CS 2731
Introduction to Natural Language Processing

Session 11: RNNs part 2, encoder-decoder

Michael Miller Yoder

October 2, 2024

University of Pittsburgh | School of Computing and Information

Course logistics

- Homework 2 is due **tomorrow, Thu Oct 3**
  - Text classification
  - Written and programming components
  - Optional Kaggle competition for best LR and NN deception classifiers

Course logistics

- Project groups have been formed
  - If you want to change groups, let me know
  - Please schedule a group meeting with me this week through my Bookings link in person in IS 604B or on Zoom
    - What questions do you have about completing your project?
- Next project deliverable is the proposal and literature on Oct 17
  - Instructions are on the project webpage
  - It's good to start the literature review early
  - Look for NLP papers in ACL Anthology, Semantic Scholar, and Google Scholar

How to do a literature review

- Look for NLP papers related to your topic in ACL Anthology, Semantic Scholar, and Google Scholar
- For each paper, note:
  - What they cite in their related work sections (find those papers, iterate)
  - Data
  - Methods
  - Findings
- For at least 3 papers, organize them into themes of approaches, datasets, findings
- Ok: X paper did this, Y paper did this, Z paper did that
- Good: X and Y papers did this, while Z improved with that
- Best: X and Y papers did this, Z improved, nobody has yet to do…

Midterm course evaluation (OMETs)

- https://go.blueja.io/NsyfmLUs5Ee4GQiFKcqQ3g
- All types of feedback are welcome (critical and positive)
- Completely anonymous, will not affect grades
- Let me know what's working and what to improve on while the course is still running!
- Please be as specific as possible
- Available until **Mon Oct 7 at 11:59pm**

Lecture overview: RNNs part 2, encoder-decoder

- RNN language modeling
- LSTMs
- RNNs for other NLP tasks
- Encoder-decoder model
- Attention

RNNs for language modeling

RNN refresher

With a neighbor, talk about the following questions:

1. What are the 2 inputs to the hidden states in a simple 2-layer RNN?
2. What do RNNs allow us to do in NLP that we can't do with feedforward neural networks?

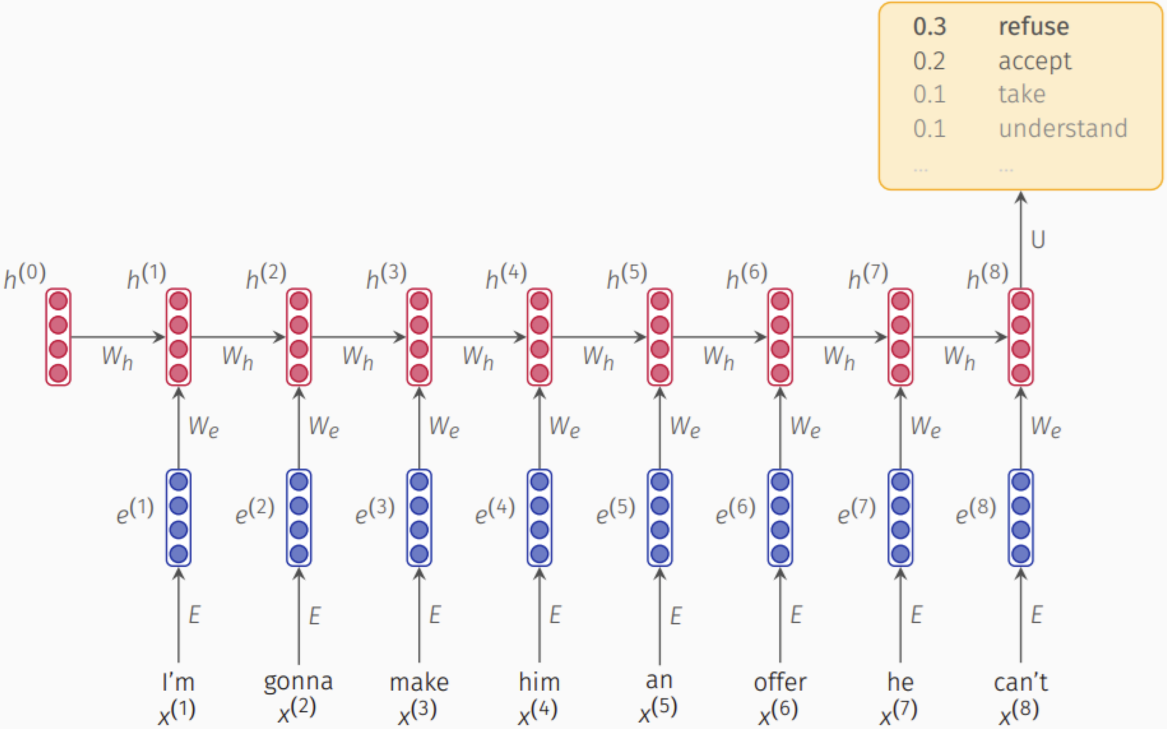
An RNN Language Model

output distribution
$\hat{y}^{(t)} = \mathrm{softmax}(Uh^{(t)} + b_2) \in \mathbb{R}^{|V|}$

hidden states
$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$
$h^{(0)}$ is the initial hidden state

word embeddings
$e^{(t)} = Ex^{(t)}$

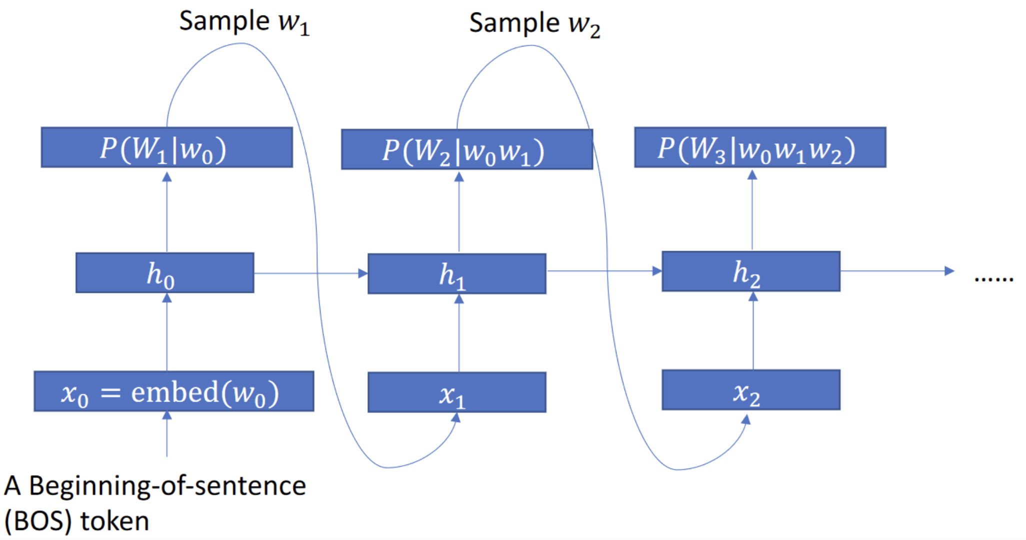
one-hot vectors
$x^{(t)} \in \mathbb{R}^{|V|}$



Slide credit: David Mortensen

Generation with RNN LMs

- At each time step t, we sample w_t from $P(W_t| \ldots)$, and feed it to the next timestep!
- LM with this kind of generation process is called autoregressive LM



Slide adapted from Tianxing He

Advantages of RNN Language Models

- Input can be of an arbitrary length
- Computation can use information from many steps back (in principle)
- Longer inputs do not mean larger model sizes
- Same weights applied at every time step—**symmetry**

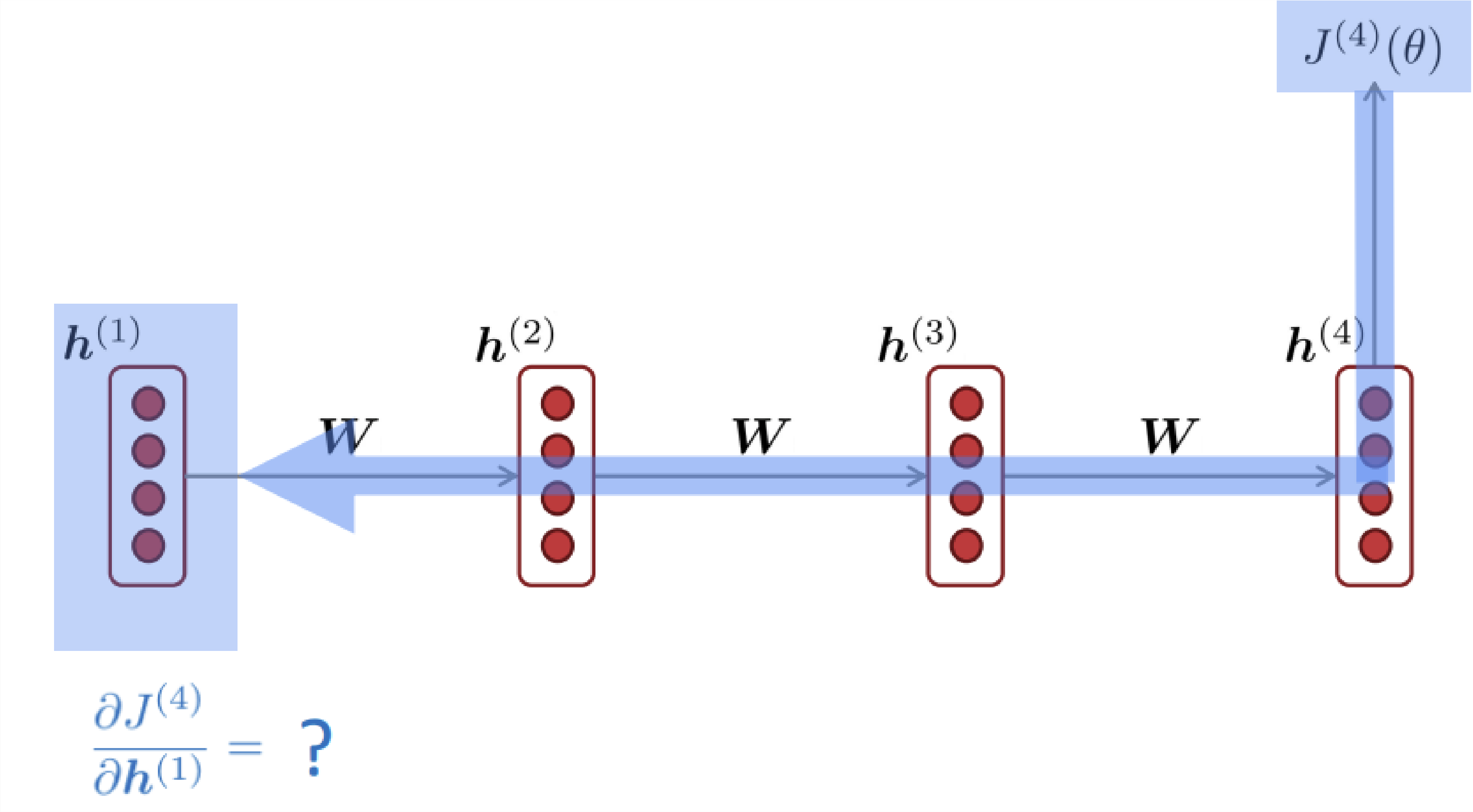
Slide credit: David Mortensen

Disadvantages of RNN LMs

- Recurrent computation is slow
  - Computing $h^{(t)}$ requires computing $h^{(t-1)}$ which requires computing $h^{(t-2)}$
  - Cannot be parallelized
- In practice, it is difficult to access information from many steps back (cf. the VANISHING GRADIENT PROBLEM)

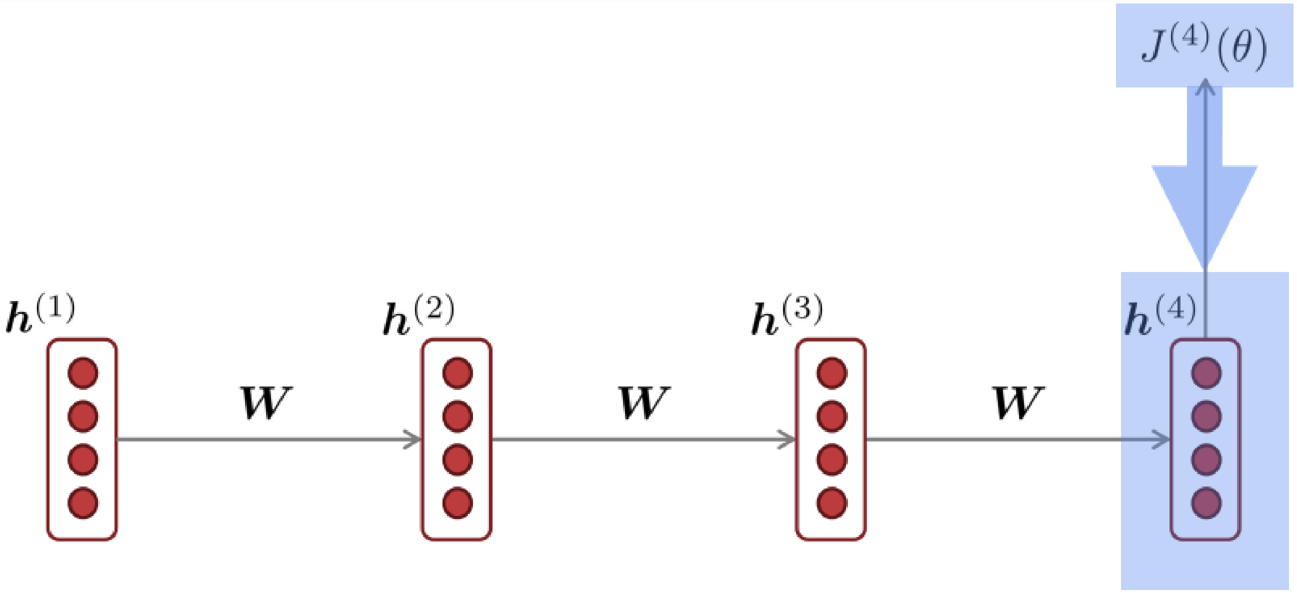
Slide credit: David Mortensen

Vanishing gradient problem



Slide adapted from Chris Manning

Vanishing gradient problem

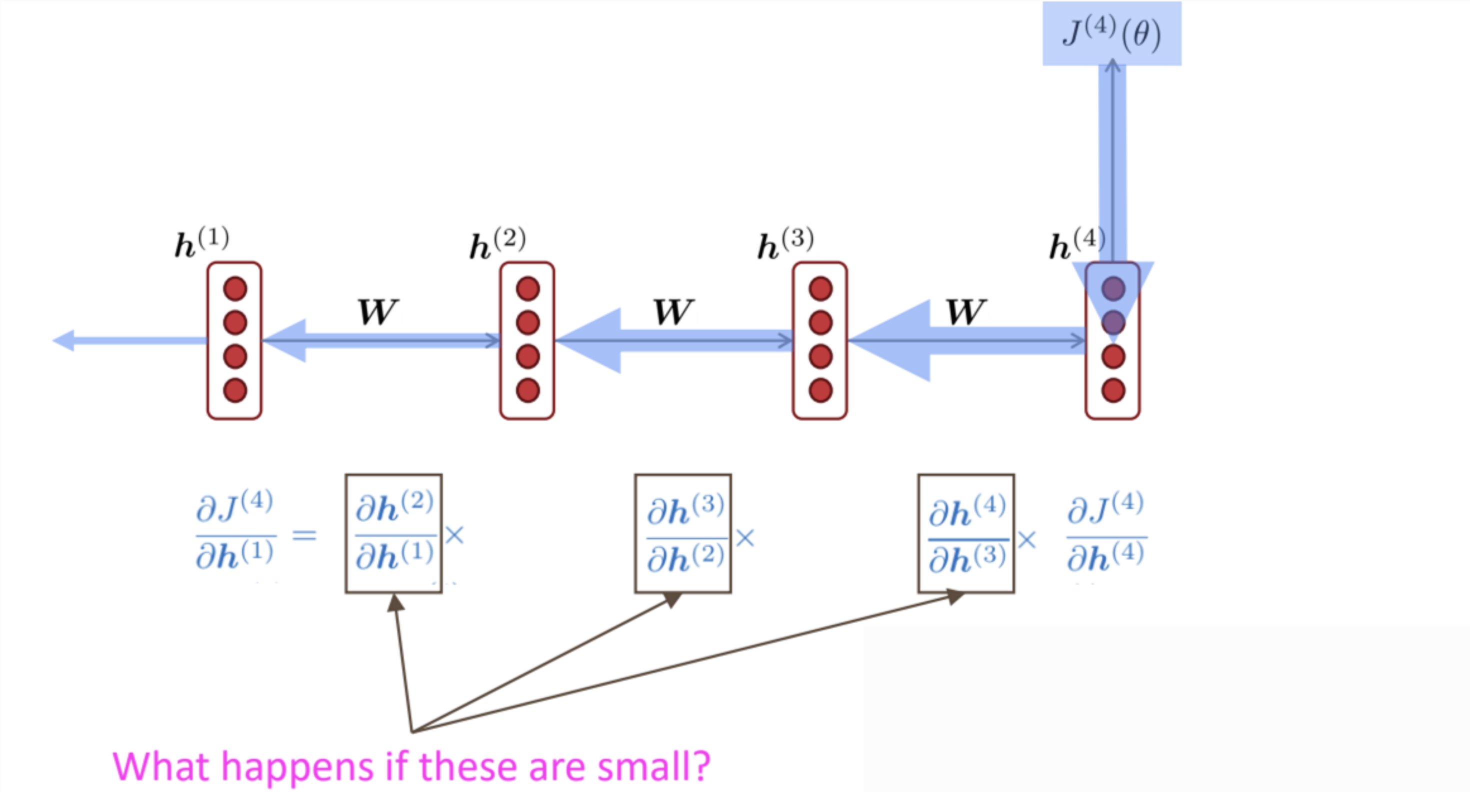


$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

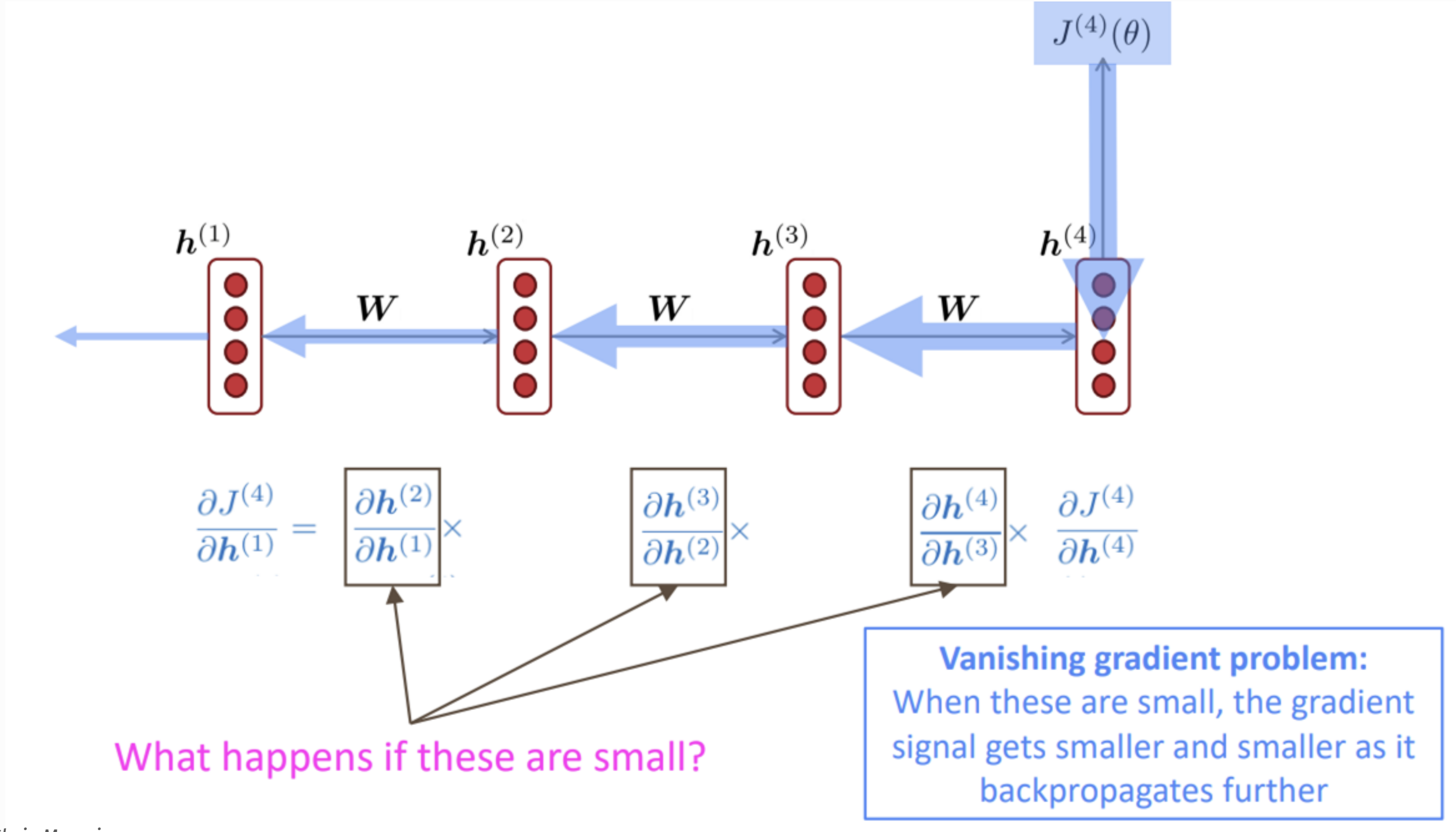
Slide adapted from Chris Manning

Vanishing gradient problem



Slide adapted from Chris Manning

Vanishing gradient problem



Slide adapted from Chris Manning

Vanishing gradient problem

- Gradient signal from far away is lost because it’s much smaller than gradient signal from close-by.
- So model weights are basically updated only with respect to near effects, not long-term effects
- **LM task:** When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _______
    - To be successful, need to model the dependency between “tickets” in the beginning and very end of the paragraph
    - If the gradient is small, can’t learn this long-range dependency

Slide adapted from Chris Manning

LSTMs

LSTMs Address (but Do not Solve) the Vanishing and Exploding Gradient Problems

- LSTMs: Long Short Term Memory
- Process data sequentially, but keep hidden state through time
- Still subject, at some level, to vanishing gradients, but to a lesser degree that traditional RNNs
- Widely used in language modeling

Slide credit: David Mortensen

LSTMs: real-world success

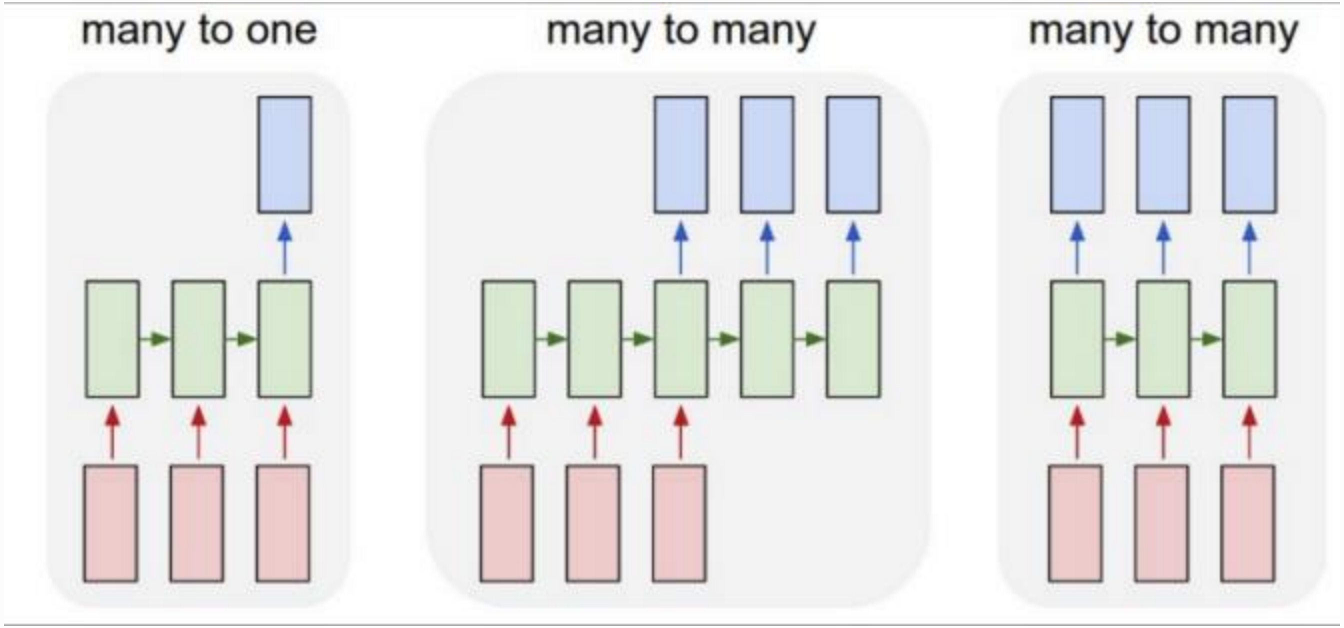
- In 2013–2015, LSTMs started achieving state-of-the-art results
    - Successful tasks include handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language models
    - LSTMs became the dominant approach for most NLP tasks
- Now (2019–2024), Transformers have become dominant for all tasks
- For example, in WMT (a Machine Translation conference + competition):
    - In WMT 2014, there were 0 neural machine translation systems (!)
    - In WMT 2016, the summary report contains “RNN” 44 times (and these systems won)
    - In WMT 2019: “RNN” 7 times, ”Transformer” 105 times

Source: "Findings of the 2016 Conference on Machine Translation (WMT16)", Bojar et al. 2016, http://www.statmt.org/wmt16/pdf/W16-2301.pdf
Source: "Findings of the 2018 Conference on Machine Translation (WMT18)", Bojar et al. 2018, http://www.statmt.org/wmt18/pdf/WMT028.pdf
Source: "Findings of the 2019 Conference on Machine Translation (WMT19)", Barrault et al. 2019, http://www.statmt.org/wmt18/pdf/WMT028.pdf

Slide adapted from Chris Manning

RNNs for other NLP tasks

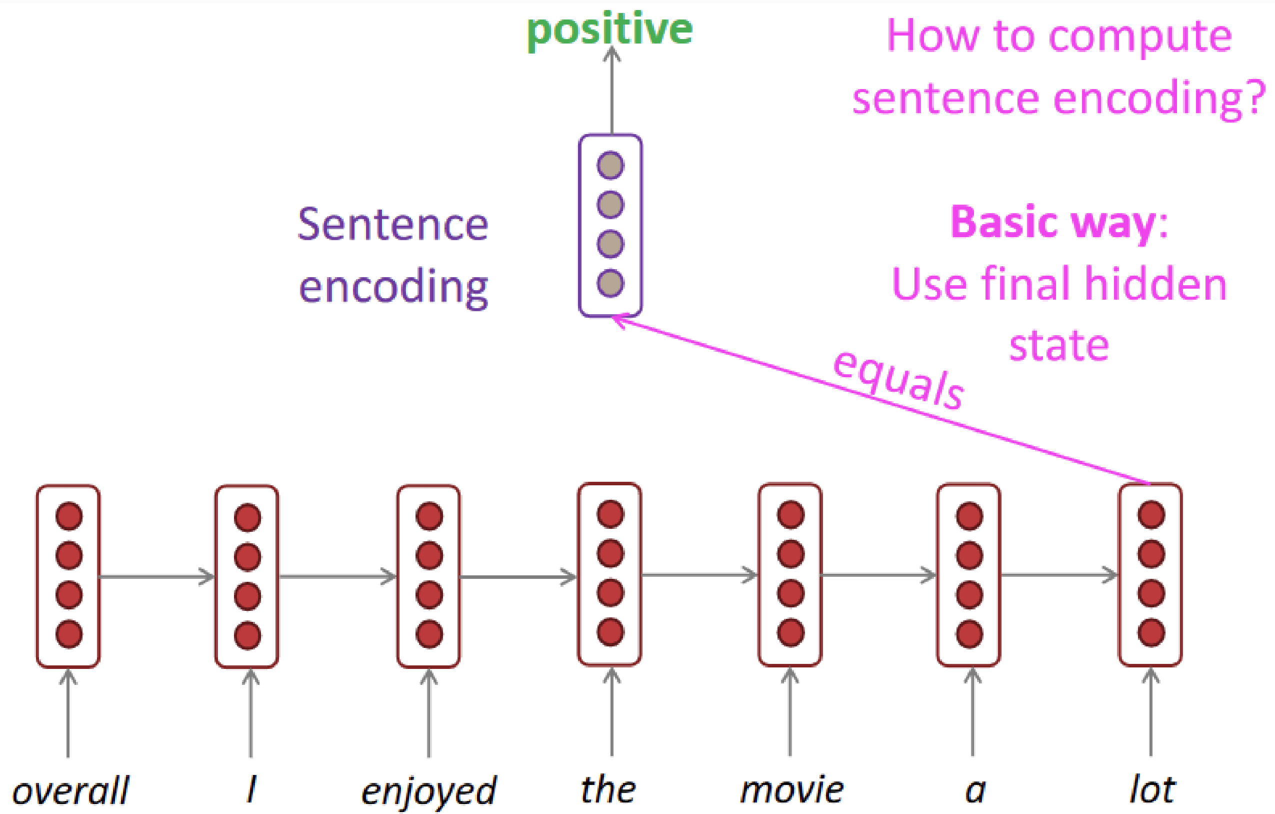
RNNs for tasks other than language modeling



- Text classification (many to one)
- Encoder-decoder, machine translation (many to many)
- Language modeling, sequence labeling (many to many)

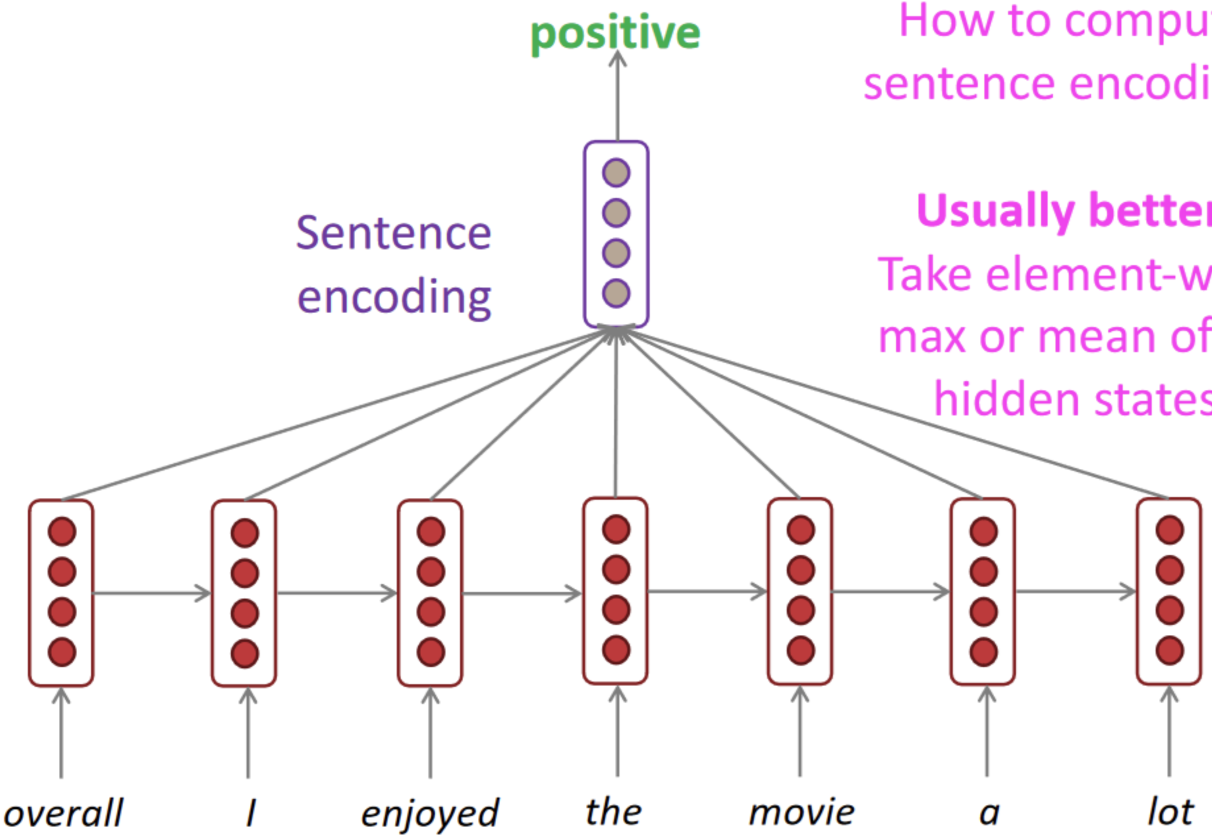
Slide adapted from David Mortensen

RNNs to encode sentences for text classification



Slide adapted from Chris Manning

RNNs to encode sentences for text classification



How to compute sentence encoding?

Usually better: Take element-wise max or mean of all hidden states

Slide adapted from Chris Manning

Encoder-decoder architecture

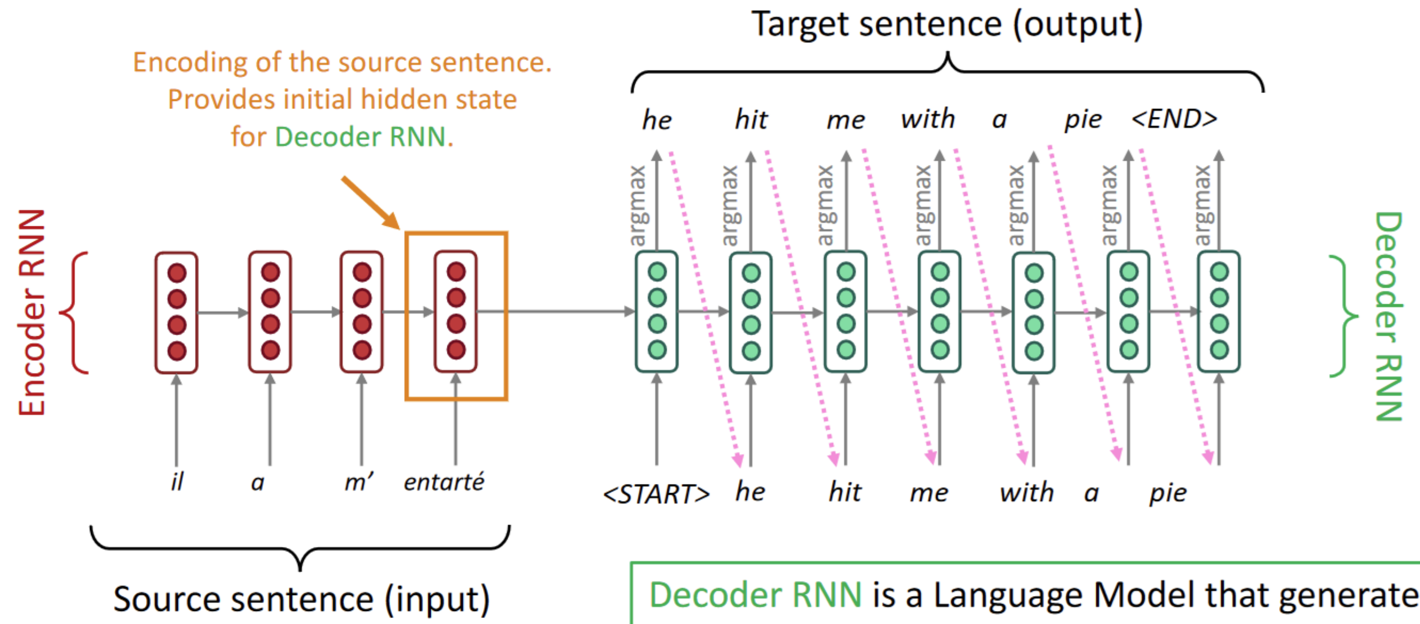
The many names of encoder-decoder

- Encoder-decoder
- Sequence-to-sequence models
  - seq2seq
- Conditional language modeling

Parts of an encoder-decoder

- **Encoder**: that accepts an input sequence, x^n, and generates a corresponding sequence of contextualized representations, h^n. LSTMs, convolutional neural networks, Transformers can all be encoders
- **Context**: c, which is a function of h^n, and conveys the essence of the input to the decoder.
- **Decoder**: which accepts c as input and generates an arbitrary length sequence of hidden states h^m , from which a corresponding sequence of output states y^m, can be obtained

Encoder-decoder (seq2seq) architecture with RNNs



Encoder RNN produces an *encoding* of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

Note: This diagram shows **test time** behavior: decoder output is fed in ·······▸ as next step's input

Slide adapted from Chris Manning

Encoder-decoder (seq2seq) is versatile!

Many NLP tasks can be phrased as sequence-to-sequence:

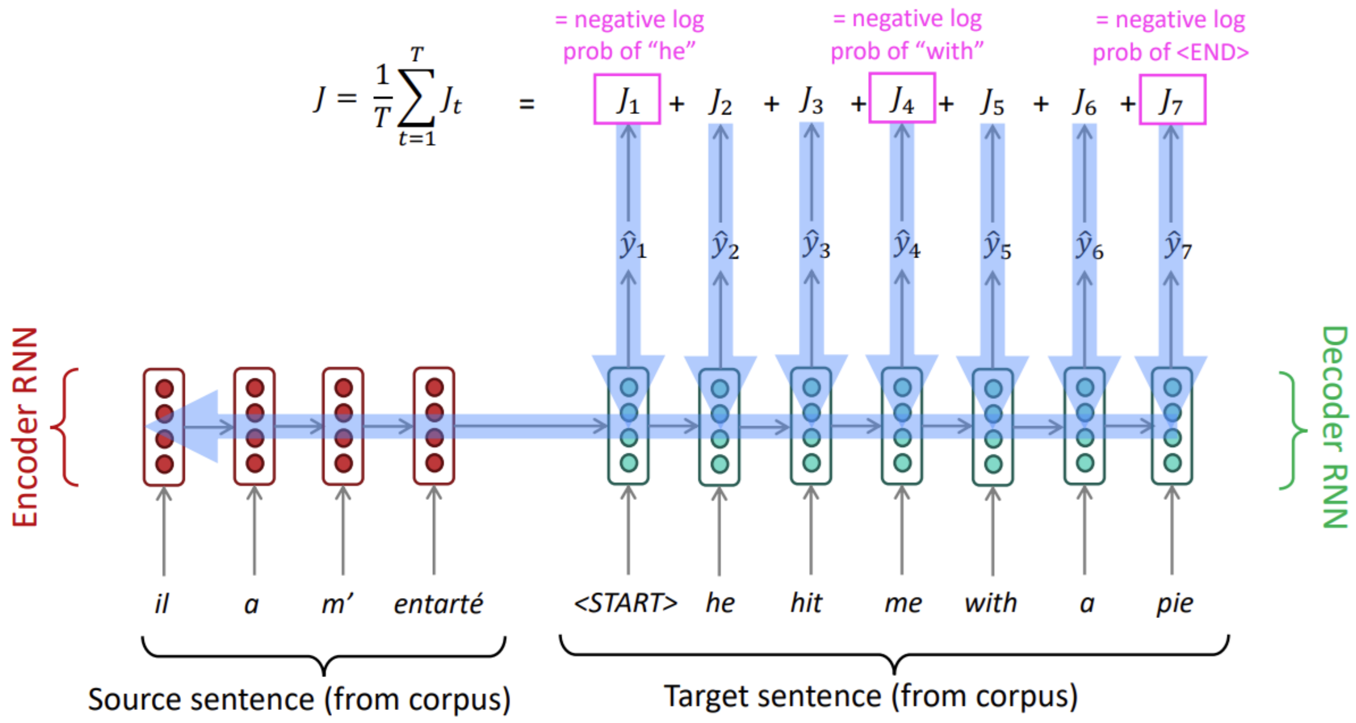
- Summarization (long text → short text)
- Dialogue (previous utterances → next utterance)
- Parsing (input text → output parse as sequence)
- Code generation (natural language → Python code)

Training corpora needed:

- input <SEPARATOR> output

Slide adapted from Chris Manning

Training an encoder-decoder RNN



Seq2seq is optimized as a **single system.** Backpropagation operates *"end-to-end"*.

Slide adapted from Chris Manning

The attention mechanism

Attention makes context available beyond the bottleneck

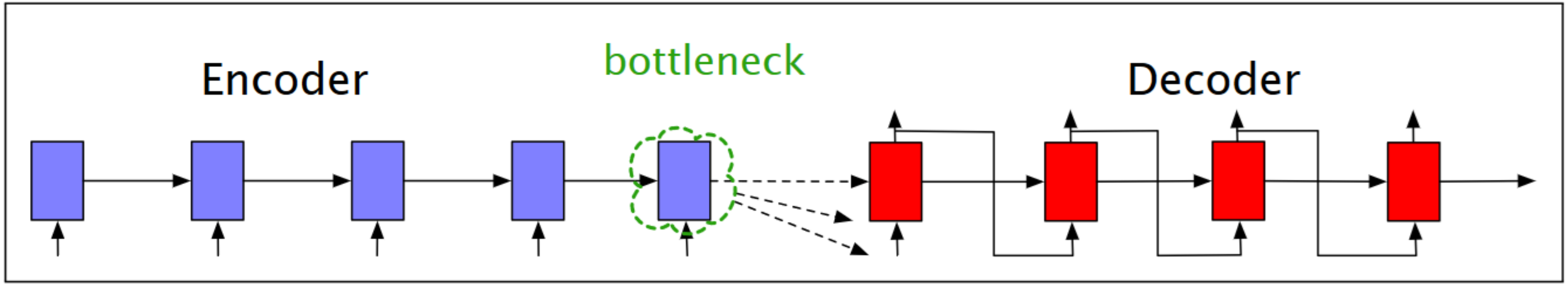
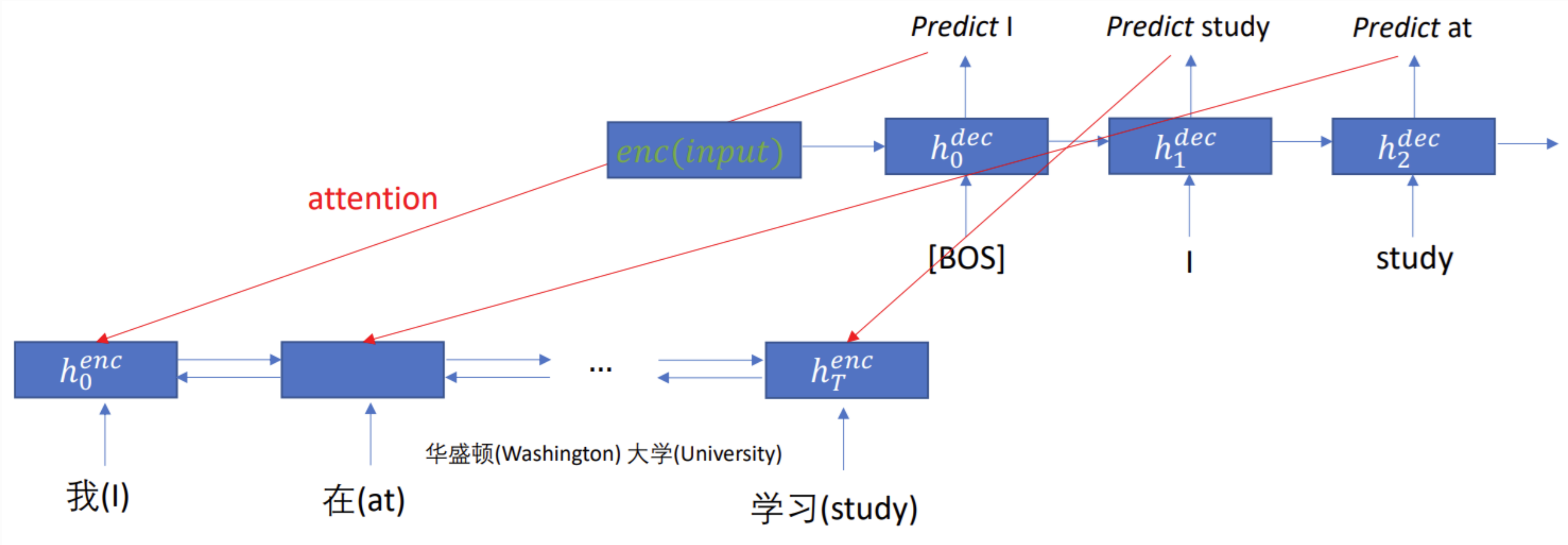


Figure 9.21 Requiring the context c to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

- Bottleneck means that early timesteps in the encoder aren't as accessible
- However, in tasks like MT, we may want to pay attention to different parts of the input in different timesteps.

Slide adapted from David Mortensen, Jurafsy & Martin, Tianxing He

Attention visualized

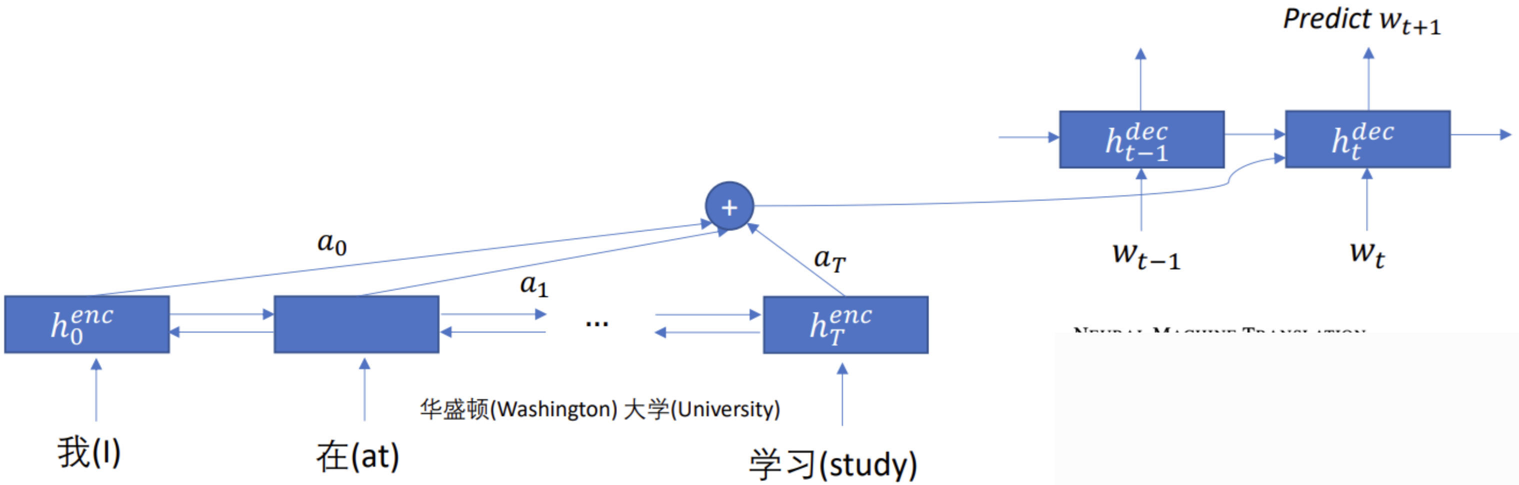


- “I study at University of Washington”
- This alignment is not trivial!
- The attention module is proposed to learn this alignment in an end-to-end fashion.

Slide adapted from Tianxing He

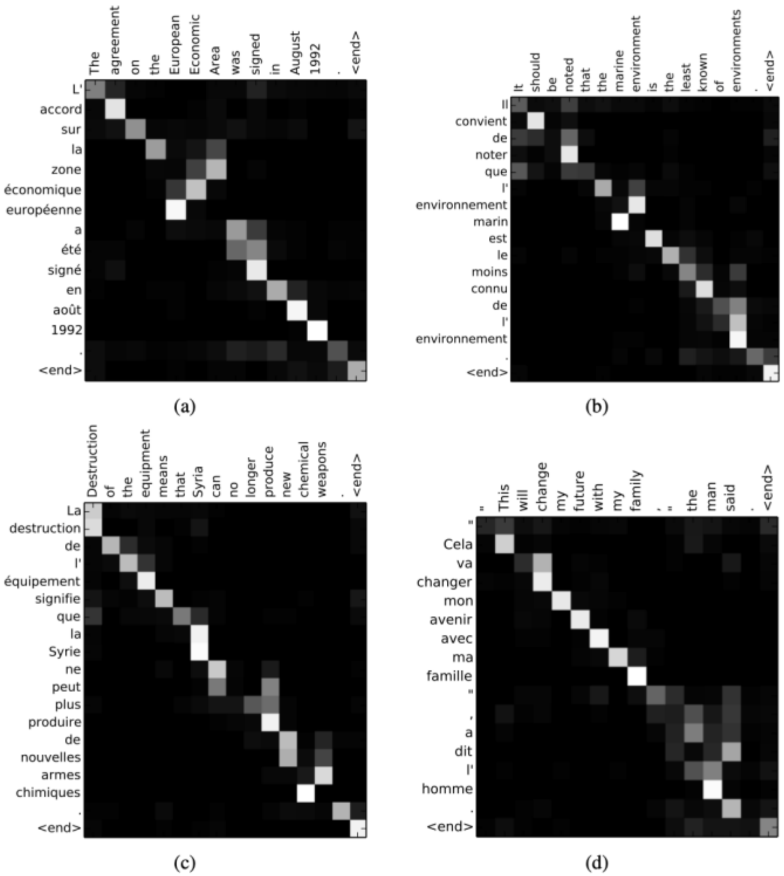
The attention mechanism [Bahdanau et al. 2014]

- We now focus on timestep t.
- For each encoder state h_i^{enc} , we compute an alignment score $\hat{a}_i = (h_i^{enc})^T W_a h_{t-1}^{dec}$.
- Then we get an attention distribution $a = softmax(\hat{a})$.
- We can then reweight the encoder states by a and pass $\sum_i a_i h_i^{enc}$ to the decoder.
- The parameter W_a is shared across time steps.



Slide adapted from Tianxing He

Attention: learned alignment example



Slide adapted from Tianxing He

Conclusion: RNNs part 2, encoder-decoder

- RNNs are effective neural networks for sequences
  - Can handle sequences of varying length
  - Can “remember” information from earlier timesteps
- RNNs can be used for language modeling and other NLP tasks
- LSTMs are a type of RNNs that handles vanishing gradient problem
- The encoder-decoder framework “encodes” sequential input and then “decodes” sequential output
- The attention mechanism weights encodings of relevant pieces of the input when producing output

Questions?