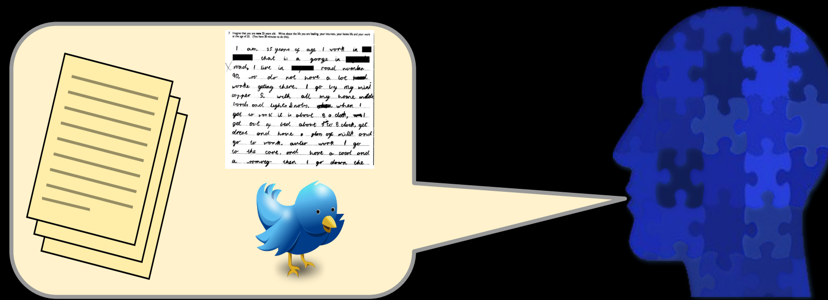
# Attention and Transformer Sequence Models

CSE354 - Spring 2021
Natural Language Processing



Most NLP Tasks. E.g.

- Sequence Tasks
  - Language Modeling
  - Machine Translation
  - Speech Recognition
  - Named Entity
- Document Classification

- Transformer Networks
  - Transformers
  - BERT

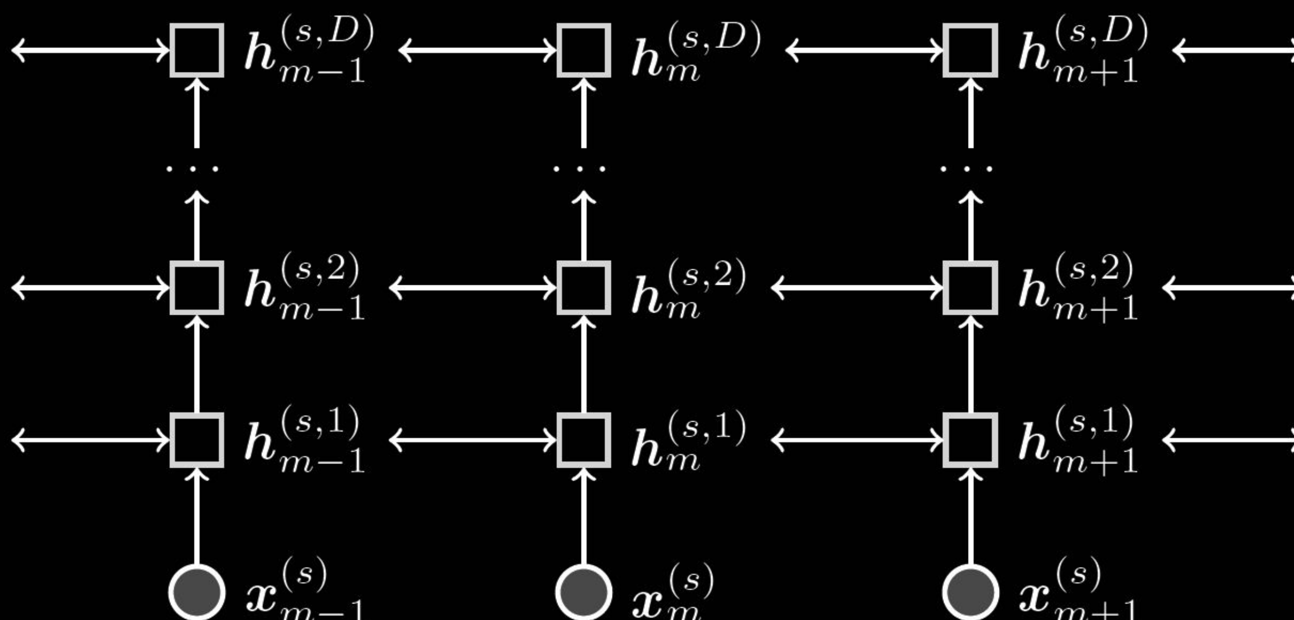
# Evolution of Sequence Modeling

RNNs LSTMs LSTMS with Attention Attention without LSTMs

# Evolution of Sequence Modeling

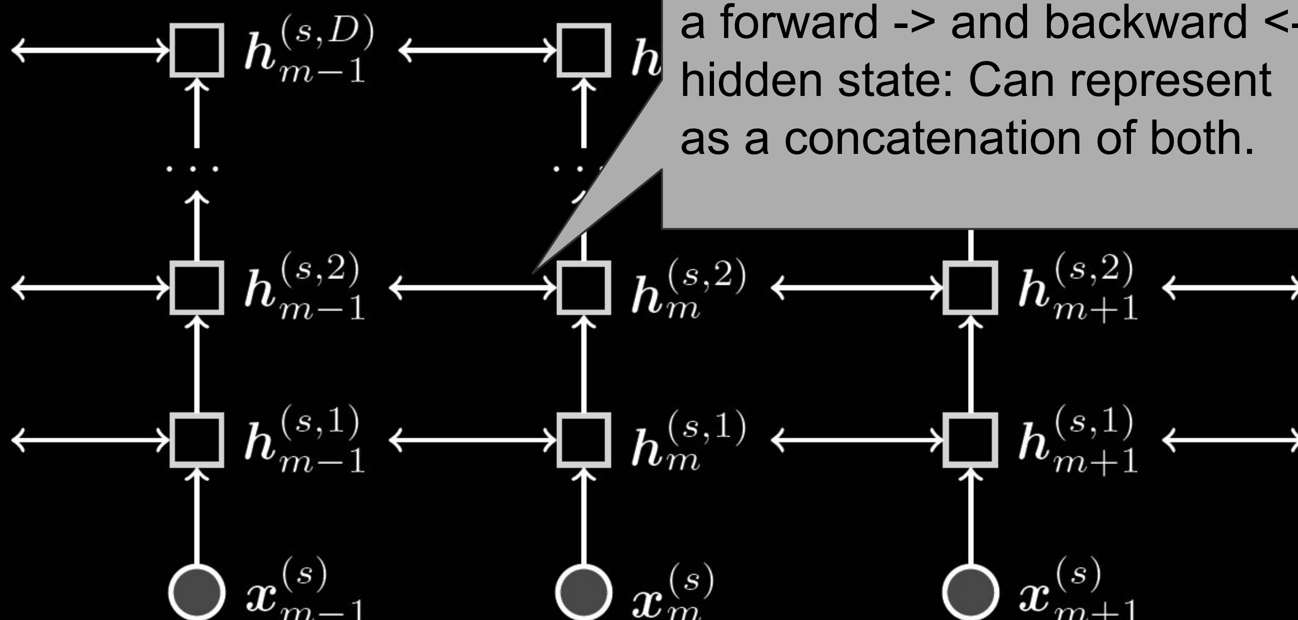
RNNs LSTMs **LSTMS with Attention** Attention without LSTMs

# Multi-level bidirectional RNN (LSTM or GRU)




(Eisenstein, 2018)

# Multi-level bidirectional RNN (LSTM or GRU)

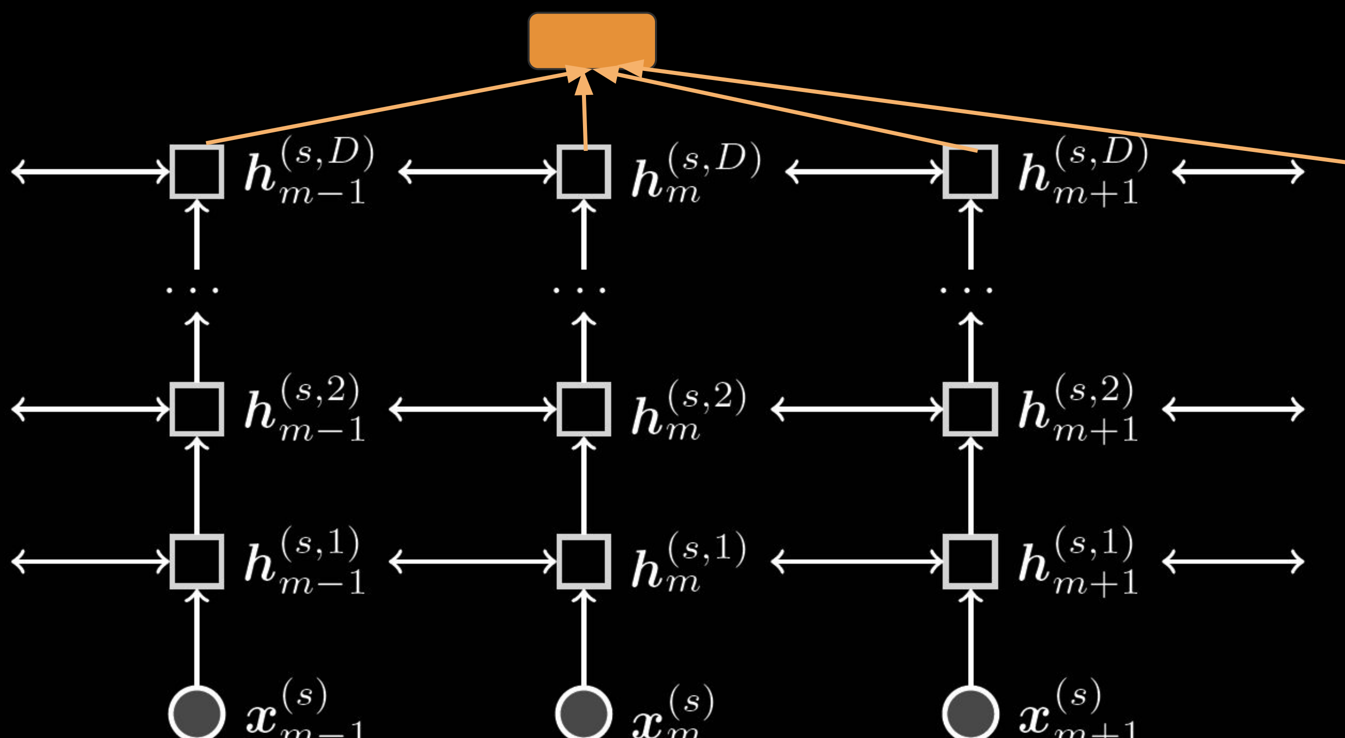



**Bidirectional:** Each node has a forward -> and backward <- hidden state: Can represent as a concatenation of both.

(Eisenstein, 2018)

# Multi-level bidirectional RNN (LSTM or GRU)

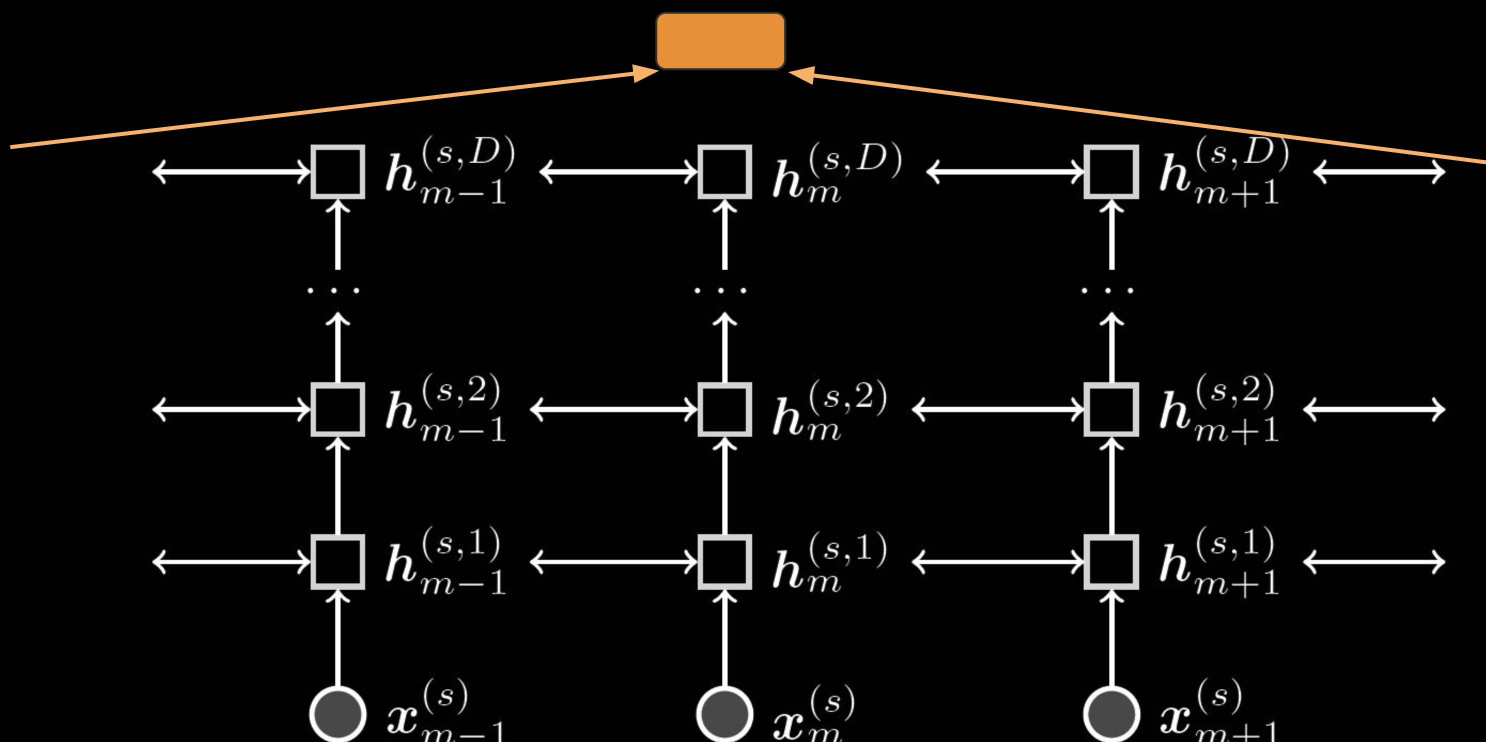
Average of top layer is an embedding (average of concatinated vectors)




(Eisenstein, 2018)

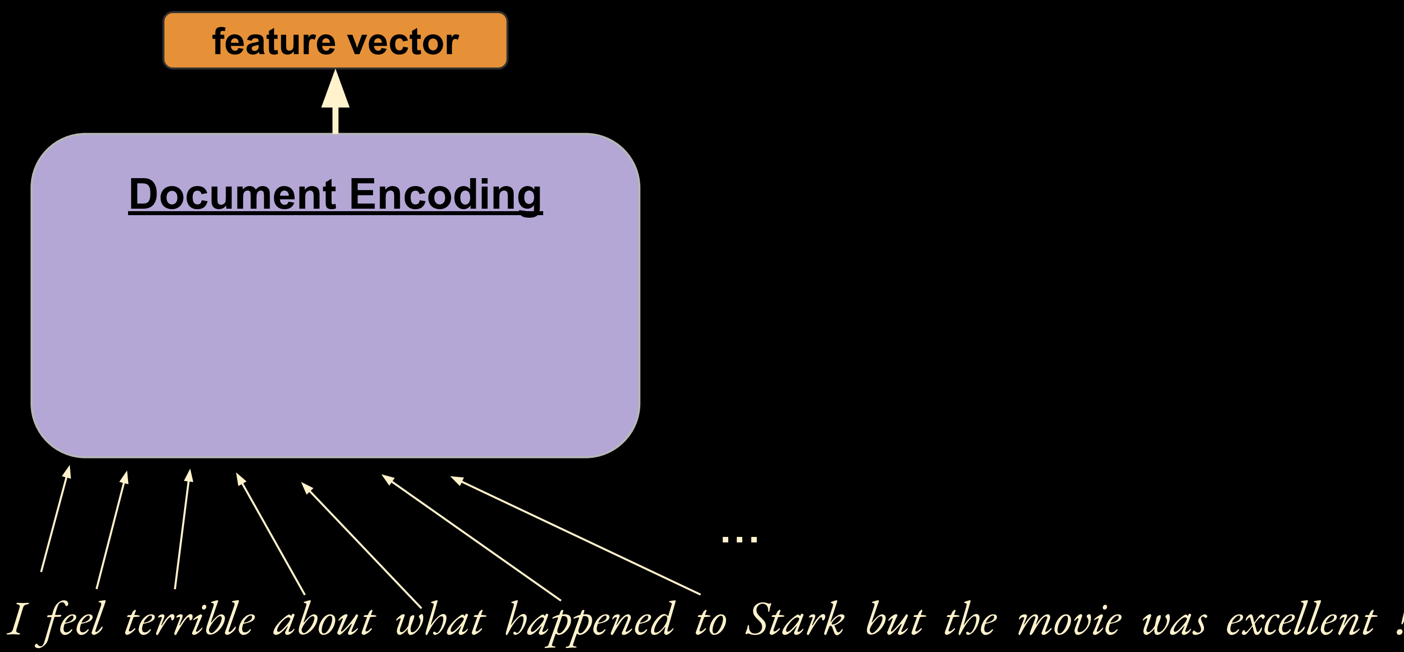
# Multi-level bidirectional RNN (LSTM or GRU)

Sometimes just use left-most and right-most hidden state instead

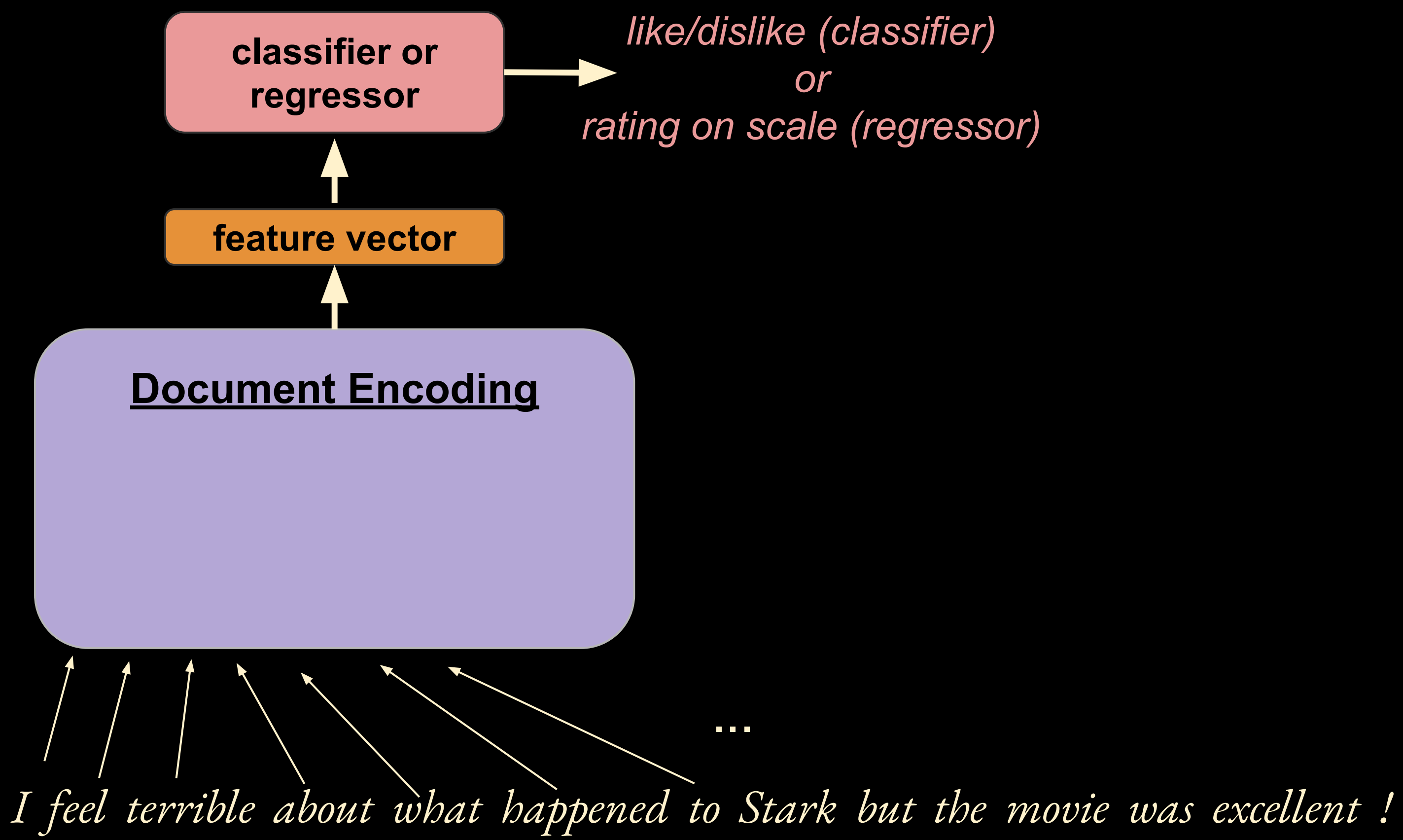



(Eisenstein, 2018)

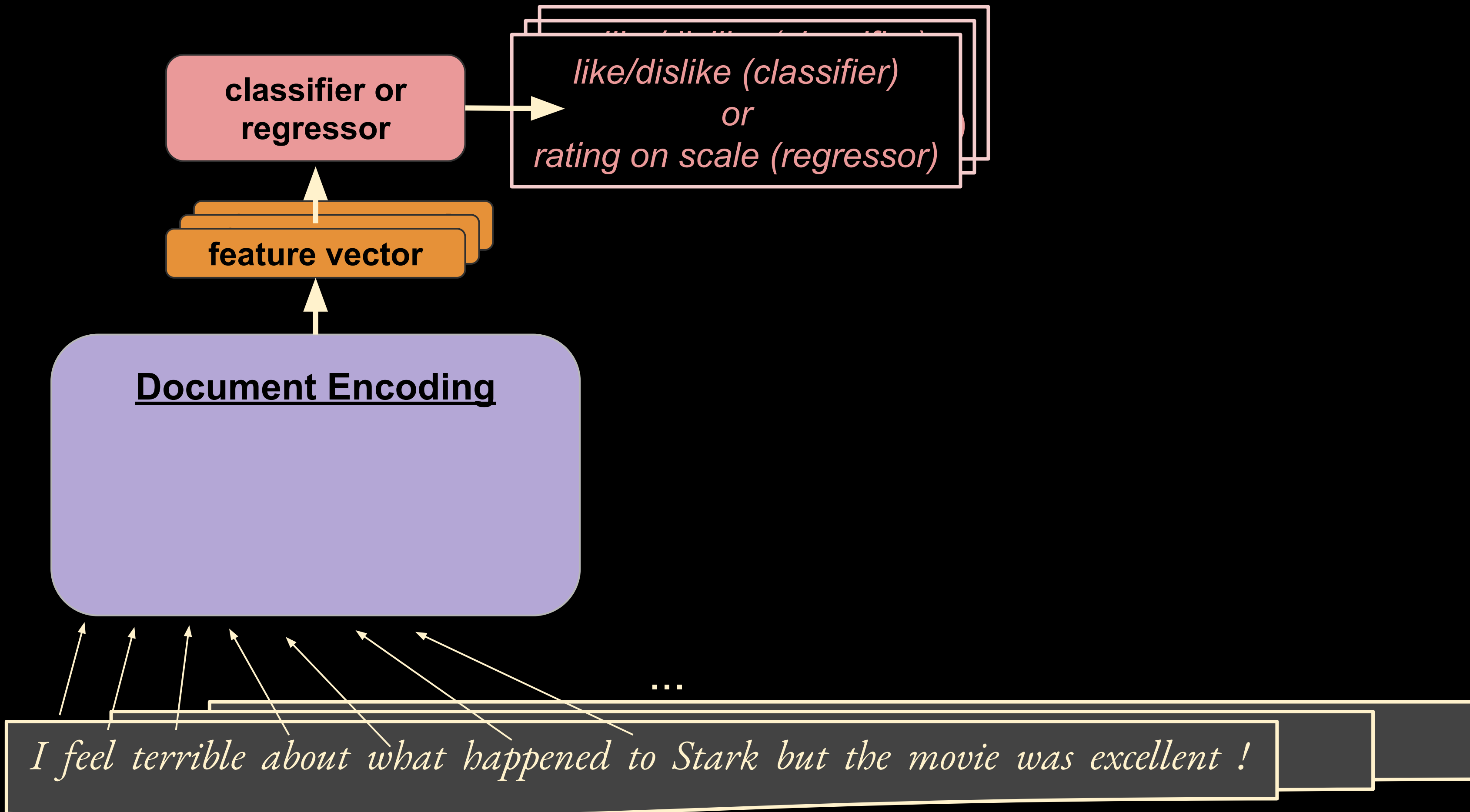
# Sentiment Analysis:
## Example Application of Single Representation of document



# Sentiment Analysis:
## Example Application of Single Representation of document



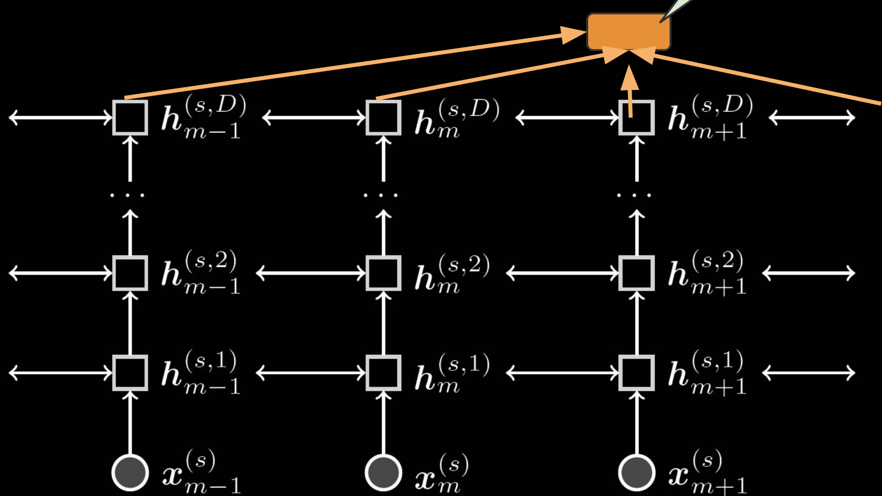
# Sentiment Analysis:
## Example Application of Single Representation of document



# Encoder

A representation of input.

Document classification only needs an ***encoder***: Goes from input into a single representation.

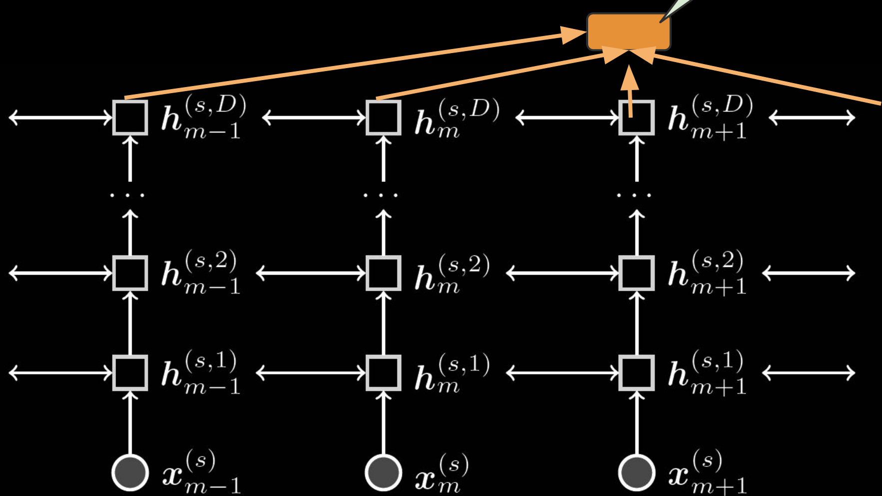



(Eisenstein, 2018)

# Encoder

A representation of input.

Document classification only needs an ***encoder***: Goes from input into a single representation.



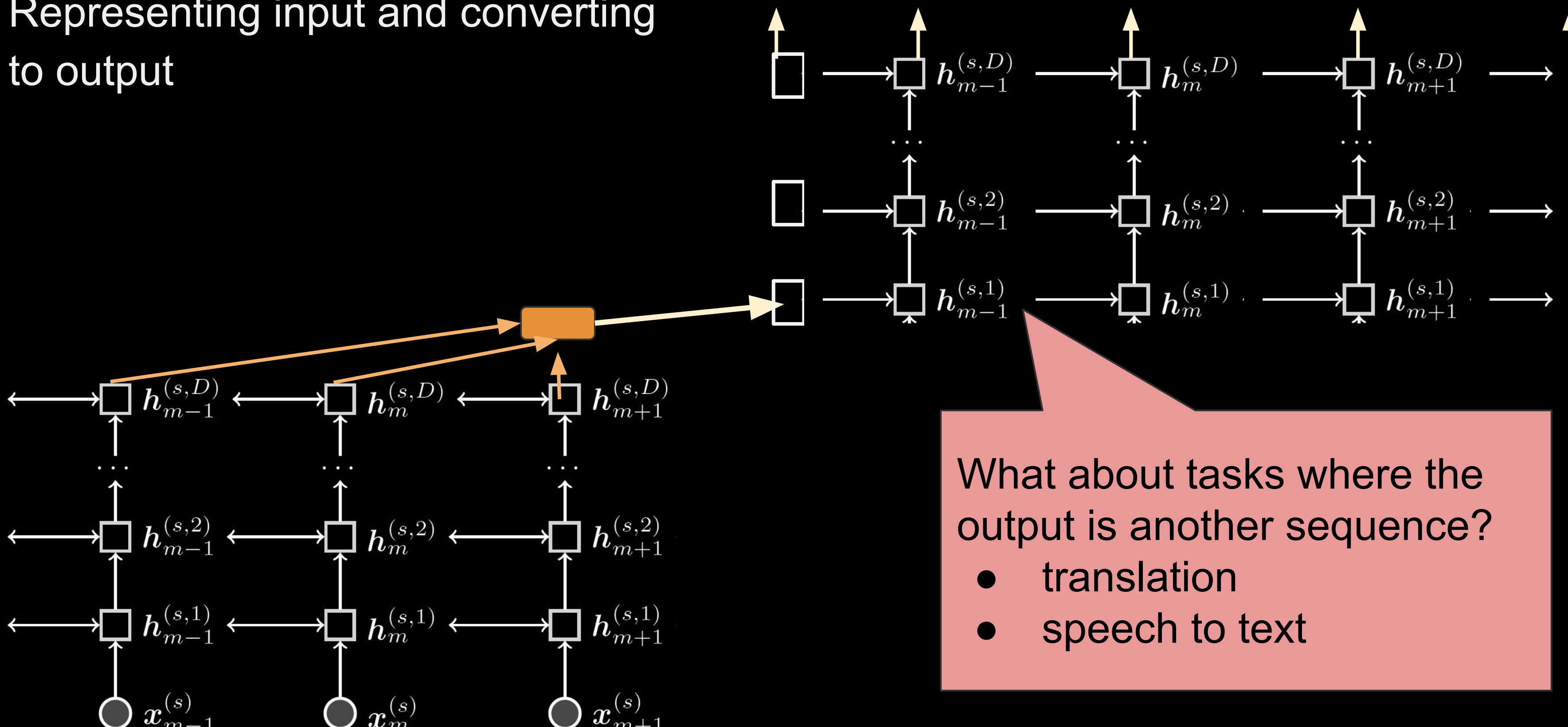

What about tasks where the output is another sequence?

- translation
- speech to text

(Eisenstein, 2018)

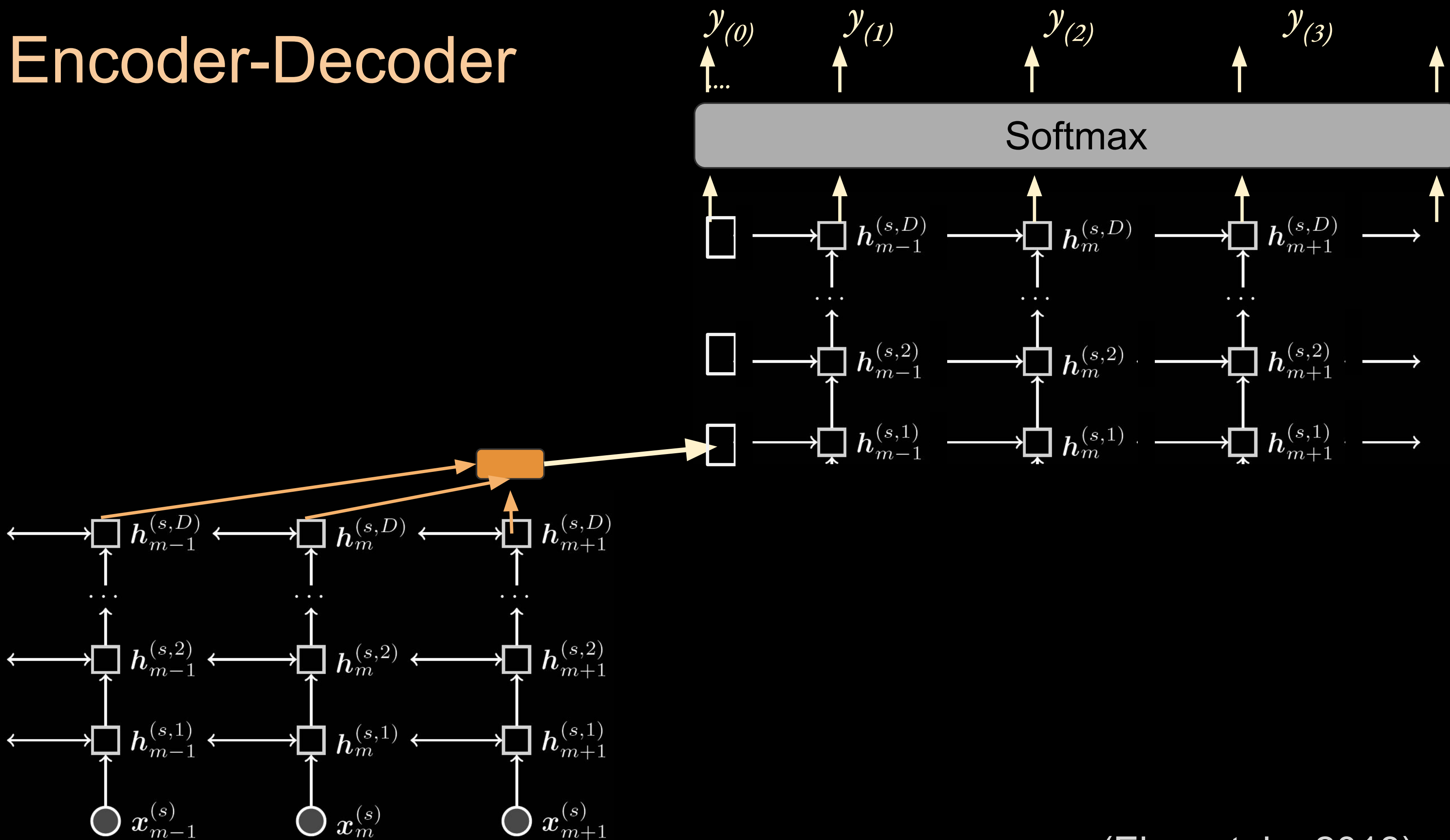
# Encoder-Decoder (seq to seq models)

Representing input and converting to output



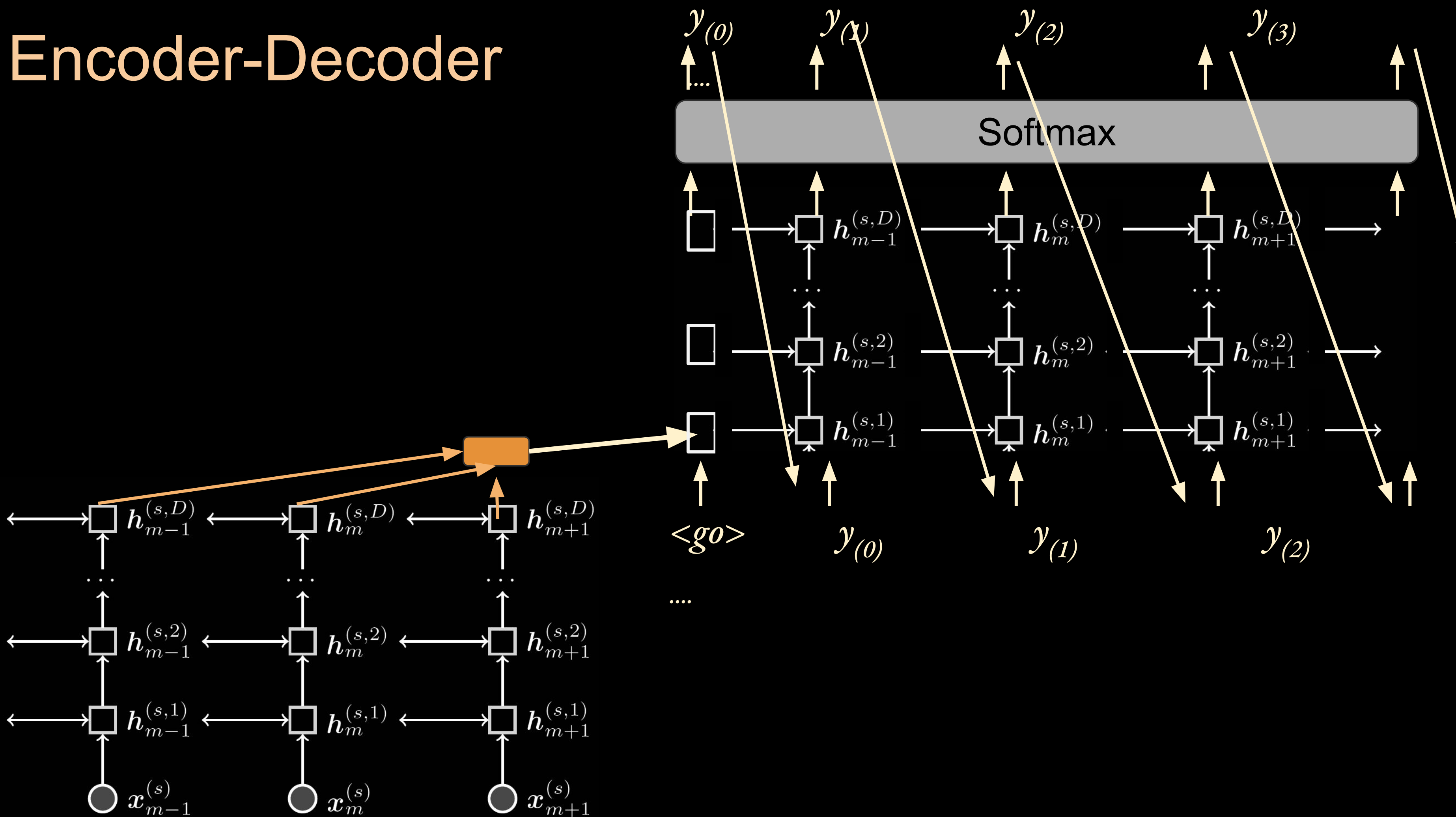

(Eisenstein, 2018)

# Encoder-Decoder

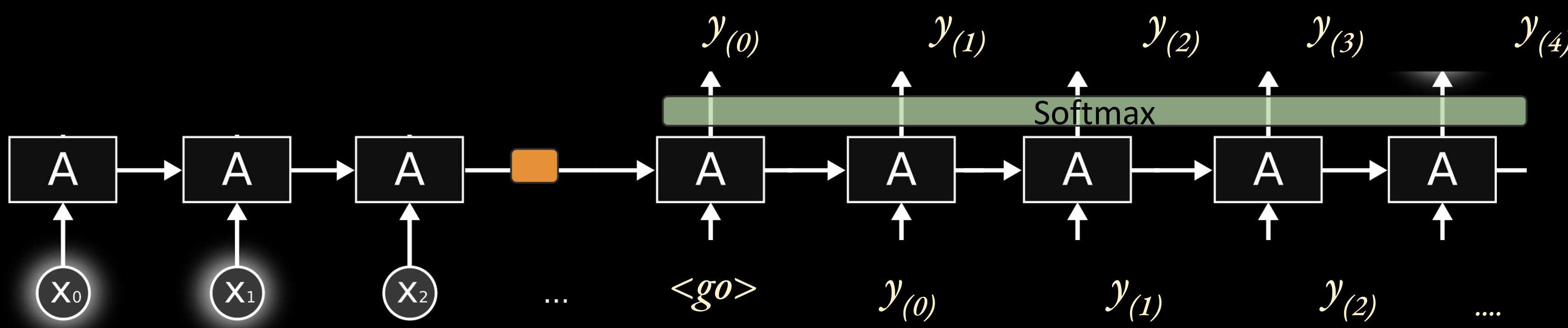



(Eisenstein, 2018)

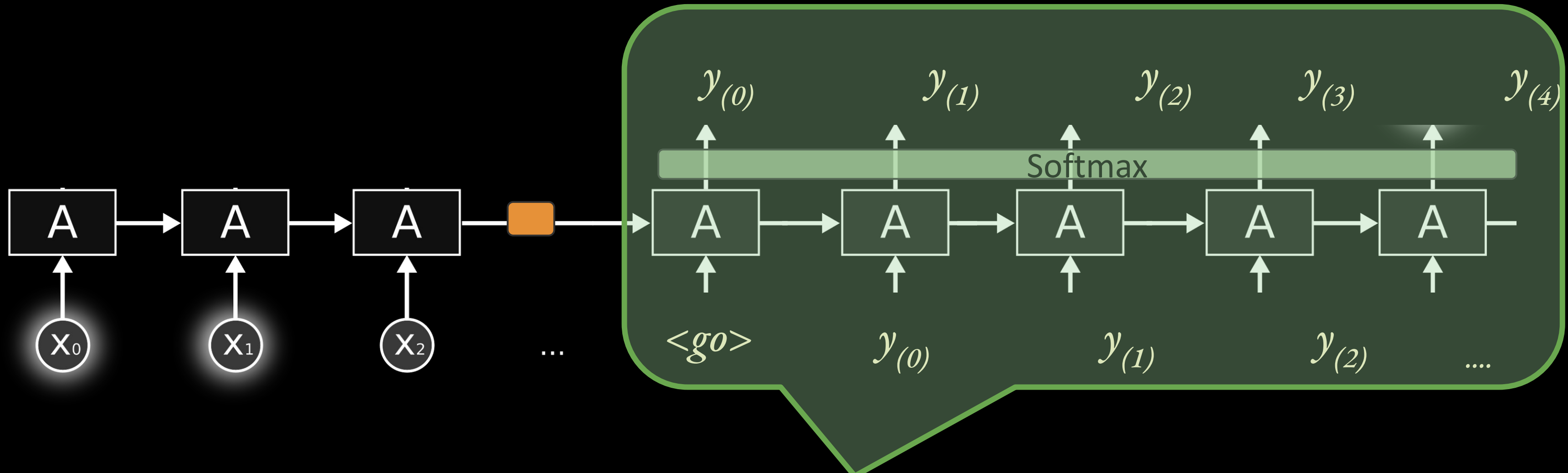
# Encoder-Decoder



# Encoder-Decoder (Simpler Representation)

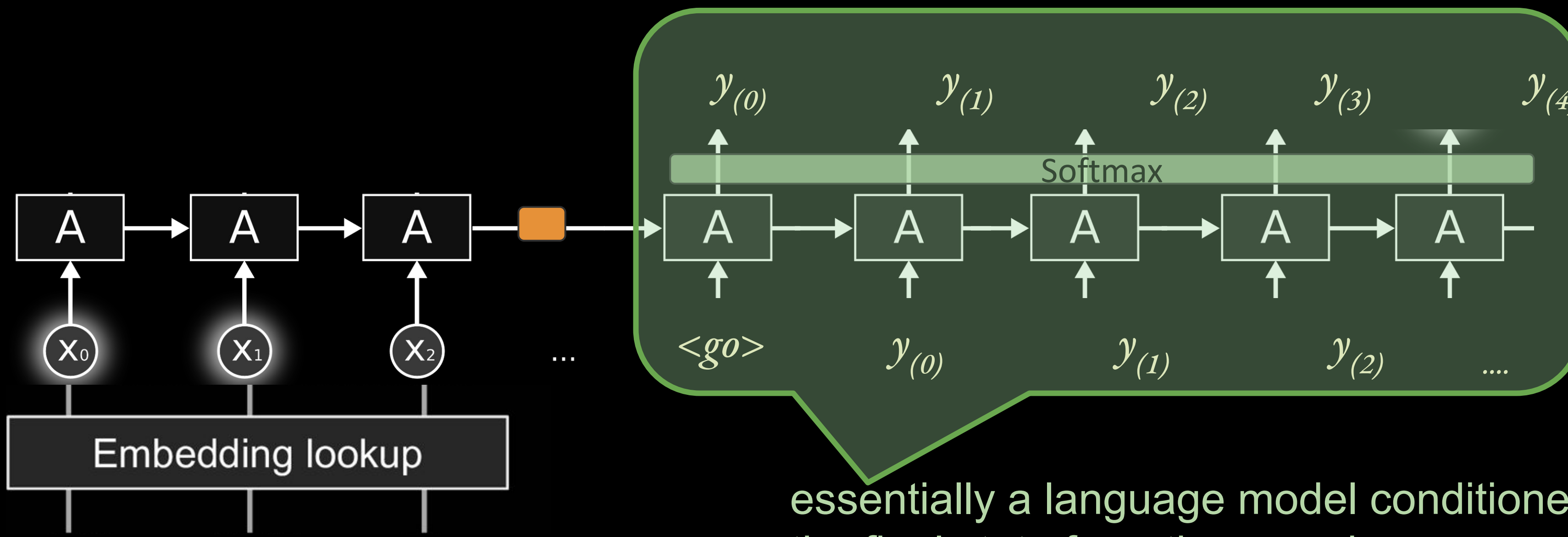


# Encoder-Decoder (Simpler Representation)



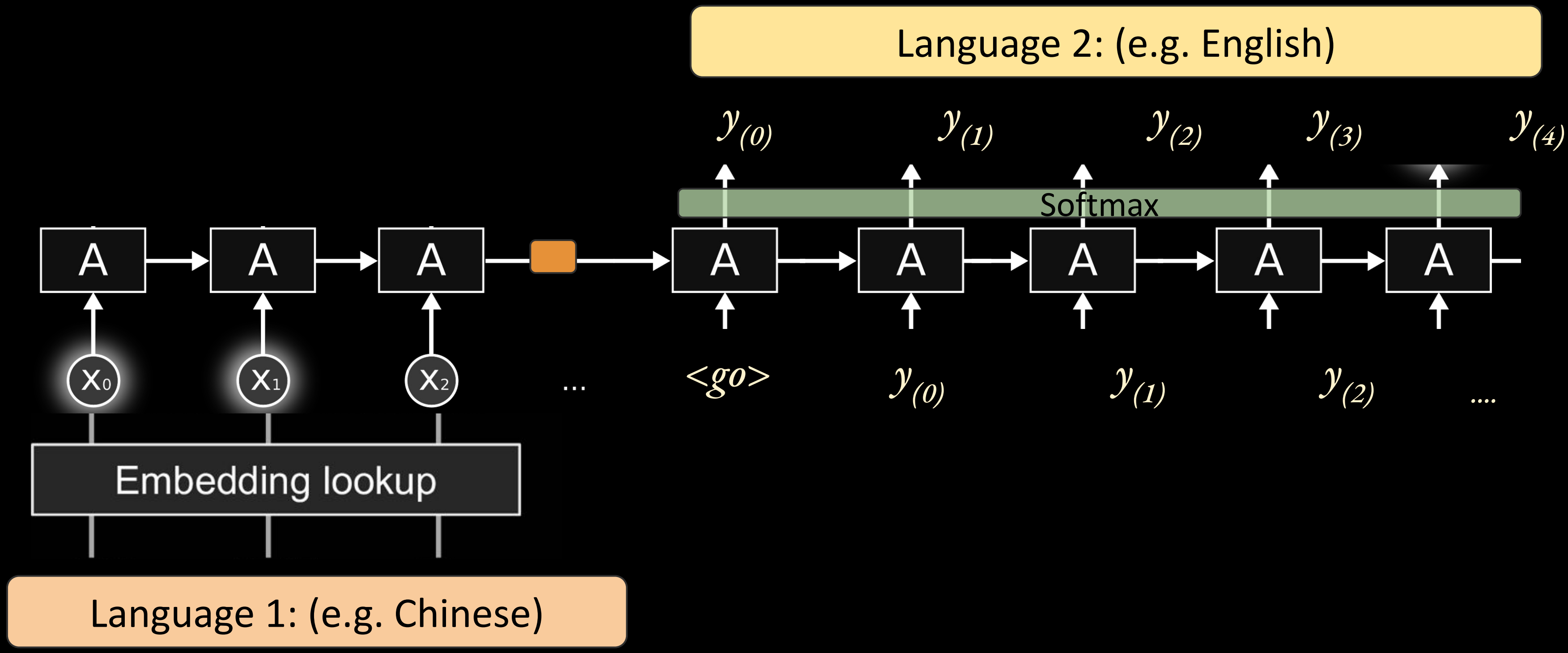

essentially a language model conditioned on the final state from the encoder.

# Encoder-Decoder (Simpler Representation)




essentially a language model conditioned on the final state from the encoder.

# Encoder-Decoder (Simpler Representation)

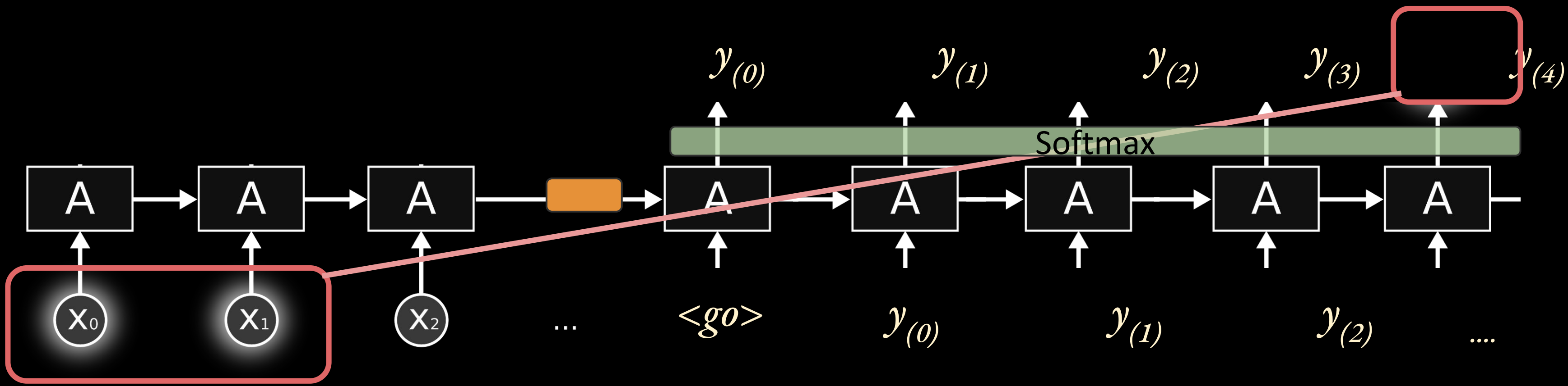


# Encoder-Decoder

Challenge:

- Long distance dependency when translating:

*The ball was kicked by kayla.*



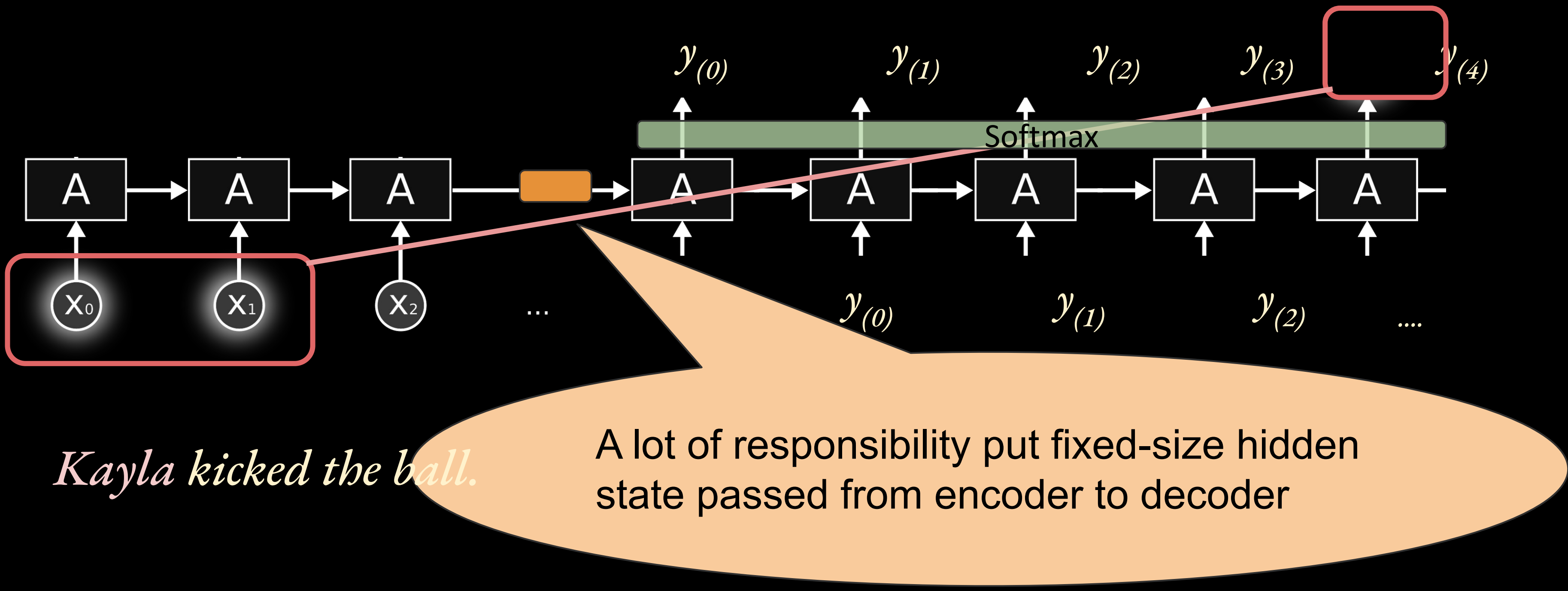

*Kayla kicked the ball.*

# Encoder-Decoder

Challenge:

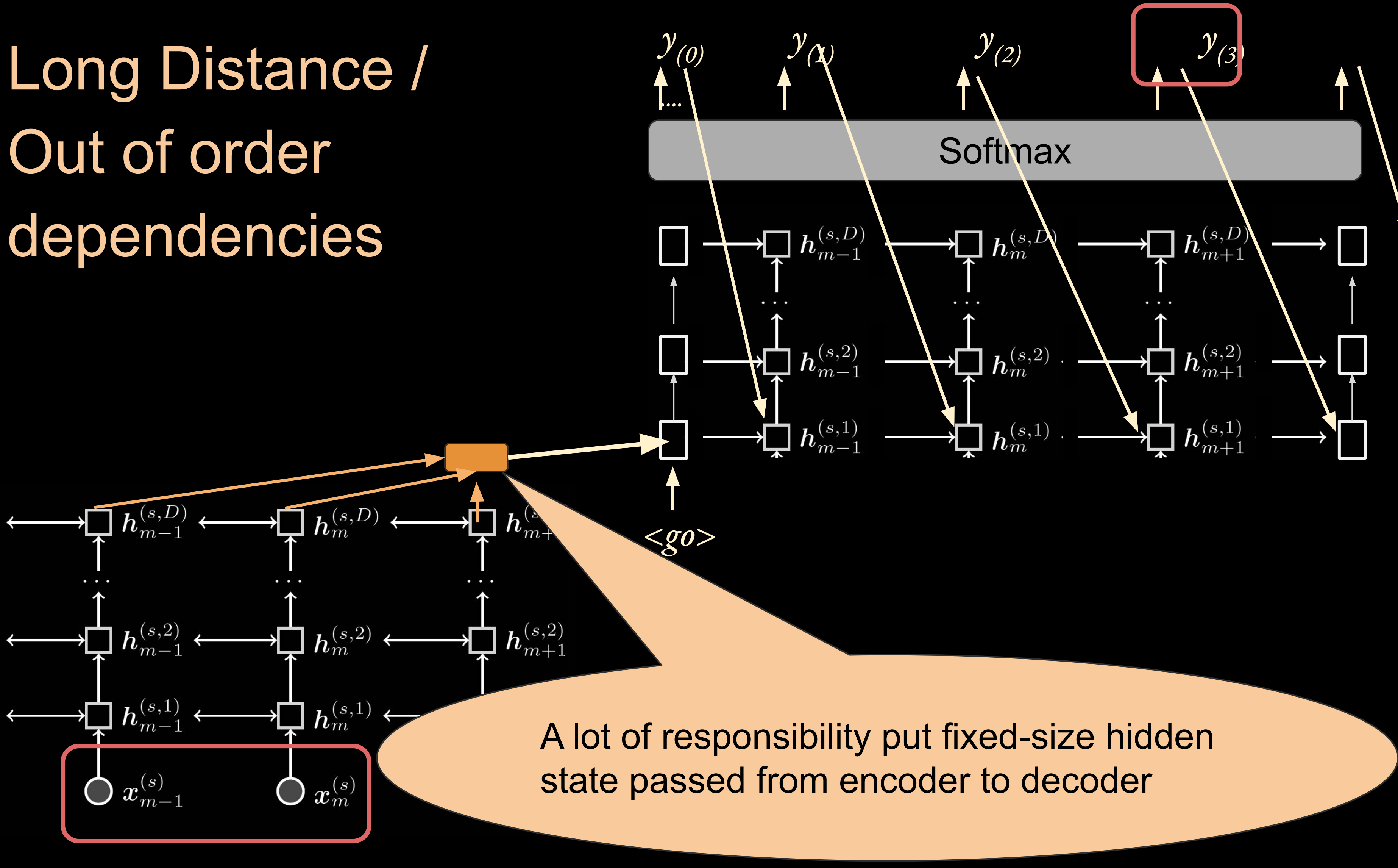
*The ball was kicked by kayla.*

- Long distance dependency when translating:

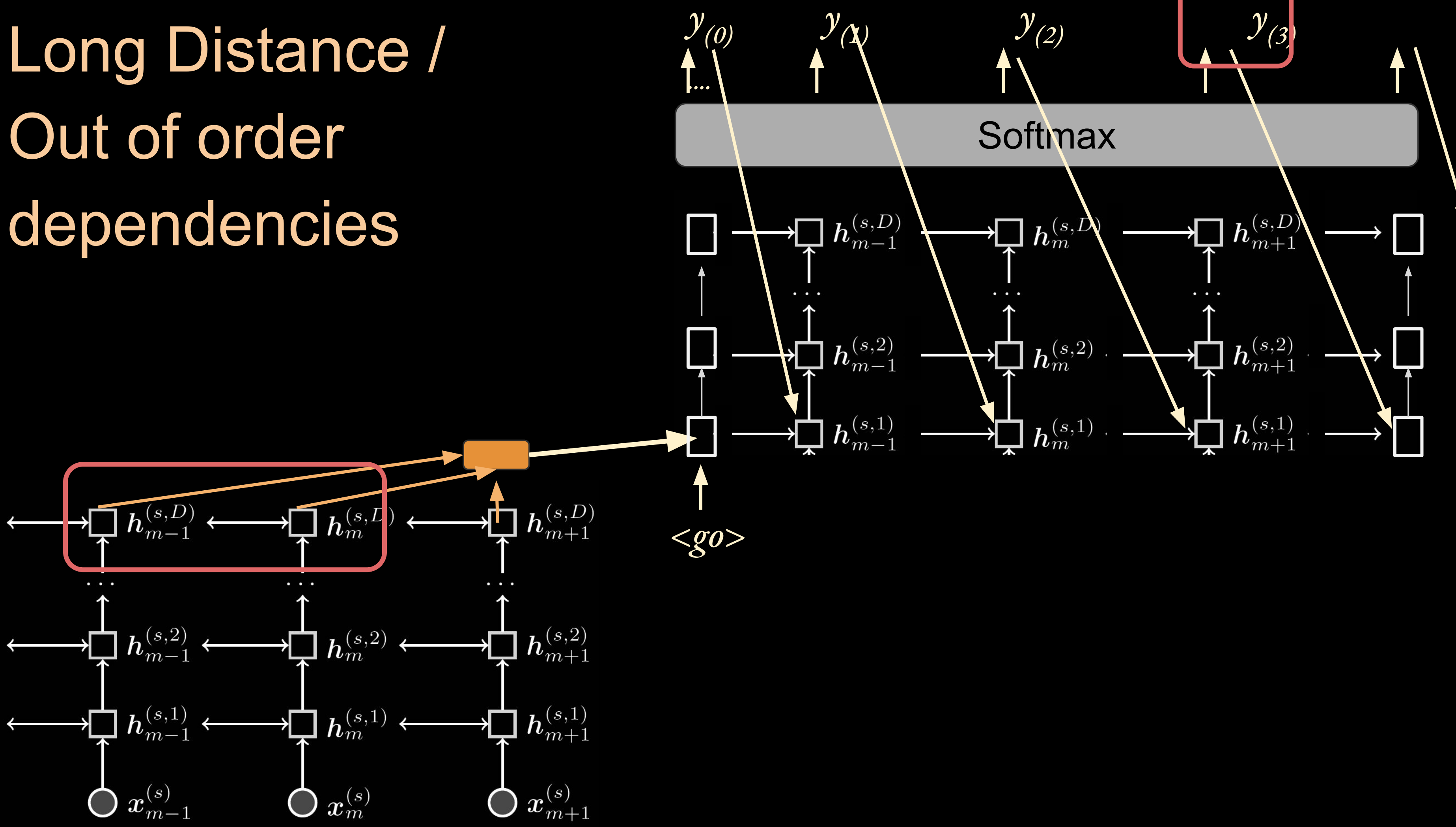



*Kayla kicked the ball.*

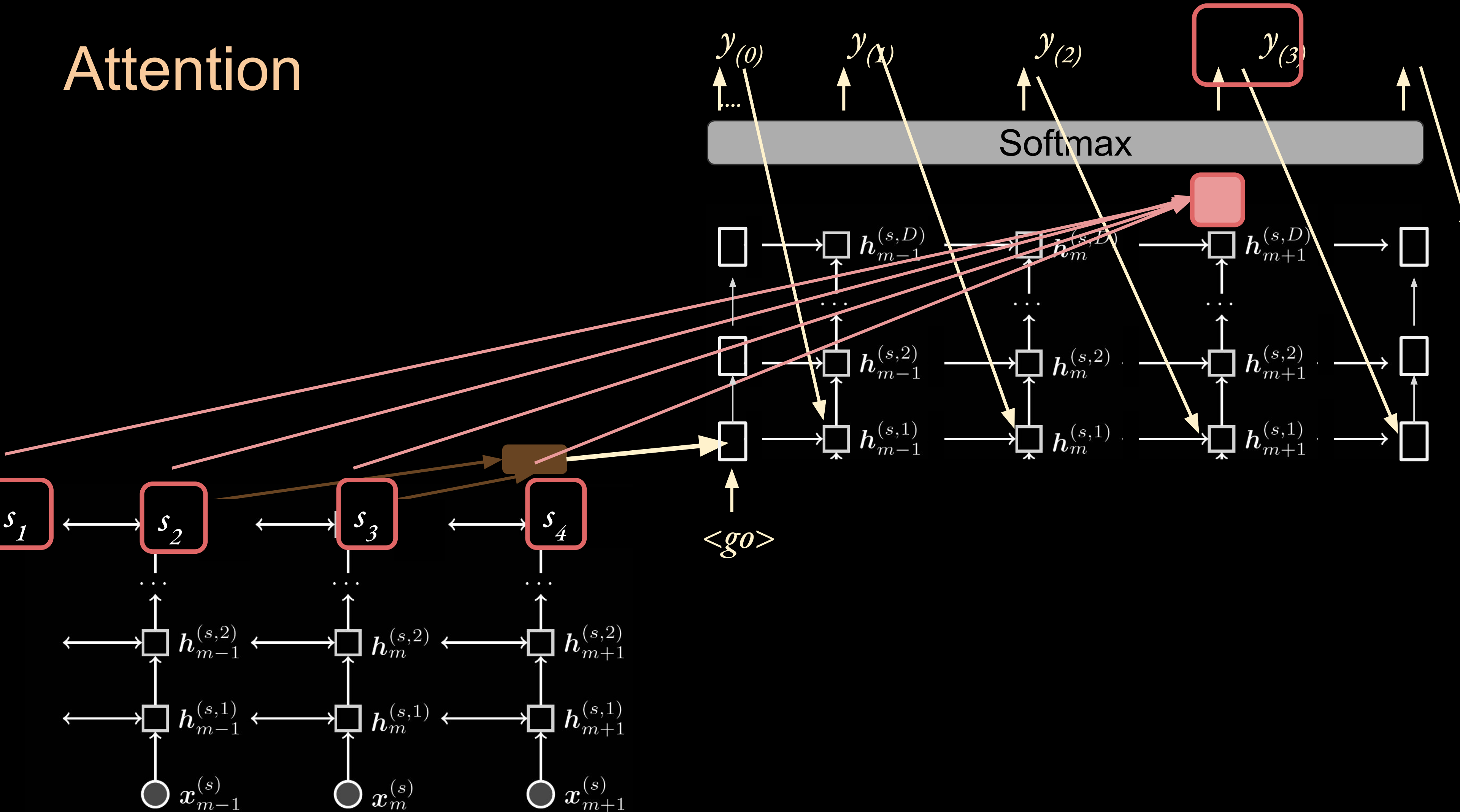
# Long Distance / Out of order dependencies



# Long Distance / Out of order dependencies



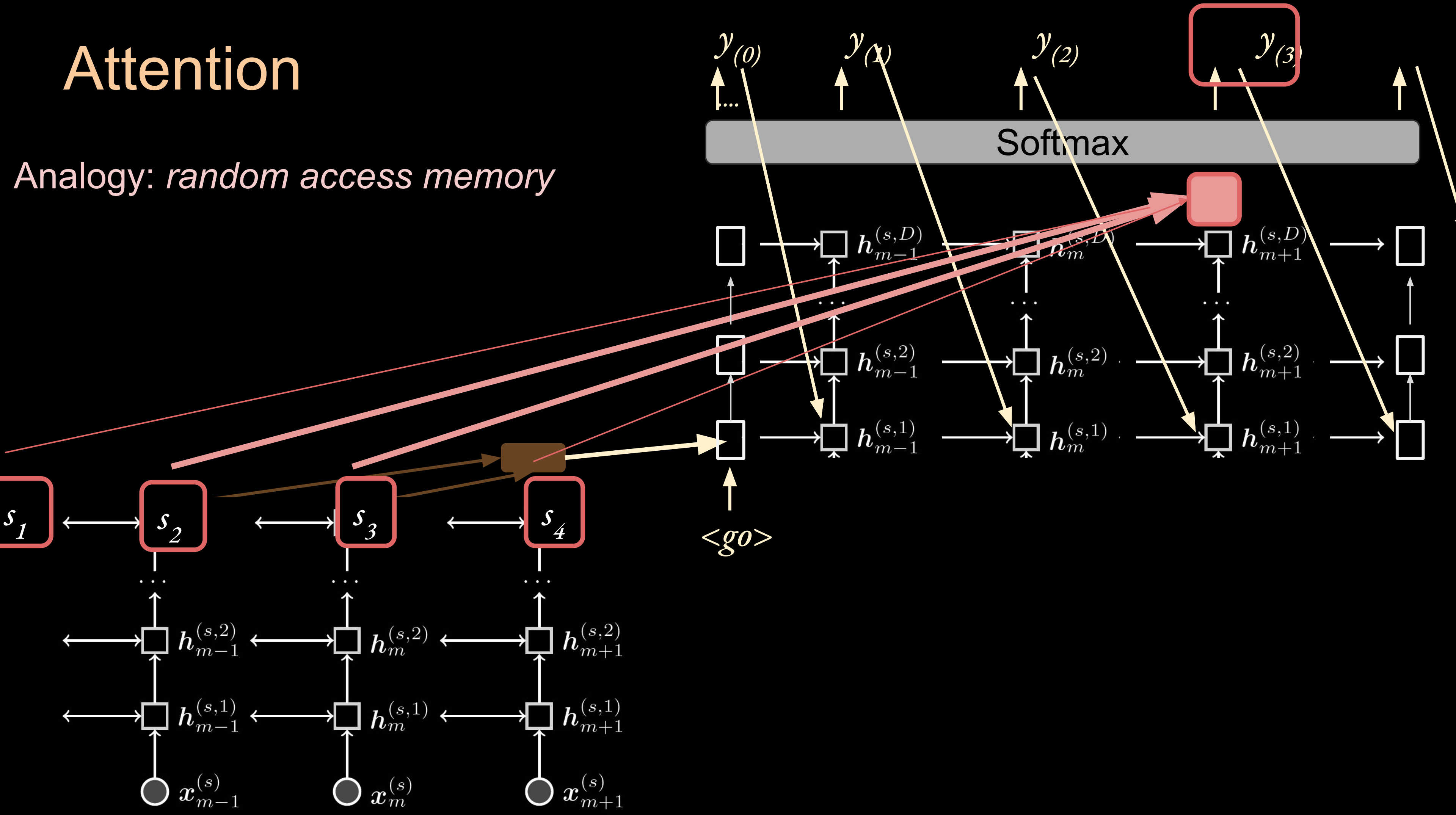
# Attention



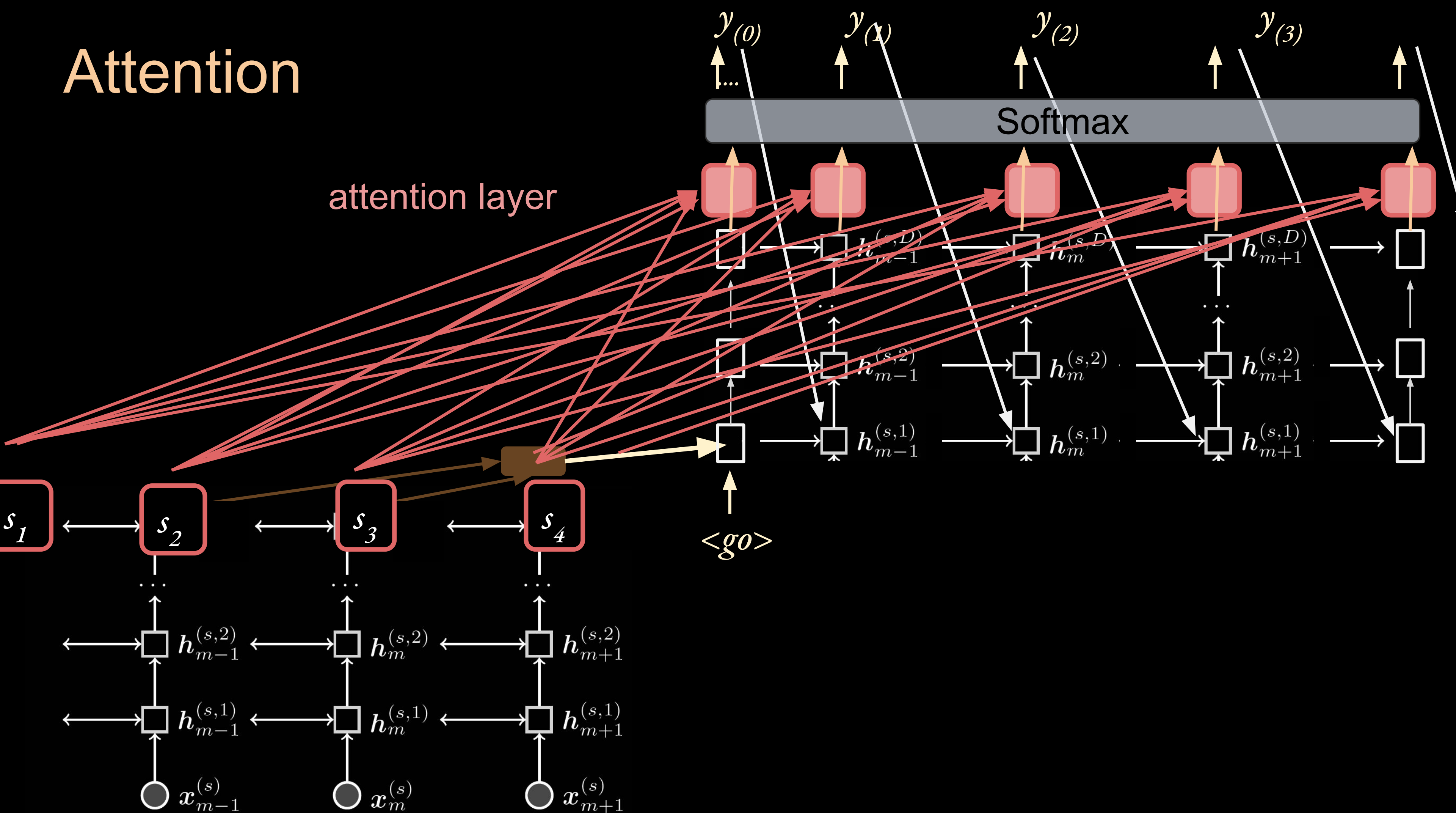
$y_{(0)}$
$y_{(1)}$
$y_{(2)}$
$y_{(3)}$
Softmax
$h_{m-1}^{(s,D)}$
$h_m^{(s,D)}$
$h_{m+1}^{(s,D)}$
$h_{m-1}^{(s,2)}$
$h_m^{(s,2)}$
$h_{m+1}^{(s,2)}$
$h_{m-1}^{(s,1)}$
$h_m^{(s,1)}$
$h_{m+1}^{(s,1)}$
<go>
$s_1$
$s_2$
$s_3$
$s_4$
$h_{m-1}^{(s,2)}$
$h_m^{(s,2)}$
$h_{m+1}^{(s,2)}$
$h_{m-1}^{(s,1)}$
$h_m^{(s,1)}$
$h_{m+1}^{(s,1)}$
$x_{m-1}^{(s)}$
$x_m^{(s)}$
$x_{m+1}^{(s)}$

# Attention

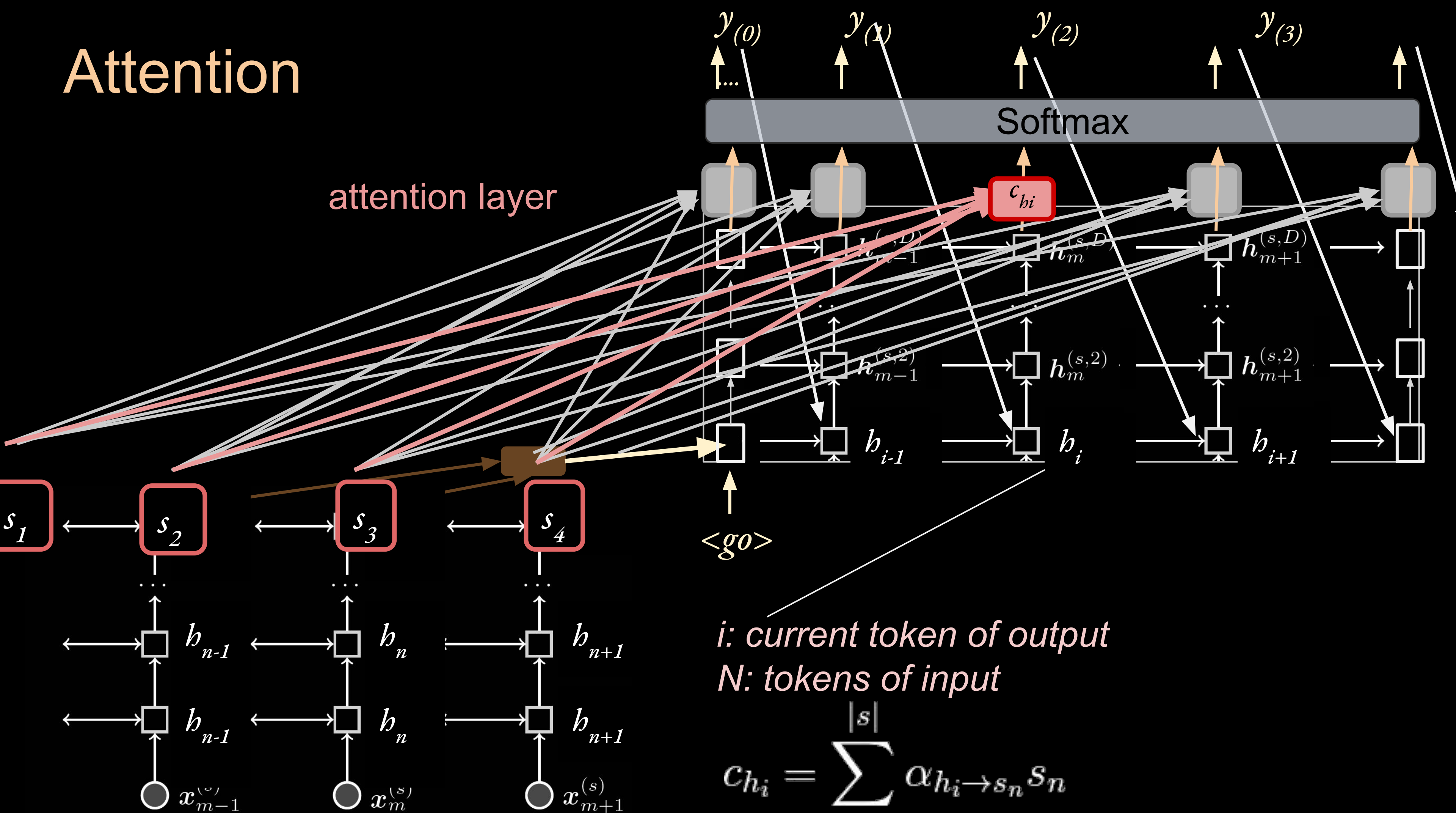
Analogy: *random access memory*



# Attention


$y_{(0)}$
$y_{(1)}$
$y_{(2)}$
$y_{(3)}$
Softmax
attention layer
$h_{m-1}^{(s,D)}$
$h_{m}^{(s,D)}$
$h_{m+1}^{(s,D)}$
$h_{m-1}^{(s,2)}$
$h_{m}^{(s,2)}$
$h_{m+1}^{(s,2)}$
$h_{m-1}^{(s,1)}$
$h_{m}^{(s,1)}$
$h_{m+1}^{(s,1)}$
<go>
$s_1$
$s_2$
$s_3$
$s_4$
$x_{m-1}^{(s)}$
$x_{m}^{(s)}$
$x_{m+1}^{(s)}$

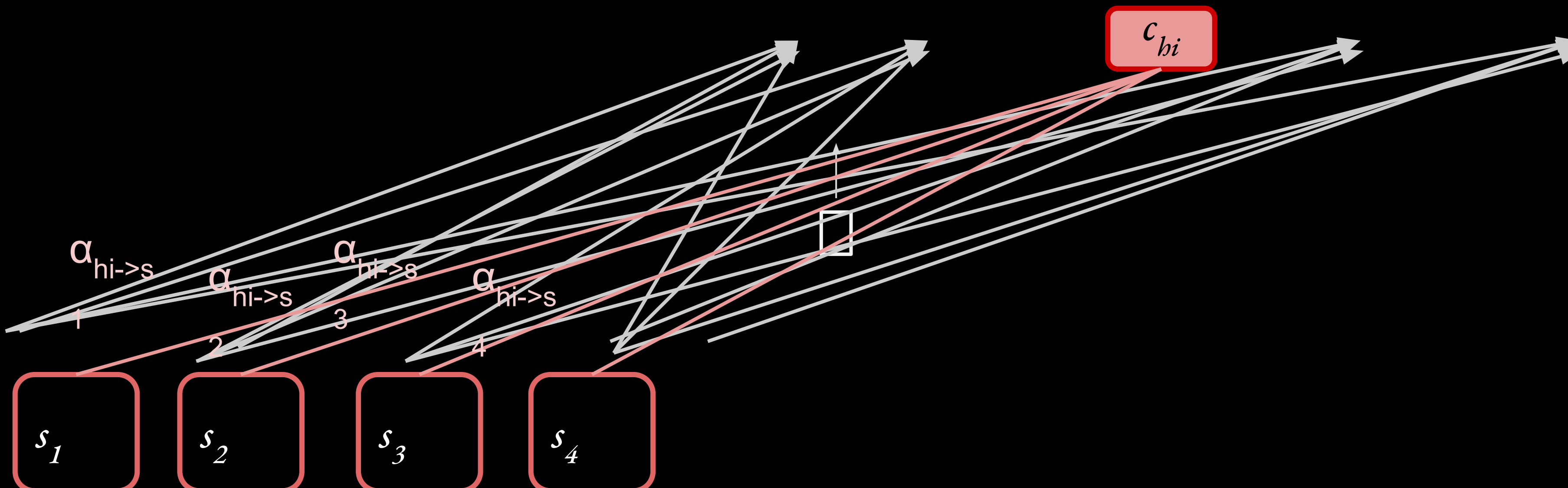
# Attention




*i: current token of output*
*N: tokens of input*

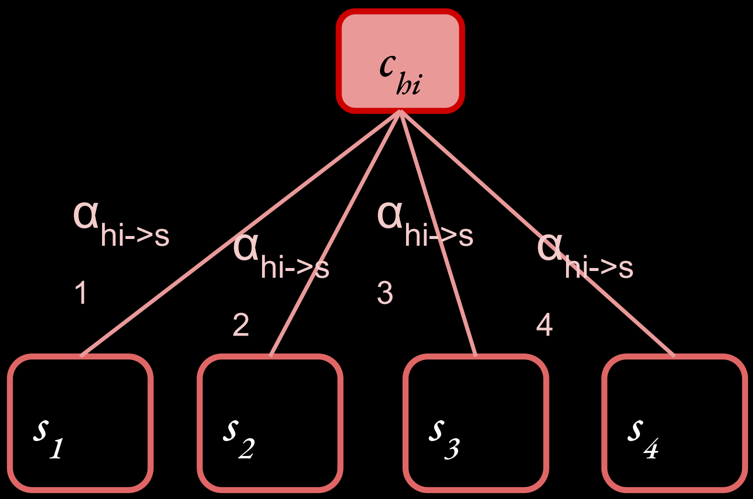
$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} s_n$$

# Attention



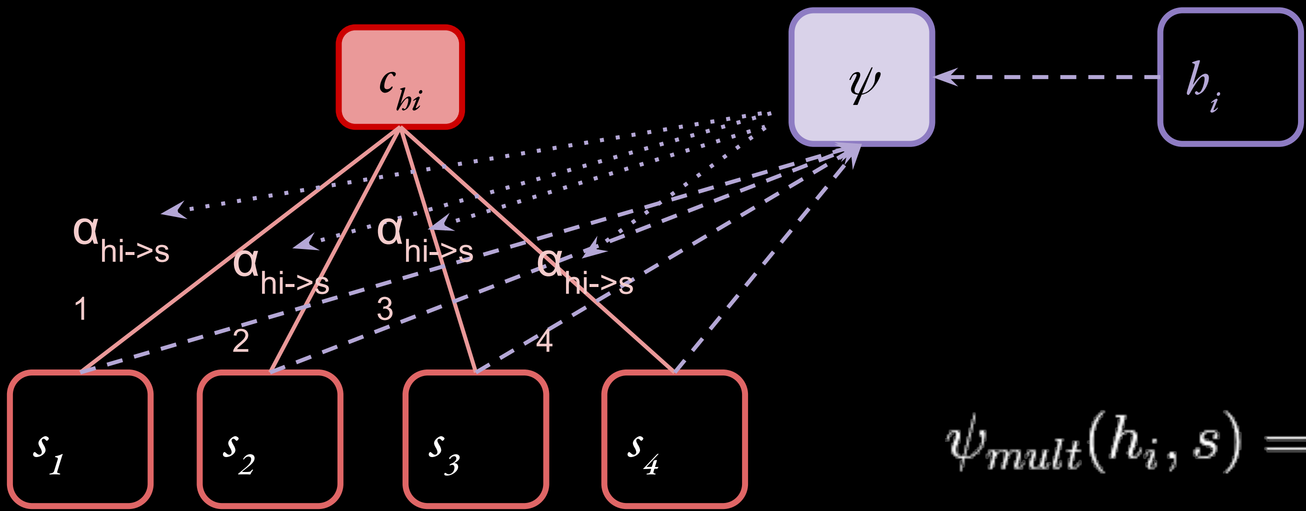

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} s_n$$

# Attention




$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \rightarrow s_n} s_n$$

# Attention

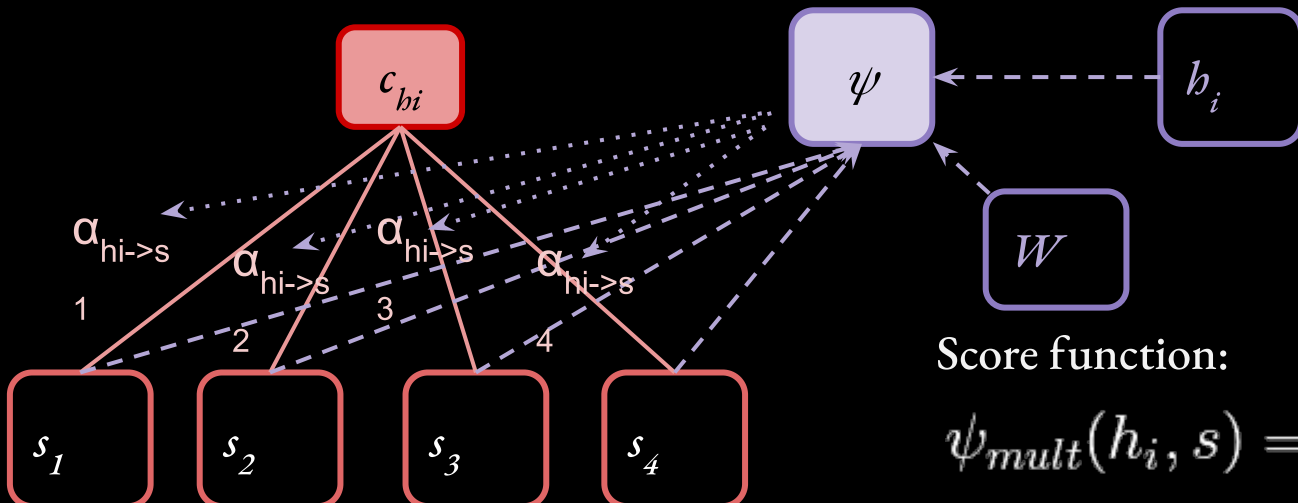



$$\psi_{mult}(h_i, s) = s^T W h_i$$

$$\alpha_{h_i \to s} = \text{softmax}(\psi(h_i, s))$$

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} s_n$$

# Attention



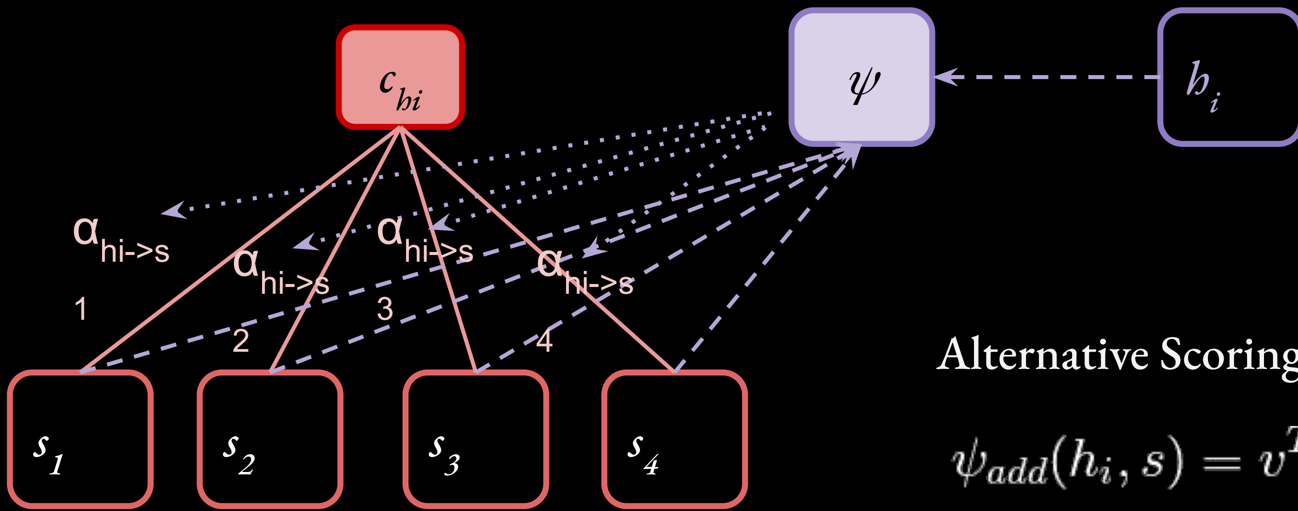

Score function:

$$\psi_{mult}(h_i, s) = s^T W h_i$$

$$\alpha_{h_i \to s} = \text{softmax}(\psi(h_i, s))$$

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} s_n$$

# Attention



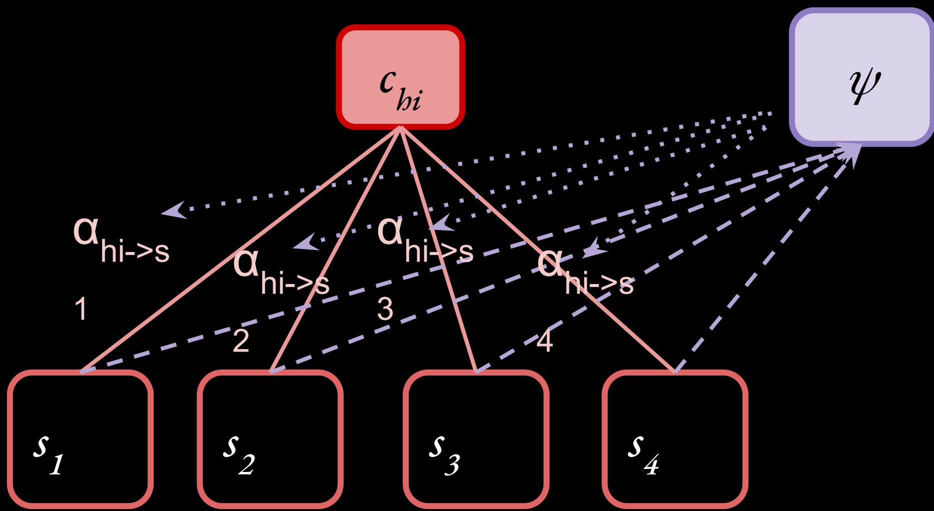

Alternative Scoring Functions

$$\psi_{add}(h_i, s) = v^T \tanh(W_h h_i + W_s s])$$

$$\psi_{dp}(h_i, s) = s^T h_i$$

$$\psi_{mult}(h_i, s) = s^T W h_i$$

# Attention




If variables are standardized, matrix multiply produces a similarity score.

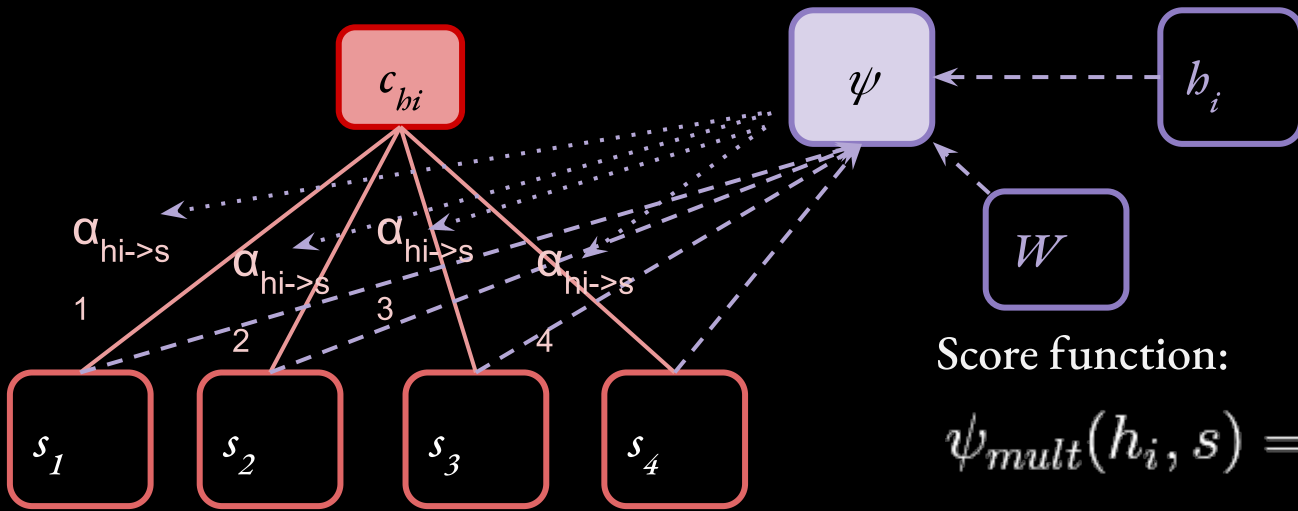
Alternative Scoring Functions

$$\psi_{add}(h_i, s) = v^T \tanh(W_h h_i + W_s s])$$

$$\psi_{dp}(h_i, s) = s^T h_i$$

$$\psi_{mult}(h_i, s) = s^T W h_i$$

# Attention



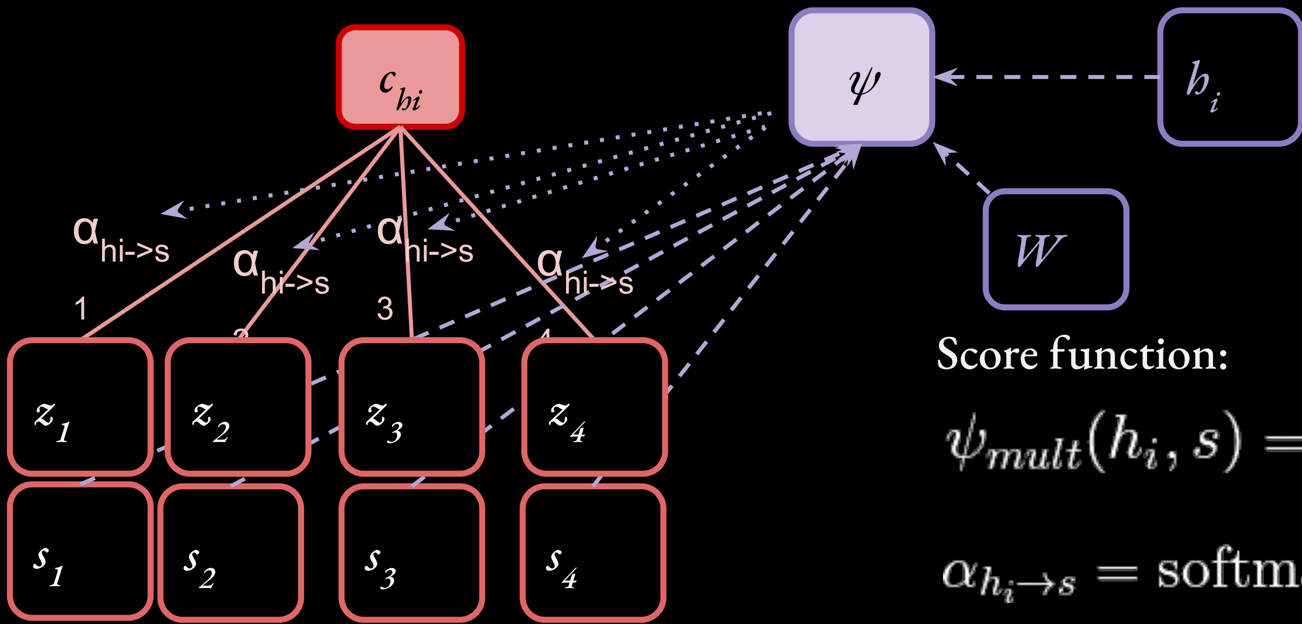

Score function:

$$\psi_{mult}(h_i, s) = s^T W h_i$$

$$\alpha_{h_i \to s} = \text{softmax}(\psi(h_i, s))$$

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} s_n$$

# Attention




Score function:

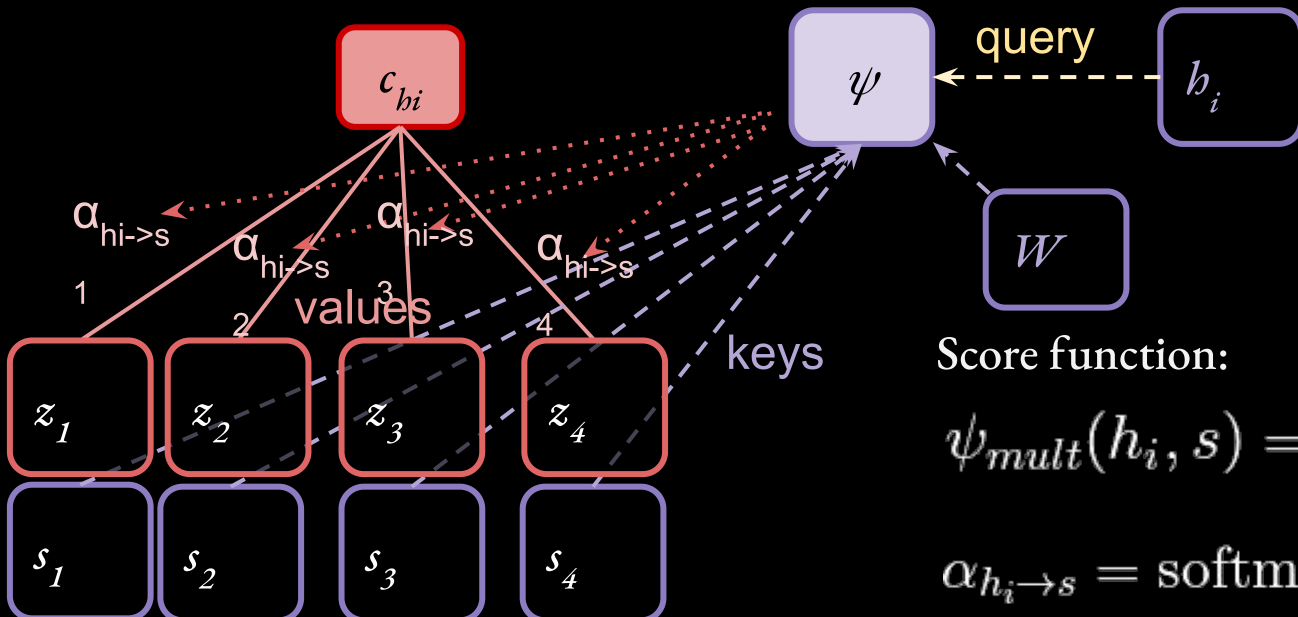
$$\psi_{mult}(h_i, s) = s^T W h_i$$

$$\alpha_{h_i \to s} = \text{softmax}(\psi(h_i, s))$$

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} z_n$$

A useful abstraction is to make the vector attended to (the "value vector", $Z$) separate than the "key vector" (s).

# Attention




Score function:

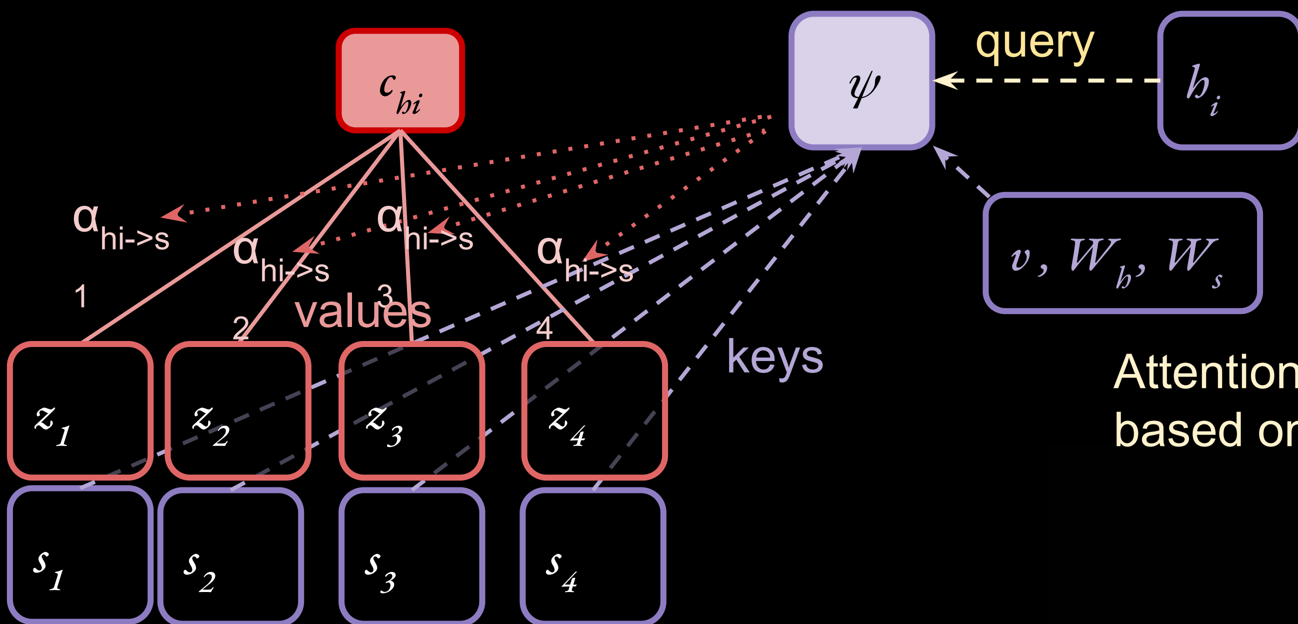
$$\psi_{mult}(h_i, s) = s^T W h_i$$

$$\alpha_{h_i \to s} = \text{softmax}(\psi(h_i, s))$$

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} z_n$$

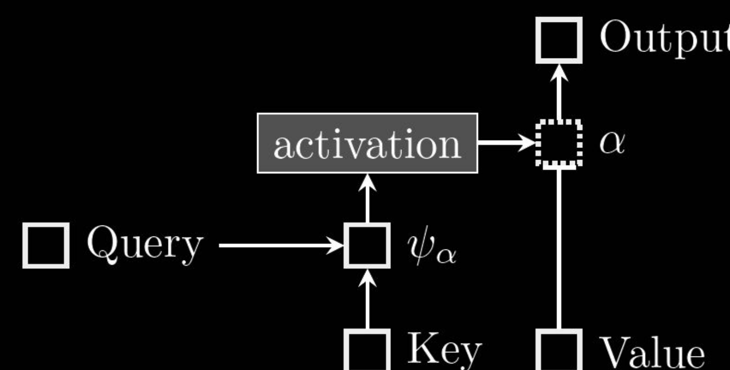
A useful abstraction is to make the vector attended to (the "value vector", *Z*) separate than the "key vector" (s).

# Attention




(Eisenstein, 2018)

Attention as weighting a value based on a query and key:



# Machine Translation

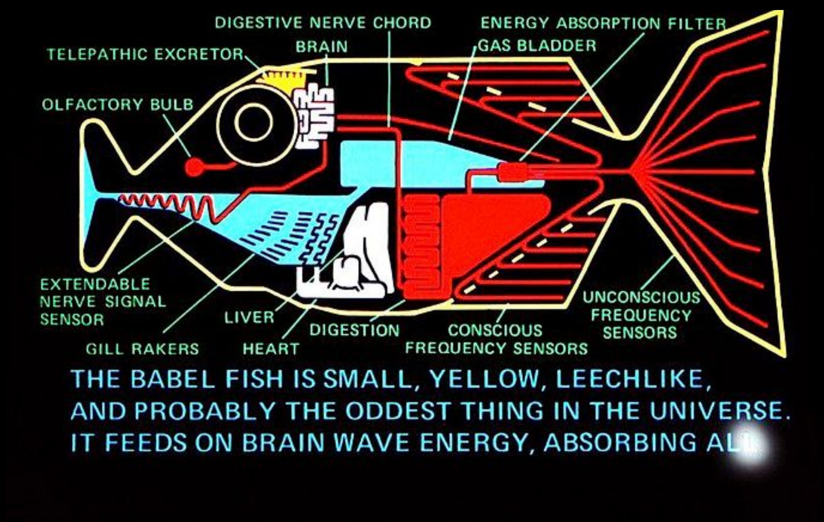
As an optimization problem (Eisenstein, 2018):

$$\hat{\boldsymbol{w}}^{(t)} = \underset{\boldsymbol{w}^{(t)}}{\operatorname{argmax}} \, \Psi(\boldsymbol{w}^{(s)}, \boldsymbol{w}^{(t)})$$

# Machine Translation

## Why?

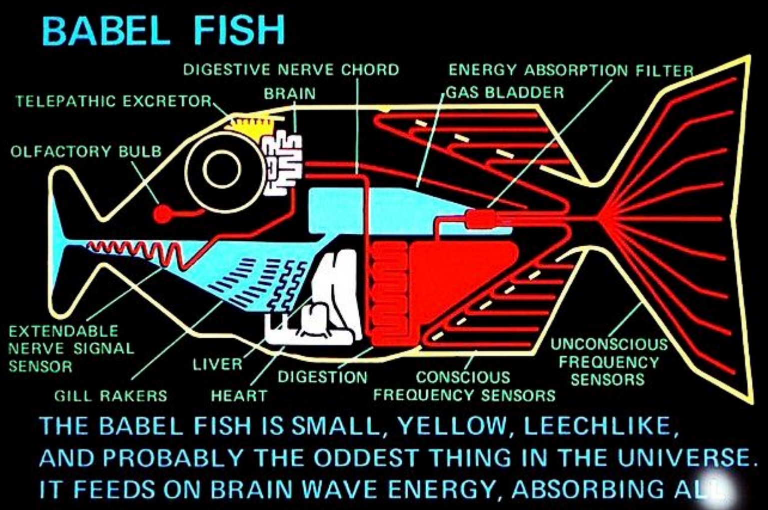
- $40billion/year industry
- A center piece of many genres of science fiction
- A fairly "universal" problem:
  - Language understanding
  - Language generation
- Societal benefits of inter-cultural communication



# Machine Translation

Why?

- $40billion/year industry
- A center piece of many genres of science fiction
- A fairly “universal” problem:
  - Language understanding
  - Language generation
- Societal benefits of inter-cultural communication



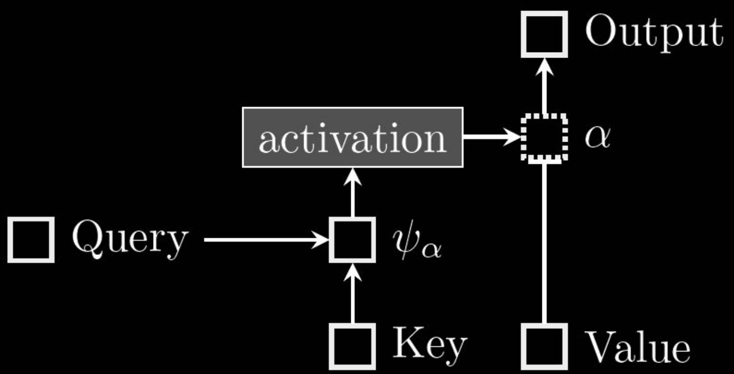

(Douglas Adams)

# Machine Translation

Why Neural Network Approach works? (Manning, 2018)

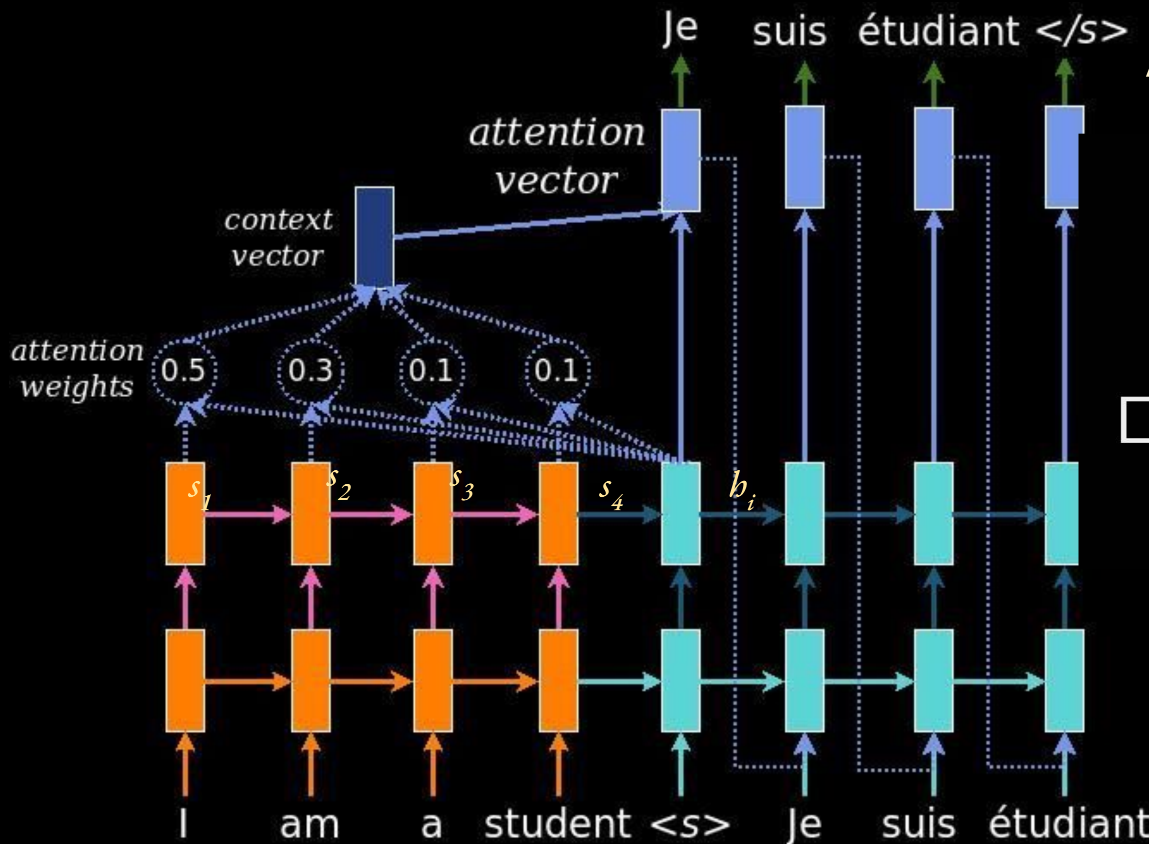
- Joint end-to-end training: learning all parameters at once.
- Exploiting distributed representations (embeddings)
- Exploiting variable-length context
- High quality generation from deep decoders - stronger language models (even when wrong, make sense)

Attention as weighting a value based on a query and key:

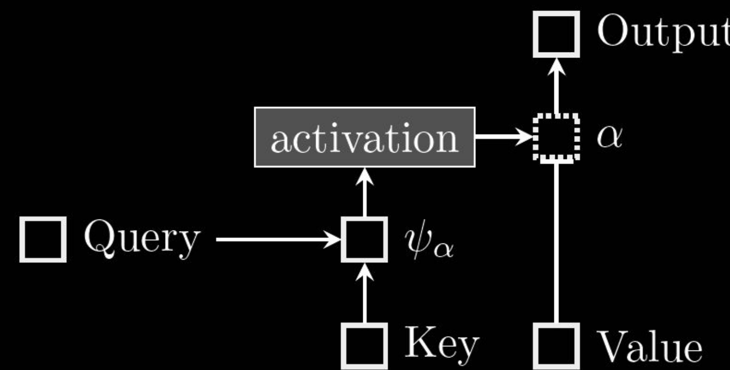



(Eisenstein, 2018)

# Attention

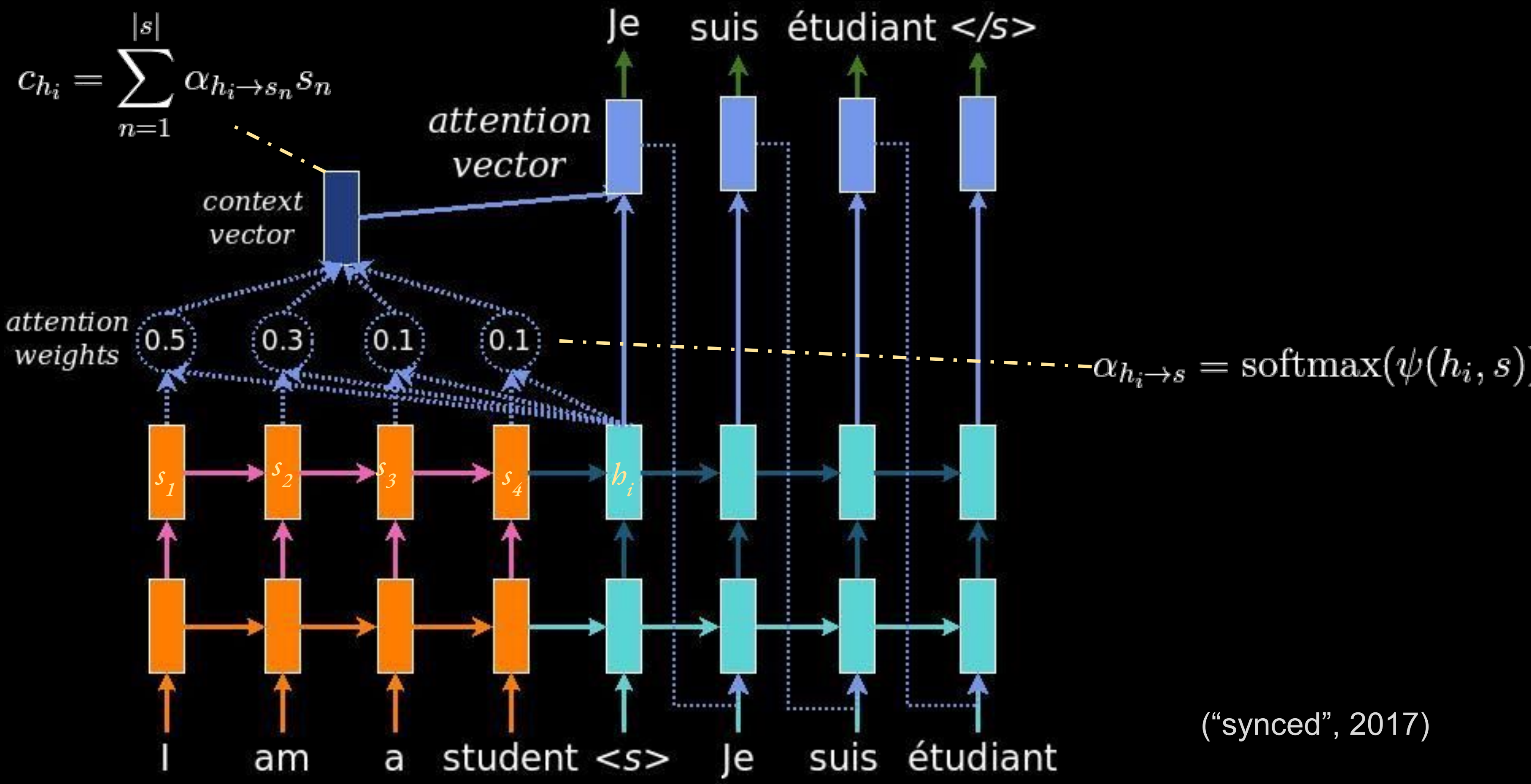



Attention as weighting a value based on a query and key:




(“synced”, 2017)

# Attention

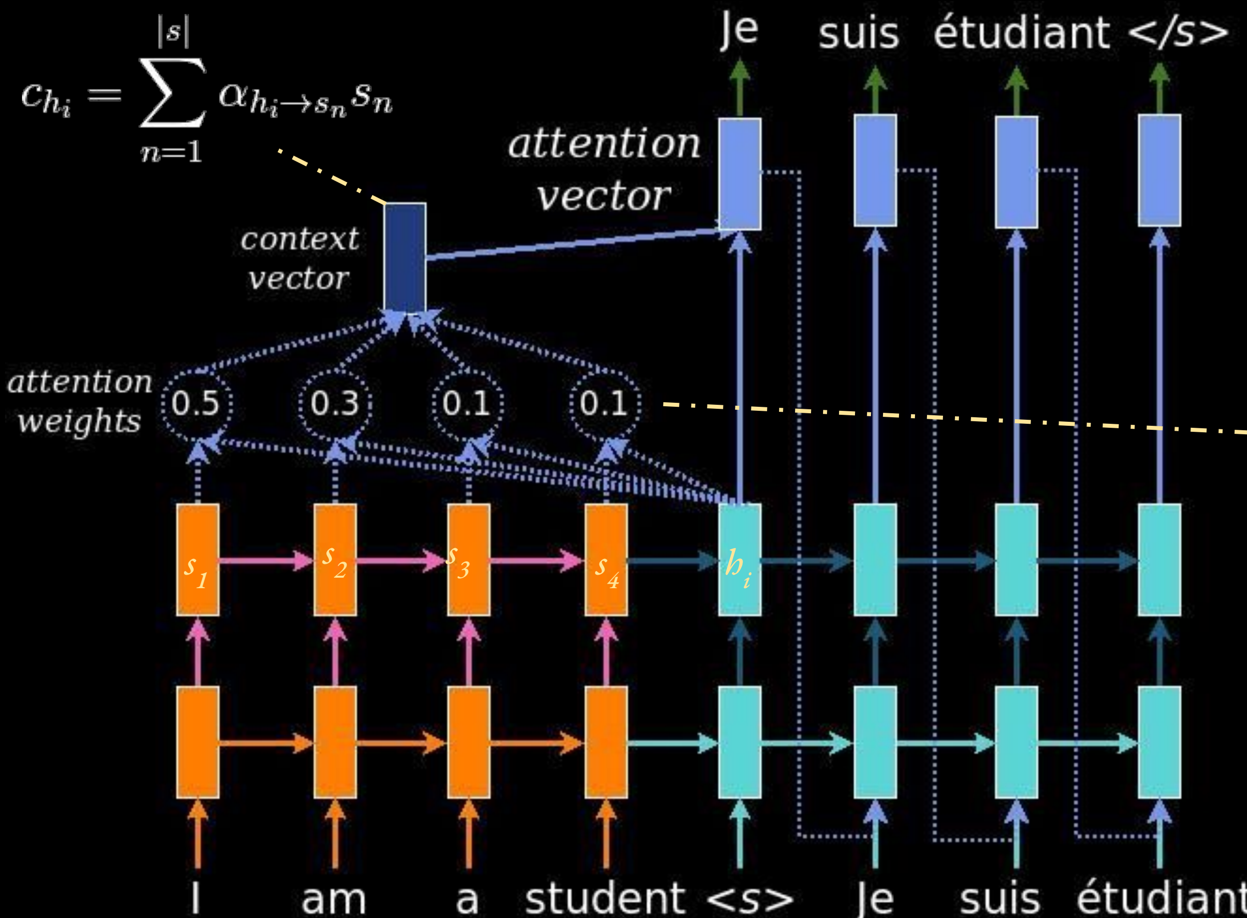



("synced", 2017)

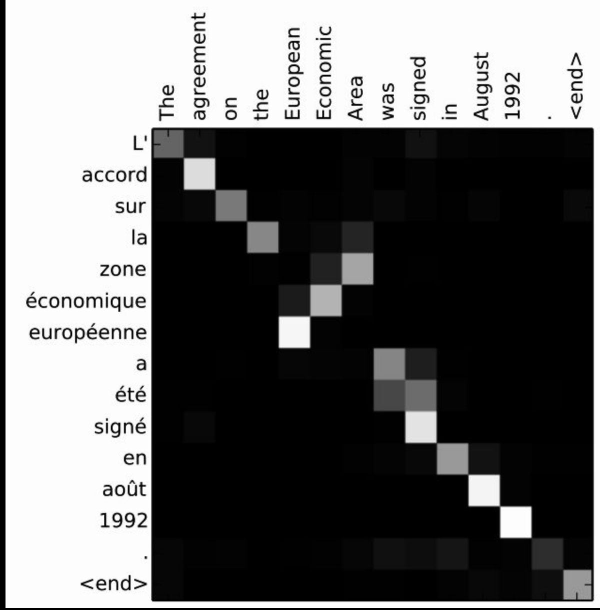
(Bahdanau et al., 2015)

# Attention

$$c_{h_i} = \sum_{n=1}^{|s|} \alpha_{h_i \to s_n} s_n$$



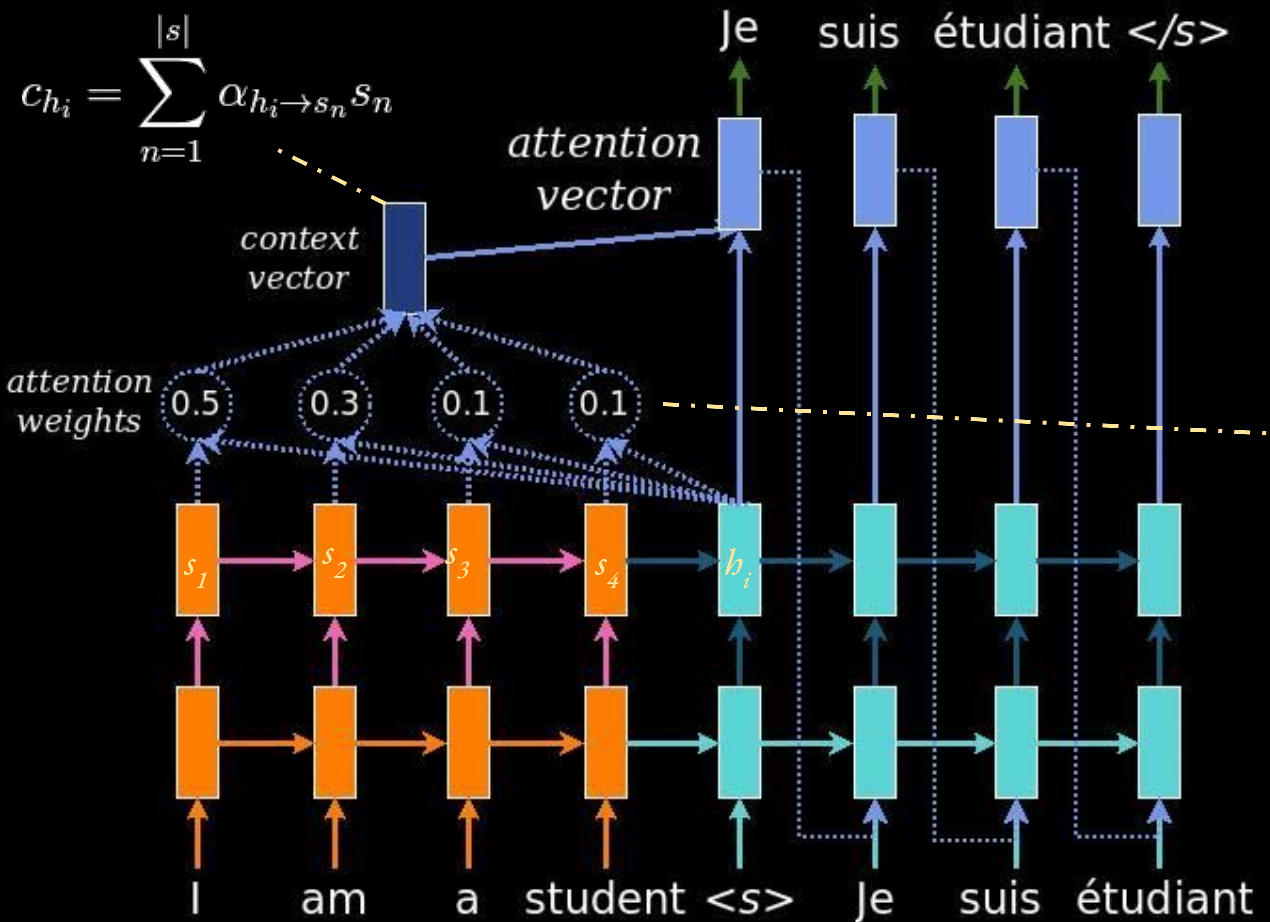

$$\alpha_{h_i \to s} = \text{softmax}(\psi(h_i, s))$$

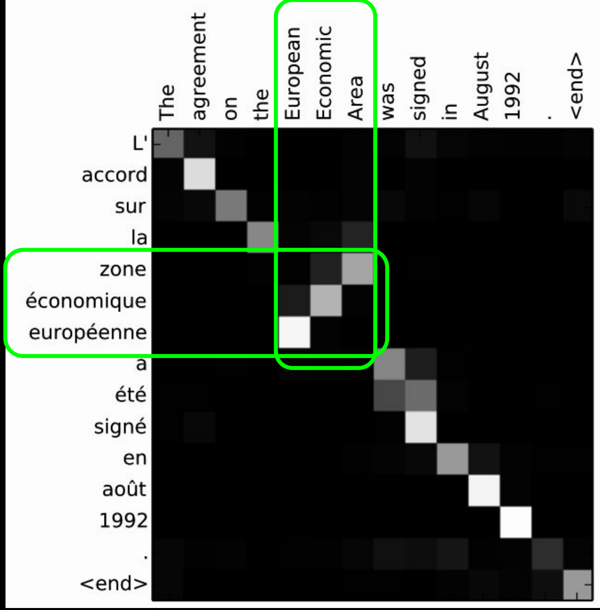



("synced", 2017)

# Attention

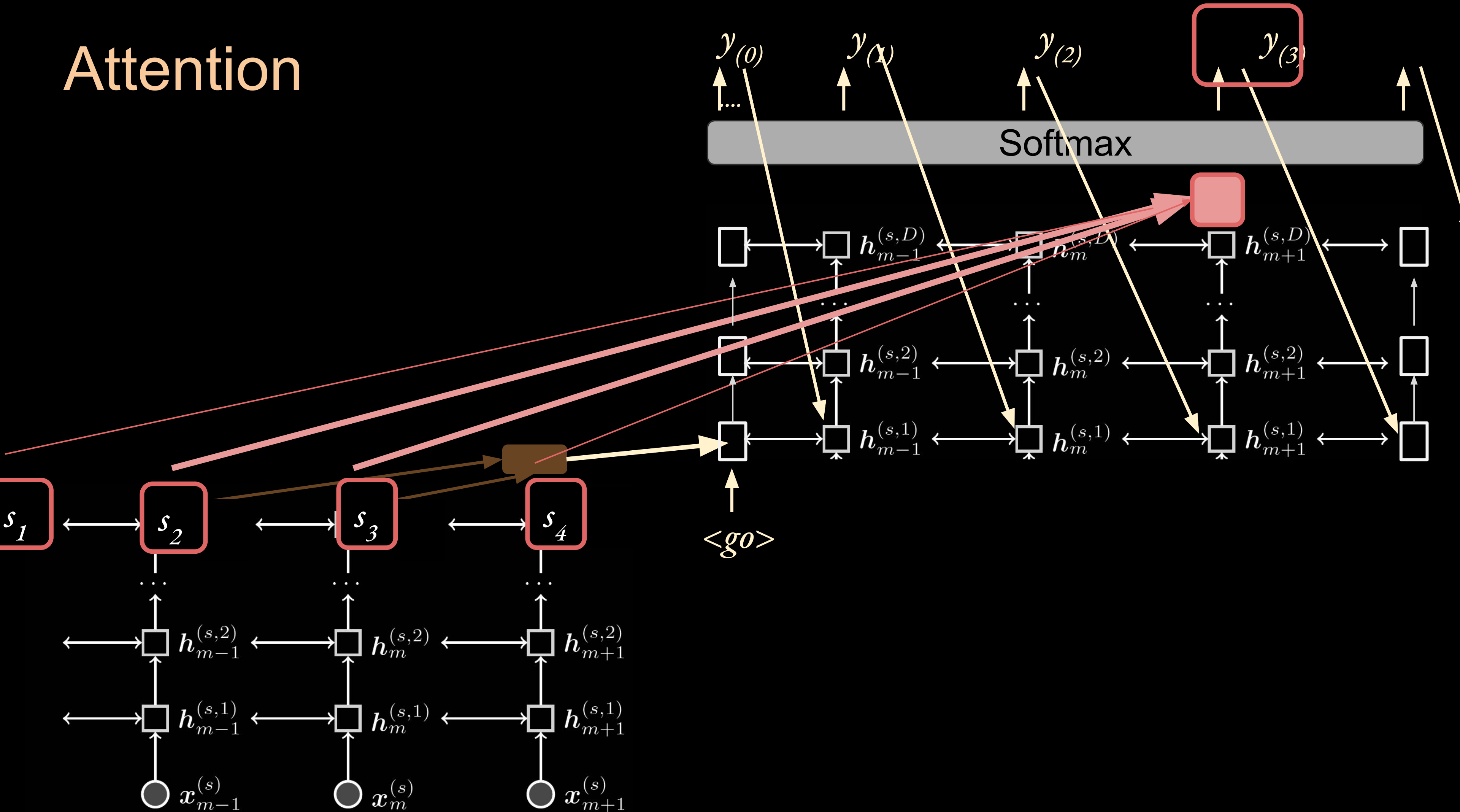
(Bahdanau et al., 2015)



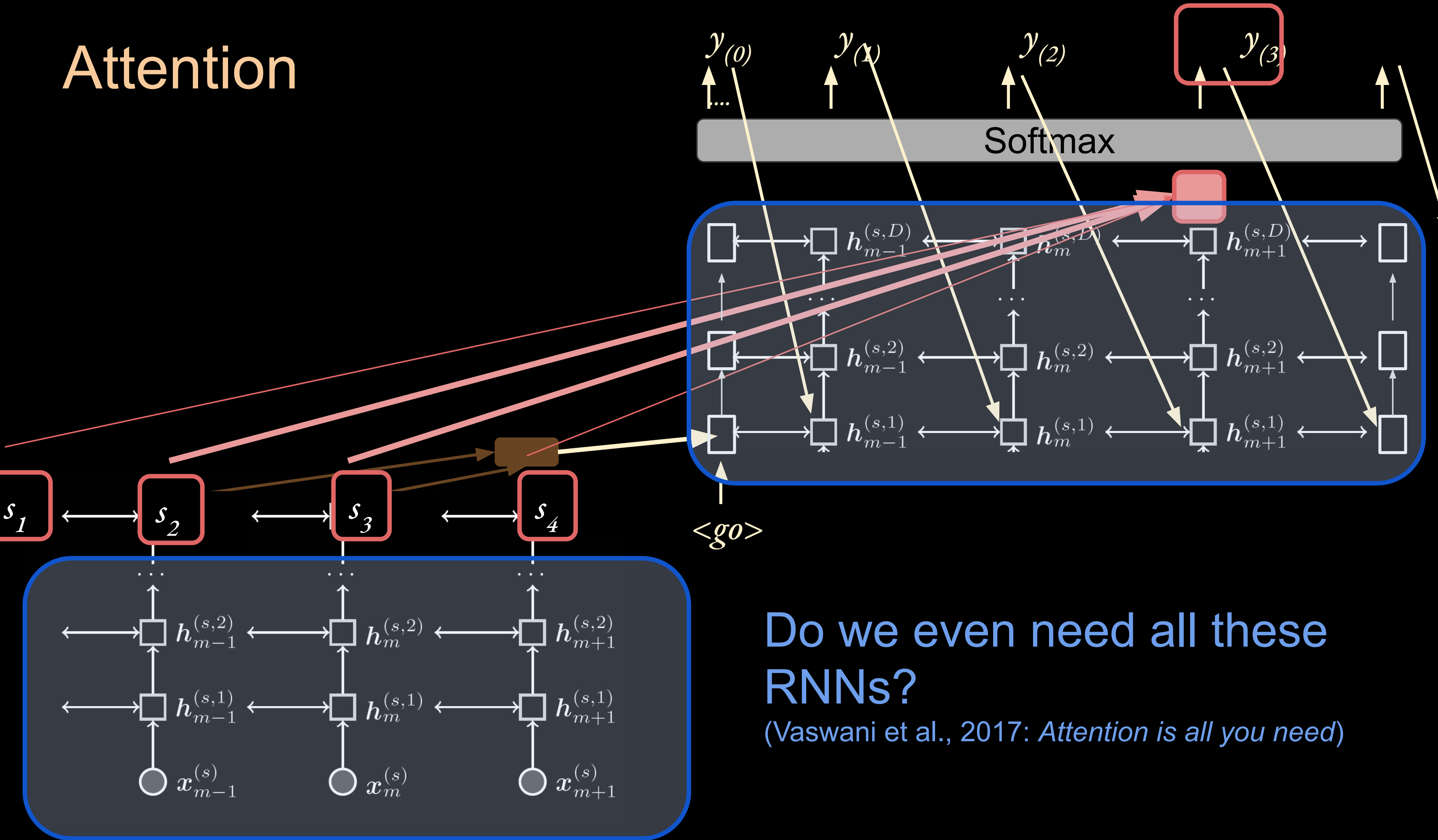

("synced", 2017)

# Attention



$y_{(0)}$
$y_{(1)}$
$y_{(2)}$
$y_{(3)}$
Softmax
$h_{m-1}^{(s,D)}$
$h_{m}^{(s,D)}$
$h_{m+1}^{(s,D)}$
$h_{m-1}^{(s,2)}$
$h_{m}^{(s,2)}$
$h_{m+1}^{(s,2)}$
$h_{m-1}^{(s,1)}$
$h_{m}^{(s,1)}$
$h_{m+1}^{(s,1)}$
<go>
$s_1$
$s_2$
$s_3$
$s_4$
$h_{m-1}^{(s,2)}$
$h_{m}^{(s,2)}$
$h_{m+1}^{(s,2)}$
$h_{m-1}^{(s,1)}$
$h_{m}^{(s,1)}$
$h_{m+1}^{(s,1)}$
$x_{m-1}^{(s)}$
$x_{m}^{(s)}$
$x_{m+1}^{(s)}$

# Attention




Do we even need all these RNNs?

(Vaswani et al., 2017: *Attention is all you need*)

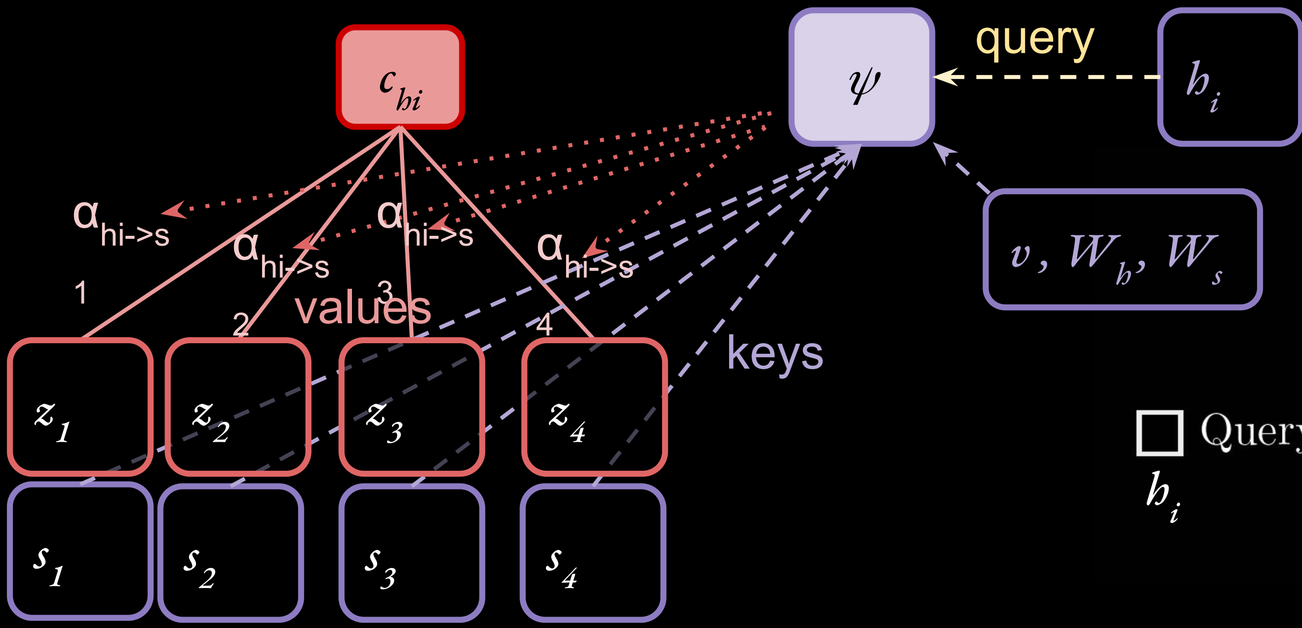
# Evolution of Sequence Modeling

RNNs LSTMs <u>LSTMS with Attention</u> Attention without LSTMs

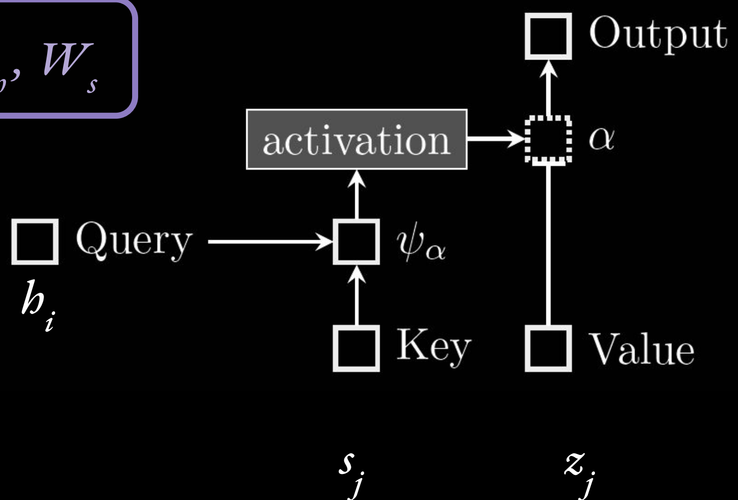
# Evolution of Sequence Modeling

RNNs LSTMs LSTMS with Attention <u>Attention without LSTMs</u>

# Attention



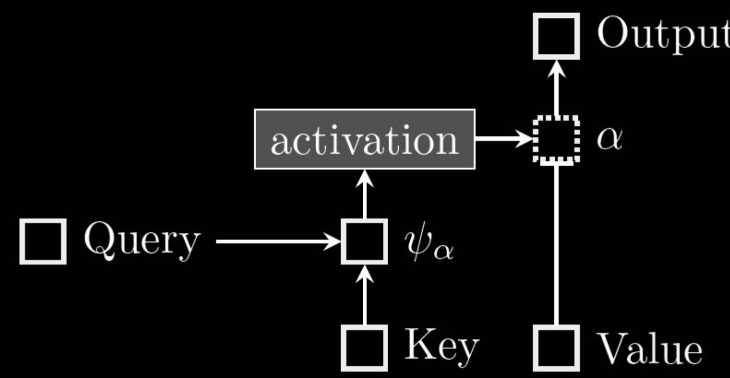

A useful abstraction is to make the vector attended to (the "value vector", $Z$) separate than the "key vector" (s).




(Eisenstein, 2018)

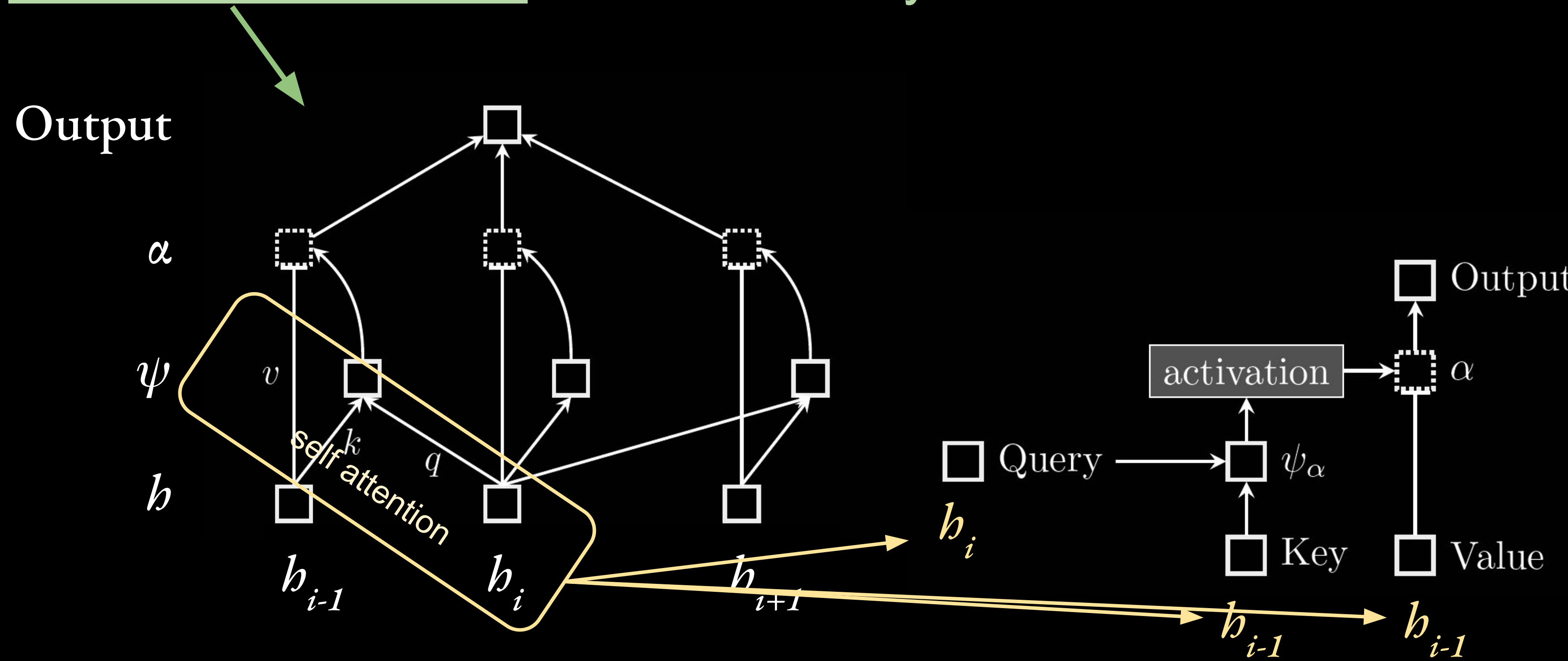
# **The Transformer:** “Attention-only” models

Attention as weighting a value based on a query and key:



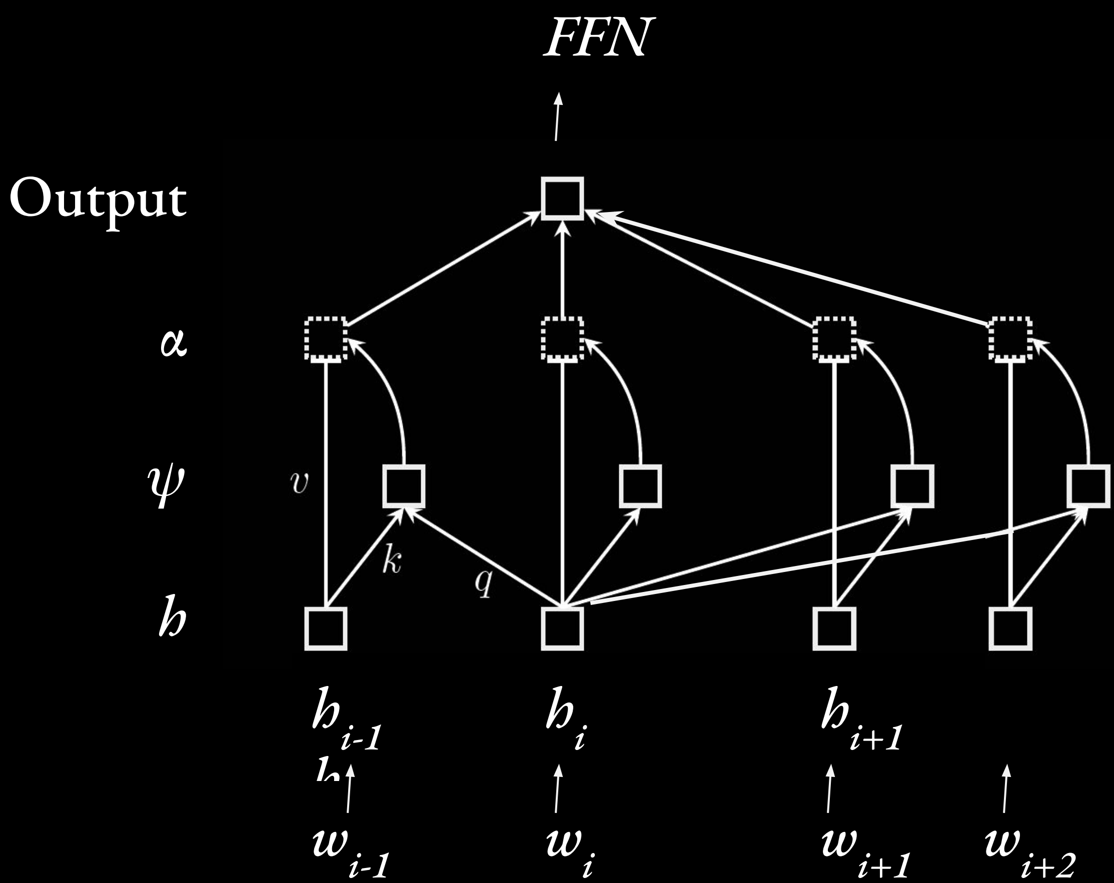

(Eisenstein, 2018)

# The Transformer: “Attention-only” models

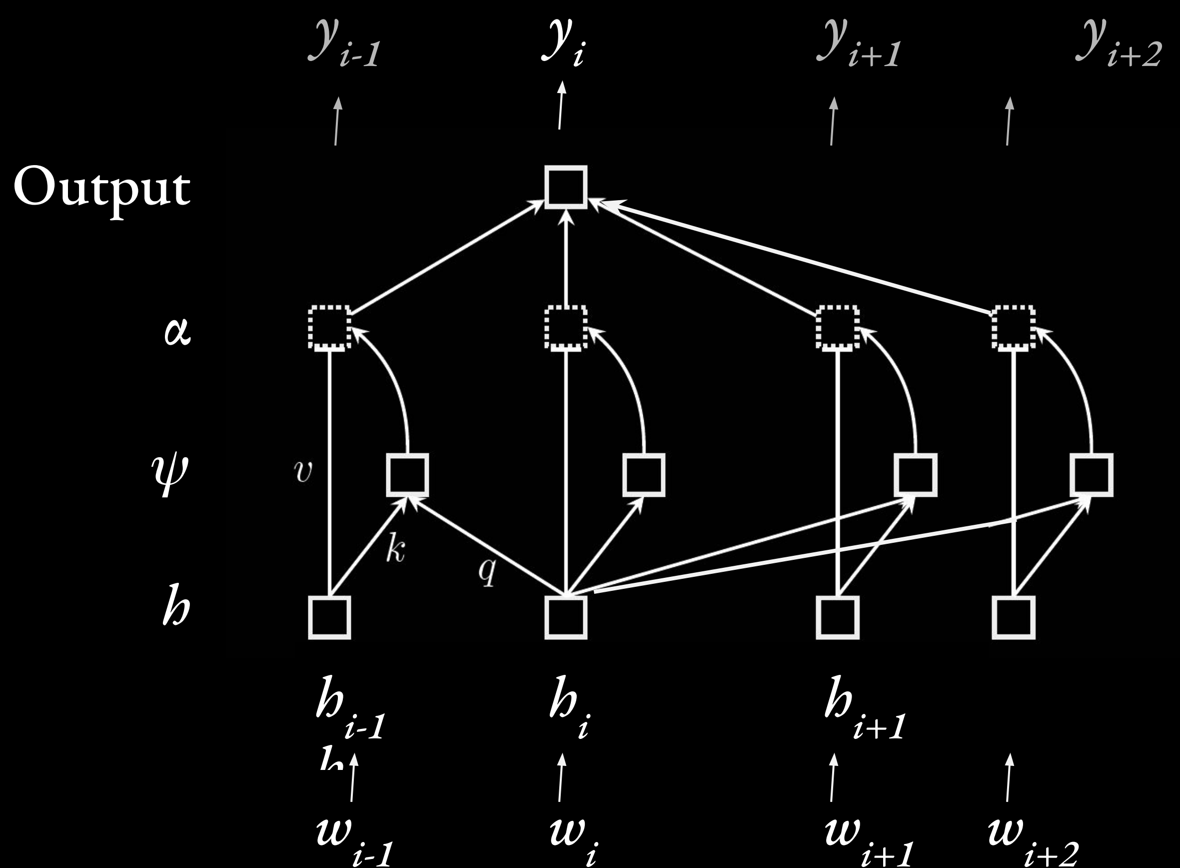



(Eisenstein, 2018)

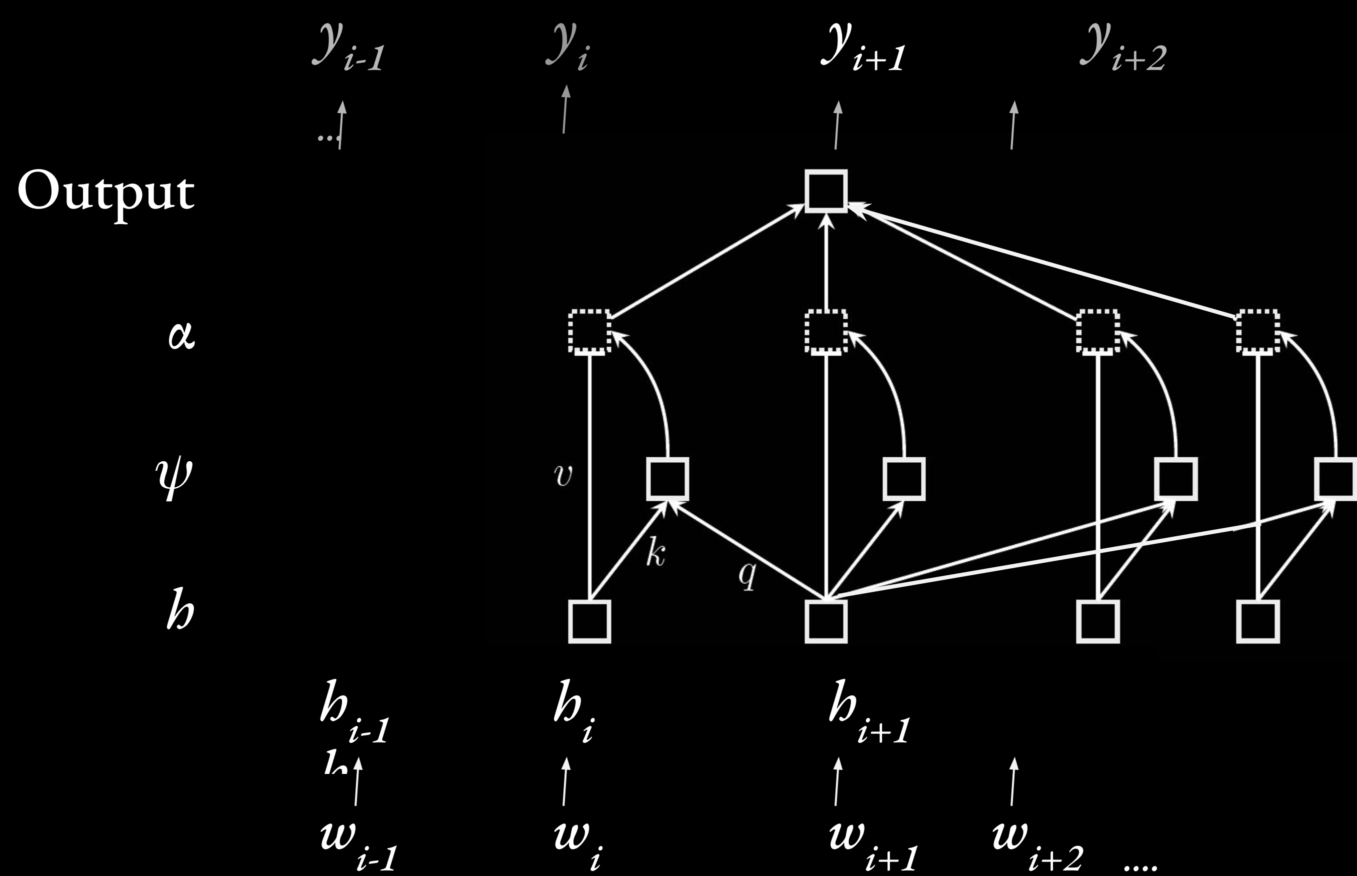
# **The Transformer:** “Attention-only” models



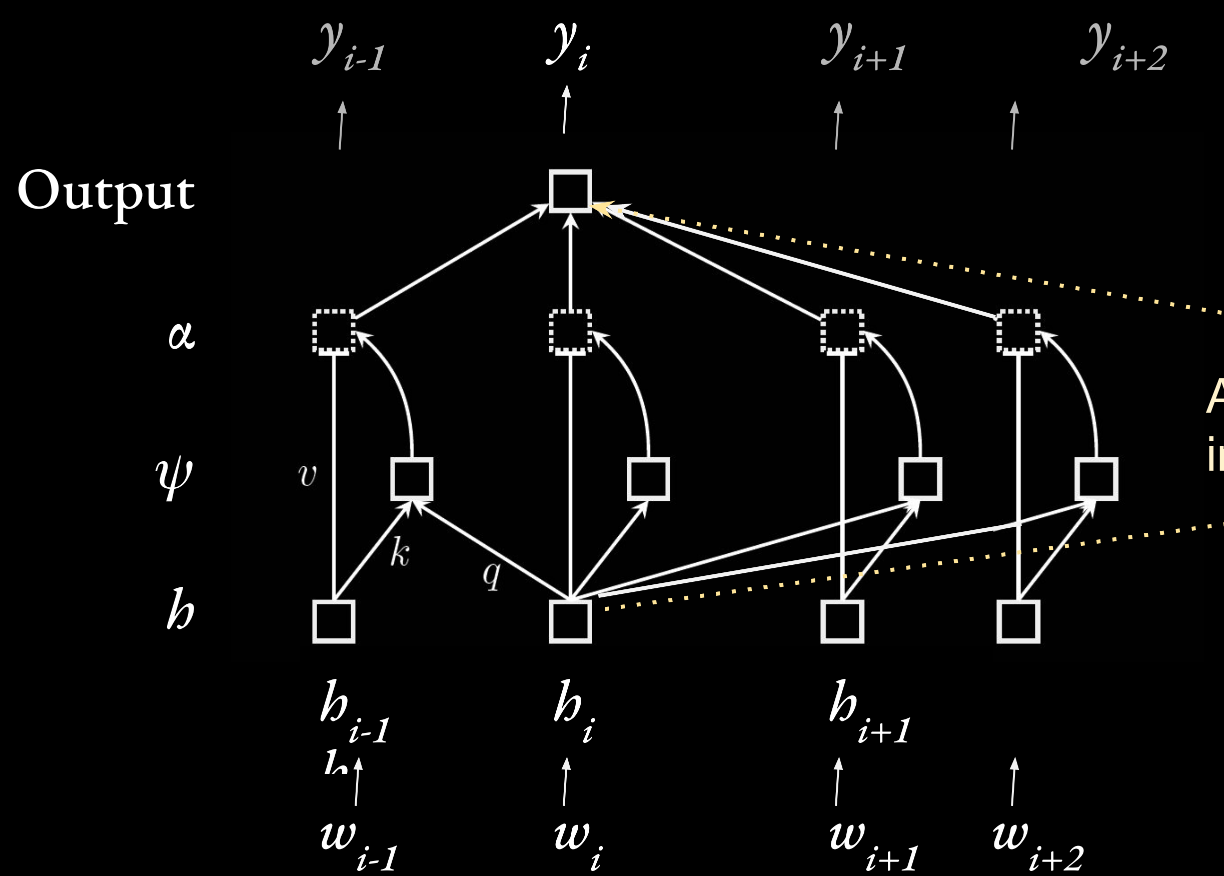
# **The Transformer:** "Attention-only" models



# **The Transformer:** "Attention-only" models

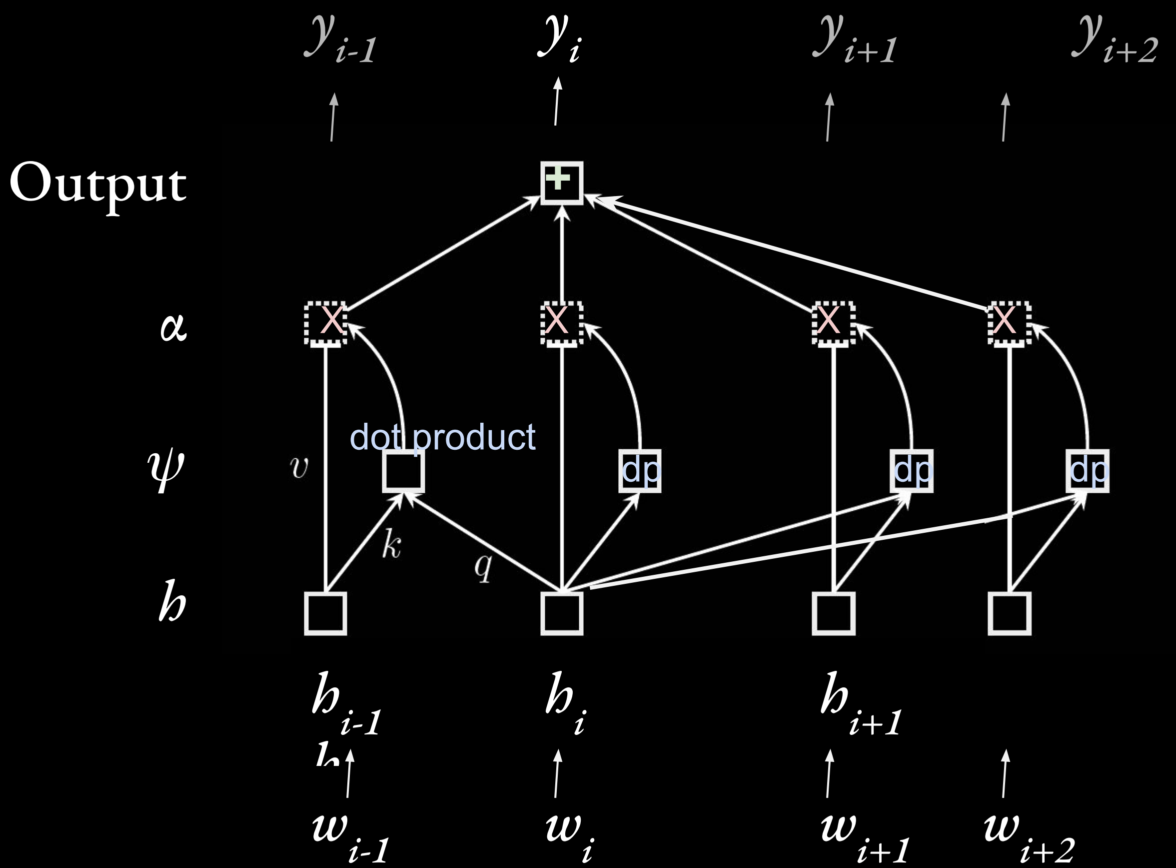


# **The Transformer:** “Attention-only” models




Attend to all hidden states in your “neighborhood”.

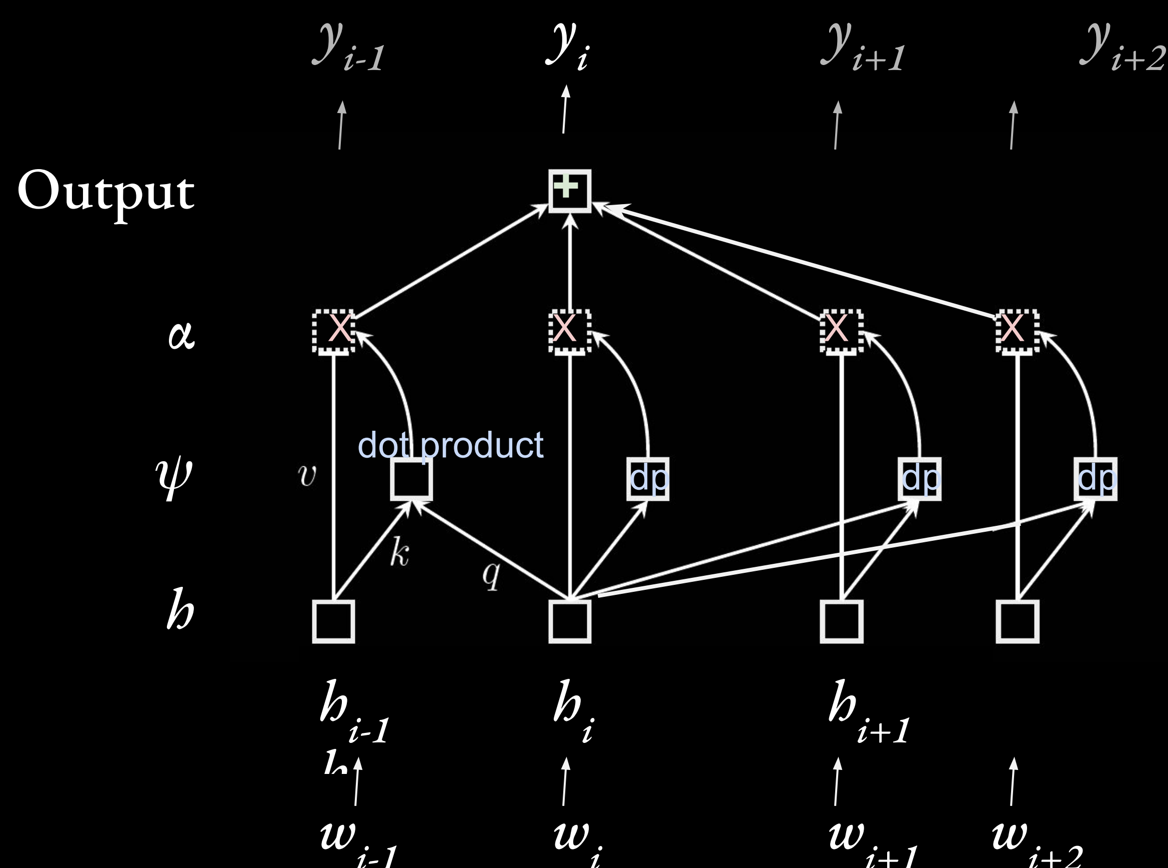
# **The Transformer:** "Attention-only" models




$$\psi_{dp}(h_i, s) = s^T h_i$$

$k^t q$

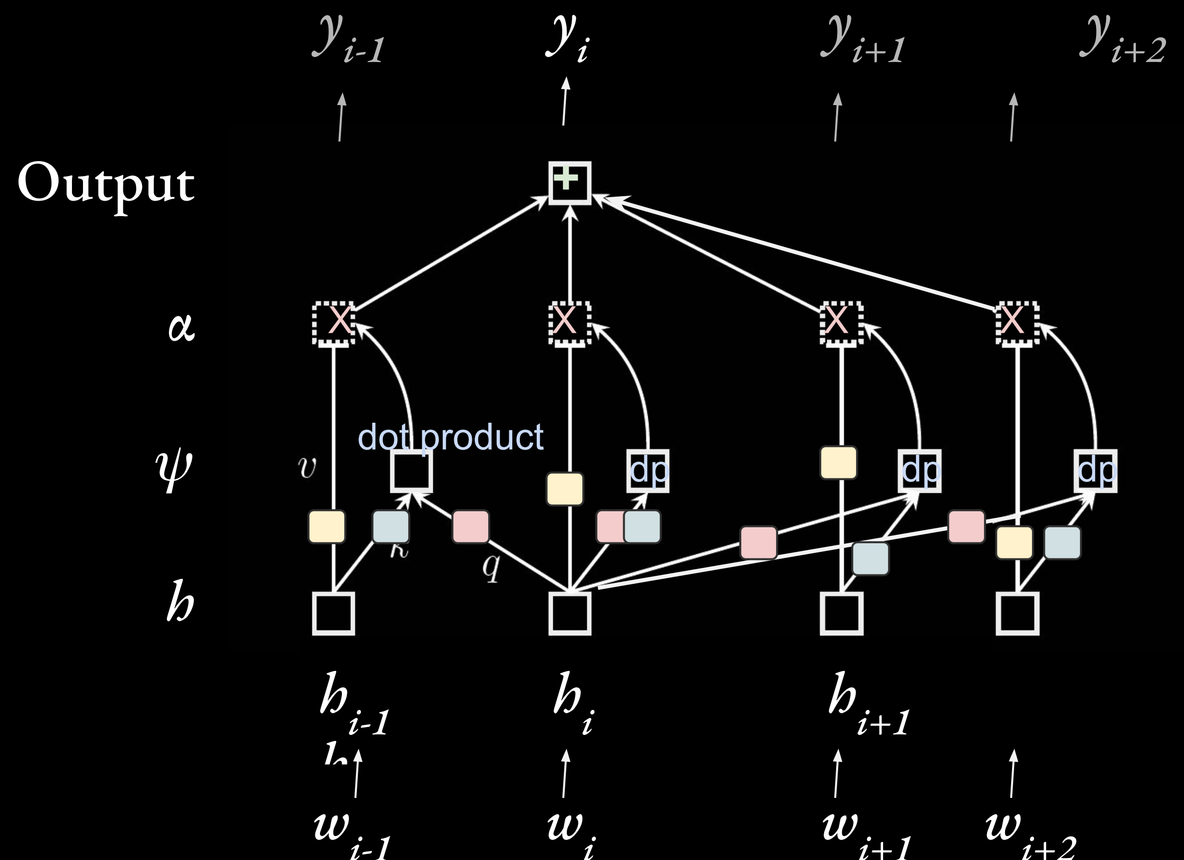
# **The Transformer:** "Attention-only" models




scaling parameter

$$\psi_{dp}(k,q) = (k^t q)\,\sigma$$

# **The Transformer:** "Attention-only" models




$$\psi_{dp}\,(k,q) \;=\; (qk^t)\,\sigma$$

Linear layer weights:

$W_{(k)}{}^T X$

$W_{(v)}{}^T X$

$W_{(q)}{}^T X$

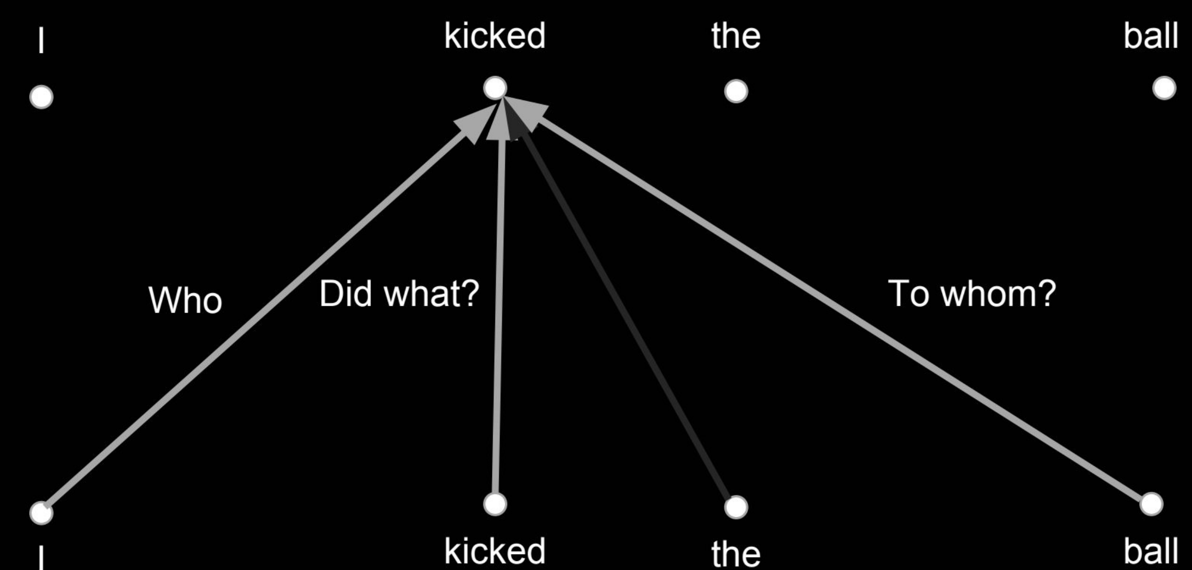
# **The Transformer:** “Attention-only” models

**Why?**

- Don’t need complexity of LSTM/GRU cells
- Constant num edges between words (or input steps)
- Enables “interactions” (i.e. adaptations) between words
- Easy to parallelize -- don’t need sequential processing.

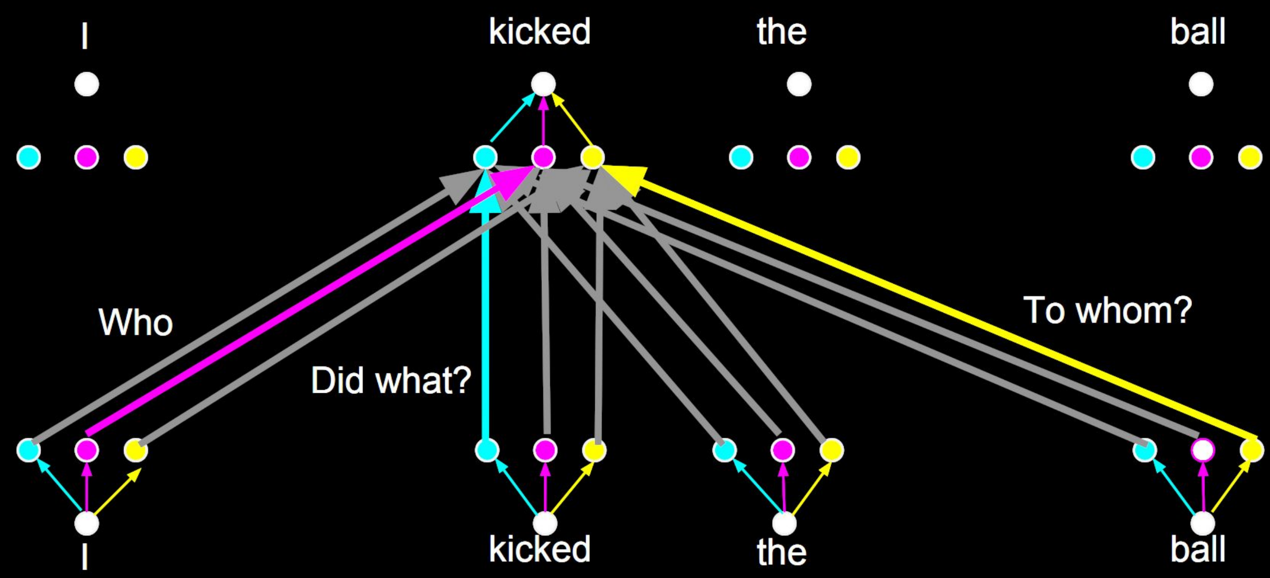
# The Transformer

Limitation (thus far): Can't capture multiple types of dependencies between words.

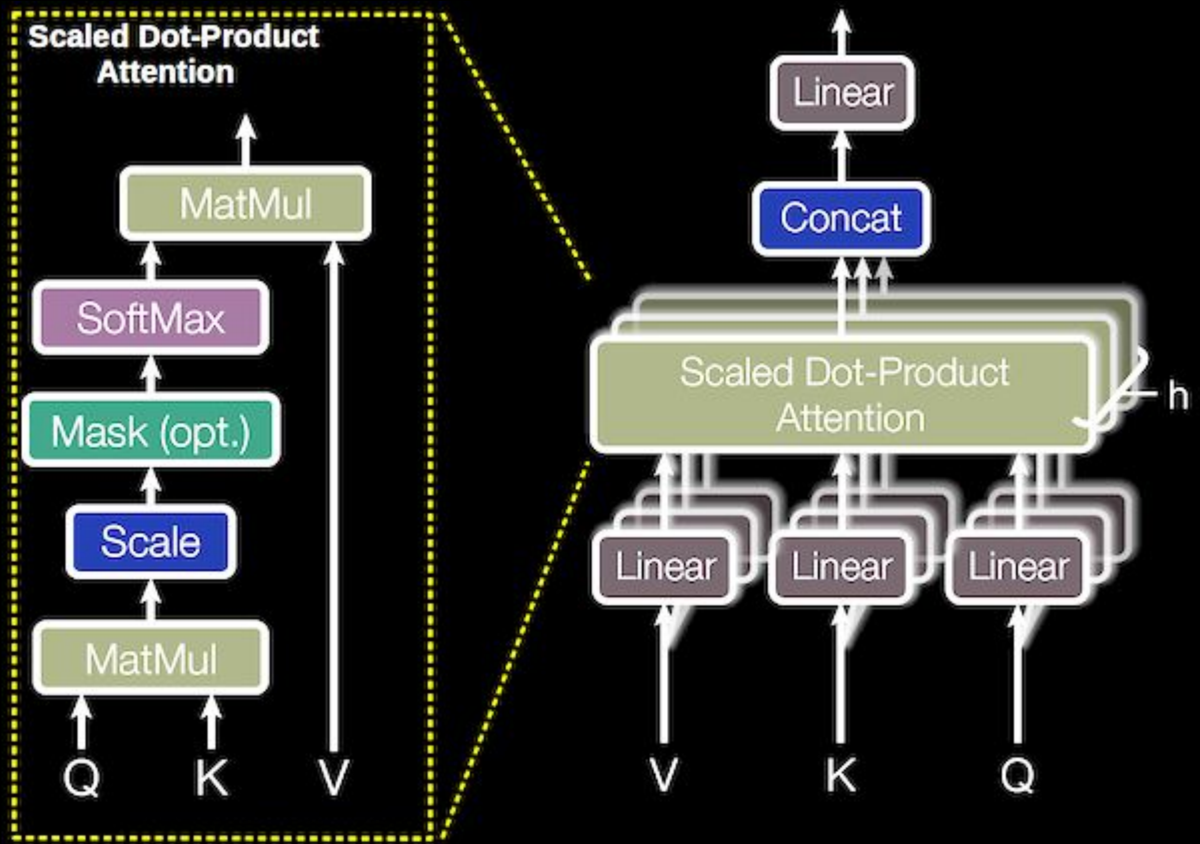


# The Transformer

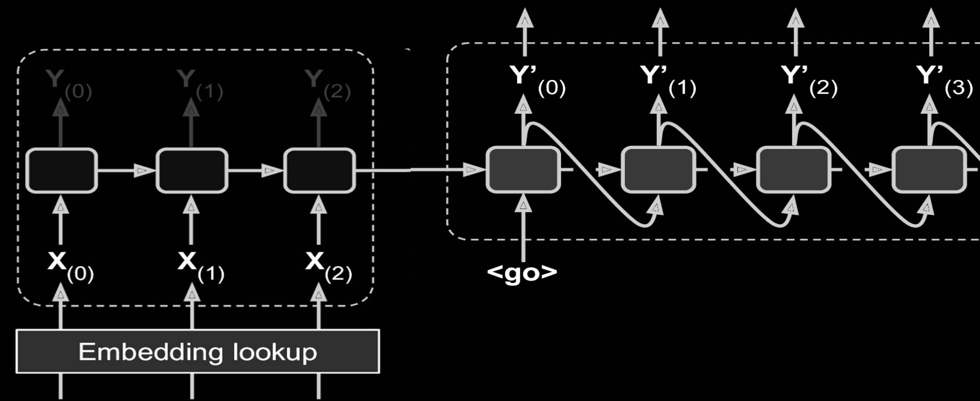
Solution: Multi-head attention

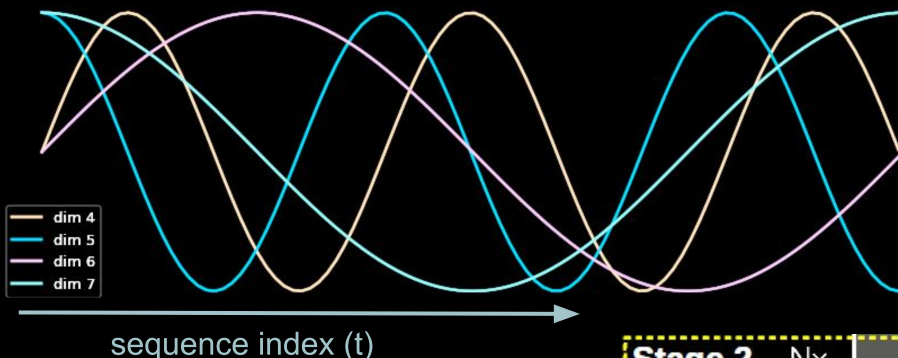


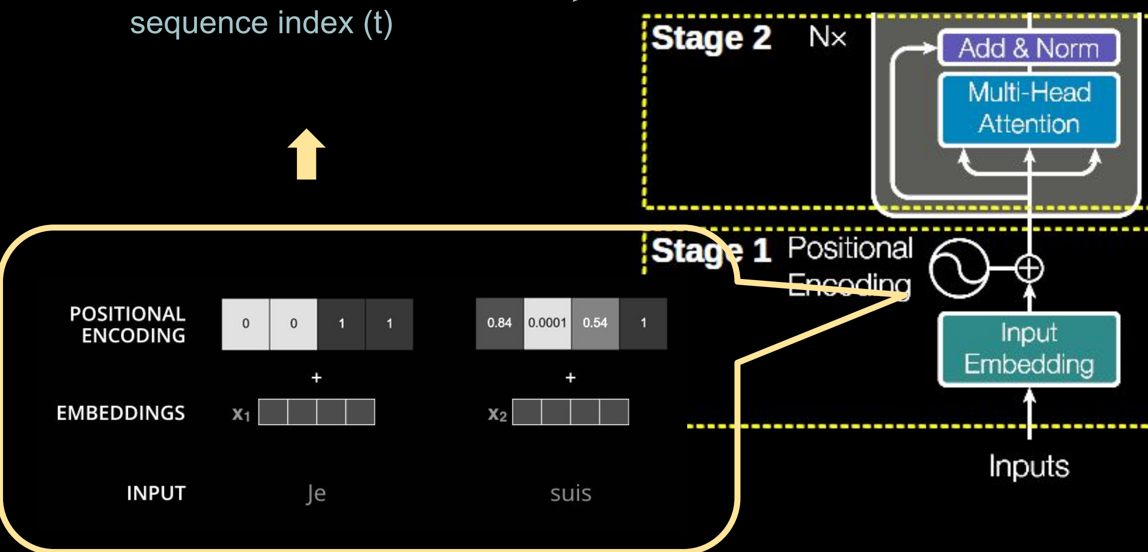
# Multi-head Attention



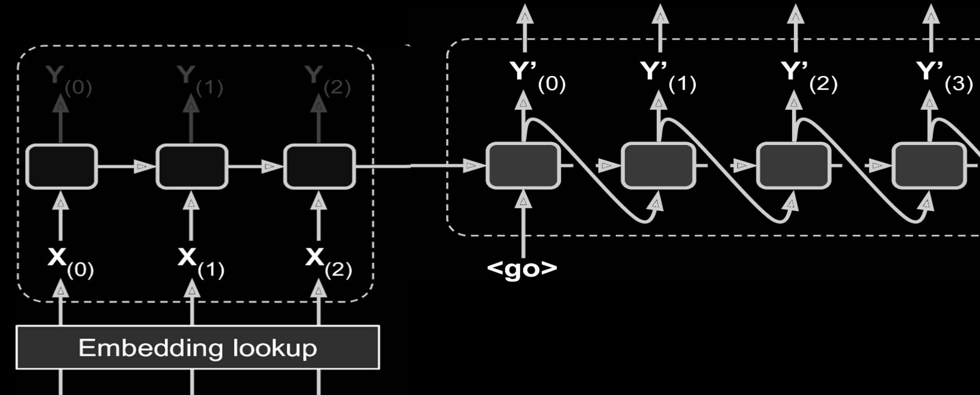
# Transformer for Encoder-Decoder

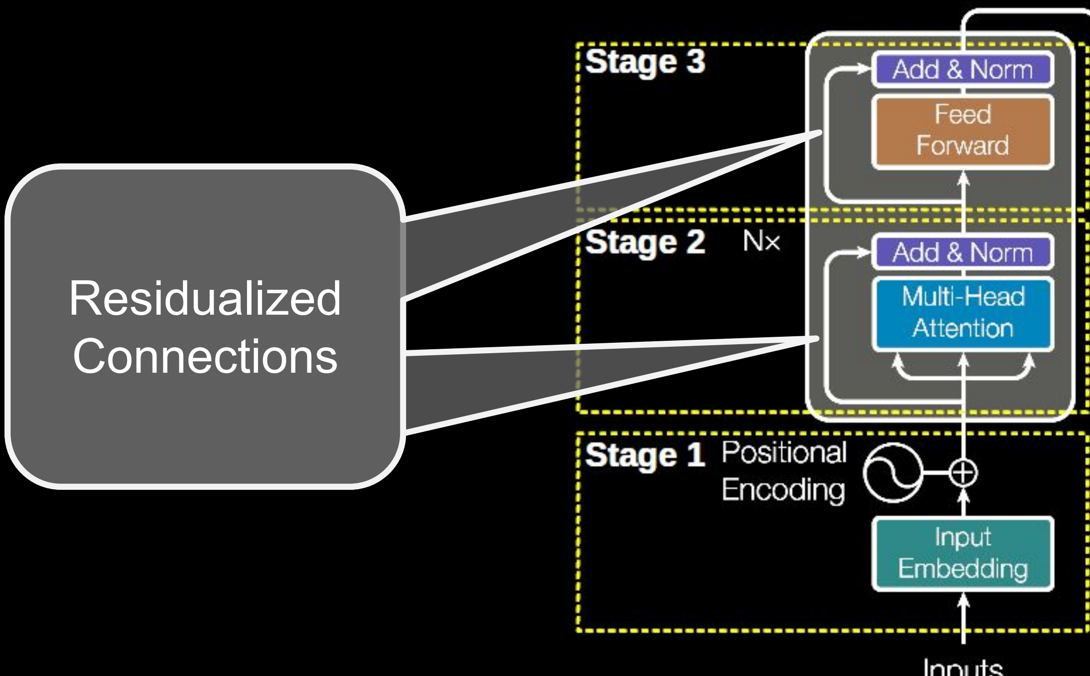


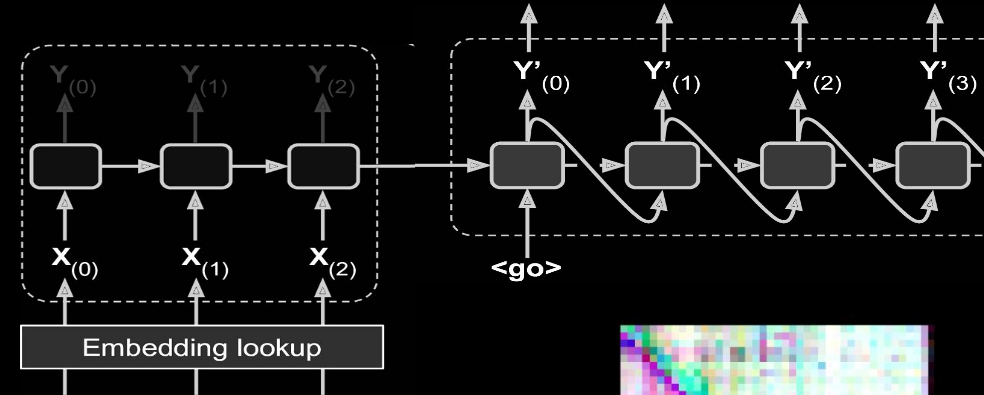



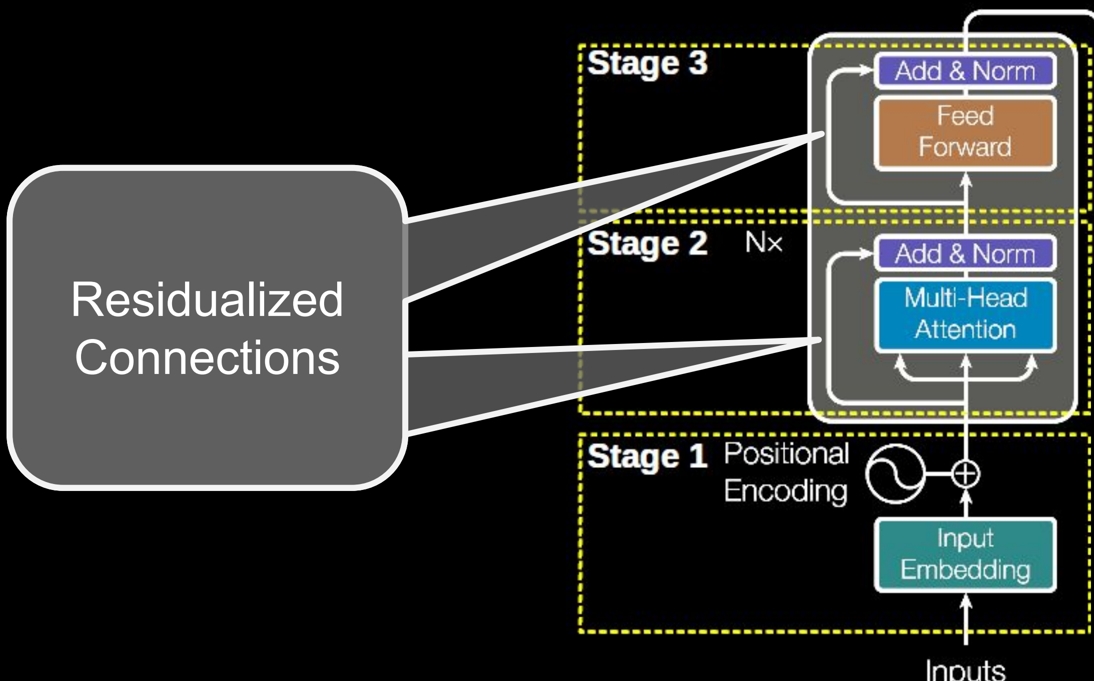
# Transformer for Encoder-Decoder



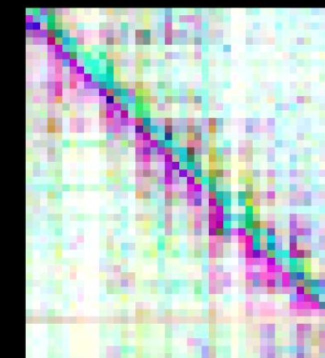



# Transformer for Encoder-Decoder

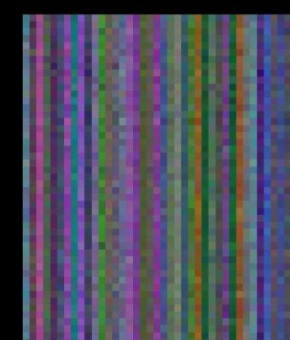



residuals enable positional information to be passed along

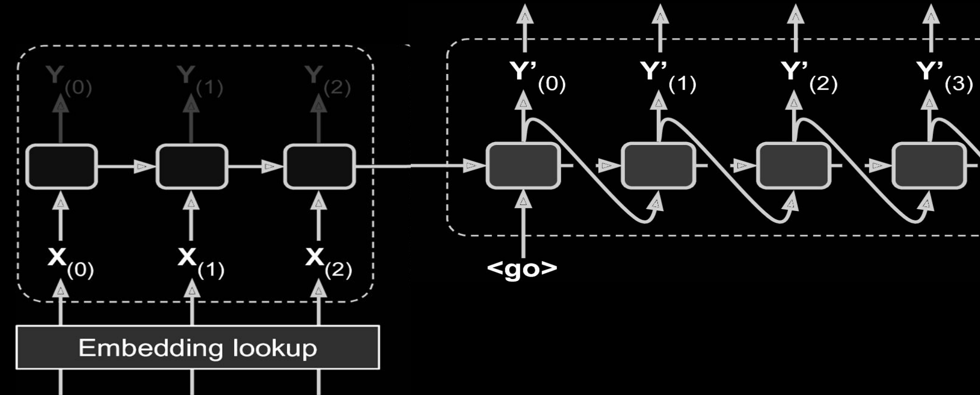


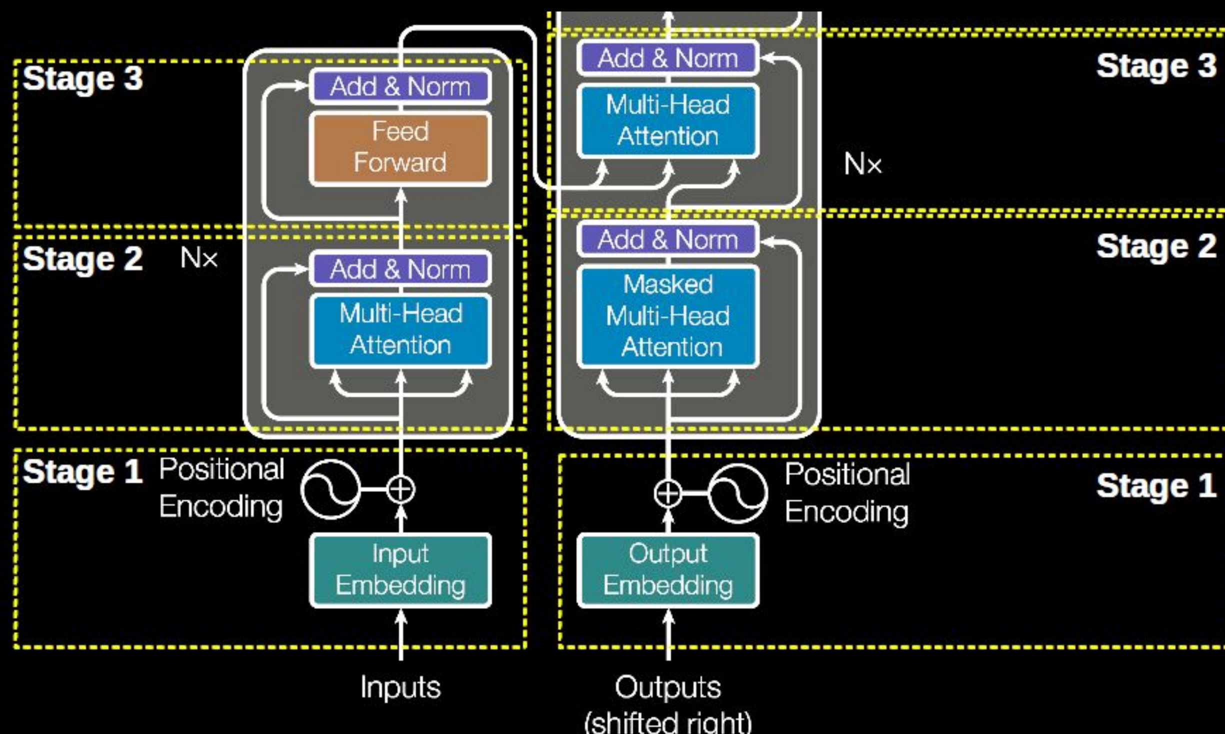
With residuals



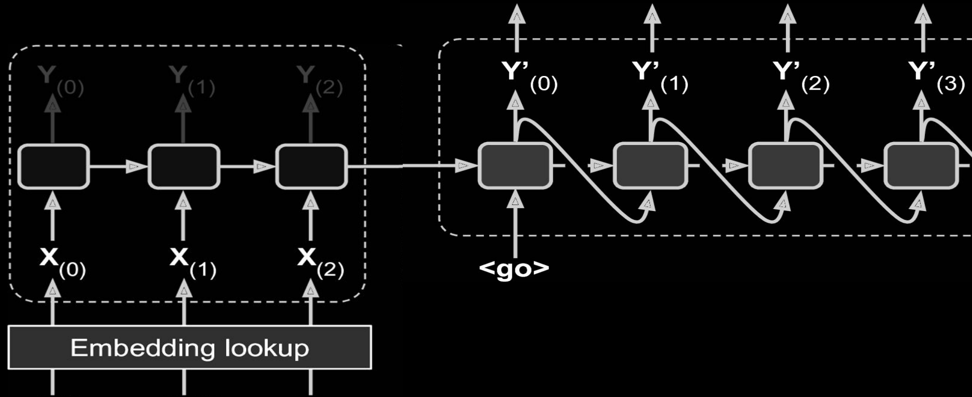
Without residuals

# Transformer for Encoder-Decoder

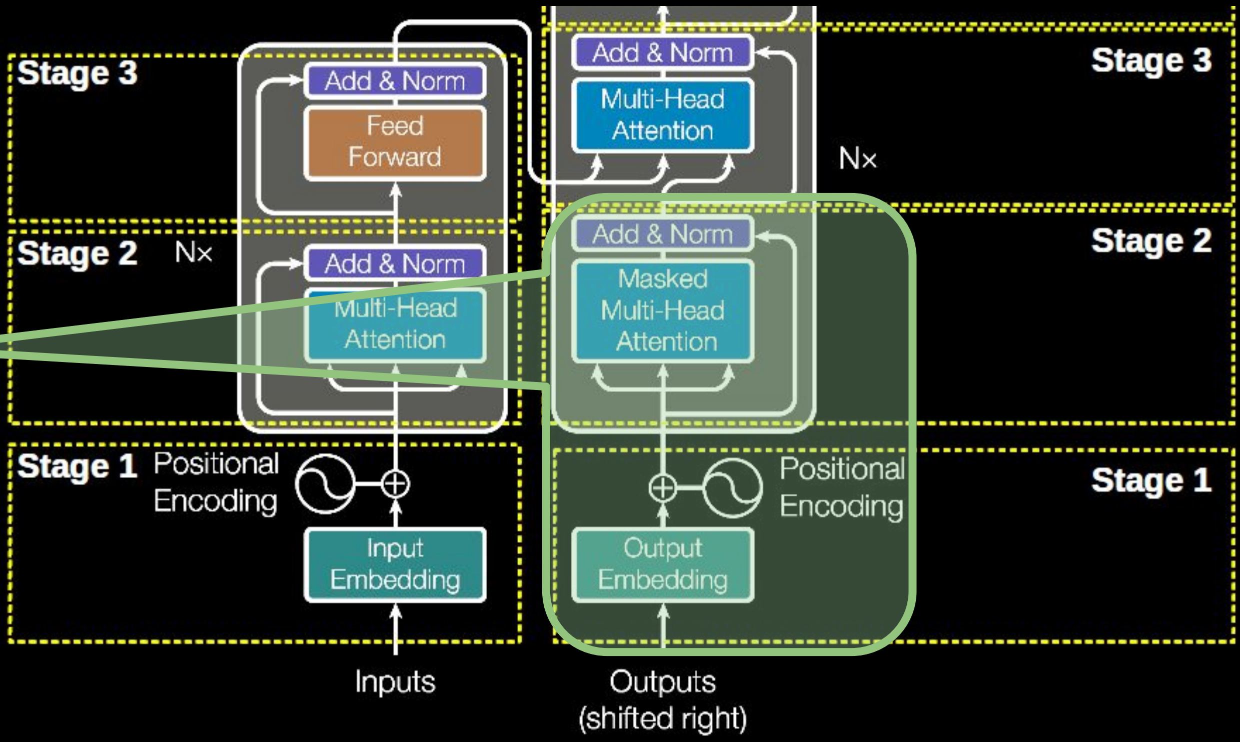


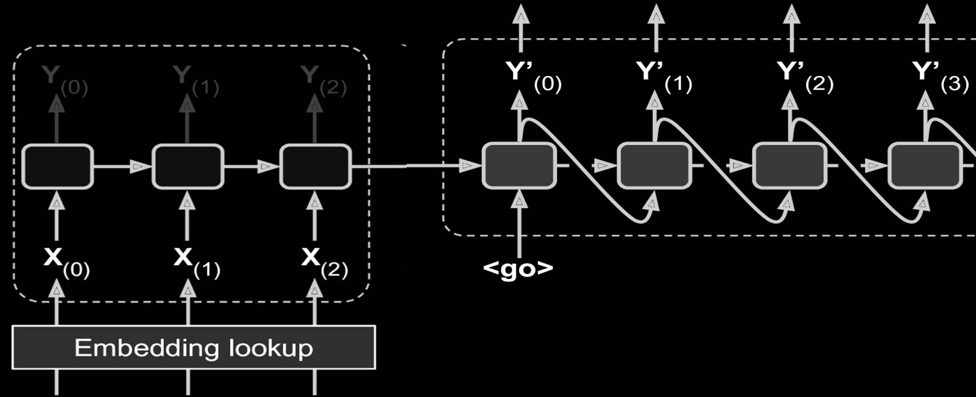
# Transformer for Encoder-Decoder



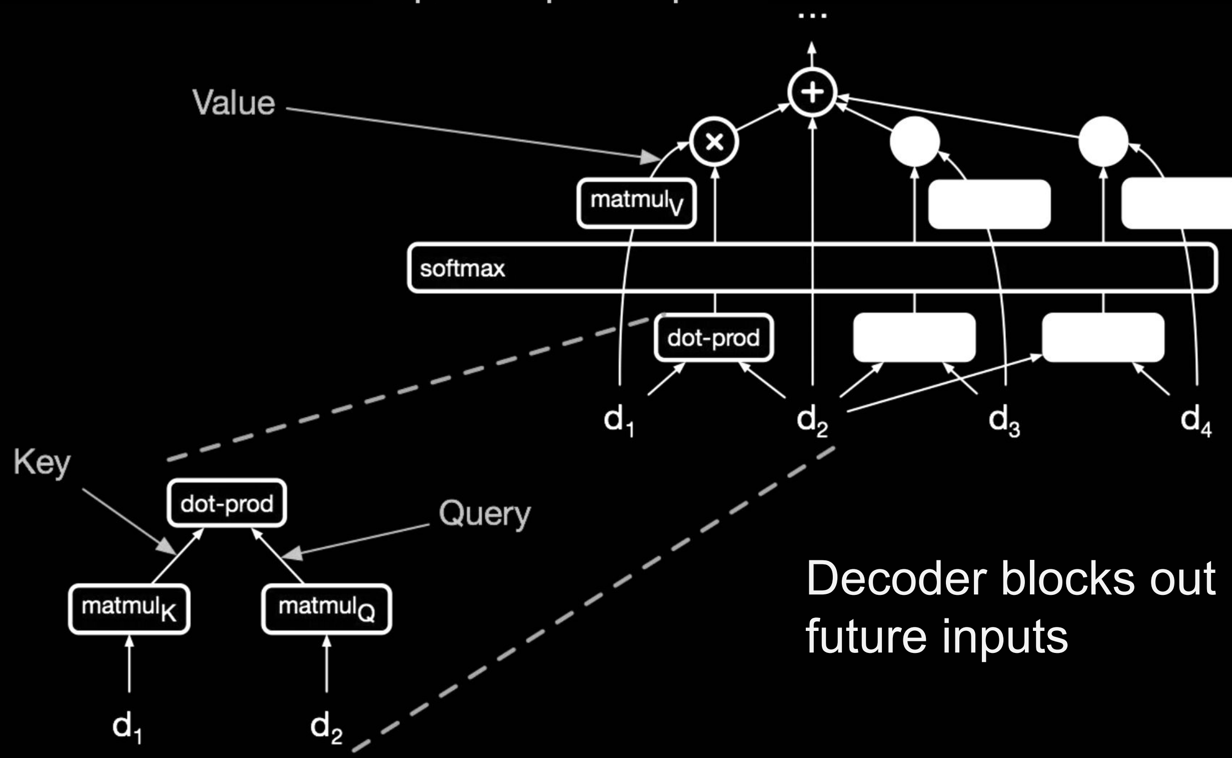

essentially, a language model



# Transformer for Encoder-Decoder




essentially, a language model

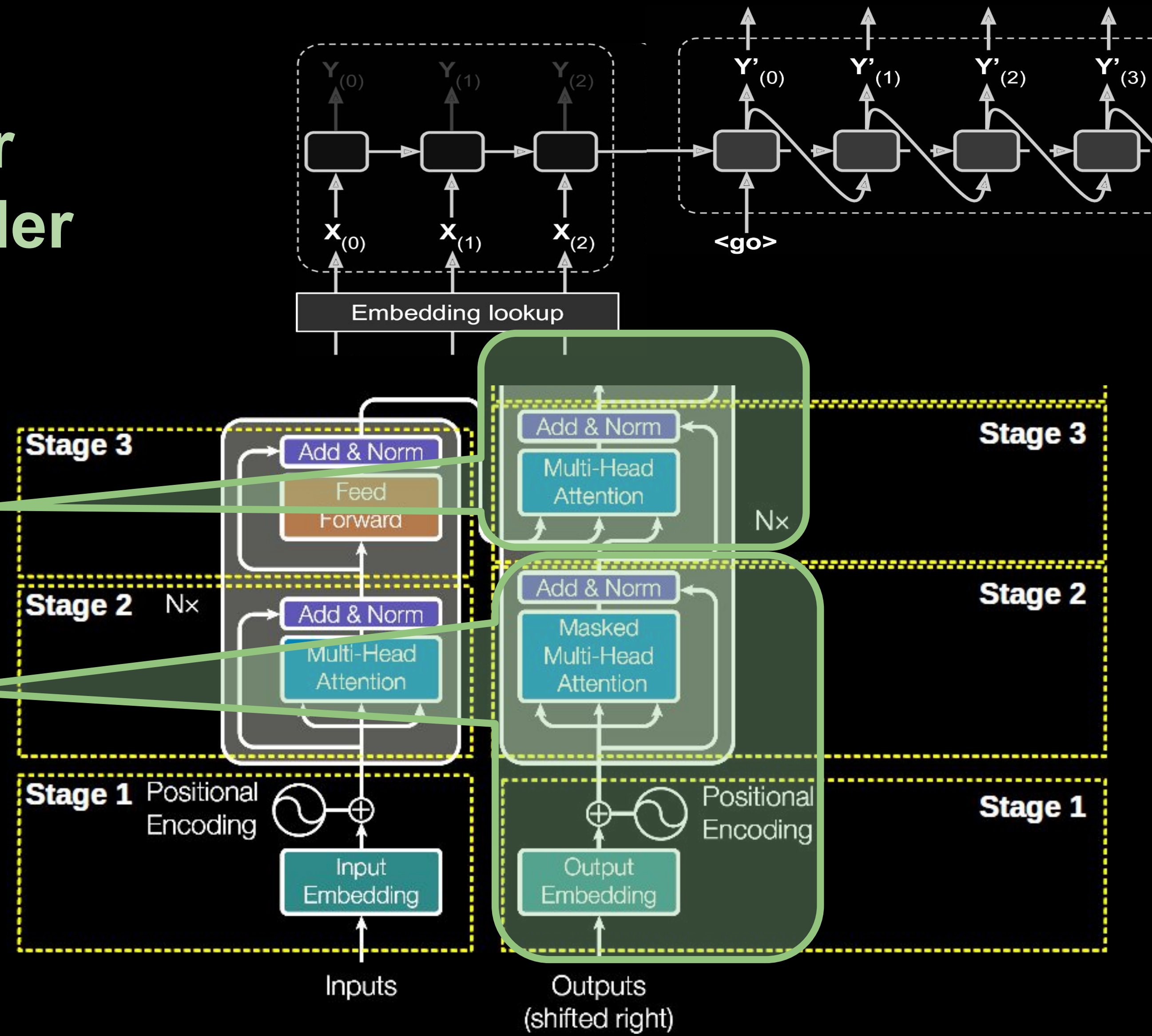



Decoder blocks out future inputs

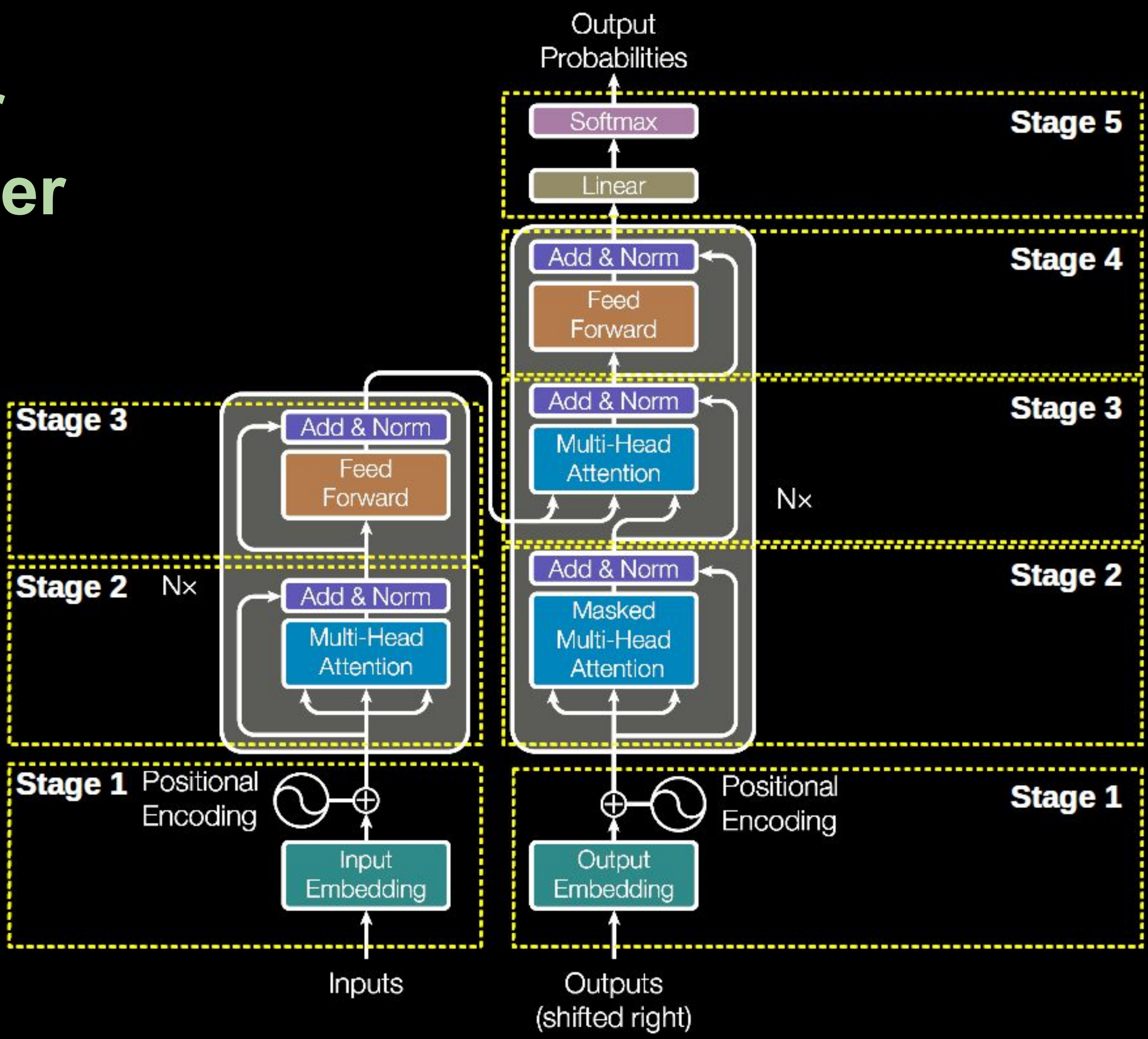
# Transformer for Encoder-Decoder

Add conditioning of the LM based on the encoder

essentially, a language model



# Transformer for Encoder-Decoder



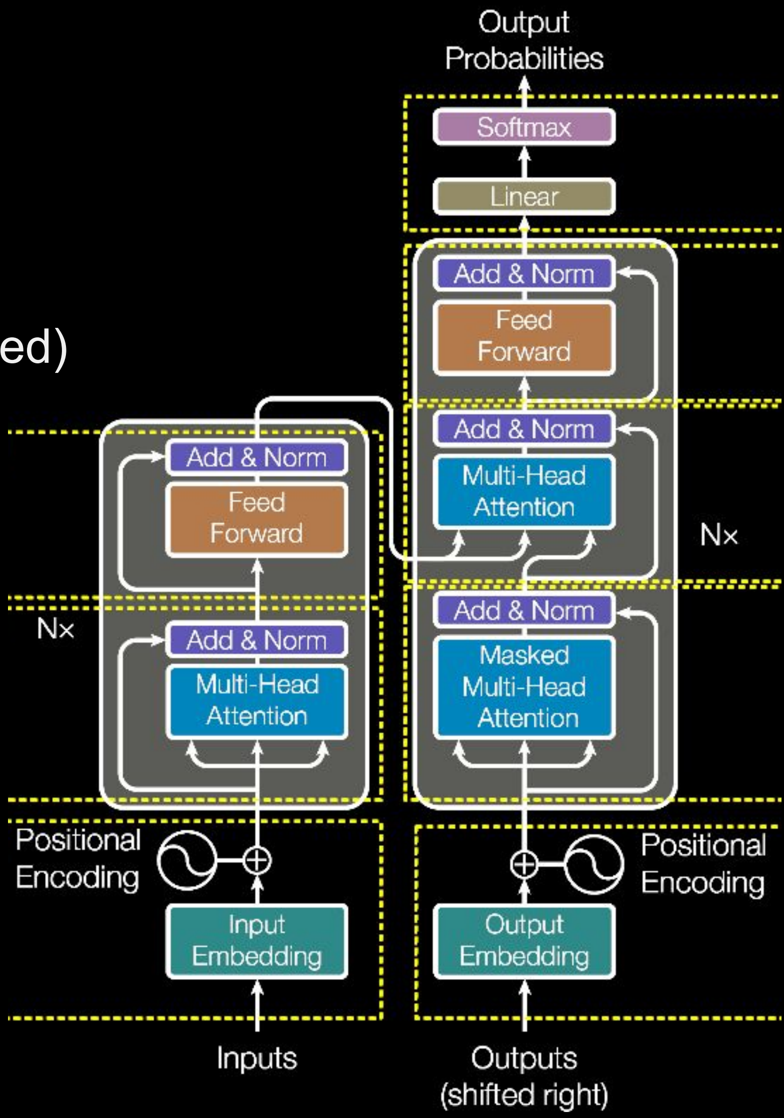
# Transformer (as of 2017)

"WMT-2014" Data Set. BLEU scores:

| | EN-DE | EN-FR |
|---|---|---|
| GNMT (orig) | 24.6 | 39.9 |
| ConvSeq2Seq | 25.2 | 40.5 |
| Transformer* | **28.4** | **41.8** |

# Transformer

- Utilize Self-Attention
- Simple att scoring function (dot product, scaled)
- Added linear layers for Q, K, and V
- Multi-head attention
- Added positional encoding
- Added residual connection
- Simulate decoding by masking



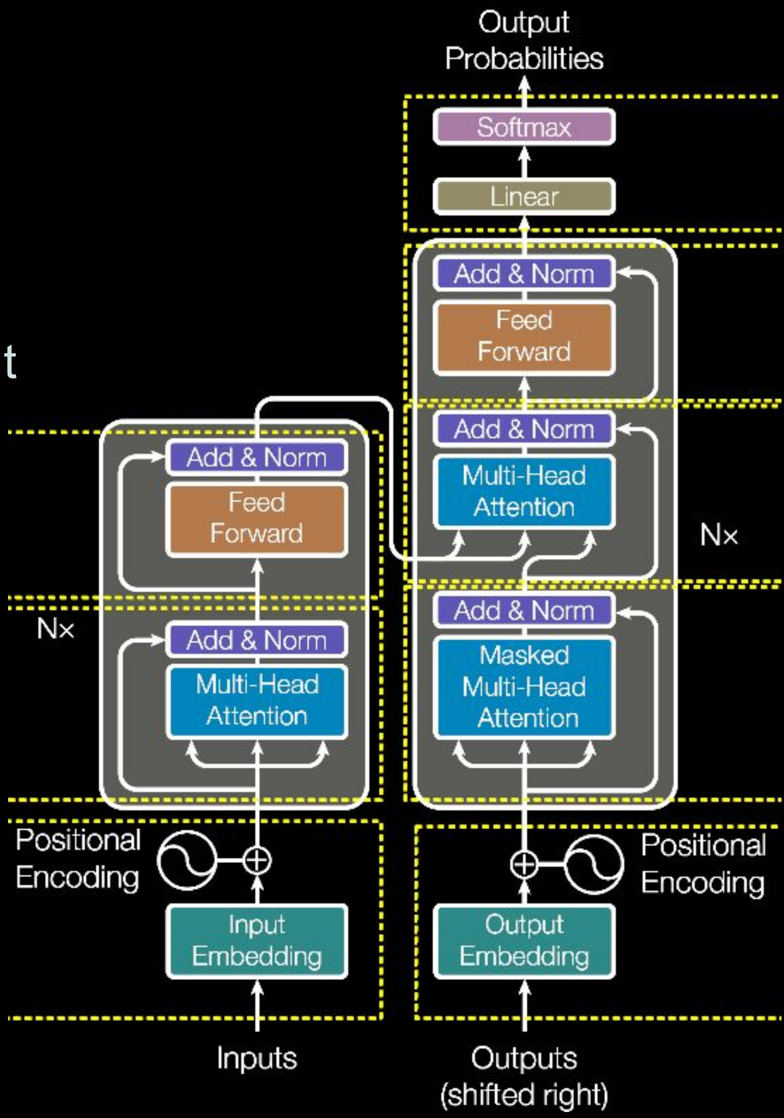
# Transformer

**Why?**

- Don't need complexity of LSTM/GRU cells
- Constant num edges between words (or input steps)
- Enables "interactions" (i.e. adaptations) between words
- Easy to parallelize -- don't need sequential processing.

**Drawbacks:**

- Only unidirectional by default
- Only a "single-hop" relationship per layer (multiple layers to capture multiple)



# BERT

**B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Produces contextualized embeddings
(or pre-trained contextualized encoder)

- Bidirectional context by “masking” in the middle
- A lot of layers, hidden states, attention heads.

**Drawbacks of Vanilla Transformers:**

- Only unidirectional by default
- Only a “single-hop” relationship per layer
  (multiple layers to capture multiple)

# BERT

**B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Produces contextualized embeddings
(or pre-trained contextualized encoder)

- **Bidirectional context by “masking” in the middle**
- A lot of layers, hidden states, attention heads.

*She saw the man on the hill with the telescope.*

*She* [mask] *the man on the hill* [mask] *the telescope.*

Mask 1 in 7 words:

- Too few: expensive, less robust
- Too many: not enough context

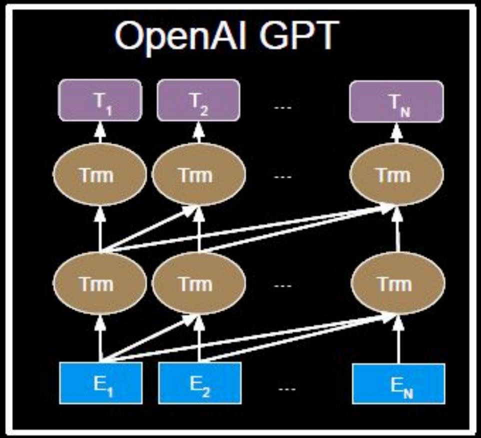
# BERT

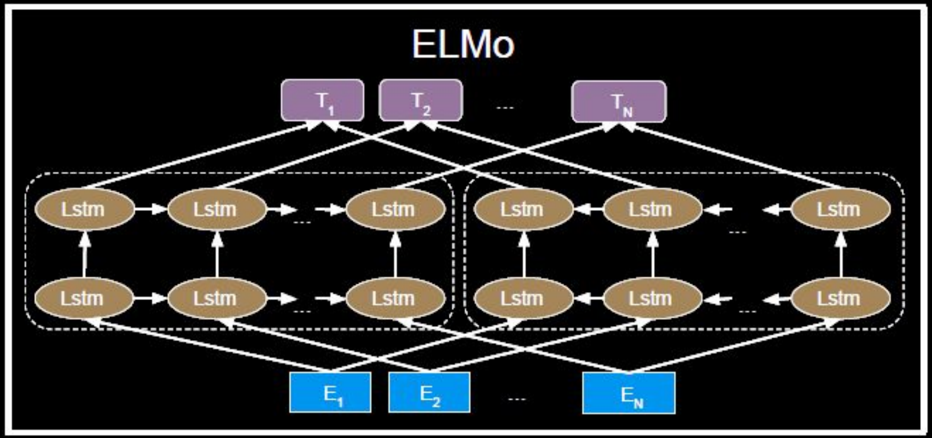
**B**idirectional **E**ncoder **R**epresentations from **T**ransformers

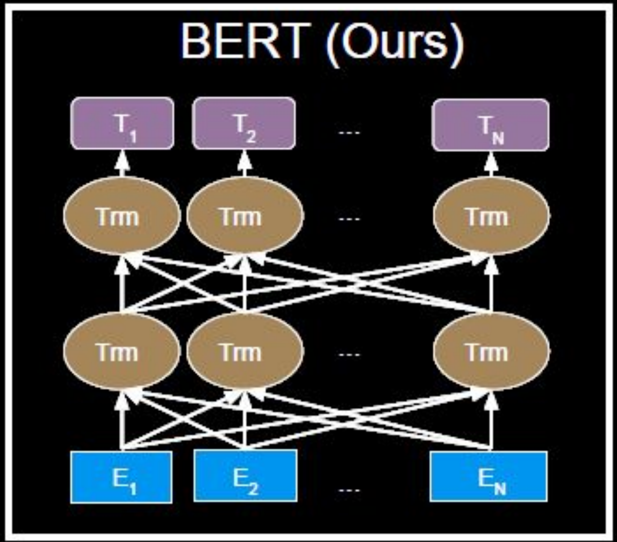
Produces contextualized embeddings
(or pre-trained contextualized encoder)

- Bidirectional context by “masking” in the middle
- **A lot of layers, hidden states, attention heads.**
  - `BERT-Base, Cased`:
    12-layer, 768-hidden, 12-heads , 110M parameters
  - `BERT-Large, Cased`:
    24-layer, 1024-hidden, 16-heads, 340M parameters
  - `BERT-Base, Multilingual Cased`:
    104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters

# BERT



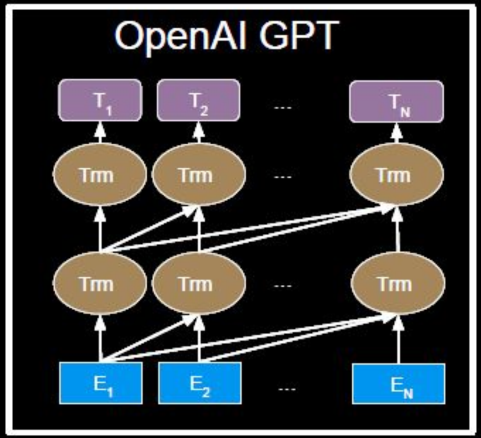



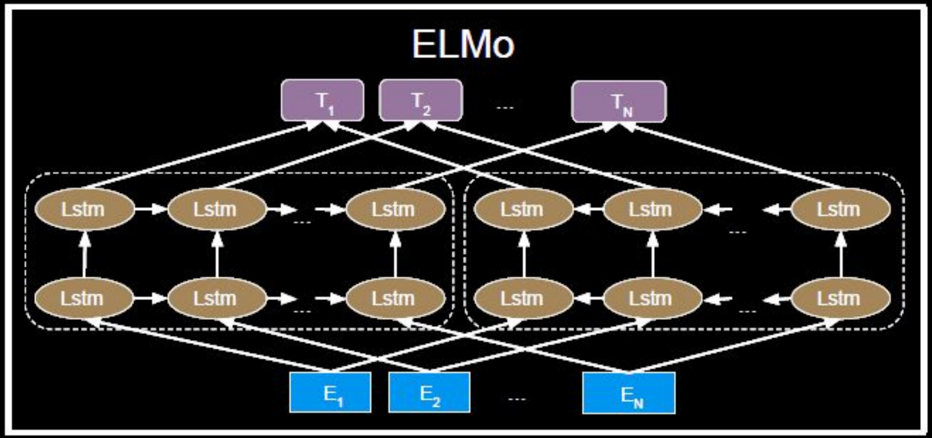

Differences from previous state of the art:

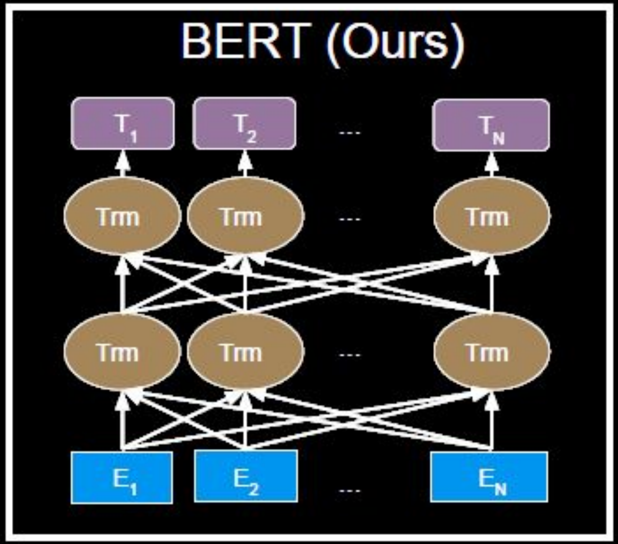
- Bidirectional transformer (through masking)
- Directions jointly trained at once.

(Devlin et al., 2019)

# BERT




Differences from previous state of the art:

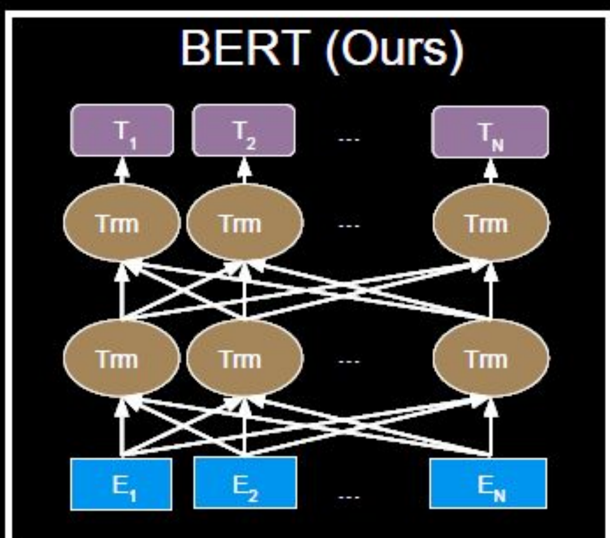
- Bidirectional transformer (through masking)
- Directions jointly trained at once.
- **Capture sentence-level relations**

(Devlin et al., 2019)

# BERT

**Sentence A** = `The man went to the store.`
**Sentence B** = `He bought a gallon of milk.`
**Label** = `IsNextSentence`

**Sentence A** = `The man went to the store.`
**Sentence B** = `Penguins are flightless.`
**Label** = `NotNextSentence`




Differences from previous state of the art:

- Bidirectional transformer (through masking)
- Directions jointly trained at once.
- **Capture sentence-level relations**

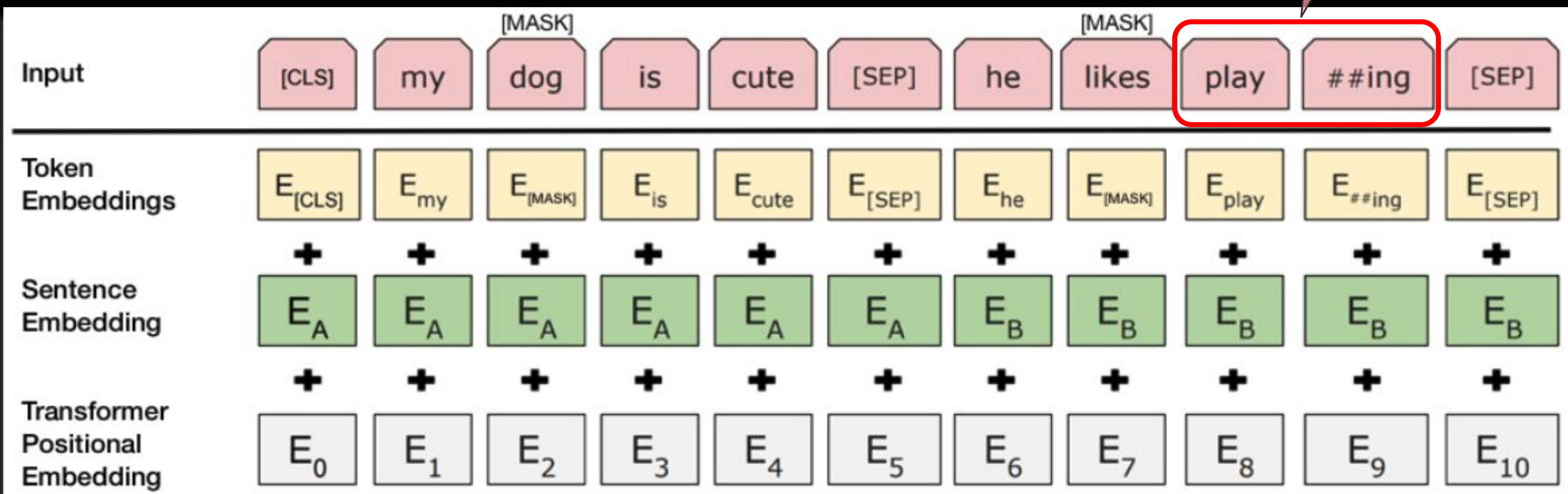
(Devlin et al., 2019)

# BERT

tokenize into “word pieces”

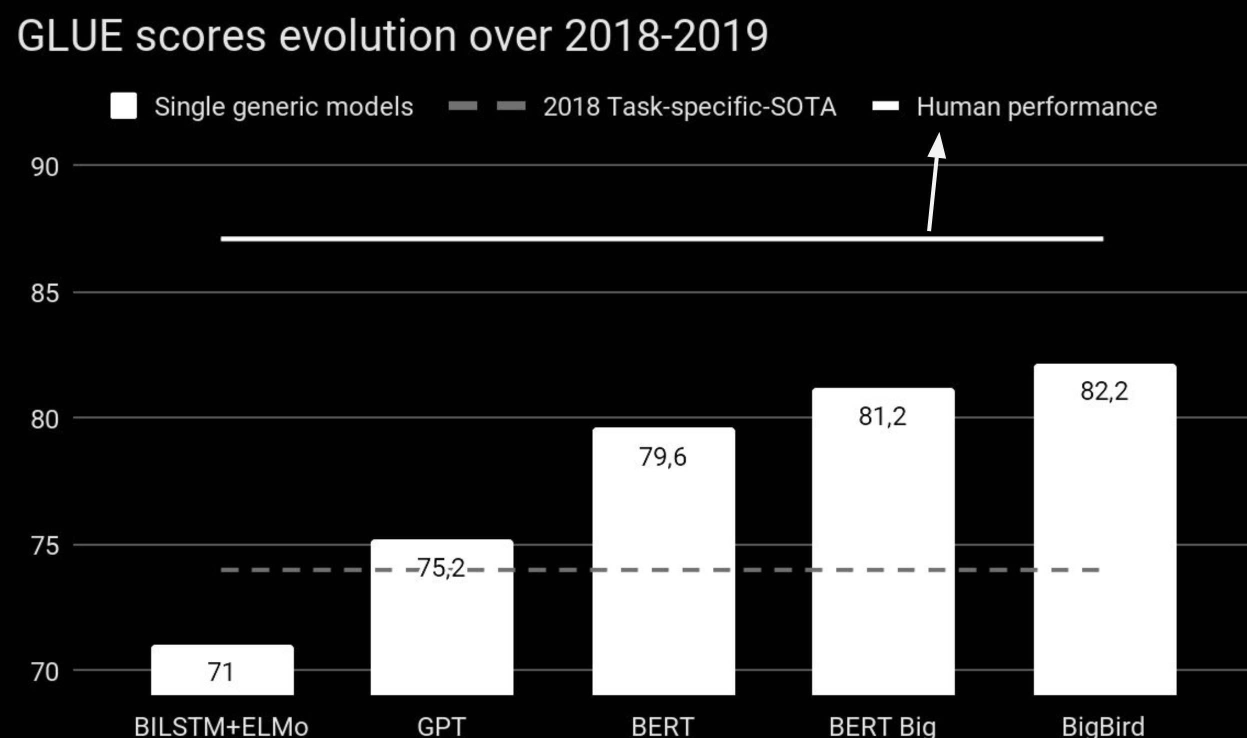
**Sentence A** = `The man went to the store.`
**Sentence B** = `He bought a gallon of milk.`
**Label** = `IsNextSentence`

**Sentence A** = `The man went to the store.`
**Sentence B** = `Penguins are flightless.`
**Label** = `NotNextSentence`




(Devlin et al., 2019)

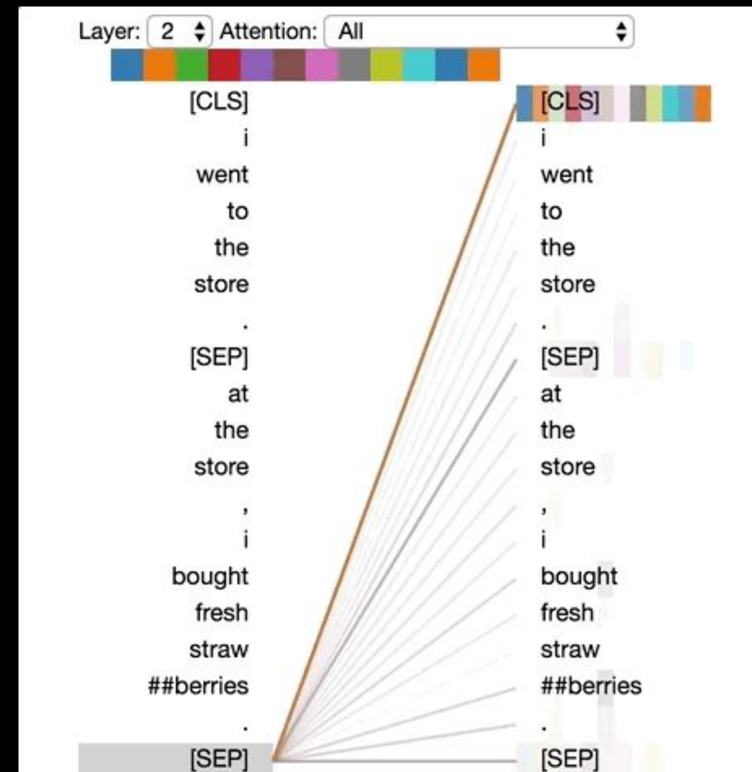
# BERT Performance: e.g. Question Answering




https://rajpurkar.github.io/SQuAD-explorer/

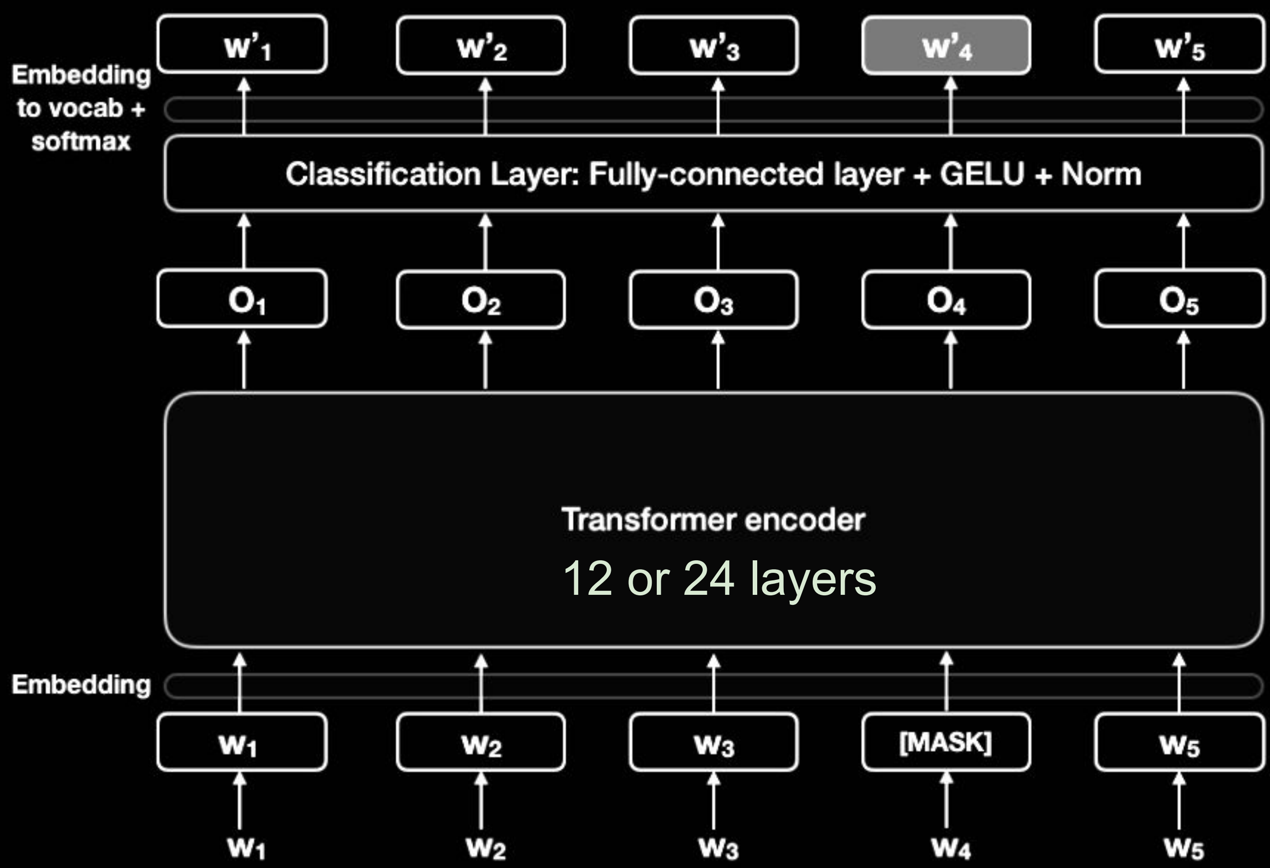
# Bert: Attention by Layers

https://colab.research.google.com/drive/1vlOJ1IhdujVjfH857hvYKIdKPTD9Kid8

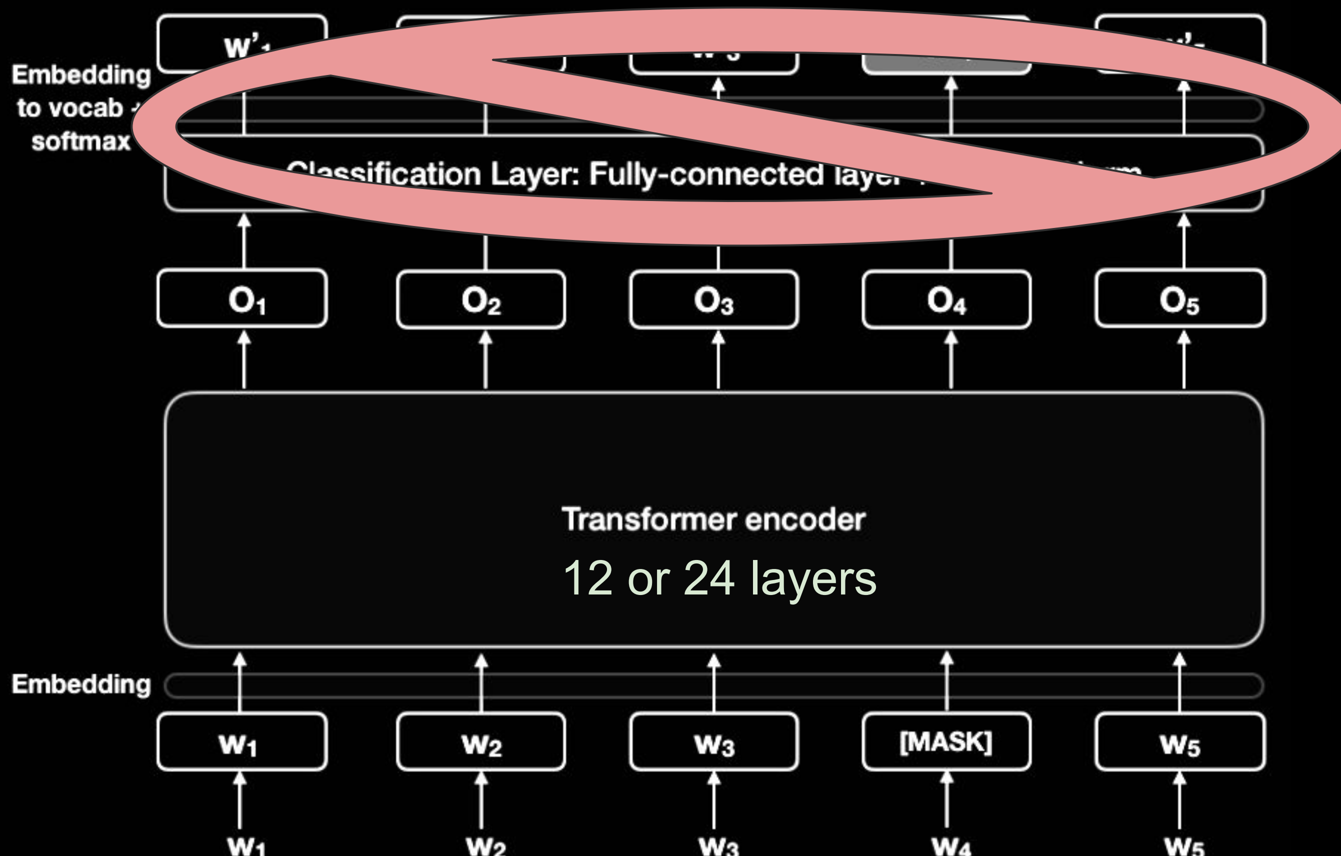



(Vig, 2019)

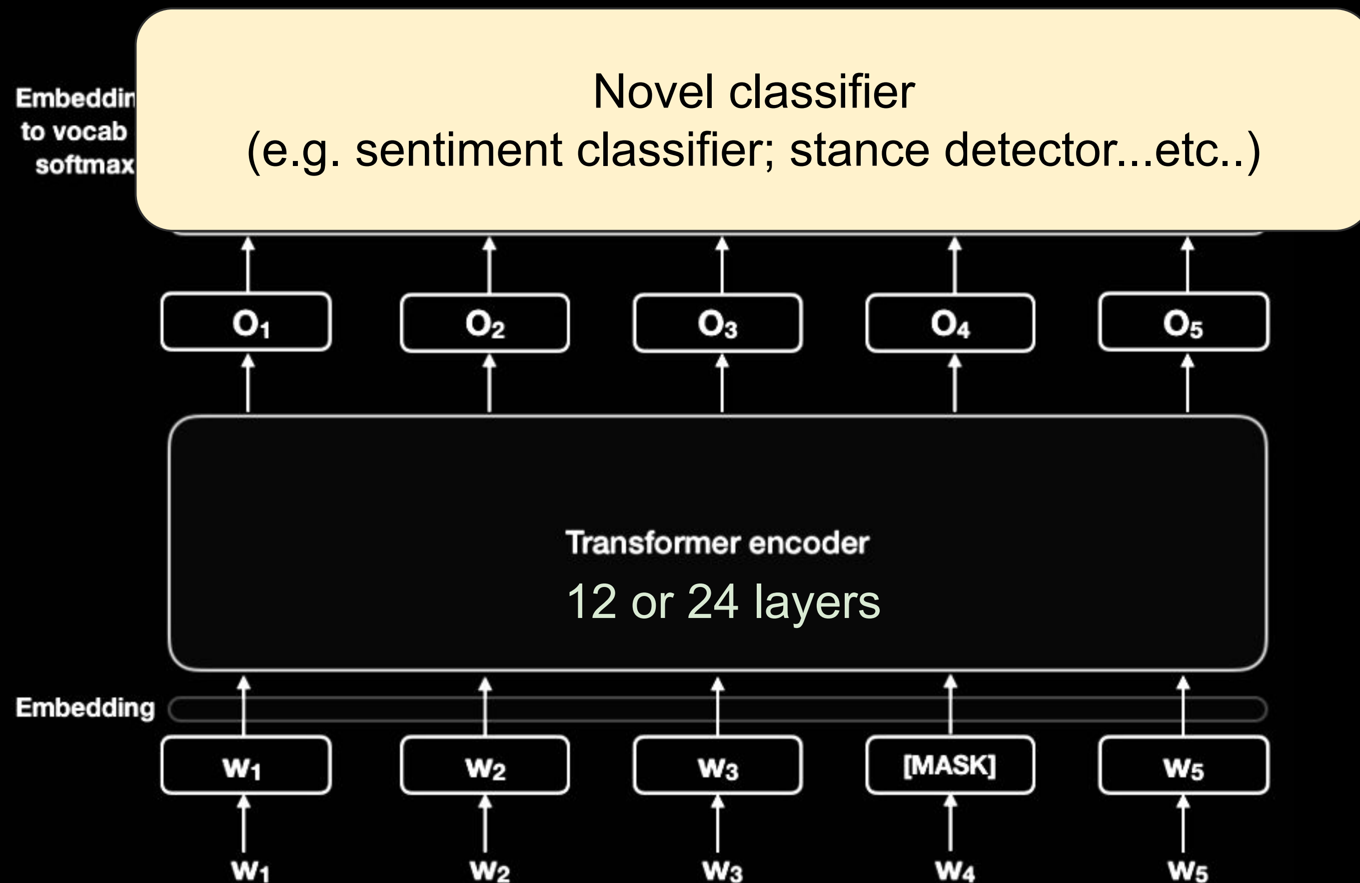
# BERT: Pre-training; Fine-tuning



# BERT: Pre-training; Fine-tuning



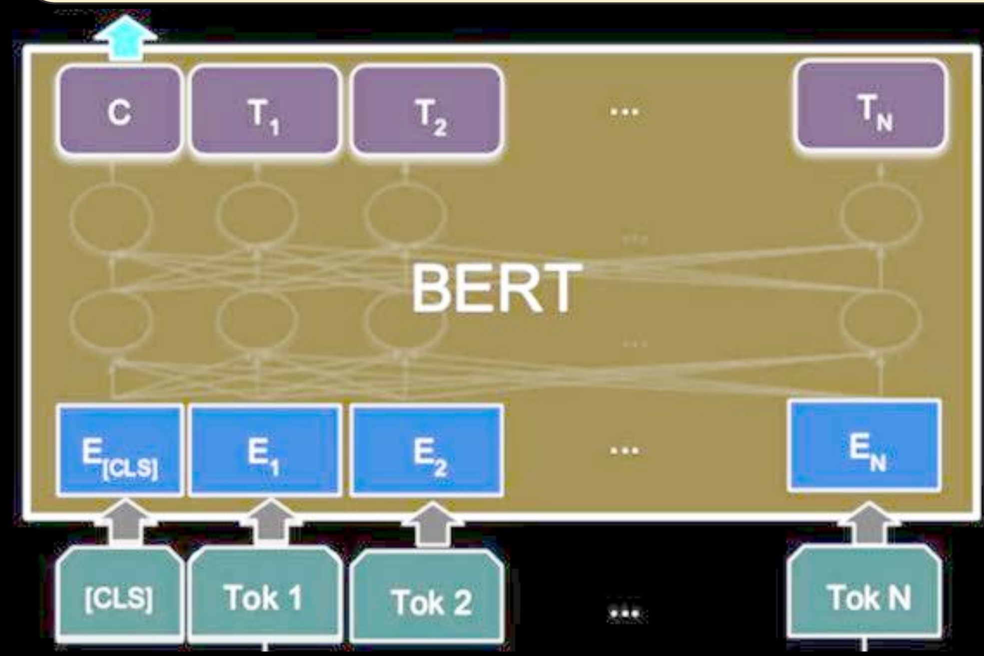
# BERT: Pre-training; Fine-tuning



# BERT: Pre-training; Fine-tuning

[CLS] vector at start is supposed to capture meaning of whole sequence.

Novel classifier
(e.g. sentiment classifier; stance detector...etc..)



# BERT: Pre-training; Fine-tuning

[CLS] vector at start is supposed to capture meaning of whole sequence.

Average of top layer (or second to top) also often used.