix C

Credits: P. Koehn John Hopkins Univ

Embeddings...

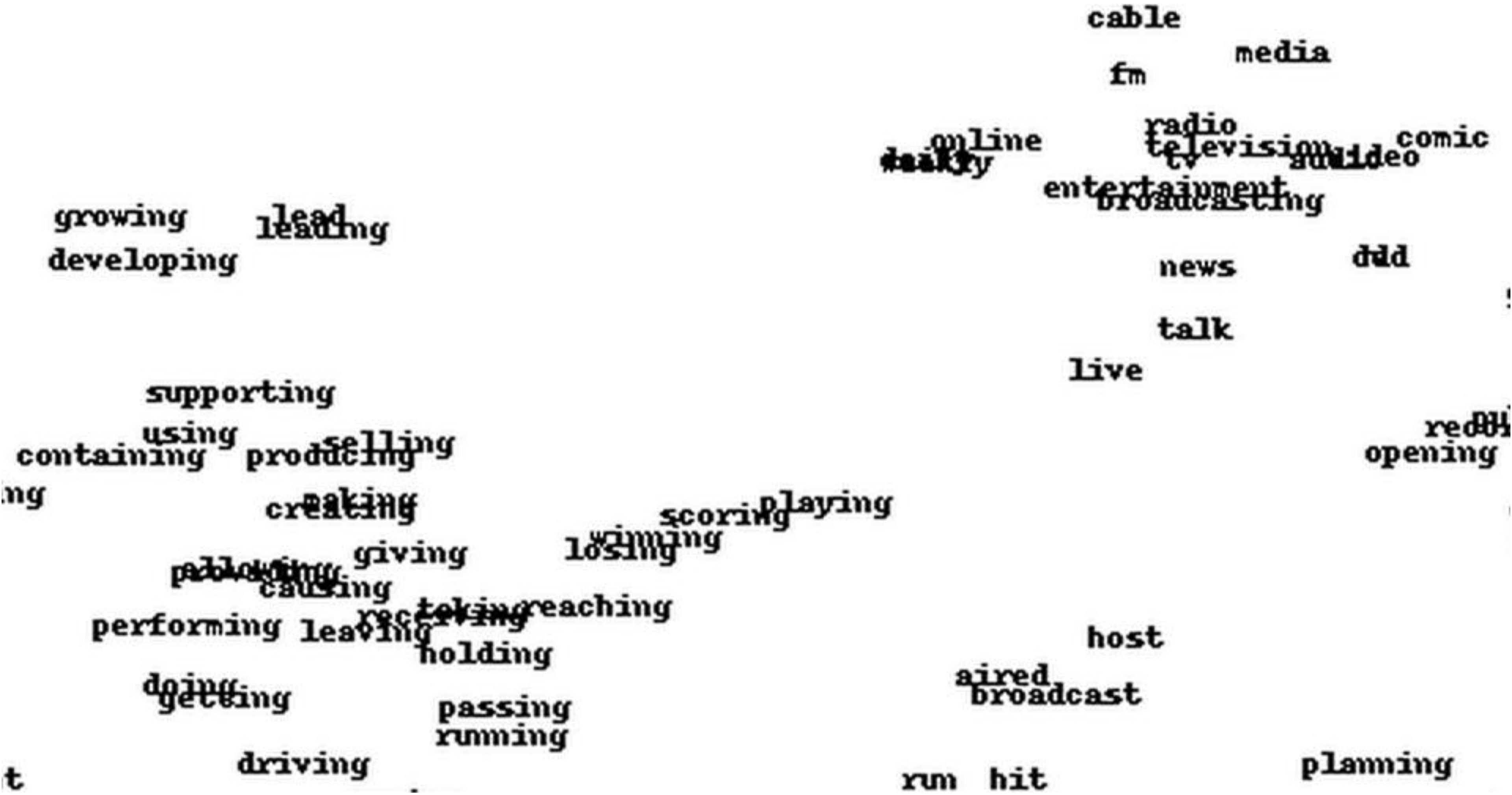


- By-product: embedding of word into continuous space
- Similar contexts → similar embedding
- Recall: distributional semantics

Credits: P. Koehn John Hopkins Univ

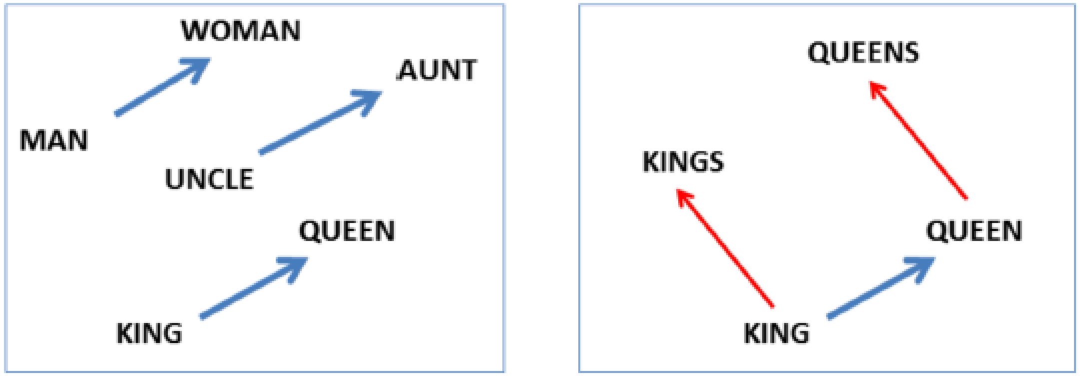
Embedding have

Credits: P. Koehn John Hopkins Univ



Credits: P. Koehn John Hopkins Univ

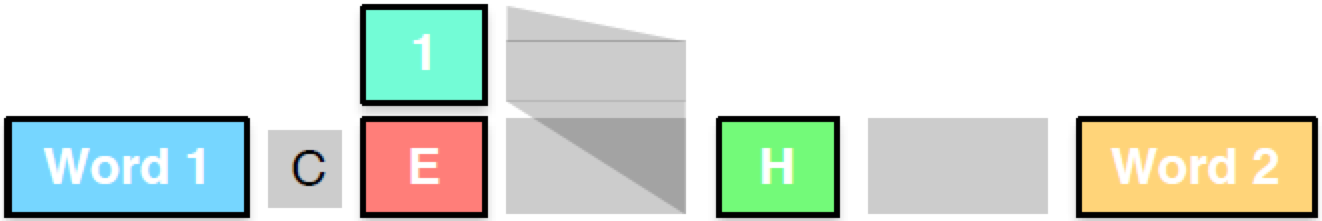
Embeddings contain a lot of information



- Morphosyntactic regularities (Mikolov et al., 2013)
  - adjectives base form vs. comparative, e.g., good, better
  - nouns singular vs. plural, e.g., year, years
  - verbs present tense vs. past tense, e.g., see, saw
- Semantic regularities
  - clothing is to shirt as dish is to bowl
  - evaluated on human judgment data of semantic similarities

Credits: P. Koehn John Hopkins Univ

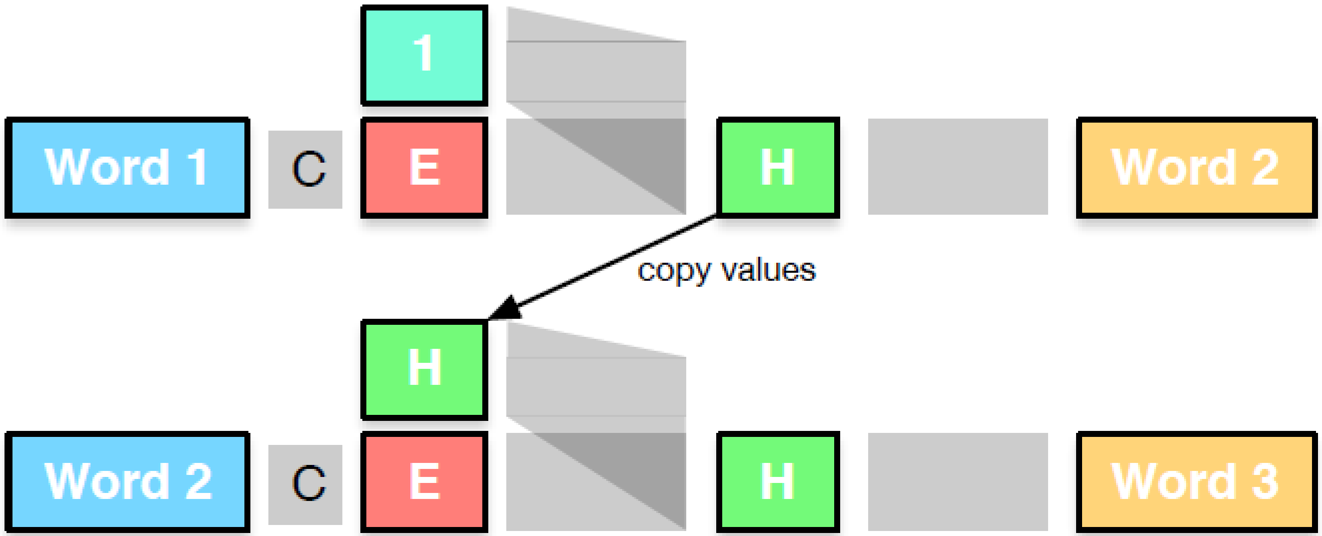
Reccurent neural networks



- Start: predict second word from first
- Mystery layer with nodes all with value 1

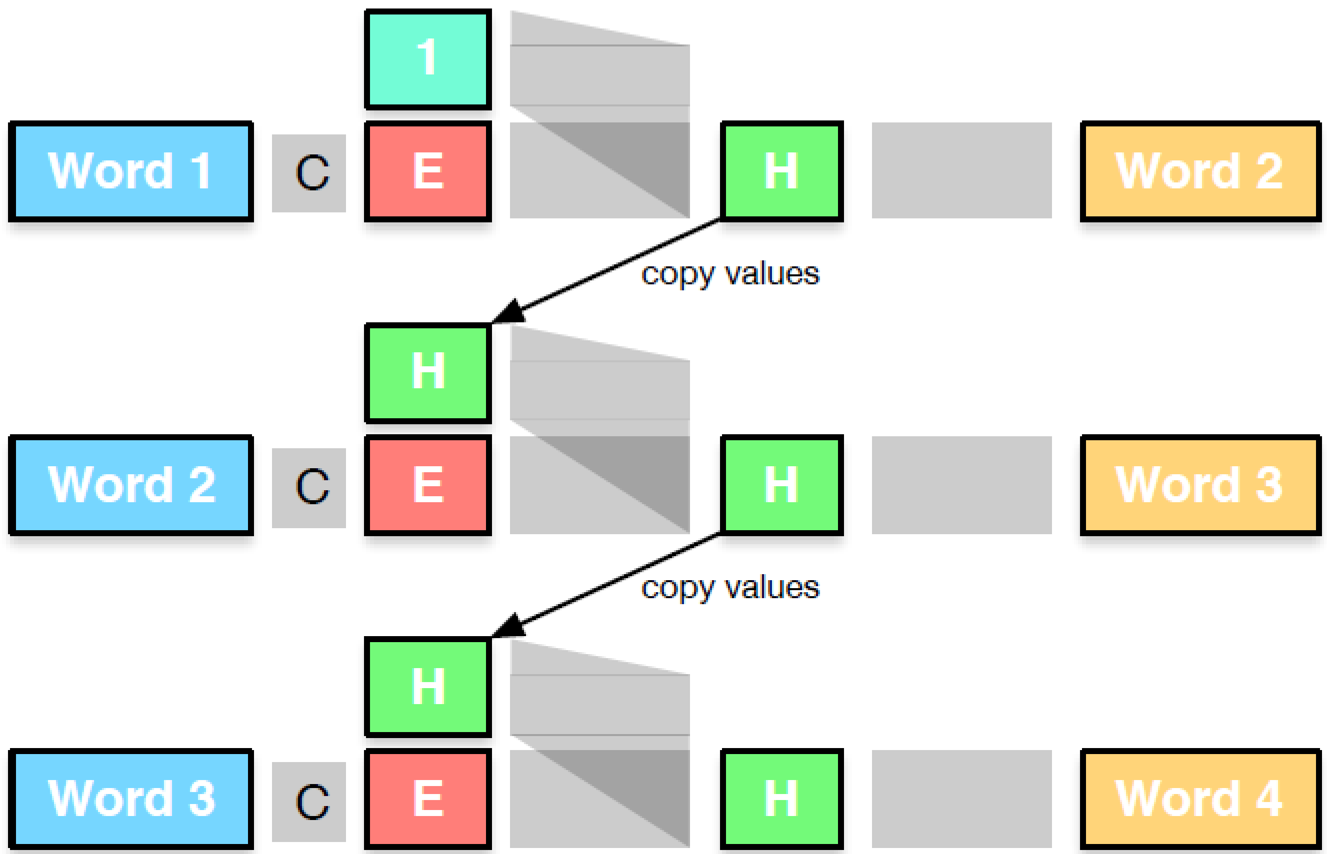
Credits: P. Koehn John Hopkins Univ

Reccurent neural networks



Credits: P. Koehn John Hopkins Univ

Reccurent neural networks



Credits: P. Koehn John Hopkins Univ

- Use LSTMs instead of “simple” RNNs to keep memory of past words
- These RNN LMs can handle sequences of *n* words while still “remembering” the first(s) word(s)