CS 378: Natural Language Processing

Lecture 12: Language Modeling

TEXAS
The University of Texas at Austin

Eunsol Choi

Slides adapted from Greg Durrett, Yoav Artzi, Yejin Choi, Princeton NLP

Language Model

- Why should we care?
- N-gram models
- Evaluating language models
- Preview: Neural language model

Recap: The Language Modeling task

- Setup: Assume a (finite) vocabulary of words, (infinite) set of sentences.

$$\mathcal{V} = \{\text{the}, \text{a}, \text{man}, \text{telescope}, \text{Beckham}, \text{two}, \text{Madrid}, ...\}$$

$$\mathcal{V}^\dagger = \{\text{the}, \text{a}, \text{the a}, \text{the fan}, \text{the man}, \text{the man with the telescope}, ...\}$$

- Data: a training set of example sentences
- Task: estimate a probability distribution over sentences

$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1$$

and $p(x) \geq 0$ for all $x \in \mathcal{V}^\dagger$

$$p(\text{the}) = 10^{-12}$$
$$p(\text{a}) = 10^{-13}$$
$$p(\text{the fan}) = 10^{-12}$$
$$p(\text{the fan saw Beckham}) = 2 \times 10^{-8}$$
$$p(\text{the fan saw saw}) = 10^{-15}$$
$$\ldots$$

Recap: N-Gram language model

- n-gram models: distribution of next word is a multinomial conditioned on previous n-1 words

$$p(x_i \mid x_1, \ldots x_{i-1}) = P(x_i \mid x_{i-n+1} \ldots x_{i-1})$$

Unigram $$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$$

Bigram $$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid x_{i-1})$$

and Trigram, 4-gram, and so on.

Instapoll

Given a unigram language model of the English language, which is estimated on raw data without any preprocessing except splitting on spaces for tokenization. The probability of P(the the the the dog) > P(the the the).

Given an audio signal A, we want to transcribe it to generate text X. The generative noisy signal model will decompose into two distributions (keep in mind that Bayes' rule was used in the decomposition):

Evaluating Language Model

Language Model Evaluation

- What we would like:
  - Would the model prefer good sentences to bad ones?
  - Bad ≠ ungrammatical!
  - Bad ≈ unlikely

Measuring the Model Quality

- The Shannon Game:
  - How well can we predict the next word?

    When I eat pizza, I wipe off the ____

    Many children are allergic to ____

    I saw a ____

grease 0.5
sauce 0.4
dust 0.05
….
mice 0.0001
….
the 1e-100

Claude Shannon

- How good are we doing?
- Compute per word log likelihood (total n words):

$$l = \frac{1}{n}\sum_{i=1}^{n} \log P(x_i \,|\, x_1, x_2 \dots x_{i-1})$$

Intrinsic Measure: Perplexity

- Evaluate LMs on the ***log likelihood*** of held-out data (averaged to normalize for length)

$$l = \frac{1}{n}\sum_{i=1}^{n} \log P(x_i \,|\, x_1, x_2 \ldots x_{i-1})$$

- Perplexity: **Lower is better!**

$$PP = 2^{-l}$$

Shannon Game intuition for perplexity

- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9' at random
  - Perplexity 10
- How hard is recognizing (30,000) names at random
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53
- Perplexity is weighted equivalent branching factor

$$PP = 2^{-\frac{1}{M}\sum_{i=1}^{m} \log_2 (\frac{1}{10})^{|X^{(i)}|}}$$

$$= 2^{-\frac{1}{M}\sum_{i=1}^{m} |X^{(i)}| \log_2 \frac{1}{10}}$$

$$= 2^{-\log_2 \frac{1}{10}} = 2^{-\log_2 10^{-1}} = 10$$

Intrinsic Measure: Perplexity

- Evaluate LMs on the ***log likelihood*** of held-out data (averaged to normalize for length)

$$l = \frac{1}{n}\sum_{i=1}^{n} \log P(x_i \mid x_1, x_2 \ldots x_{i-1})$$

- Perplexity: **Lower is better!**

$$PP = 2^{-l}$$

Perplexity

- Language with higher perplexity —> Language with high branching factor

- The difference between model's perplexity and the true perplexity of language estimates the quality of the model

Perplexity

- Would it be possible to cheat this measure?

- What would happen if we ever give a test n-gram zero probability?

$$l = \frac{1}{n} \sum_{i=1}^{n} \log P(w_i | w_1, \ldots, w_{i-1}) \qquad PP = 2^{-l}$$

Zeros

- Training set:

… denied the allegations
… denied the reports
… denied the claims
… denied the request

P("offer" | denied the) = 0

- Test set

… denied the offer
… denied the loan

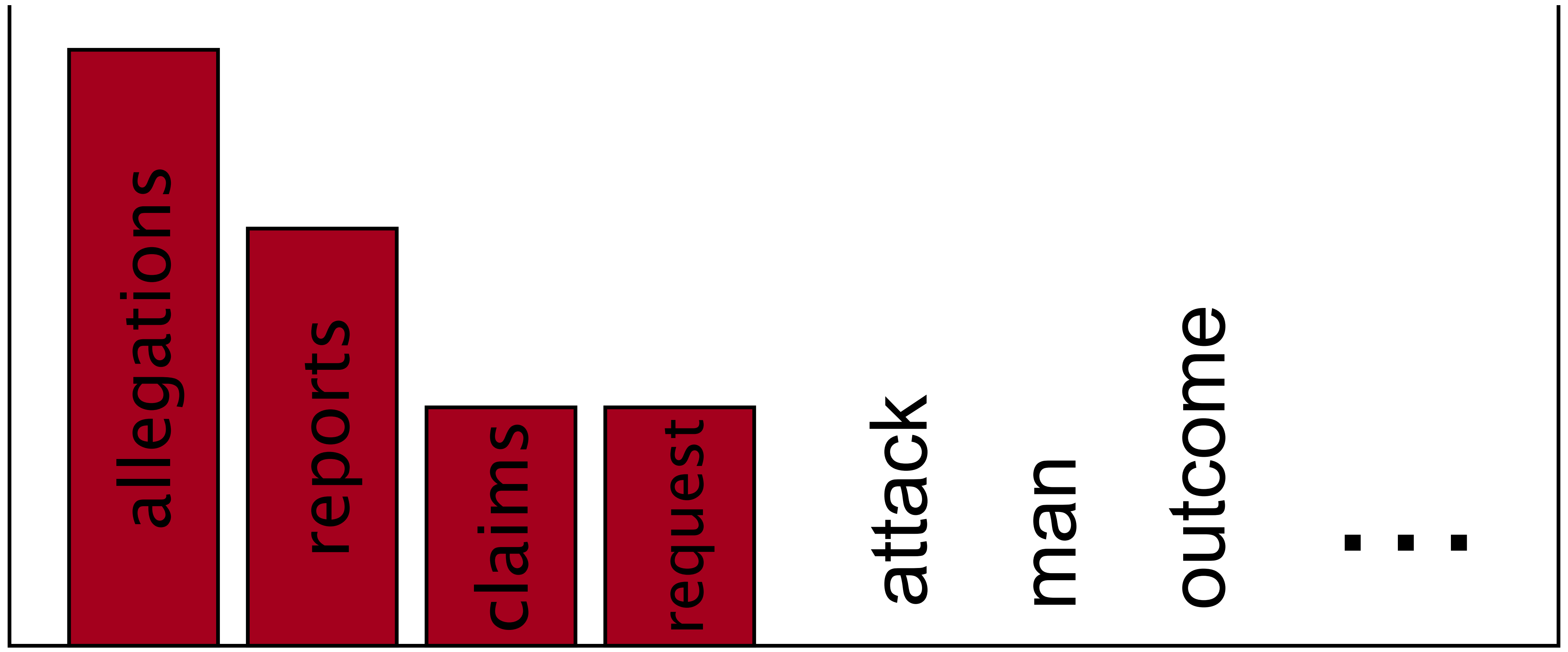
Another reason why smoothing is important!

Smoothing

When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

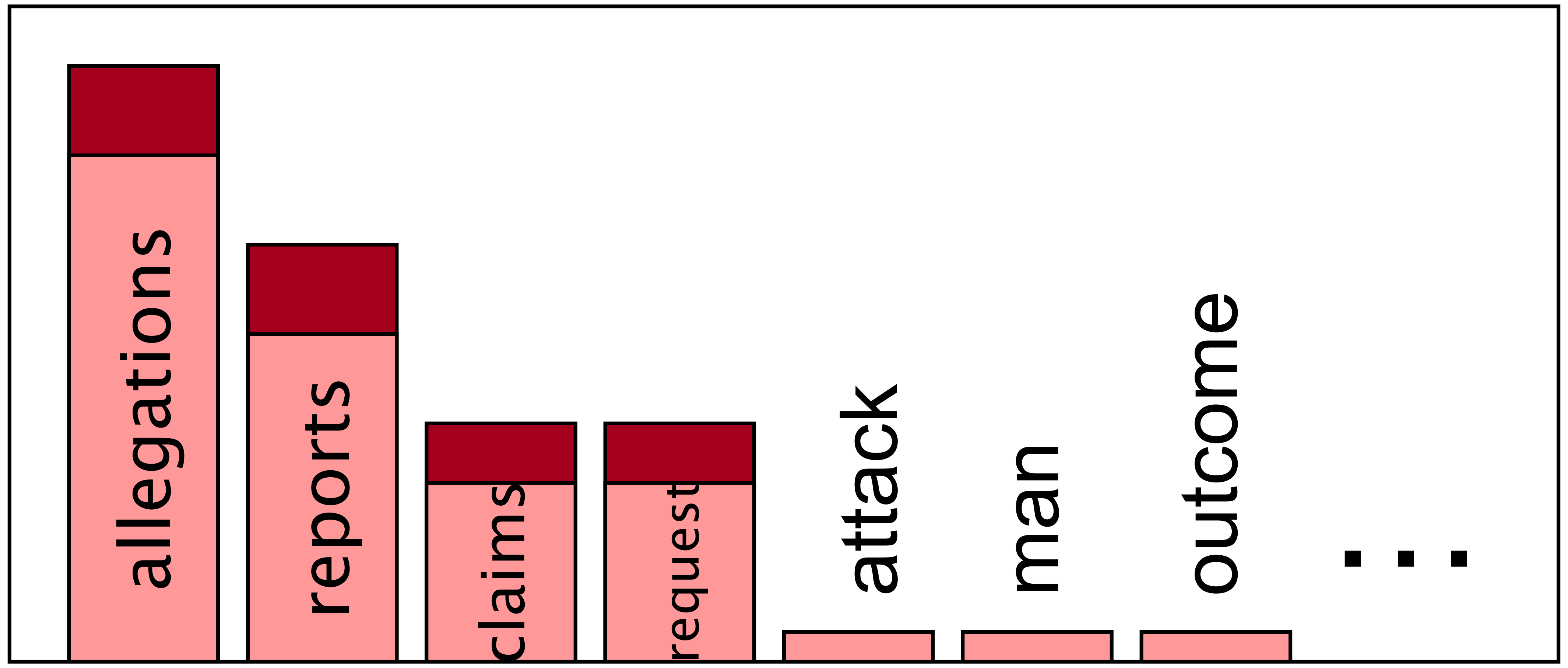
7 total



Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



Slide credit: Dan Klein

Perplexity

Pros	Cons
Easy to compute	Domain match between train and test
standardized (as long as you keep same vocab!)	Limited to sequence models
nice theoretical interpretation - matching distributions	might not correspond to end task performances
	log 0 undefined
	can be cheated by predicting common tokens

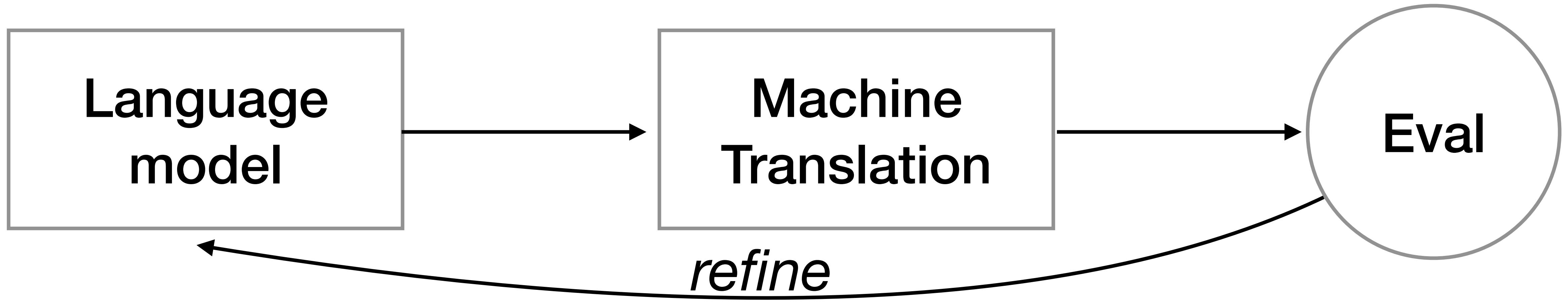
Extrinsic vs. Intrinsic

- How well this language model fit into the end tasks?

- Perplexity - model's estimate of the probability of unseen text

Extrinsic Measure

- Application driven



- Ex: Speech recognition
  - Word Error Rate (WER):

$$\frac{\text{insertions} + \text{deletions} + \text{substitutions}}{\text{true sentence size}}$$

Correct answer: Andy saw a part of the movie

Recognizer output: And he saw apart of the movie

WER: 4/7
= 57%

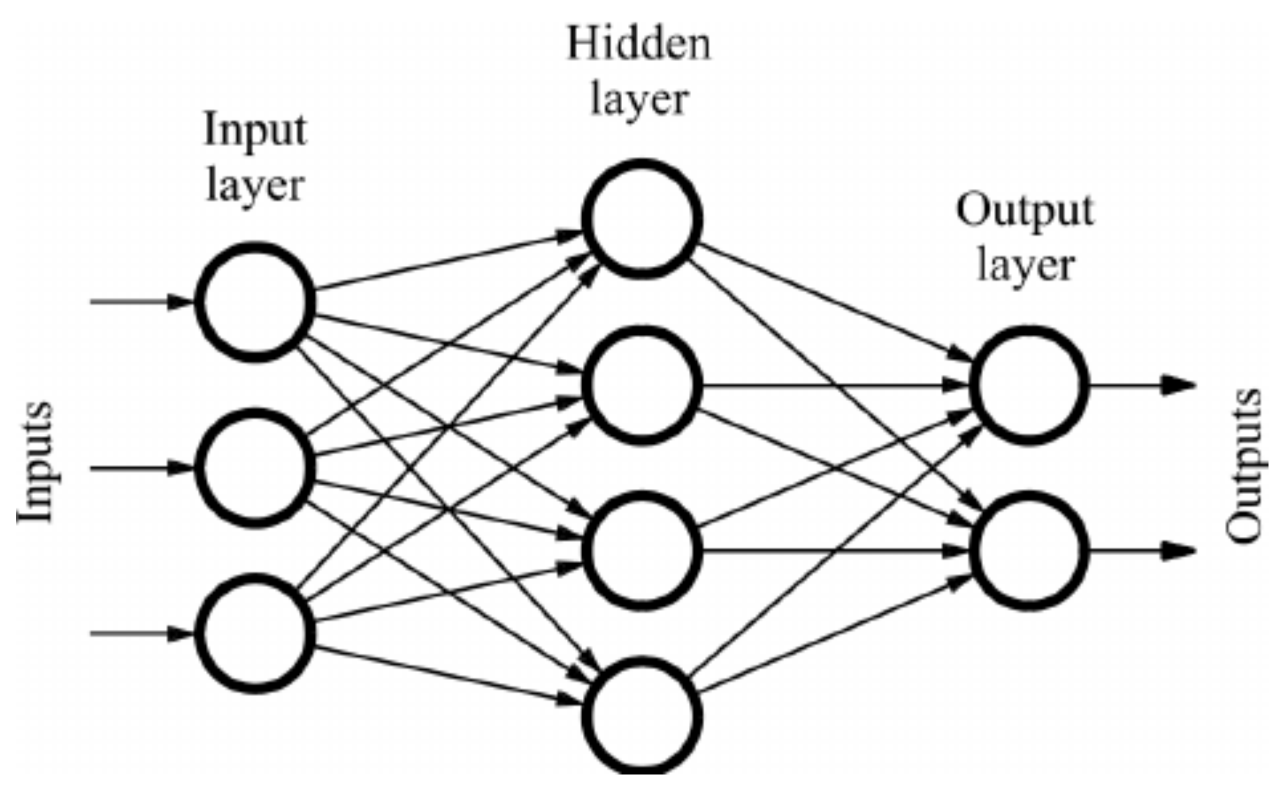
Limitations of N-gram models

- Computationally quite expensive
  - Memory requirements
- Does not consider context beyond N preceding words

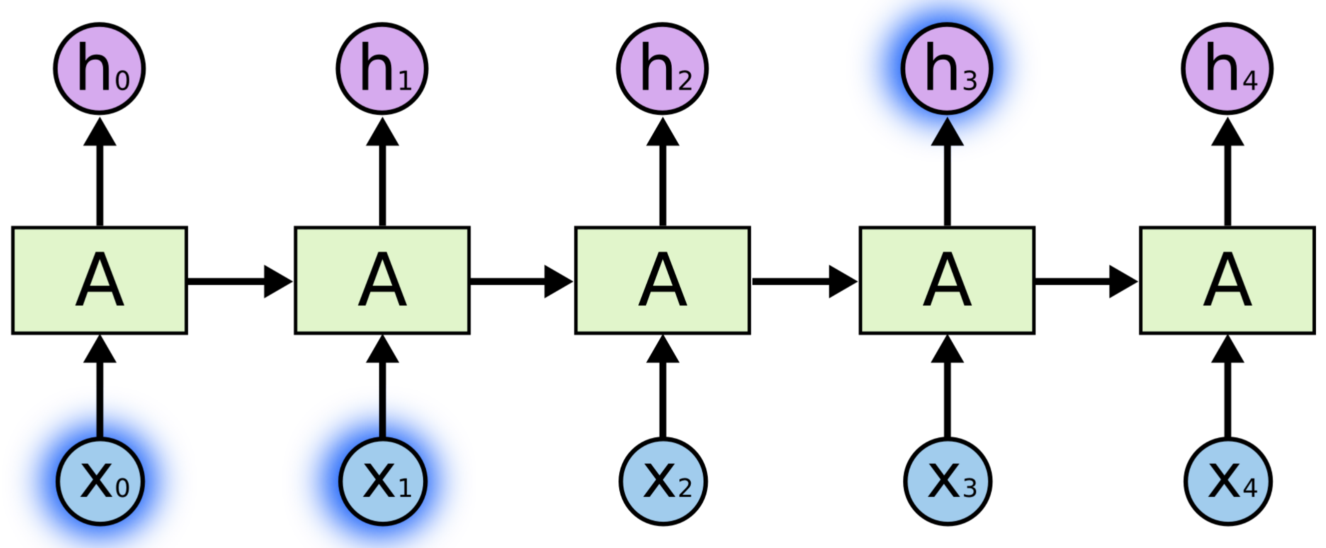
Preview: Neural Language Models

Neural Networks in NLP

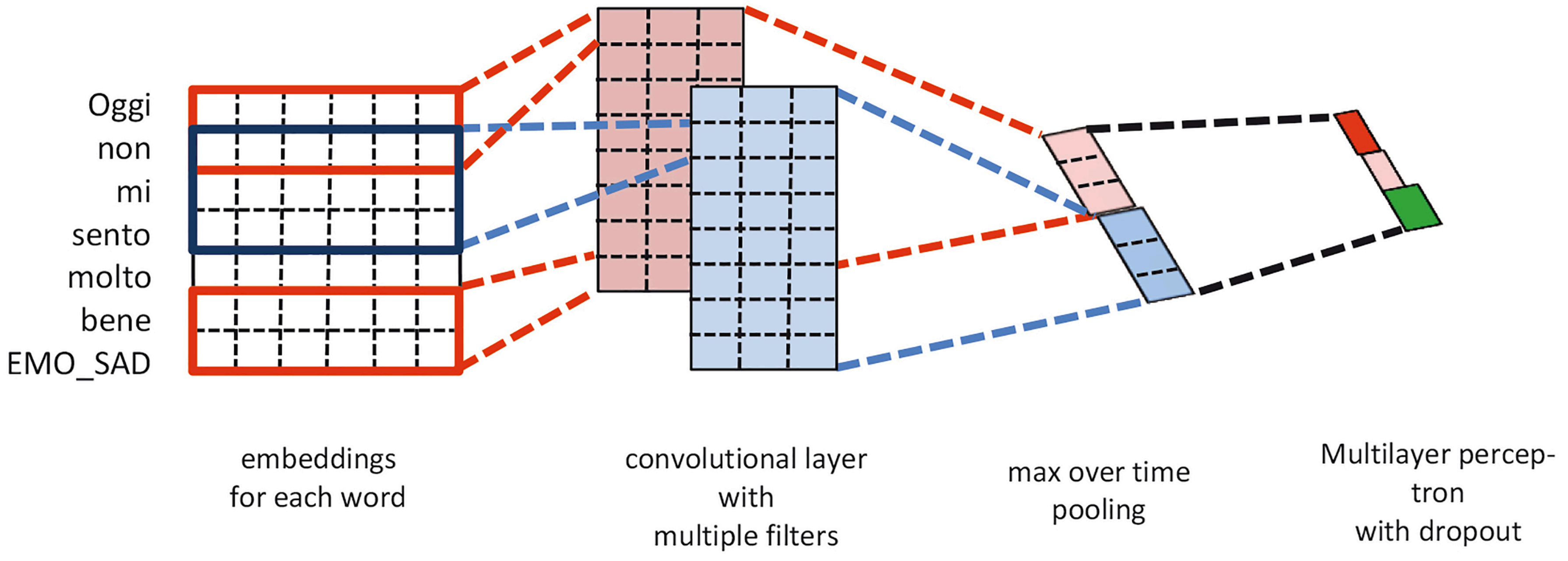
Feed-forward NNs



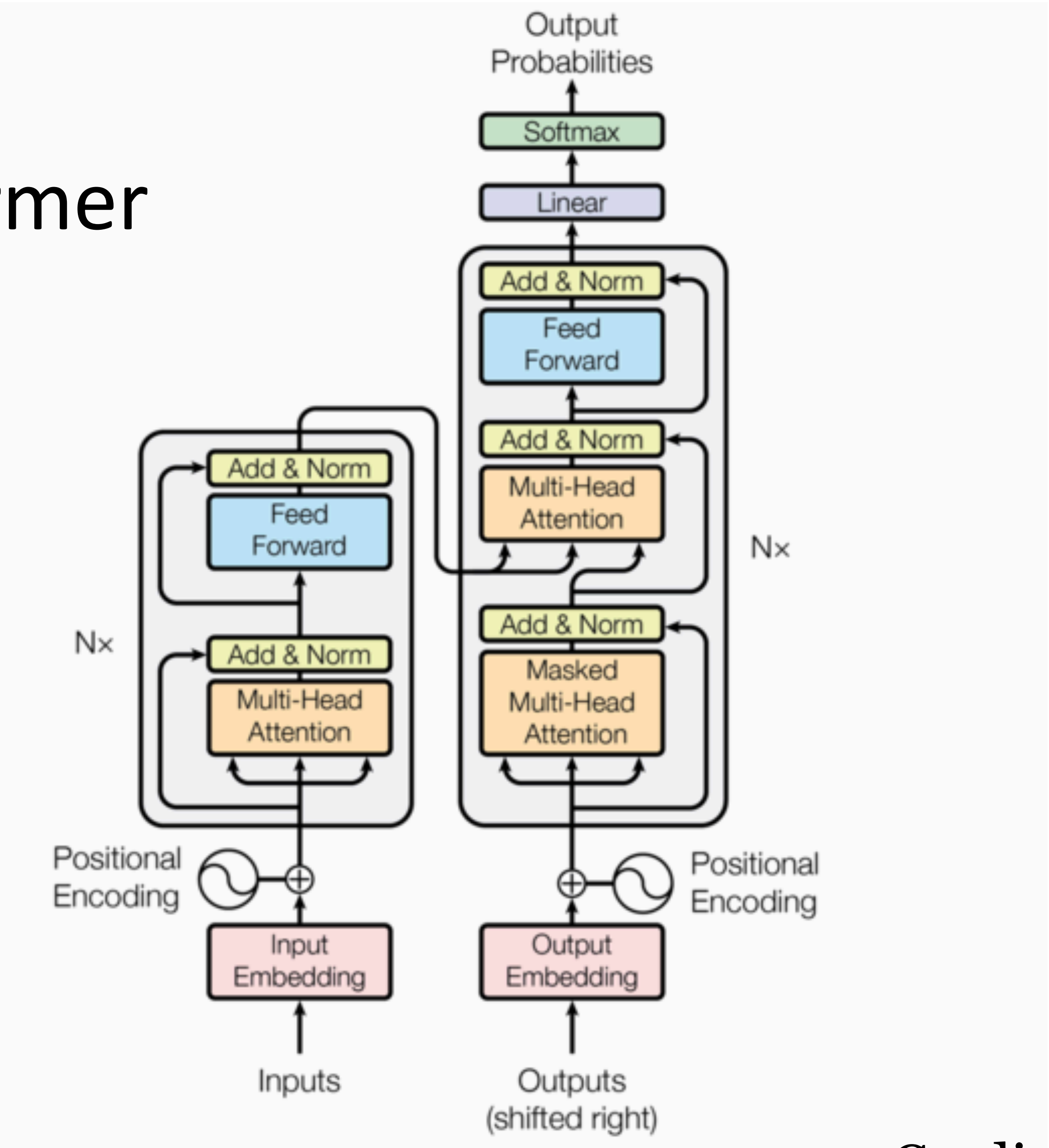
Recurrent NNs



Convolutional NNs



Transformer

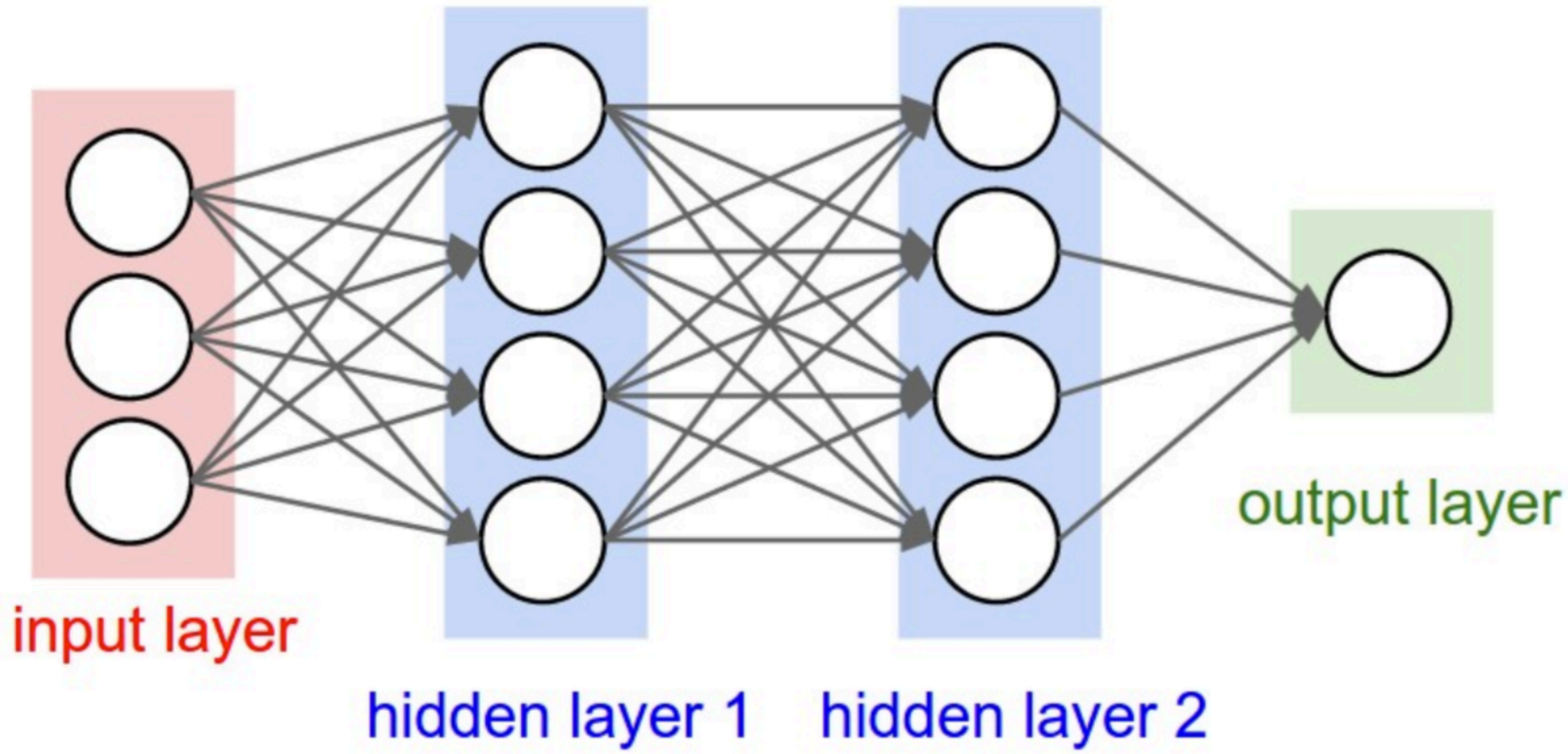


Always coupled with word embeddings…

Credits: Princeton NLP course

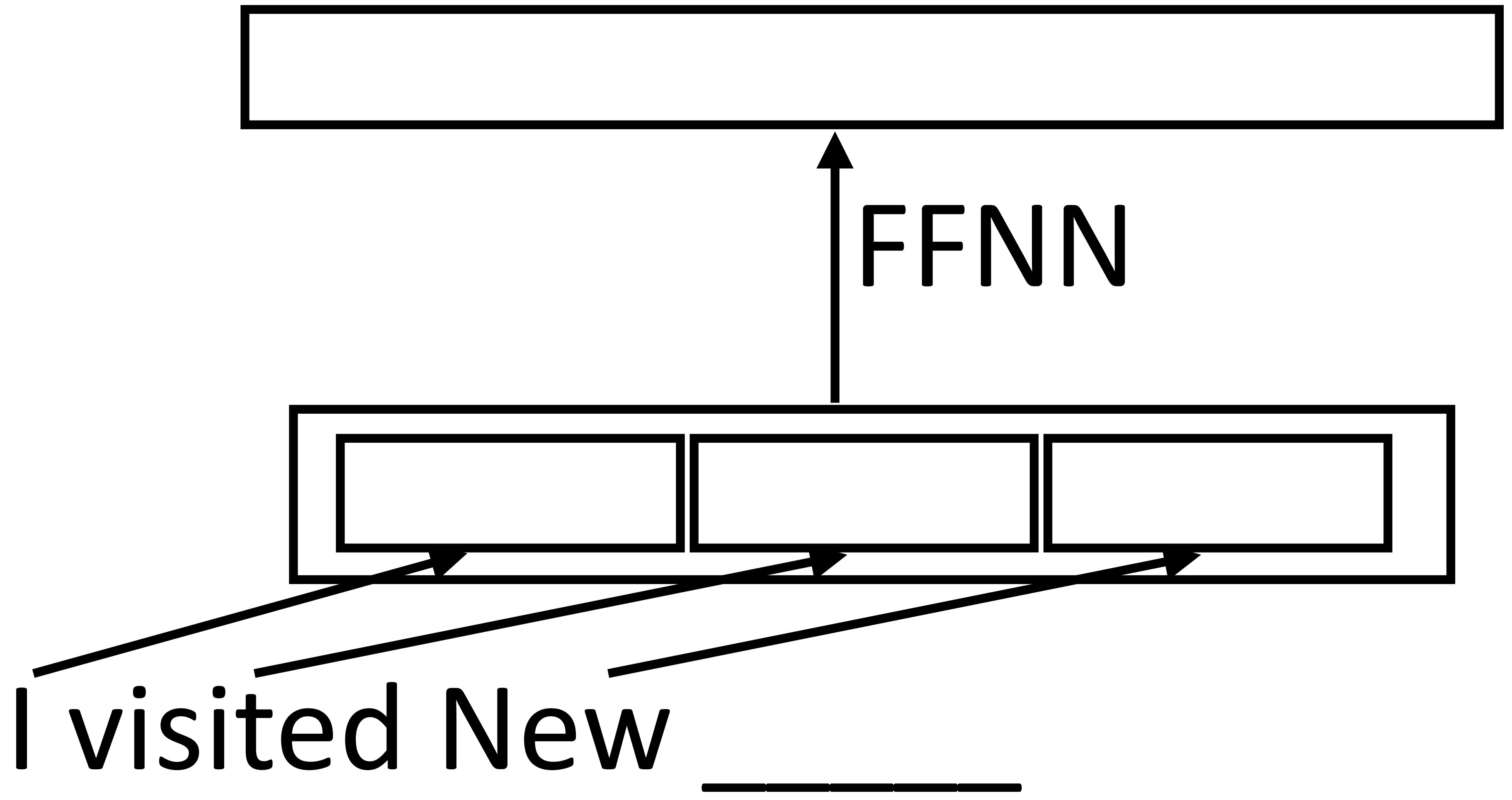
A feed-forward neural network

- If we feed inputs through multiple logistic regression functions, then we can construct a output vector…
- which we can feed into another logistic regression function as an input.



Neural Language Models

- Early work: feedforward neural networks looking at context



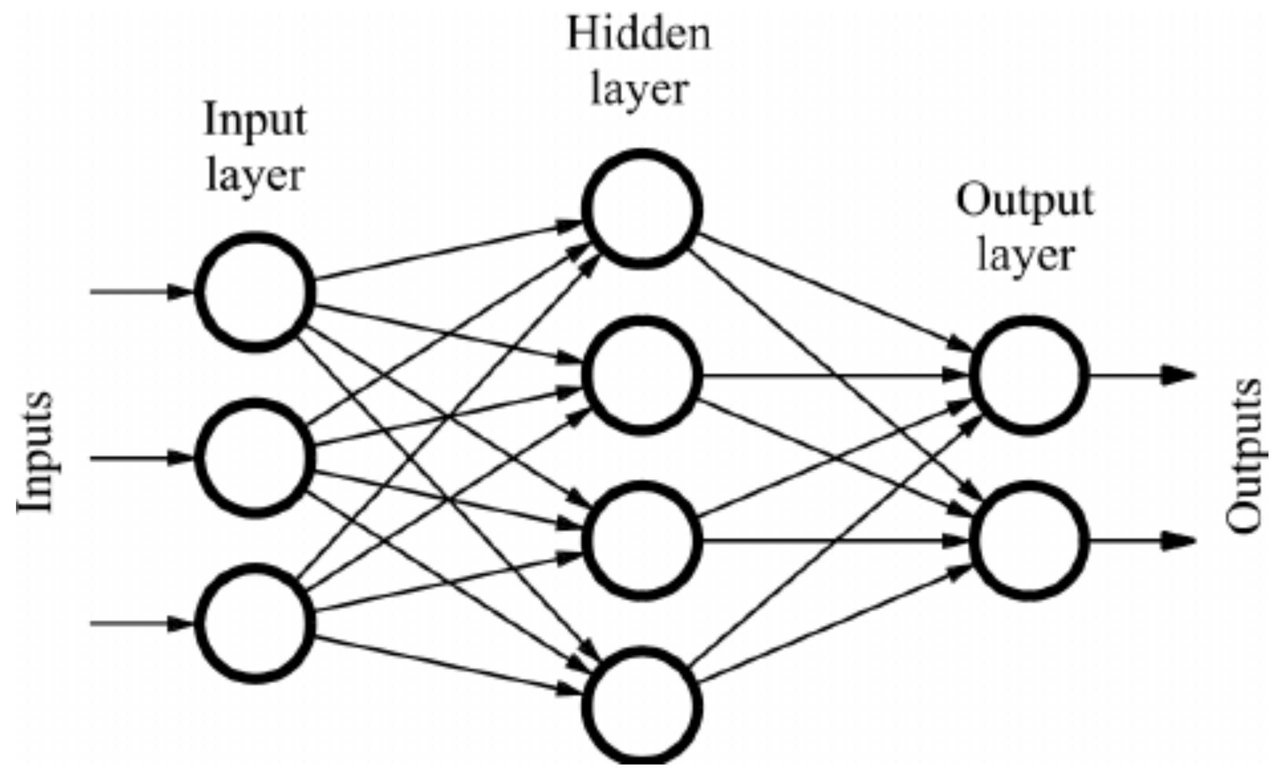
$$P(x_i \,|\, x_{i-n+1}, x_{i-n+2} \ldots, x_{i-1})$$

- Slow to train over lots of data!
- Still only look at a fixed window of information…can we use more?

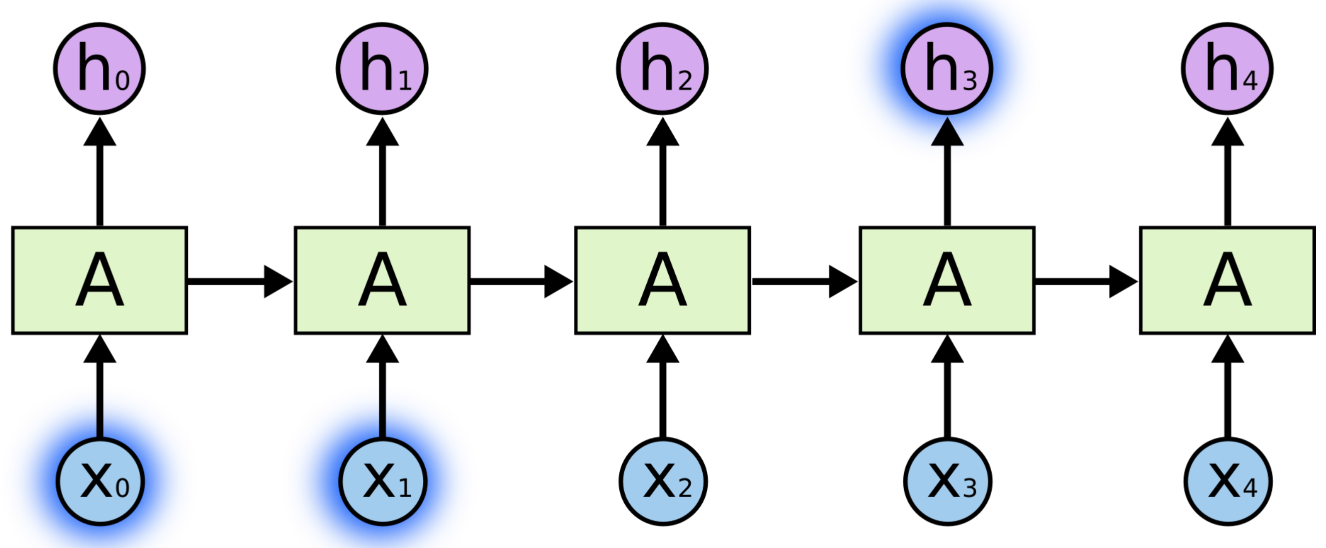
Mnih and Hinton (2003)

Neural Networks in NLP

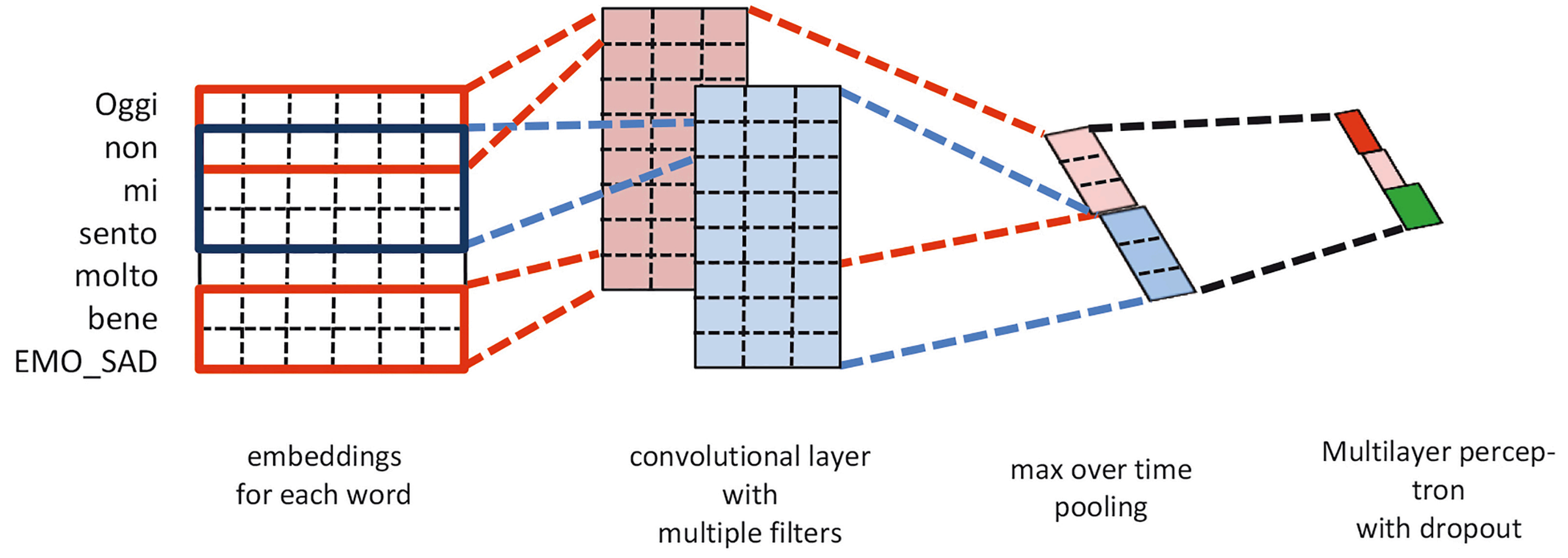
Feed-forward NNs



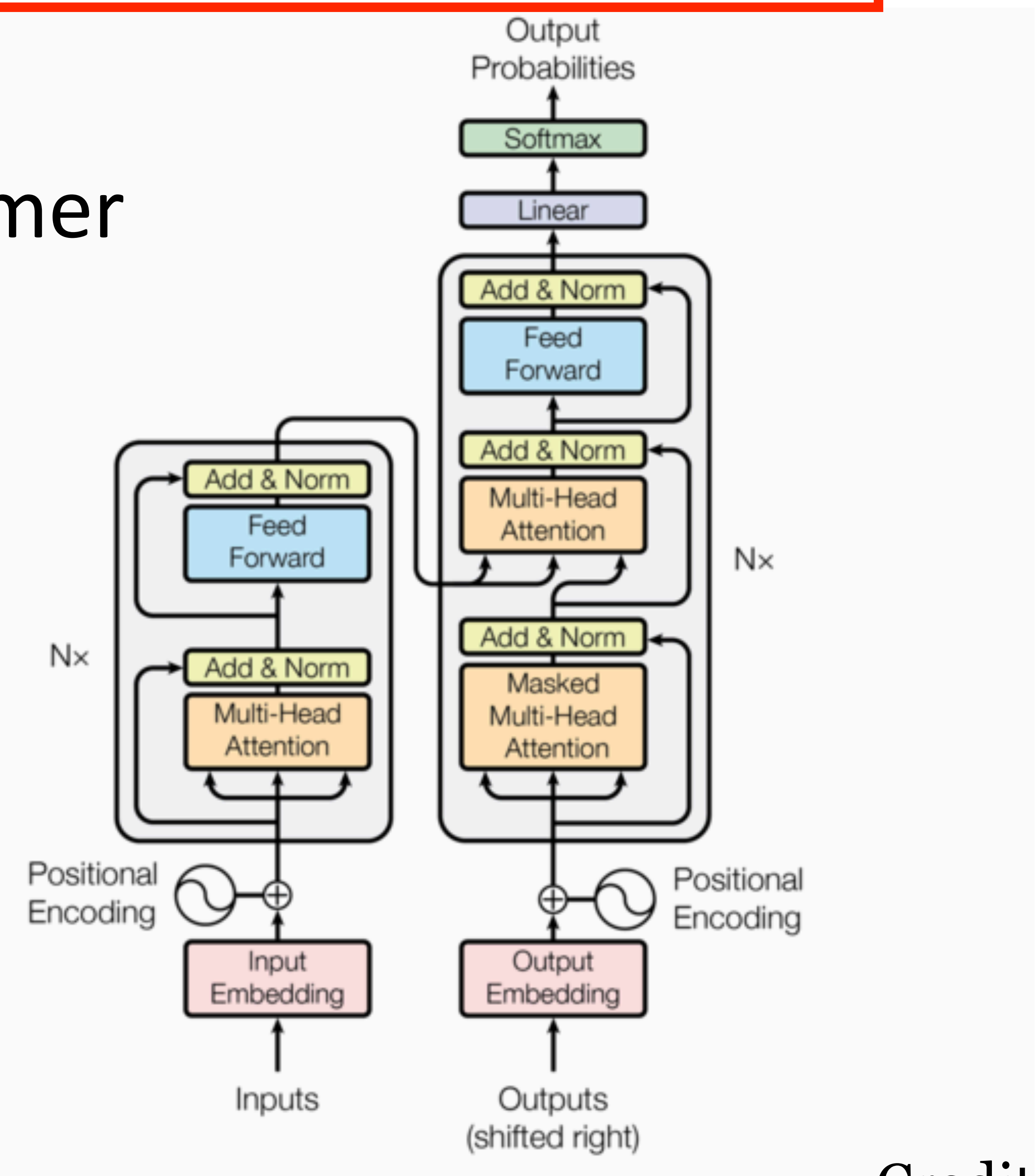
Recurrent NNs



Convolutional NNs



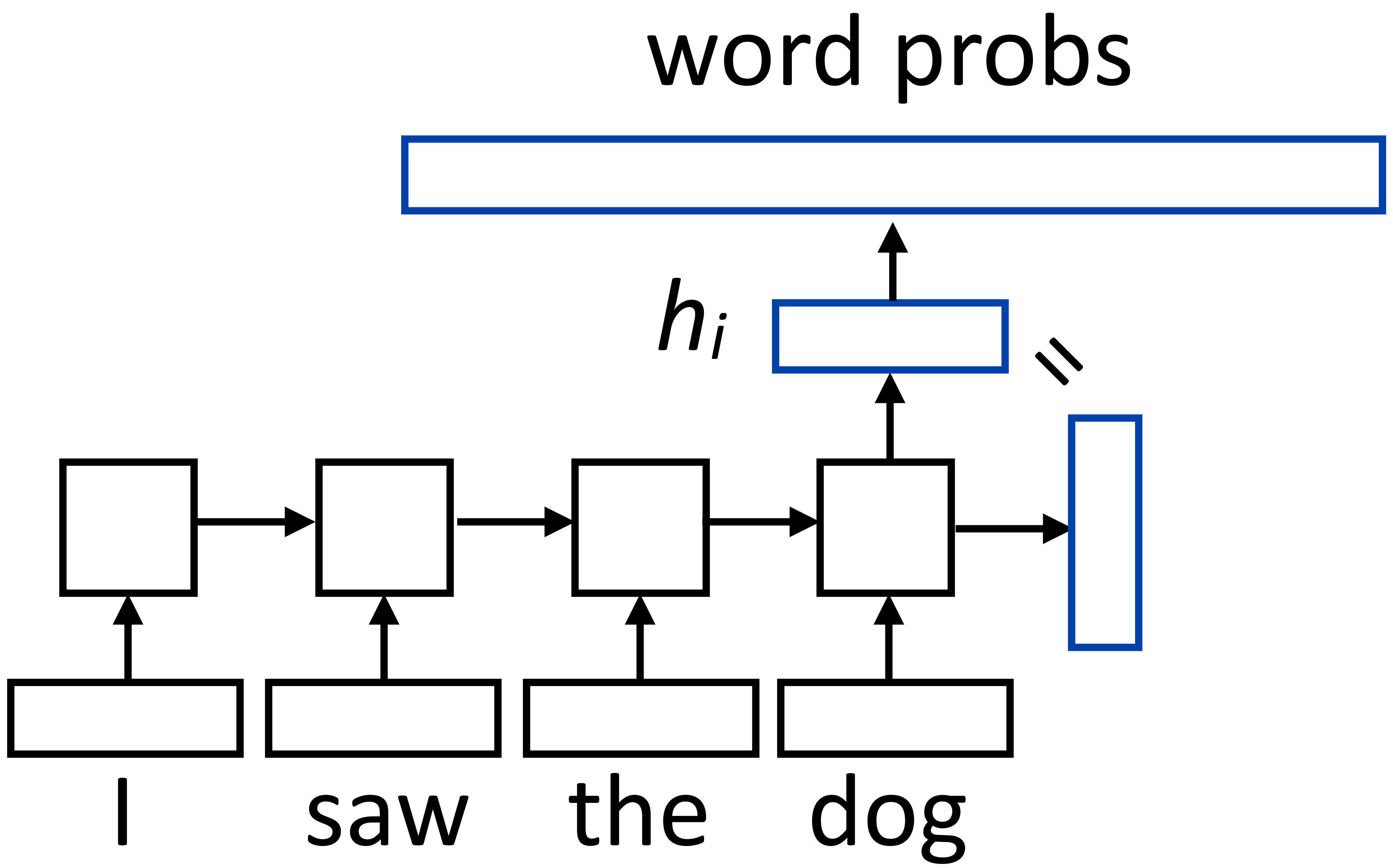
Transformer



Always coupled with word embeddings…

Credits: Princeton NLP course

RNN Language Models



$$P(w|\text{context}) = \text{softmax}(W\mathbf{h}_i)$$

- W is a (vocab size) x (hidden size) matrix

Results

- Evaluate on Penn Treebank: small dataset (1M words)
- Kneser-Ney 5-gram model with cache: PPL = 125.7
- RNN: PPL ~ 60-80 (depending on how much you optimize it)
- RNN character-level: PPL ~1.5 (205 character vocab)

Merity et al. (2017), Melis et al. (2017)

CS378: Natural Language Processing

Lecture 13: Neural Network (Sequence)

The University of Texas at Austin

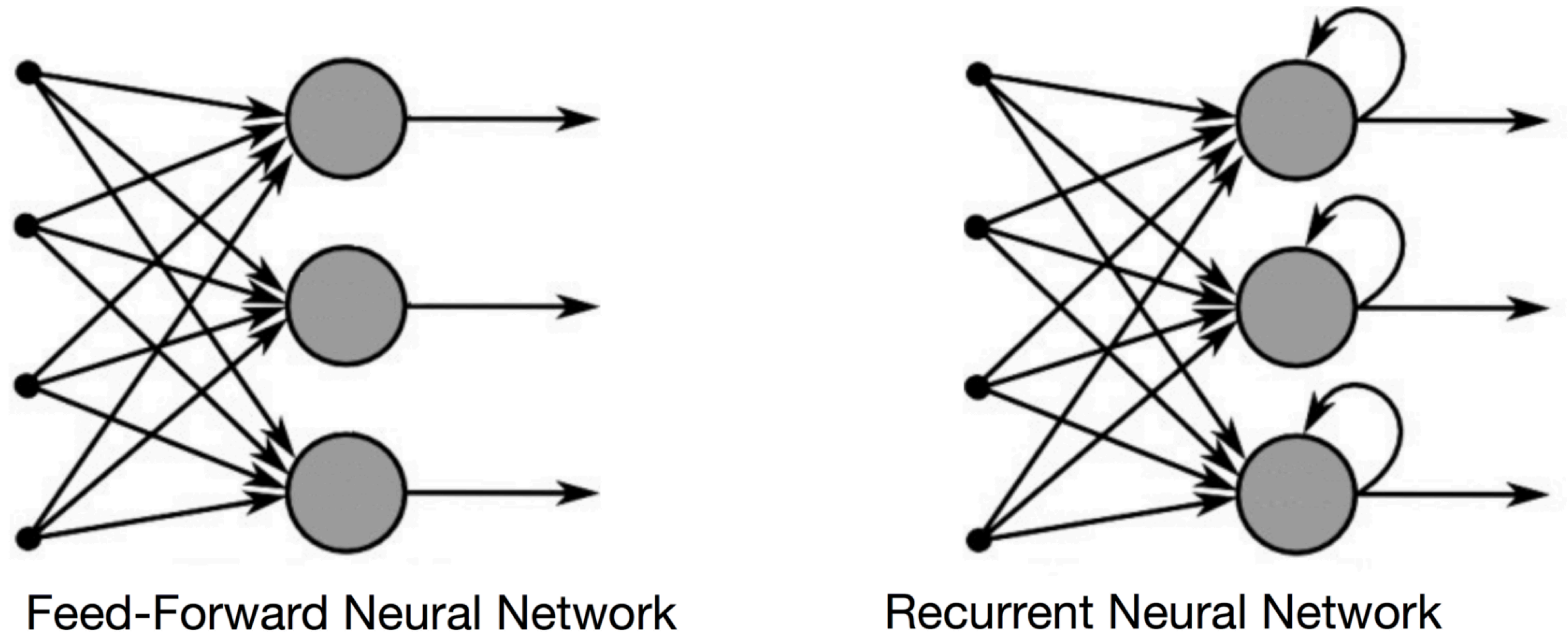
Eunsol Choi

Slides from Greg Durrett, Yoav Artzi, Yejin Choi.

Motivation

- Consider the following examples for sentiment classification:
  - *How can you not see this movie?*
  - *You should not see this movie.*
- Would unigram, bigram models work?
- Would feedforward neural network (deep averaging network) work?
- We need a model to maintain a state to capture influences among words

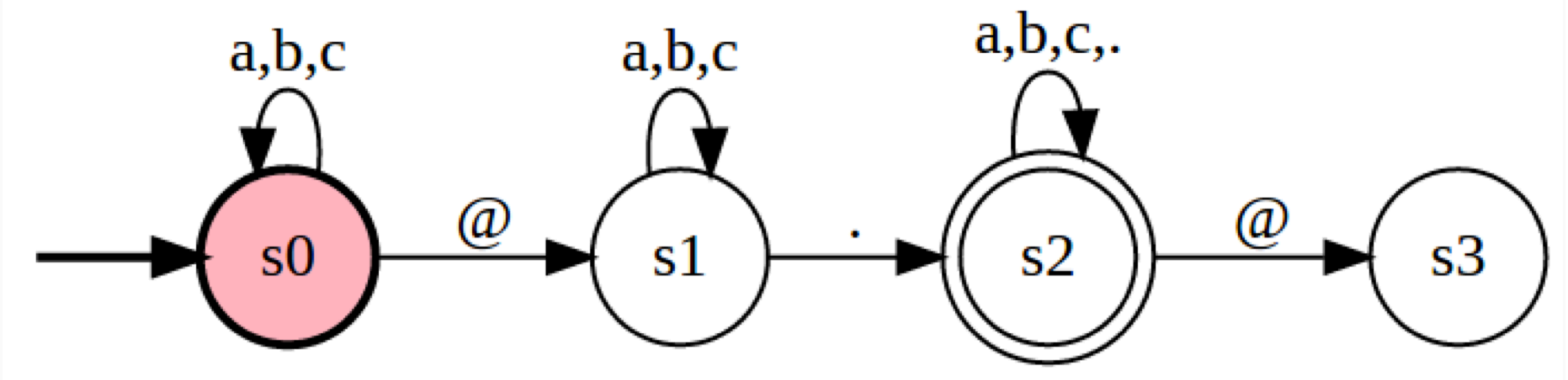
FNN vs RNN



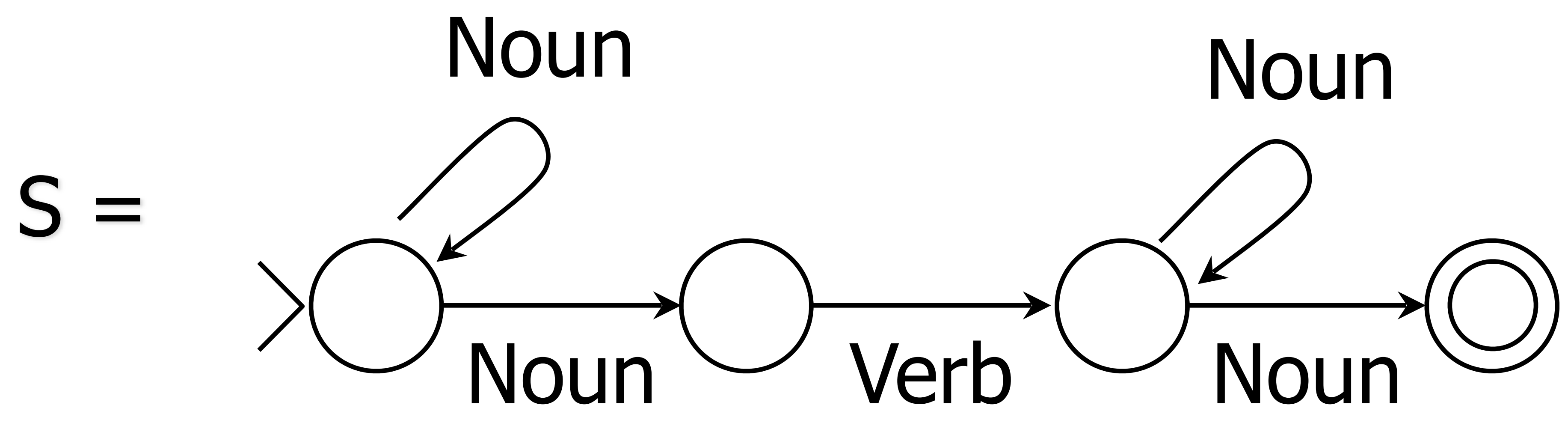
Classic Method: Finite State Machines

- Simple way of representing state in a sequential data
- You can think of it as regular expression
- Current state: saves necessary past information

- S – states
- Σ – vocabulary
- $s_0 \in S$ – start state
- $R: S \times \Sigma \rightarrow S$ – transition function

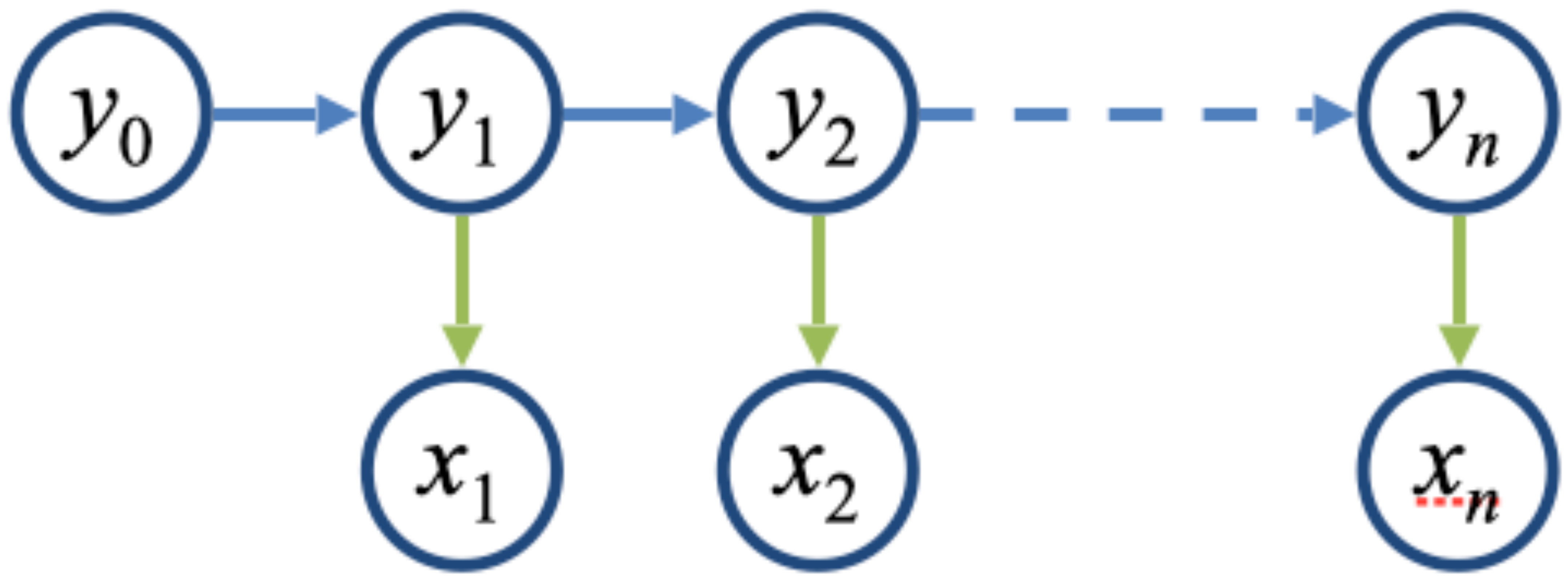


Email address parsing



Language Model

Recall: Hidden Markov Models



$$p(x_1 \dots x_n, y_1 \dots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

- Variants of a finite state machine, where you do not observe states but each states produces an observed output (emission probability)

RNNs

- Maps from dense input sequence to dense hidden state representation sequence

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \rightarrow h_1, \ldots, h_n$$

- $S = \mathbb{R}^{d_{hid}}$ - hidden state space $(h_1, h_2 \ldots)$
- $\Sigma = \mathbb{R}^{d_{in}}$ - input state space $(x_1, x_2 \ldots.)$
- $s_0 \in S$ - initial state vector (h_0)
- $R : \mathbb{R}^{d_{in}} \times \mathbb{R}^{d_{hid}} \rightarrow \mathbb{R}^{d_{hid}}$ - transition function

- For all $i \in \{1, \ldots, n\}$,
  - $h_i = R\left(h_{i-1}, \boldsymbol{x}_i\right)$
  - Simple definition of R: $R(h_{i-1}, x_i) = \tanh(Wx_i + Vh_{i-1} + b)$

- R is parameterized, where the parameters are shared across all steps.

$$h_4 = R\left(h_3, \boldsymbol{x}_4\right) = \ldots = R(R\left(R\left(R\left(h_0, \boldsymbol{x}_1\right), \boldsymbol{x}_2\right), \boldsymbol{x}_3\right), \boldsymbol{x}_4)$$

RNNs

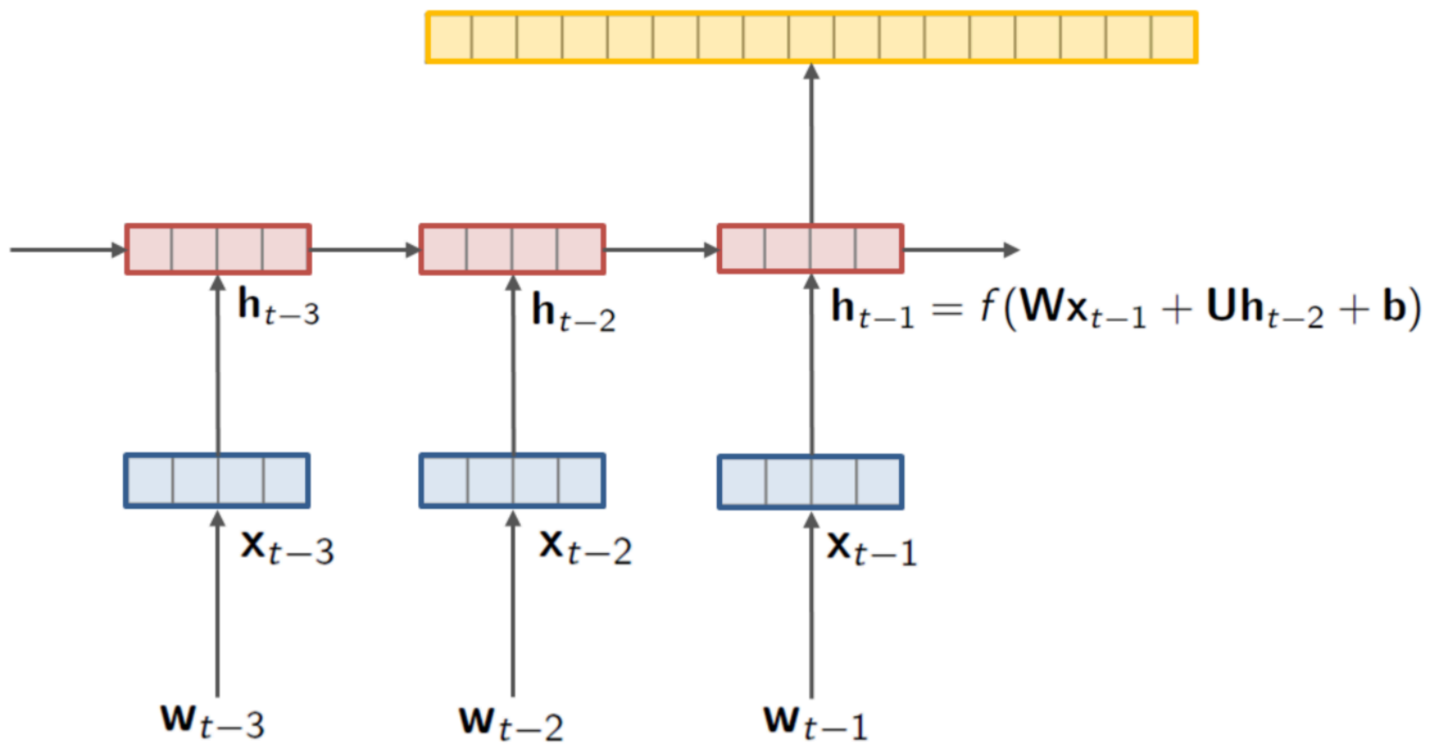
- Hidden states h_i can be used in different ways
- Output function maps hidden state vectors to symbols:

$$O : \mathbb{R}^{d_{hid}} \rightarrow \mathbb{R}^{d_{out}}$$

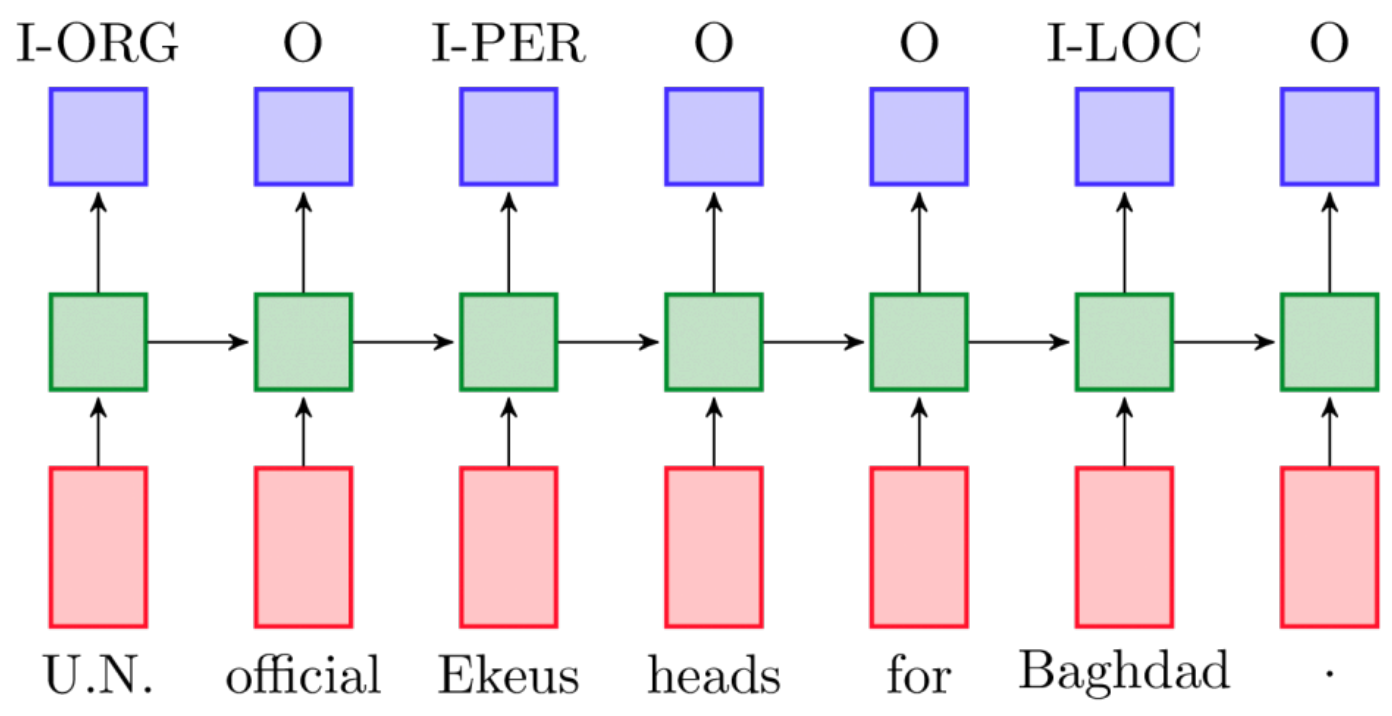
  - For example: single layer + softmax

$$O(h_i) = \text{softmax}(h_i \boldsymbol{W} + \boldsymbol{b})$$

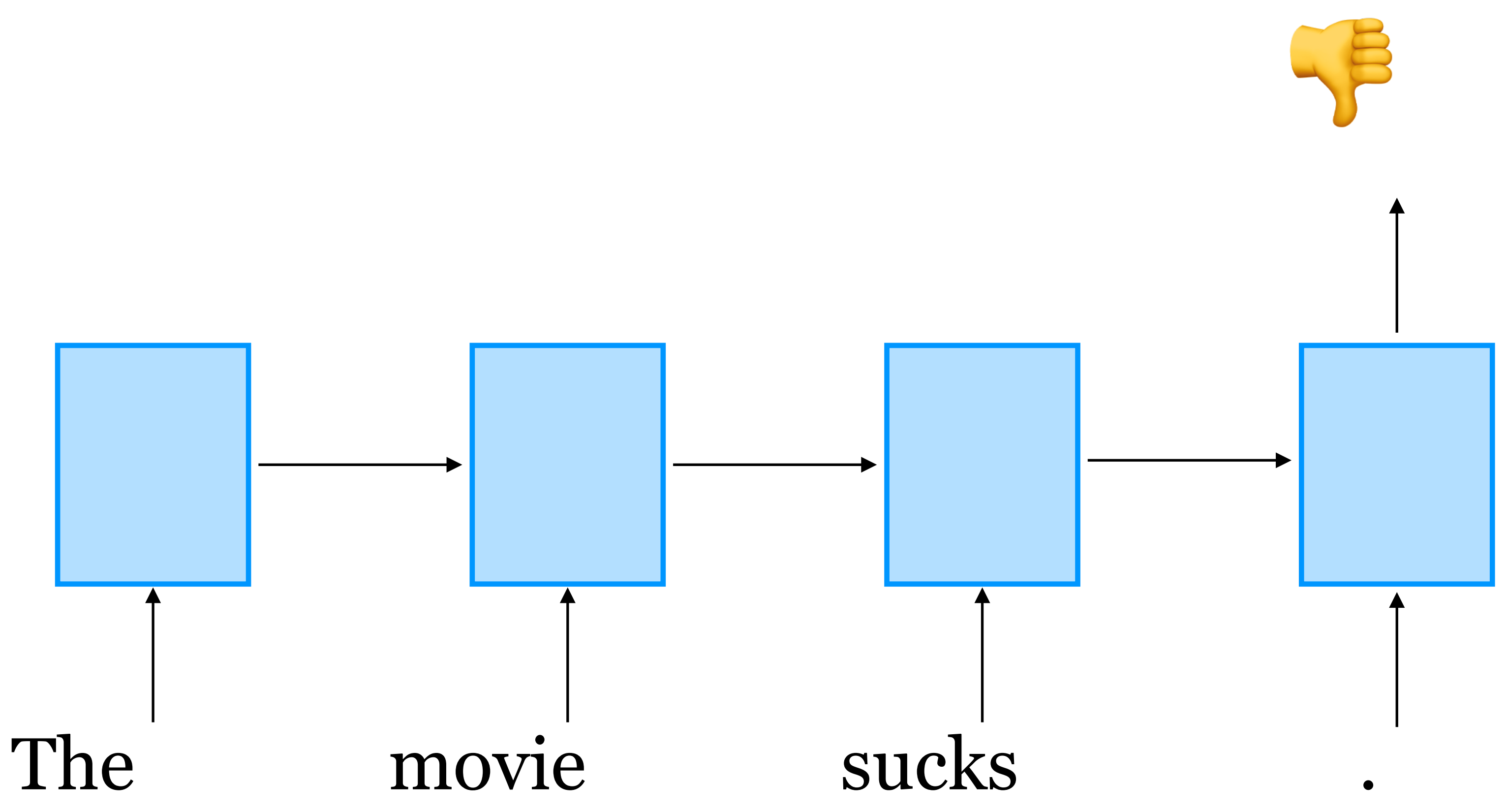
Language modeling



Sequence tagging

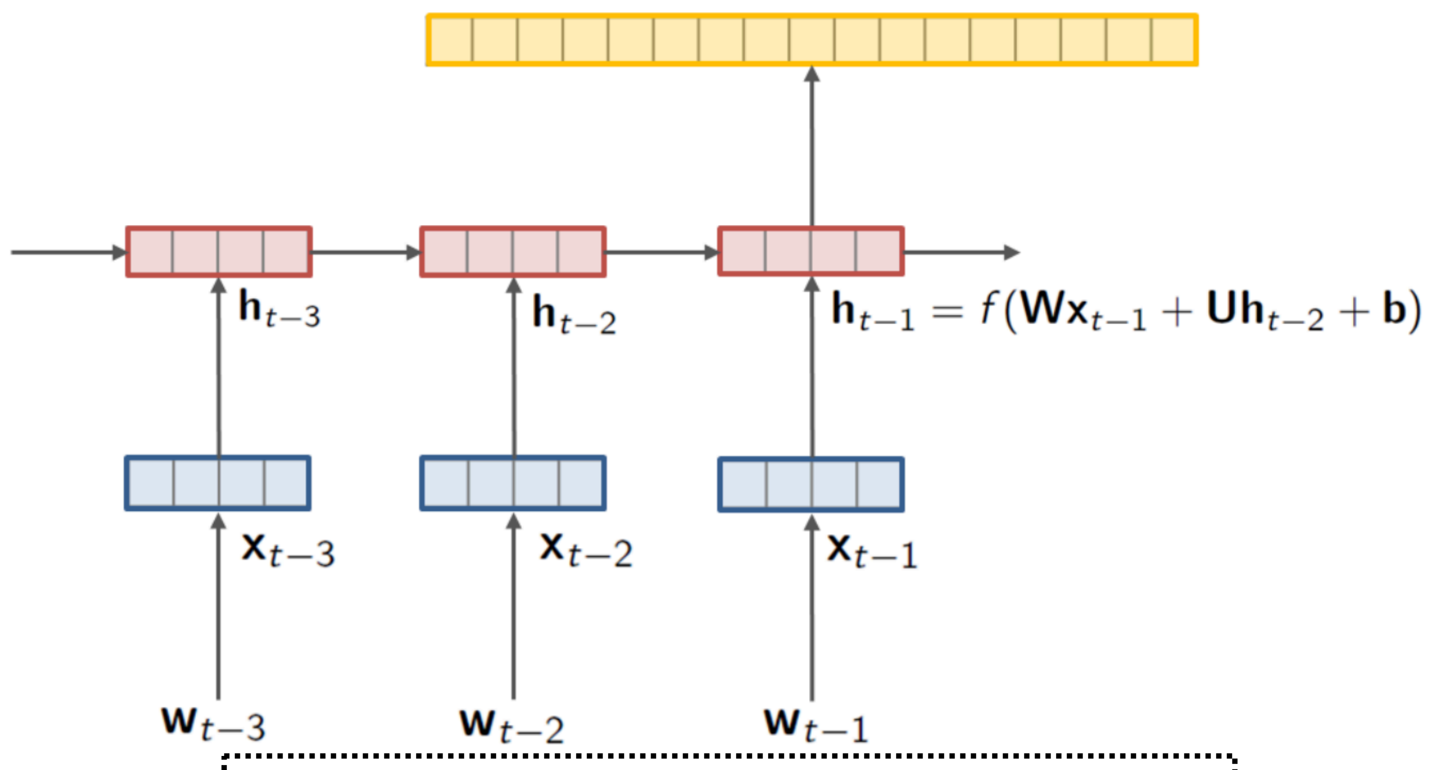


Text classification

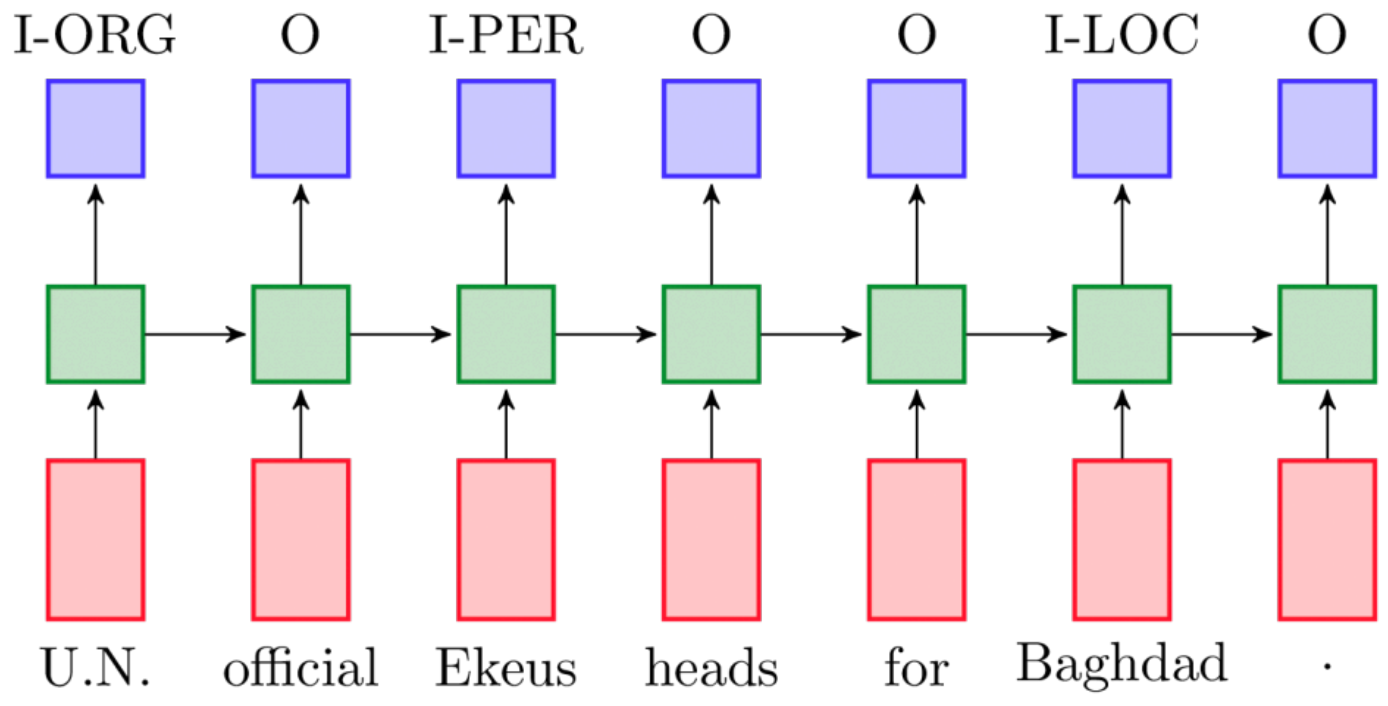


Output functions can be

- Transducer: make some prediction for each element in a sequence



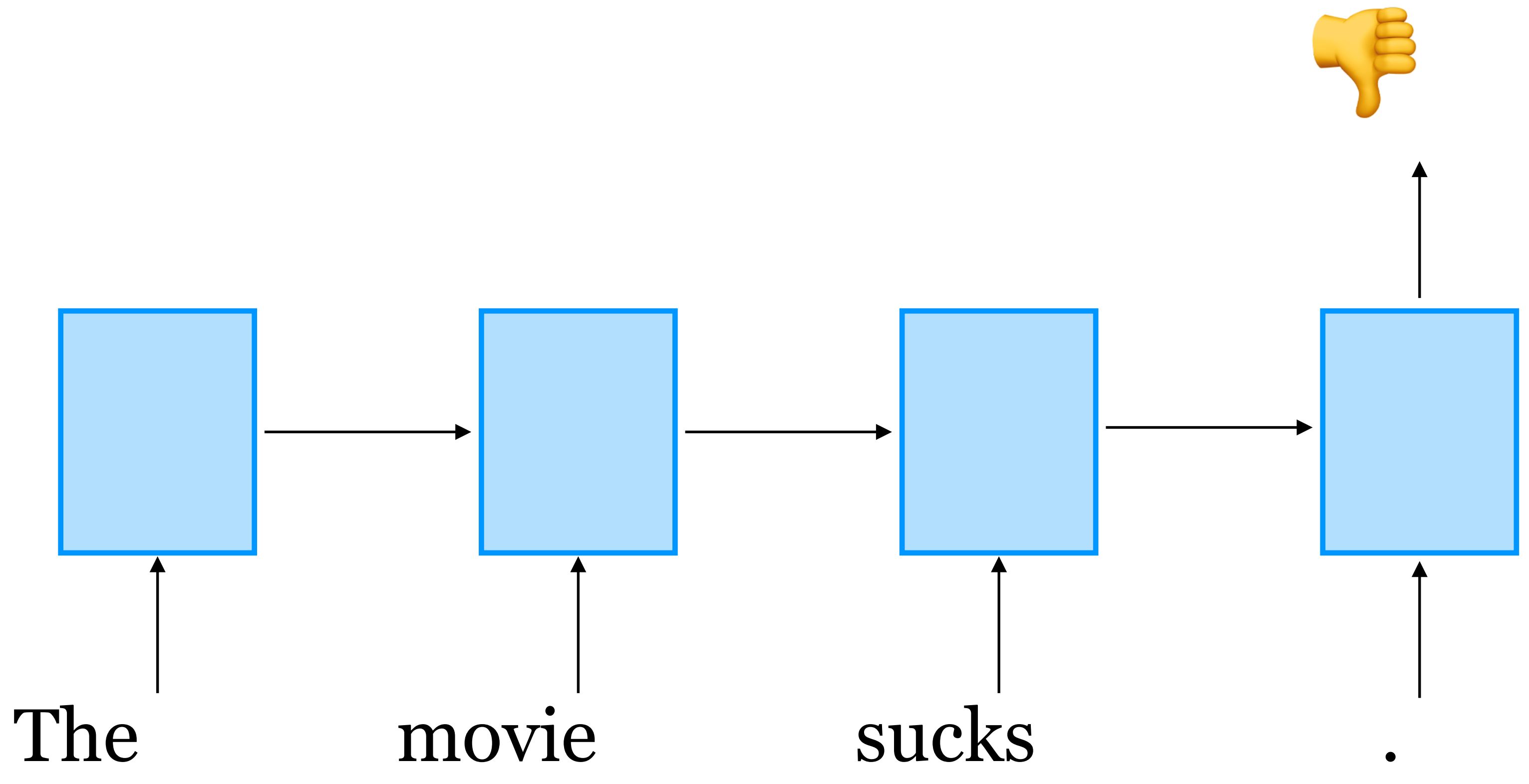
Language modeling



Sequence tagging

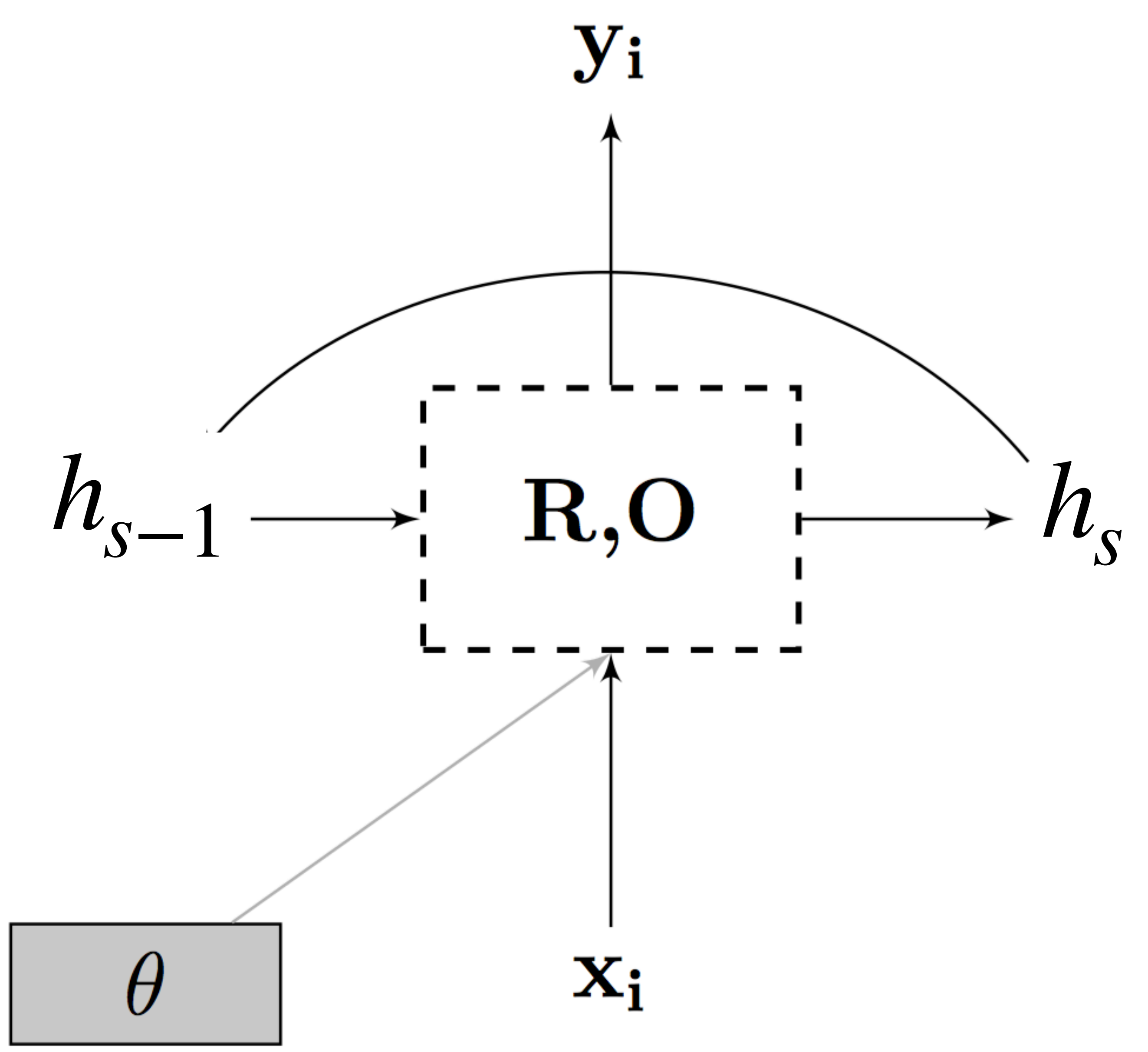
- Acceptor/encoder: encode a sequence into a fixed-sized vector and use that for some purpose

Text classification

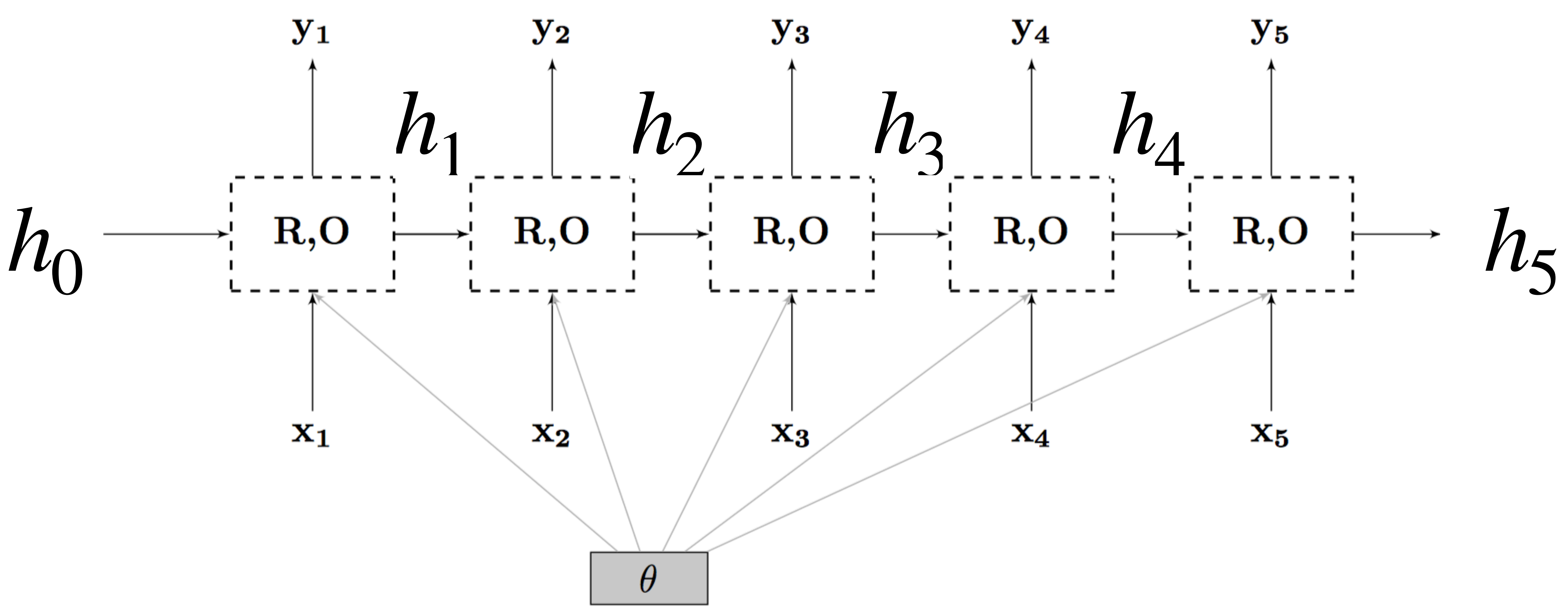


Recurrent Neural Networks (RNN)

- Neural network model, but now with states!
- Handle variable length **inputs**

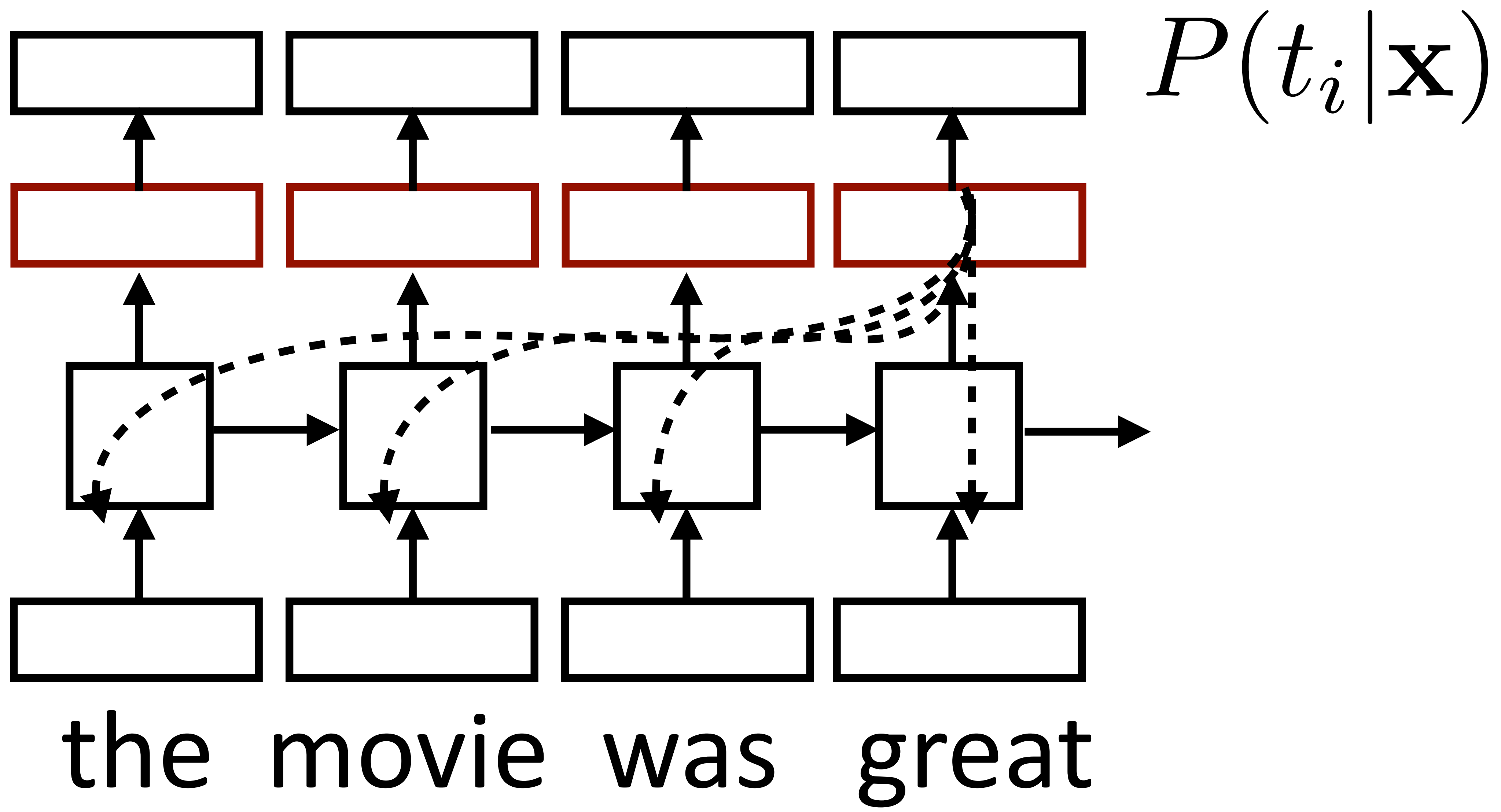


Recursive Representation



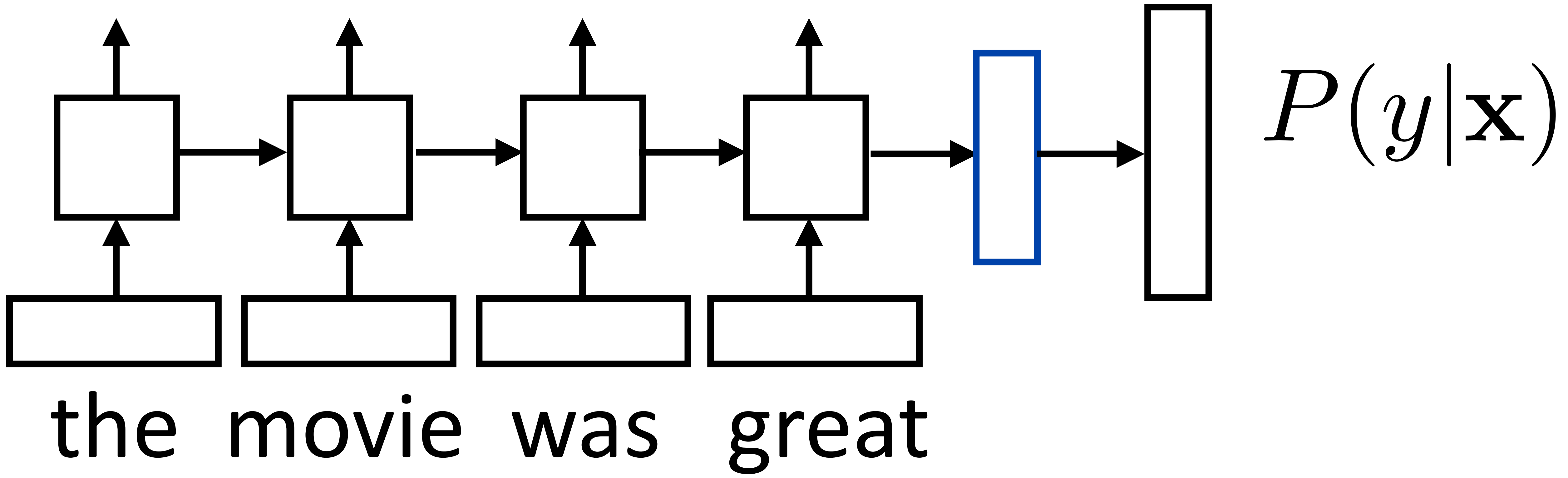
Unrolled Representation

Training RNNs: Transducer



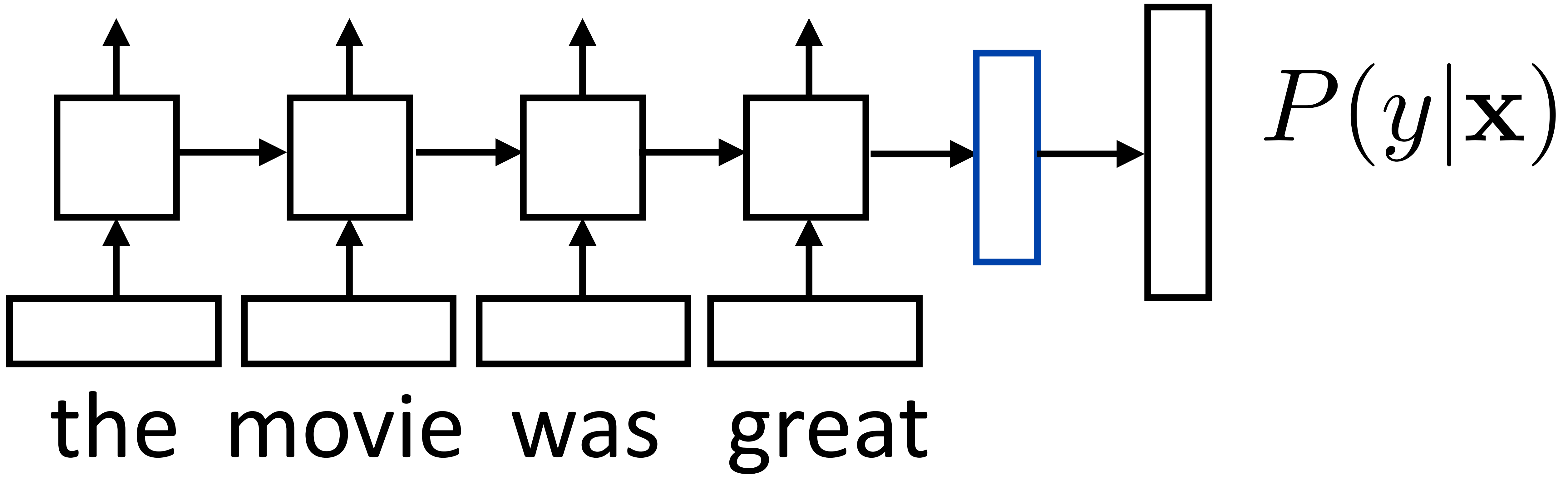
- Loss = negative log likelihood of probability of gold prediction tag
- Loss terms filter back through network

Training RNNs: Acceptor / Encoder



- Loss = negative log likelihood of probability of gold label
- Backpropagate through entire network
- Example: sentiment analysis

Training RNNs



- RNN potentially needs to learn how to "remember" information for a long time!

it was my favorite movie of 2016, though it wasn't without problems -> +