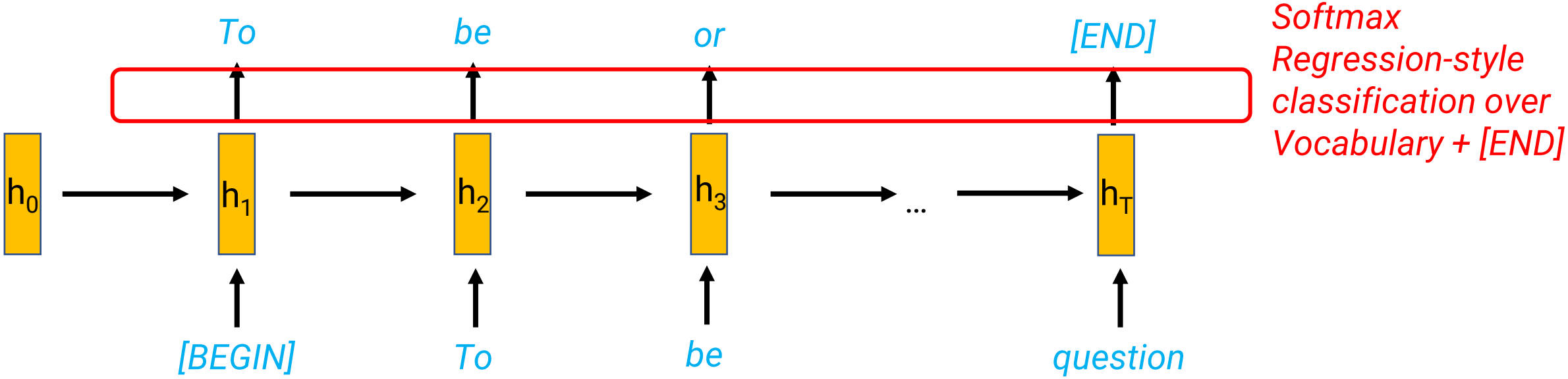
Deep Learning for Language, Part 2: Sequence-to-sequence, Attention

Robin Jia

USC CSCI 467, Fall 2023

October 5, 2023

Language Modeling ("Decoder only")



- At each step, predict the next word given current hidden state
  - Essentially a softmax regression "head"—takes in hidden state, outputs distribution over Vocabulary + [END]
- Start with special *[BEGIN]* token (so the first word model generates is first real word)
- One step's output becomes next step's input ("autoregressive")
- To mark end of sequence, model should predict the *[END]* token
- Called a "Decoder" because it looks at the hidden state and "decodes" the next word

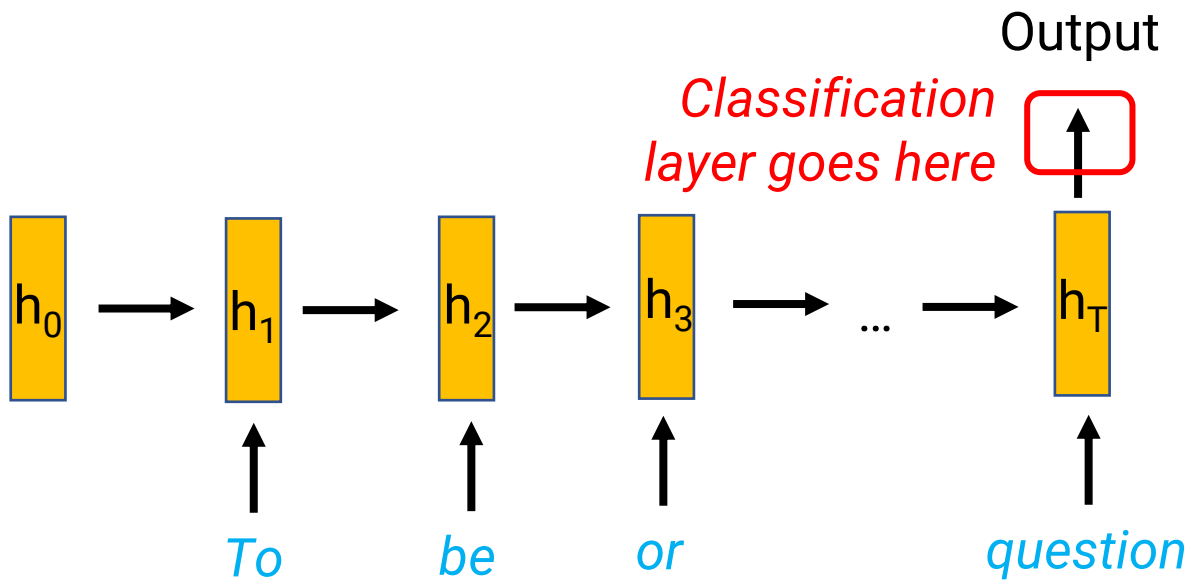
Autoregressive Language Model Training

- Training example: “Convolutional neural networks are good for image classification”
- Want to maximize P(“Convolutional neural networks are good for image classification”)
- Take log and decompose by chain rule:

  log P(“Convolutional”)
  + log P(“neural” | “Convolutional”)
  + log P(“networks” | “Convolutional neural”)
  + log P(“are” | “Convolutional neural networks”) + …

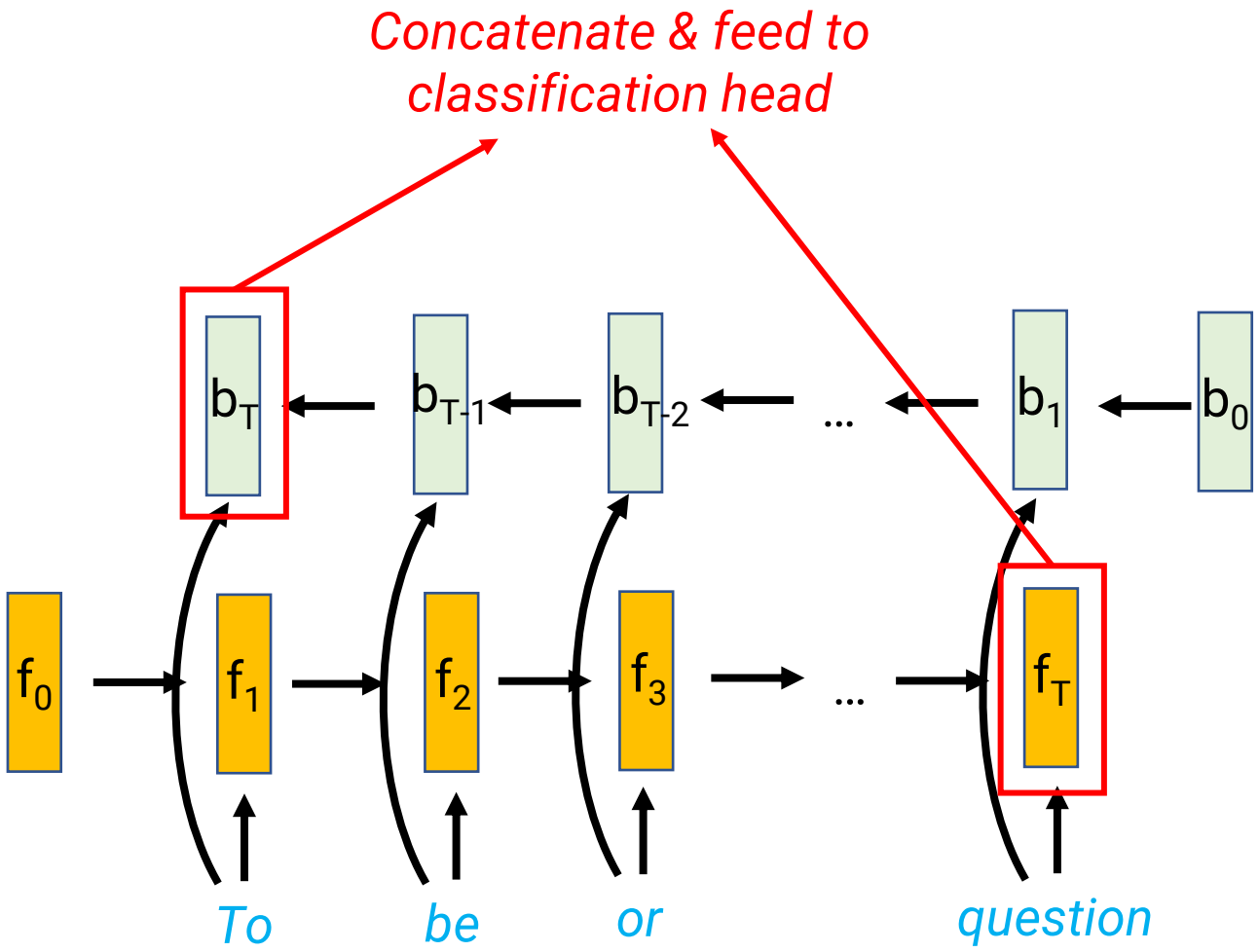
- Decomposes into a bunch of **next-word-classification** problems
  - I will also write this as P(word | prefix)

Text classification ("Encoder only")



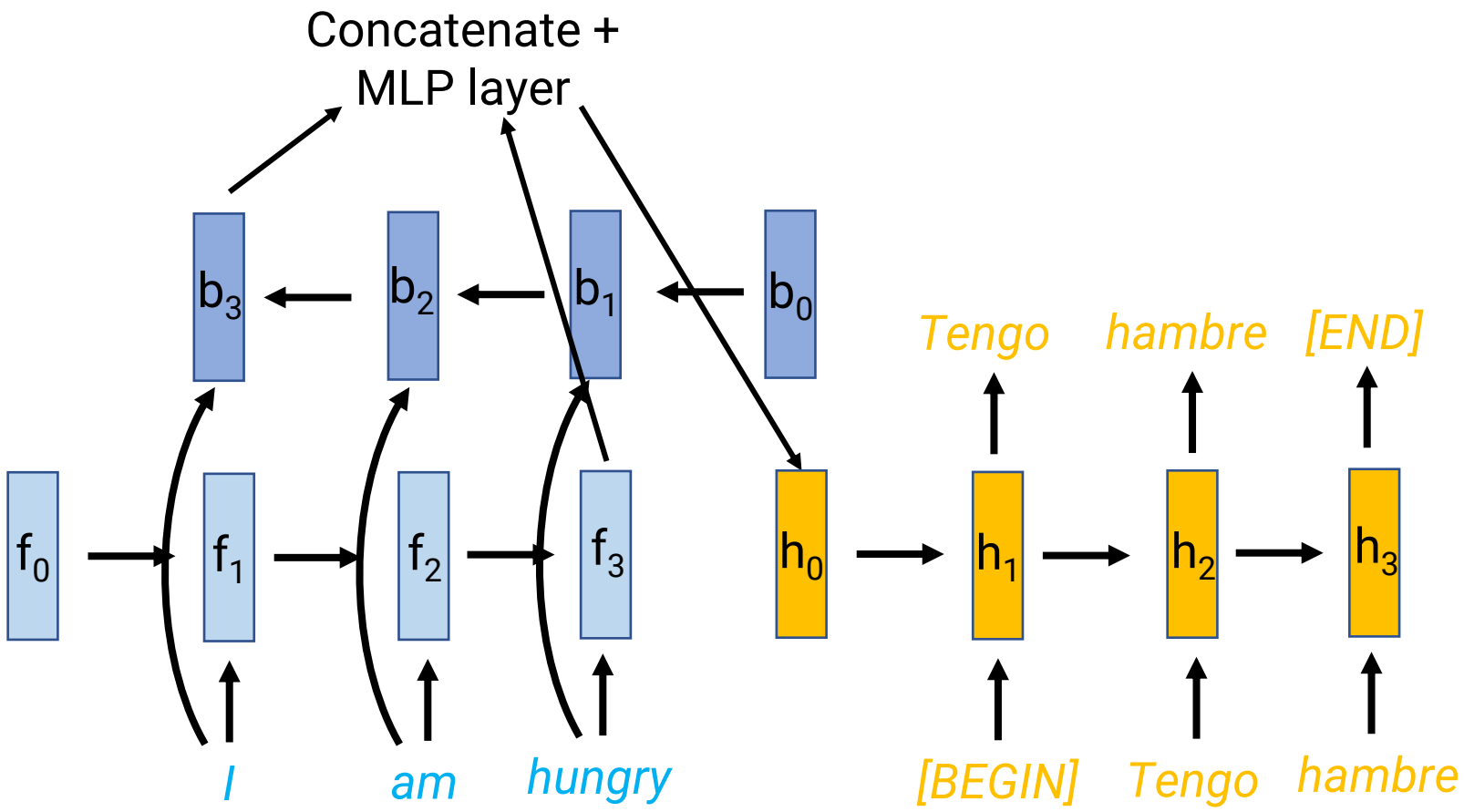
- First run an RNN over text
- Use the final hidden state as an "encoding" of the entire sequence
- Use this as features, train a classifier on top
- Downside: Later words processed better than early words (long range dependency issues)

Bi-directional encoders



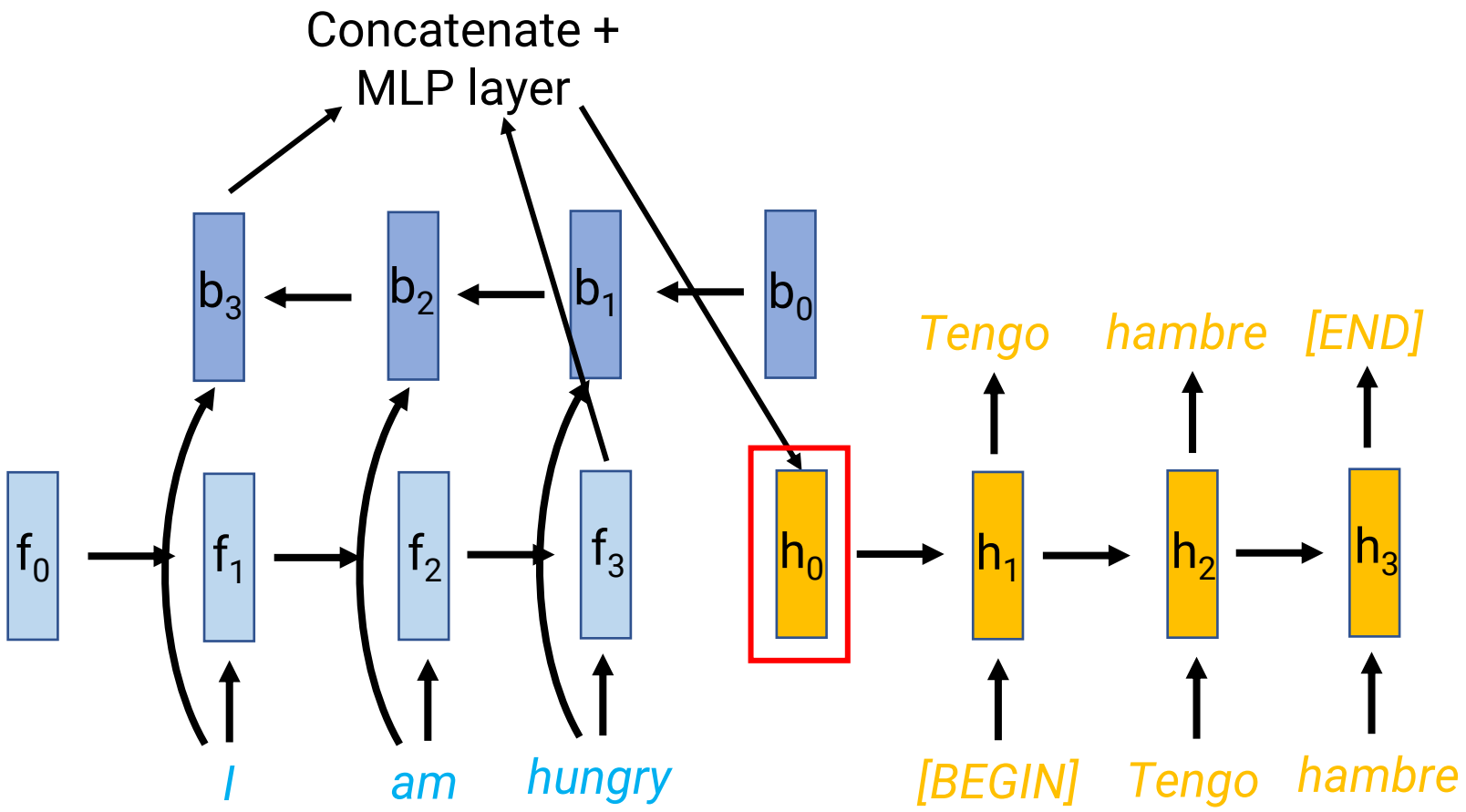
- Run one RNN left-to-right, and another one right-to-left
  - (I'll call forward-direction hidden states f_t, backward-direction hidden states b_t)
- Concatenate the 2 final hidden states as final representation
  - Note: This encoding is twice as large now—we've doubled the number of features passed to the final classifier

Sequence-to-sequence ("Encoder-decoder")



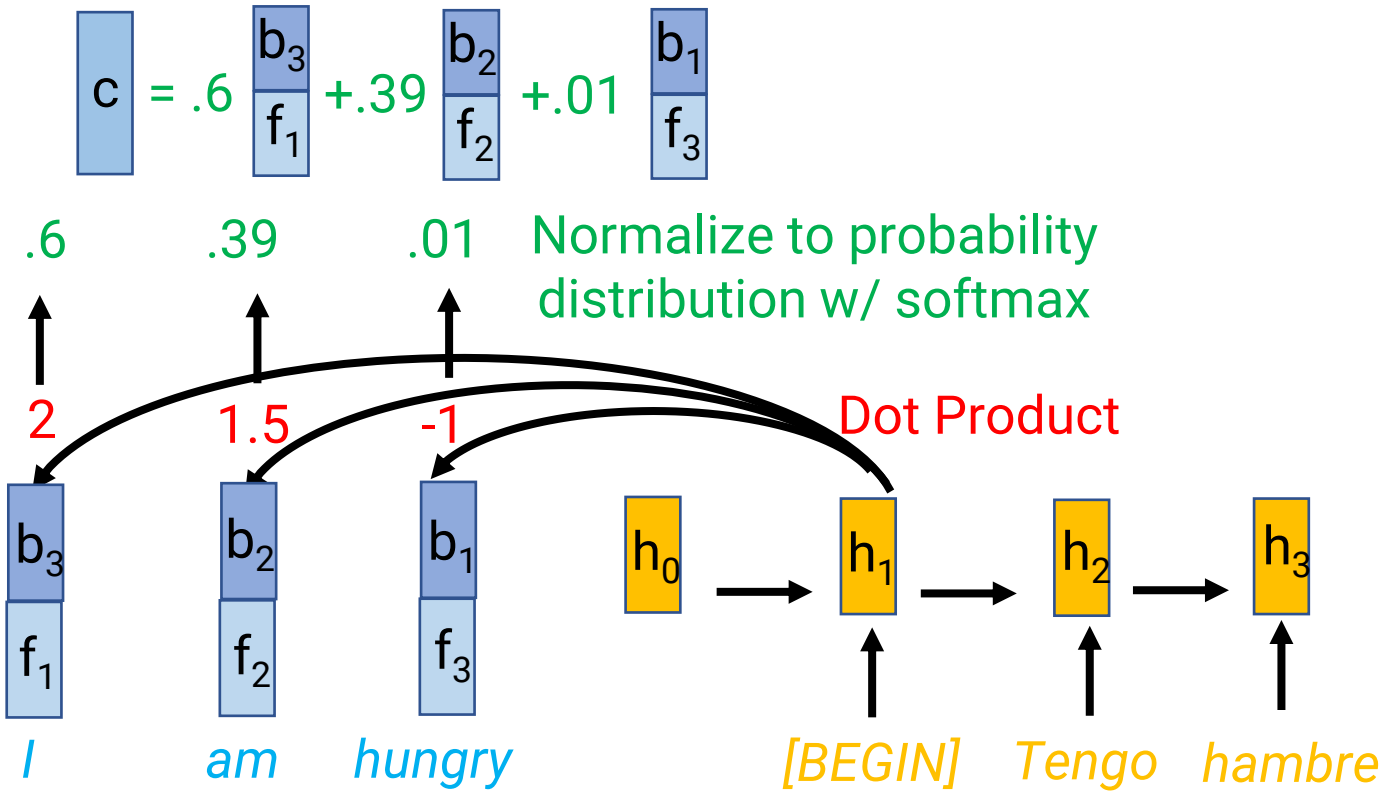
- Example: Machine Translation
  - Input = English text
  - Output = Spanish text
- Encoder: Process English sentence into vector
  - E.g. Bidirectional encoder + MLP layer to generate decoder's initial state
- Decoder: Use vector as initial hidden state and start doing language modeling in Spanish
- Vector space acts as a "shared language"

What's missing? Alignment



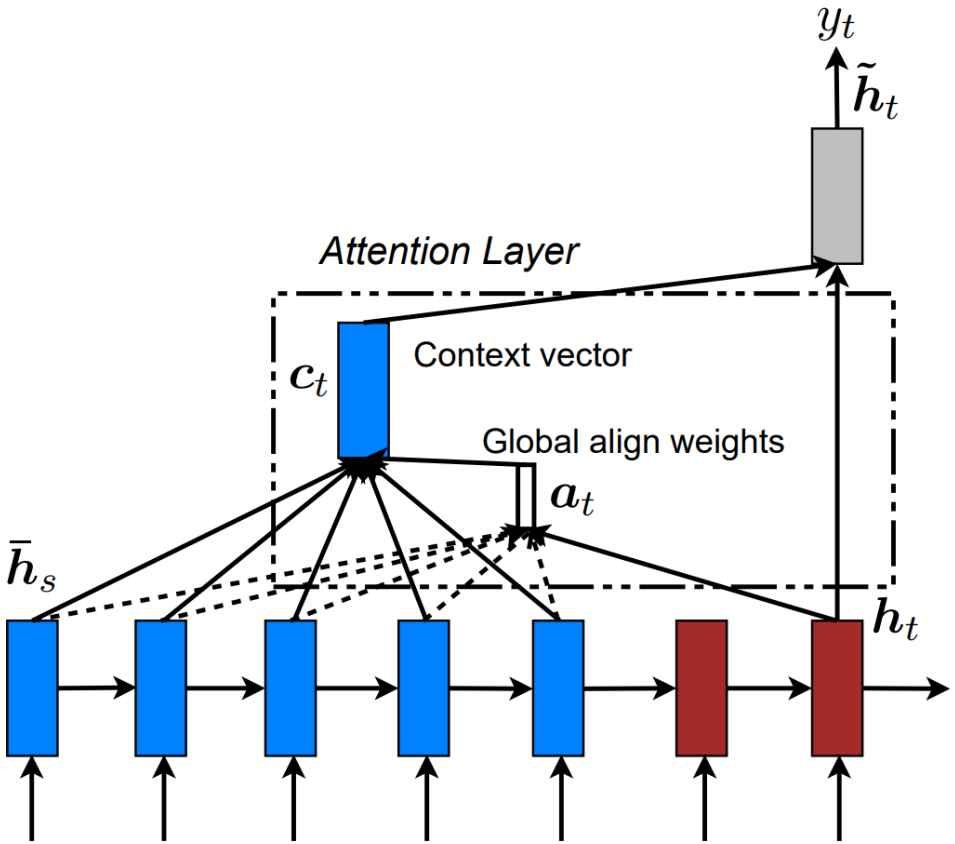
- Challenge: The single encoder output has to store information about the entire sentence in a single vector
- Would be much easier if we can "refer to our notes"
- Traditional MT: Alignment between input & output sentences
- Can we get a neural network to model alignments?

Attention



- Compute similarity between decoder hidden state and each encoder hidden state
  - E.g., dot product, if same size
- Normalize similarities to probability distribution with softmax
- Output: "Context" vector *c* = weighted average of encoder states based on the probabilities
  - No new parameters (like ReLU/max pool)
- Use *c* when computing decoder outputs or transitions
- Intuition
  - Step 1: Find similar input words
  - Step 2: Grab the encoder representation of those words
  - Step 3: Tell the decoder that this is relevant

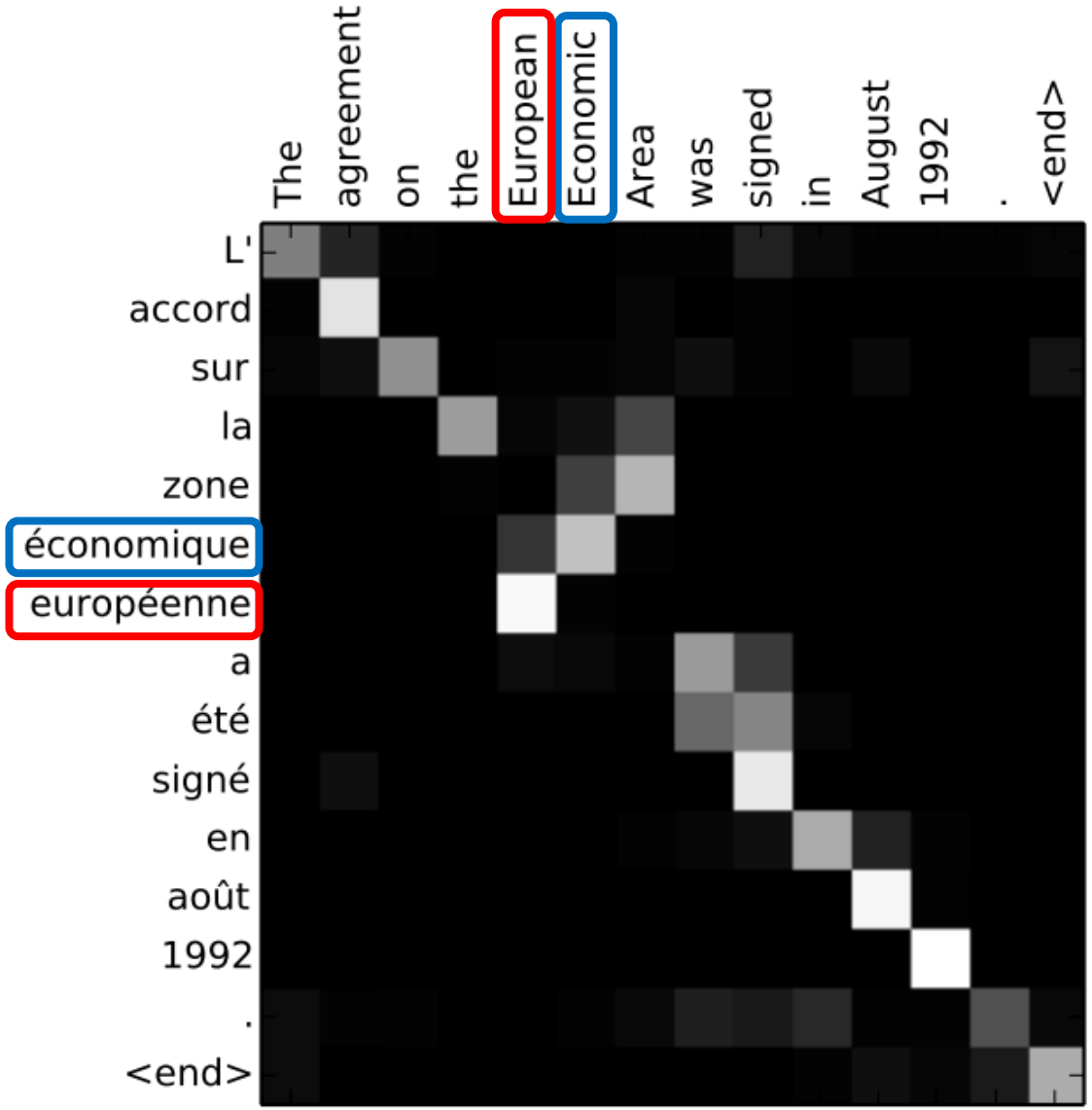
Example Attention Implementation



- Many similar ways one could implement an attention mechanism
- Example from a well-known 2015 paper by Luong et al. on machine translation
  - Blue = encoder states
  - Red = decoder states
  - Note: Encoder was unidirectional here
- Dot-product decoder state h_t with encoder states, then apply softmax to produce weights a_t
- Weighted sum of encoder states yields context vector c_t
- Context vector c_t concatenated with decoder state h_t, fed through 1 MLP layer to generate $\tilde{h}_t$
- $\tilde{h}_t$ used to make prediction y_t

Luong, Pham, and Manning. Effective Approaches to Attention-based Neural Machine Translation. EMNLP 2015.

Visualizing attention



- Source is English, Target is French
- Each row is a probability distribution over the English text
- Alignment makes sense, overcomes word order differences
  - When generating “économique” attend to “Economic”
  - When generating “européenne” attend to “European”

Conclusion

- Ways to use RNNs
  - As a decoder: To generate text
  - As an encoder: To produce feature vectors for text
  - Sequence-to-sequence: Use 2 RNNs, one for each purpose
- Attention: Know which part of the input matters when generating each word of the output

Announcements

- HW2 due today @ 11:59pm
  - Q4: Don’t worry about differences in accuracy numbers when running same code on different computers
- Section Friday: Midterm Review (practice exam + questions)
- Midterm exam next Tuesday, October 10
  - In-class, 80 minutes, one double-sided 8.5x11 sheet of notes
  - Room assignments (also on Piazza)
    - Last name A-O: LVL 17 (this room)
    - Last name P-Z: THH 116