Natural Language Processing

Lecture 18:
Recurrent Neural Nets

11/29/2018

COMS W4705
Daniel Bauer

Application: Sentiment Detection

- Goal: identify the opinion expressed in a text
    - Product reviews (movies, ...)
    - Social media.
    - ...
- Output: positive/negative/(neutral)

Positive or Negative Reviews?

- *unbelievably disappointing!*

- *Full of zany characters and richly applied satire, and some great plot twists.*

- *This is the greatest screwball comedy ever filmed*

- *It was pathetic. The worst part about it was the boxing scenes.*

How would you solve this problem?

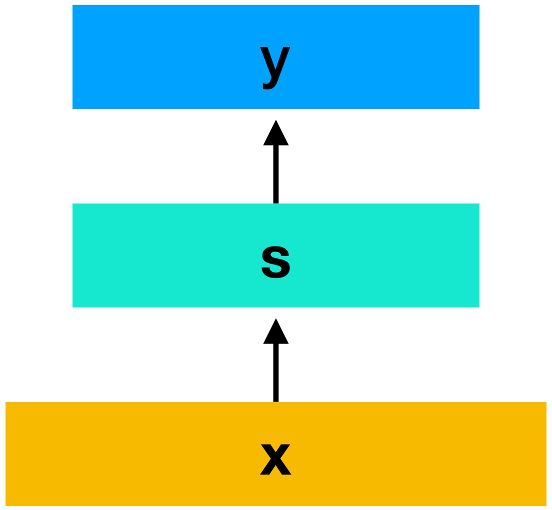
Sequence Modeling with Neural Networks

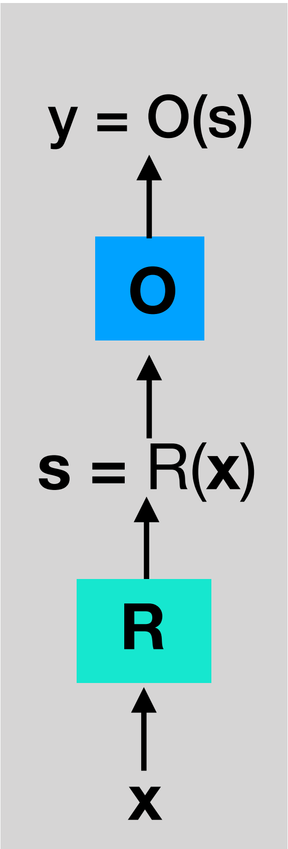
- Many NLP tasks require models of sequences (language models, text classification, sentiment analysis, POS tagging, machine translation).
- Neural Language Model represent the context as a sliding window.
  - Input word-representations for context words are concatenated.
  - Shared weights across contexts.
- Recurrent Neural Networks take **entire** history into account.

Feed-forward neural network

- Neural network with one hidden layer.
- Network computes two functions:
  - *R* maps input vector **x** to hidden representation **s**.
  - *O* maps hidden representation **s** to output vector **y**.

$$R(\mathbf{x}) = tanh(\sum_i w_i x_i) = tanh(\mathbf{w} \cdot \mathbf{x})$$

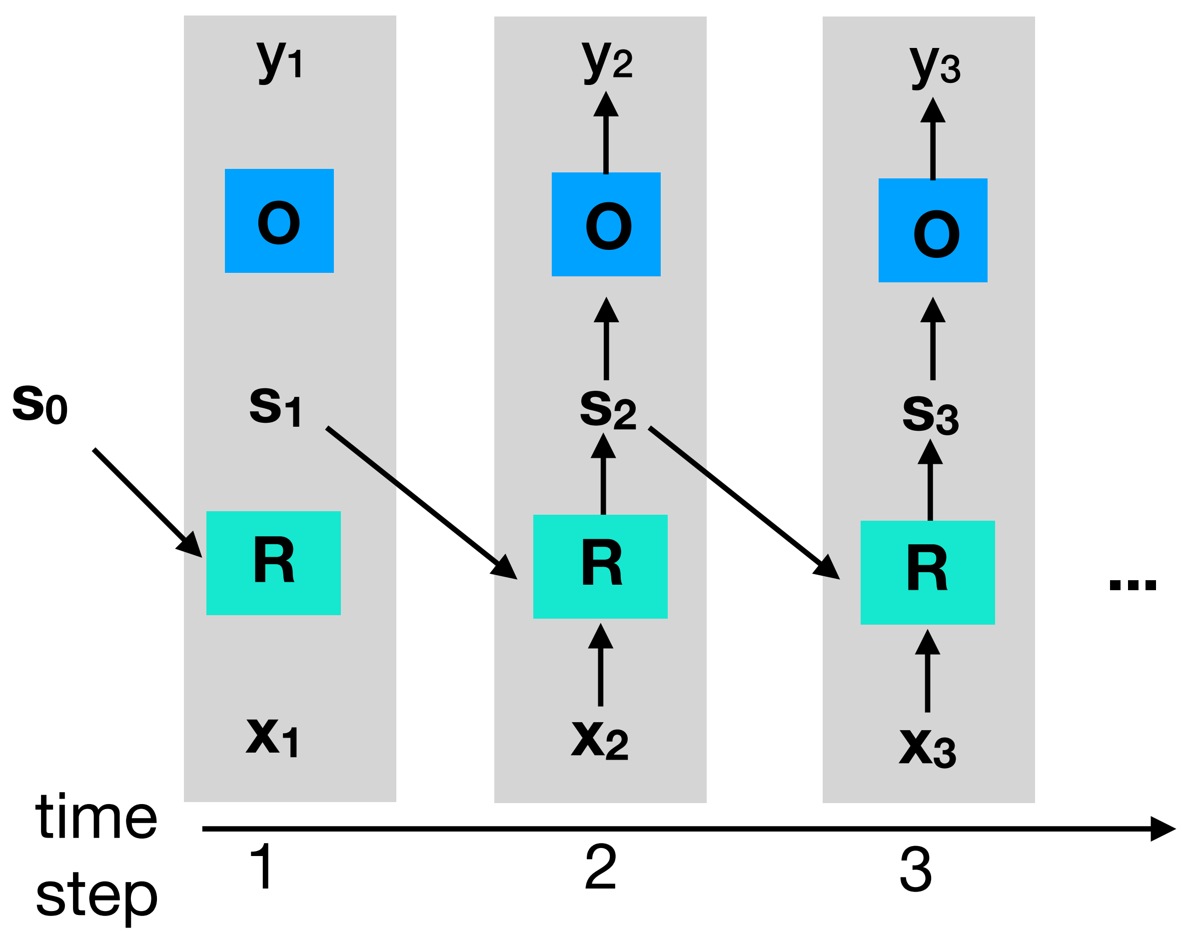


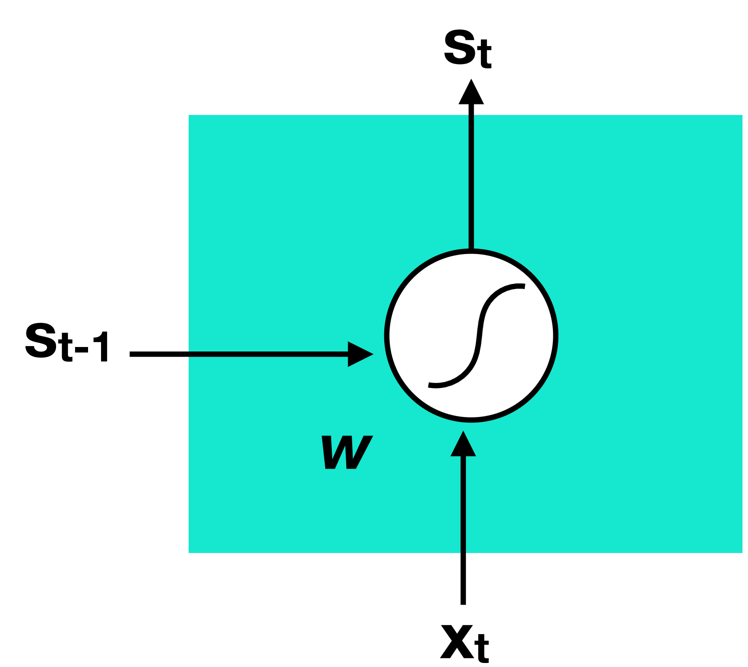


Basic RNN

- Basic idea: Hidden layer represents a state.
  State representation is fed back into the function R.

- Weights are shared across all time steps!

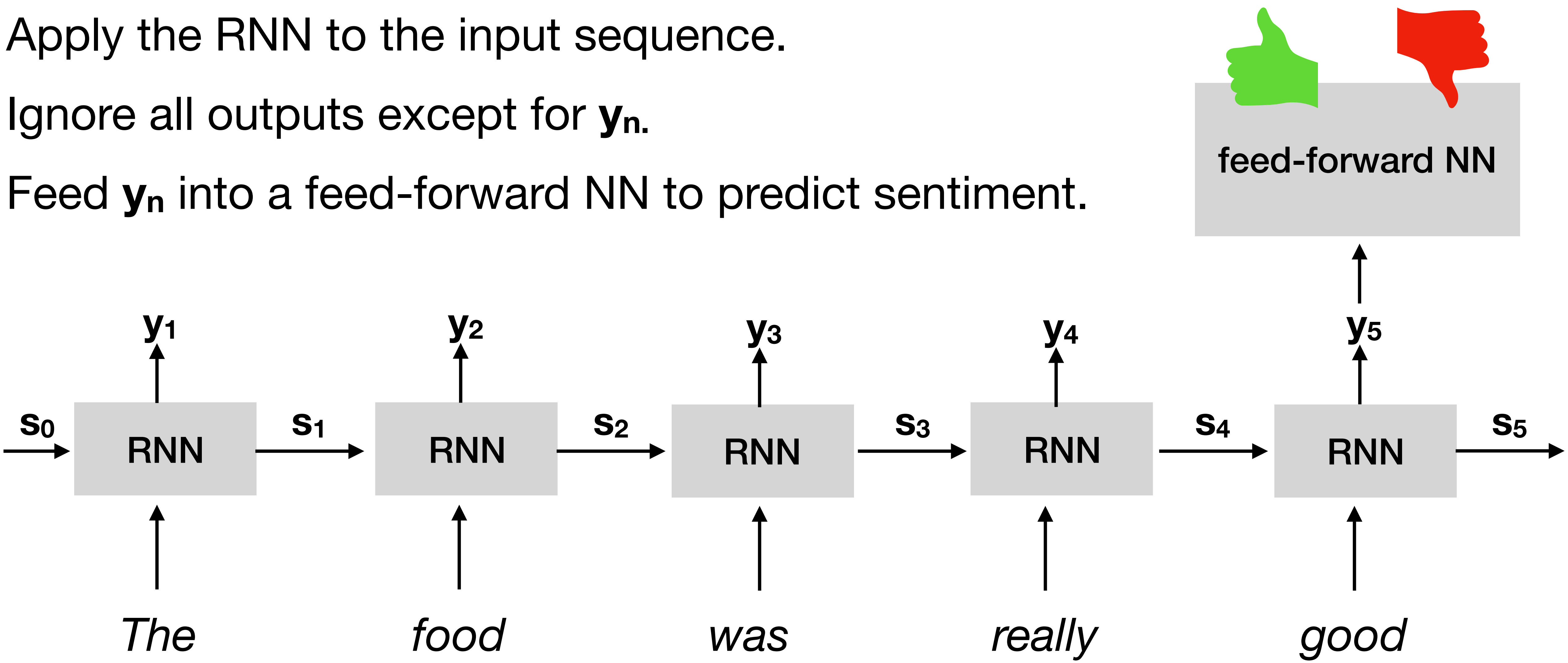




$$s_t = R\left(\begin{bmatrix} \mathbf{x}_t \\ \mathbf{s}_{t-1} \end{bmatrix}\right) = tanh\left(\mathbf{w} \cdot \begin{bmatrix} \mathbf{x}_t \\ \mathbf{s}_{t-1} \end{bmatrix}\right)$$

Sentiment Analysis with RNNs

- Apply the RNN to the input sequence.
- Ignore all outputs except for $\mathbf{y_n}$.
- Feed $\mathbf{y_n}$ into a feed-forward NN to predict sentiment.



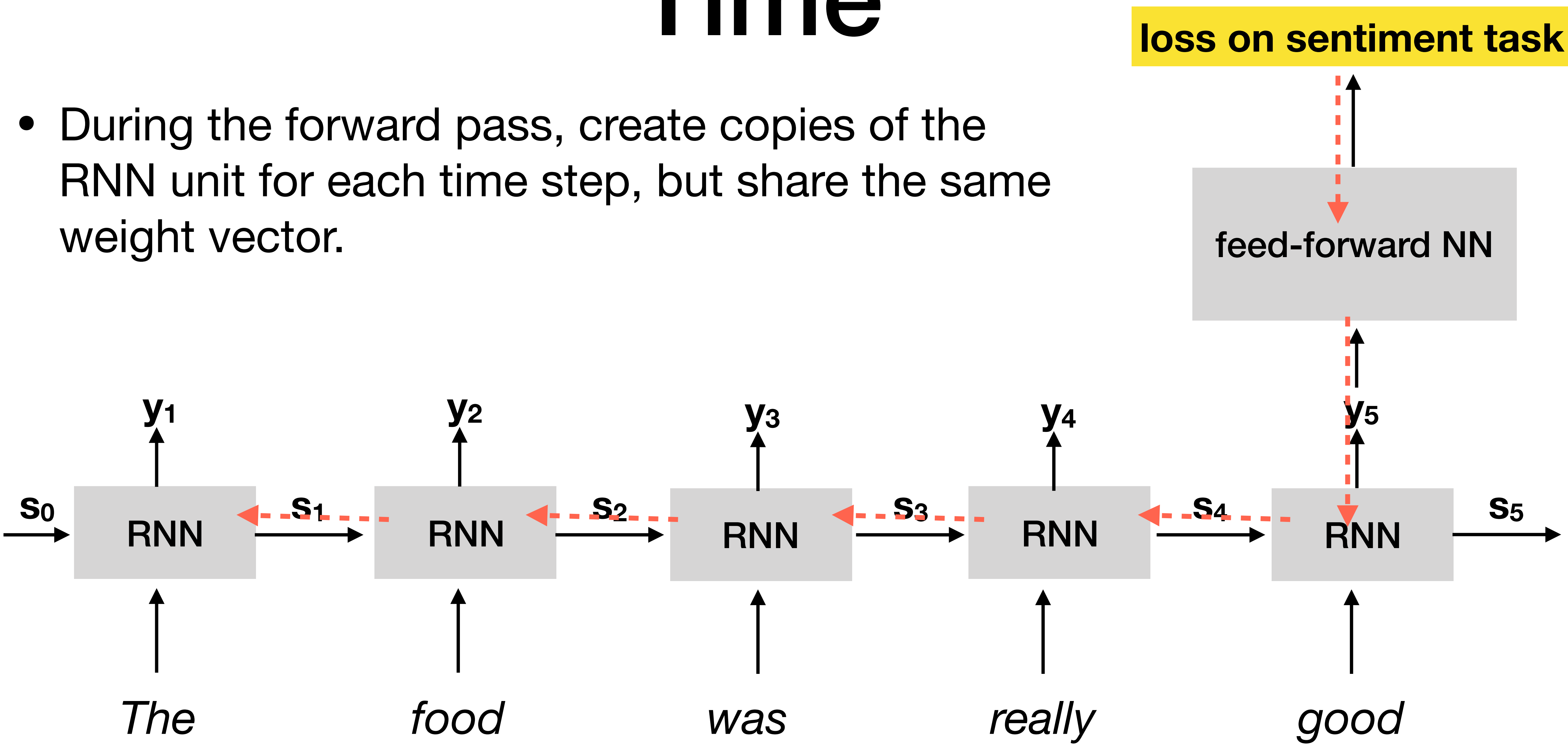
- How do we represent the words?
- How do you train such a model?

Training RNNs

- One approach: Backprogation Through Time (BPTT)
- For each input, treat the unfolded network (copies of all units for each time step) as one big feed-forward network.
  - But with shared weights across time steps.
- Compute loss and run backpropagation as usual.
- This way the RNN is optimized for some task (e.g. sentiment analysis).

Backpropagation Through Time

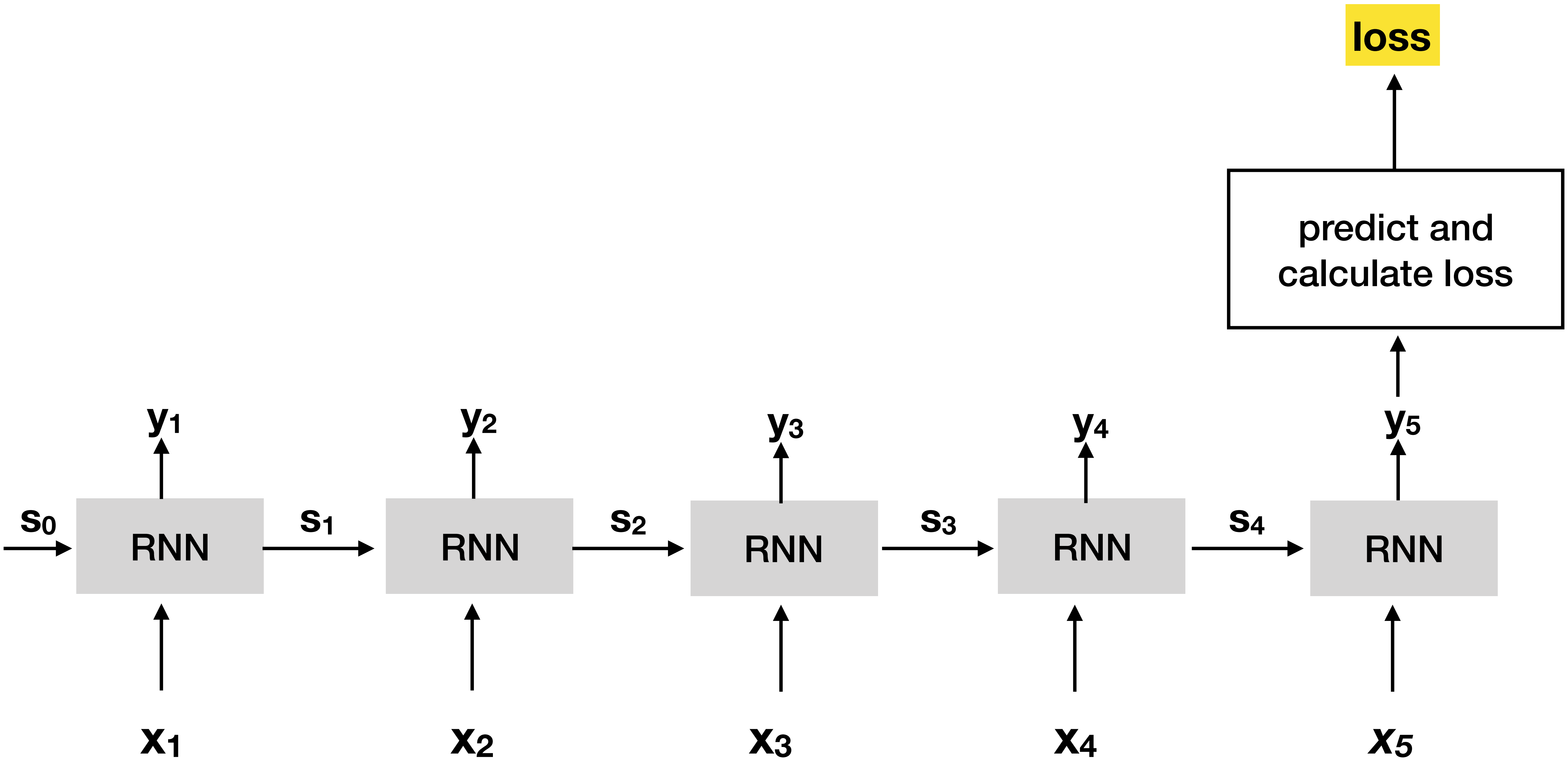
- During the forward pass, create copies of the RNN unit for each time step, but share the same weight vector.



Common Usage Patterns

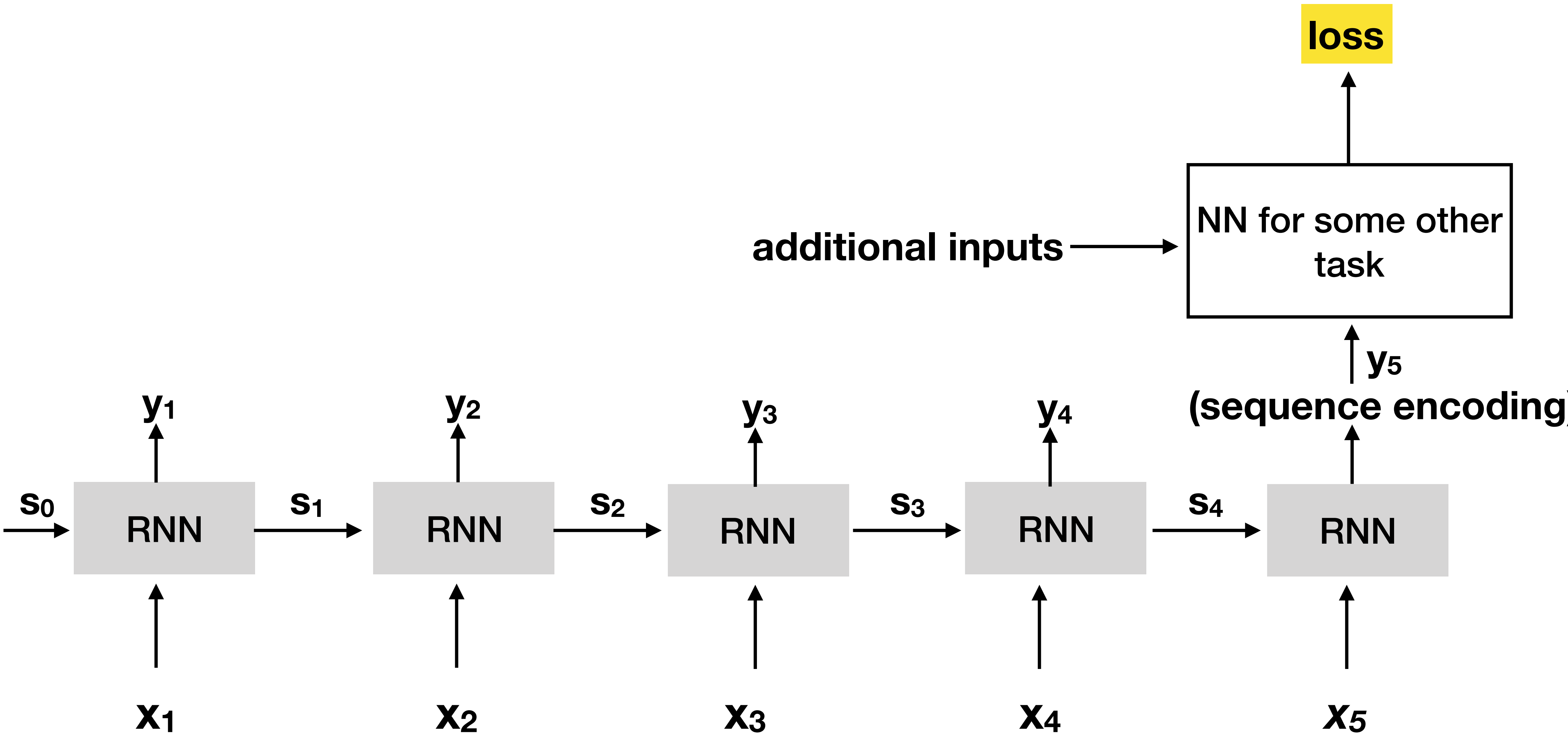
- RNNs can be used in a number of usage patterns:
  - Acceptor / Encoder
  - Transducer
    - Transducer as Generator
    - Conditioned Transduction
      - Encoder-Decoder models

Acceptor



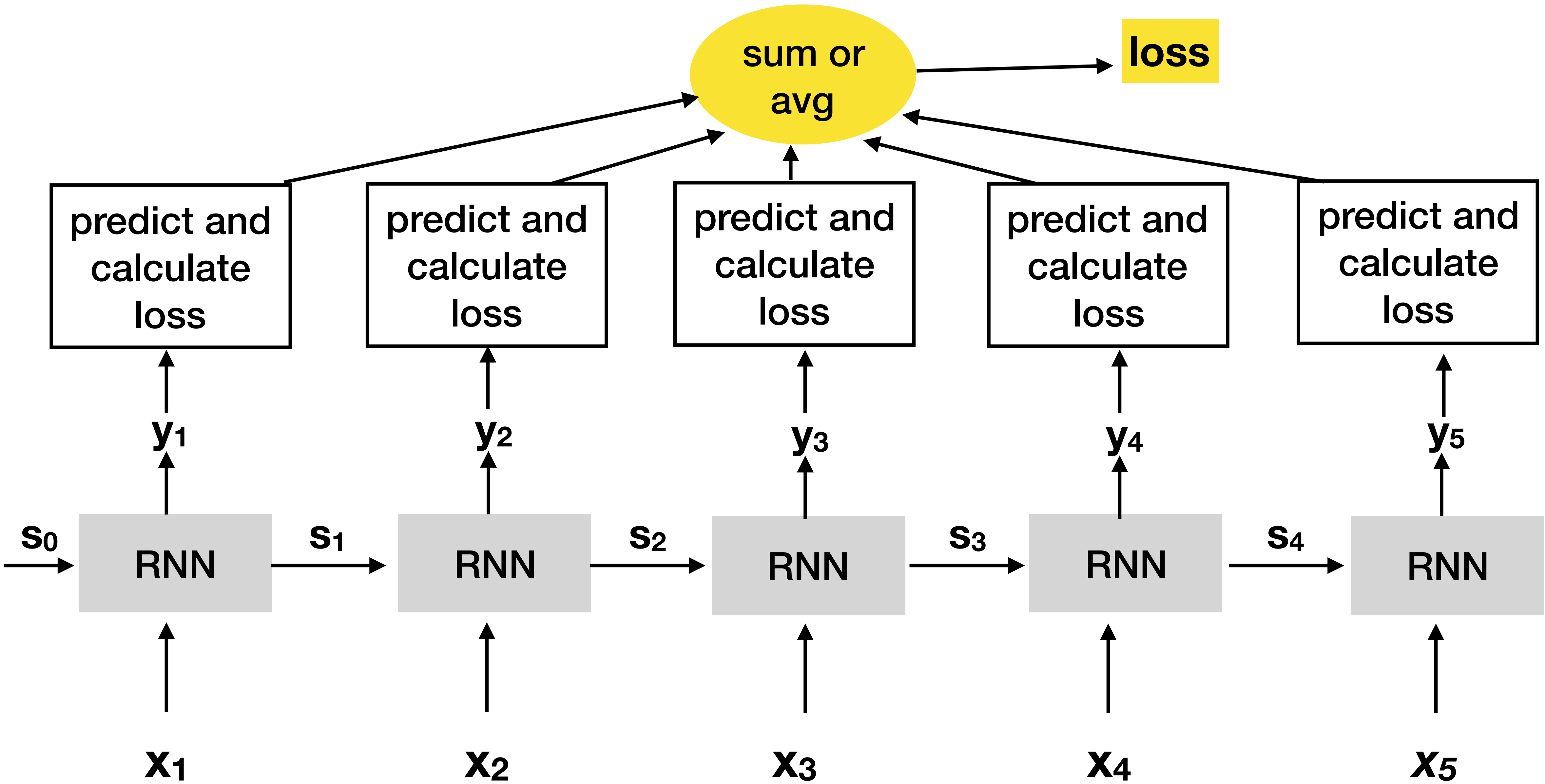
- Applications: Text classification, sentiment detection, ...

Encoder



- RNN used to compute sequence encoding (for example, to compute a sentence representation).
  This representation is then used in some other task.

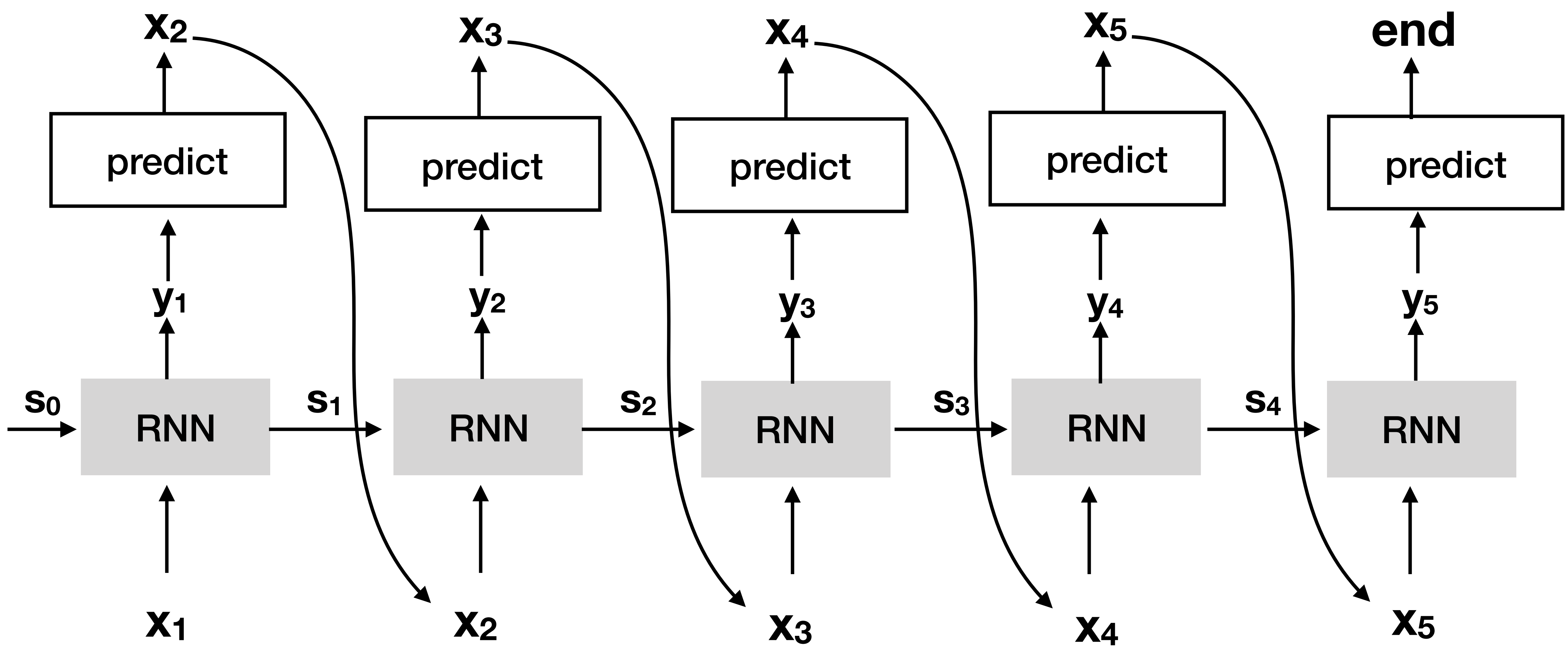
Transducer



- One output for each input (time step).
- During training, loss for each time step is combined.
- Applications: sequence tagging (e.g. POS).

Generator

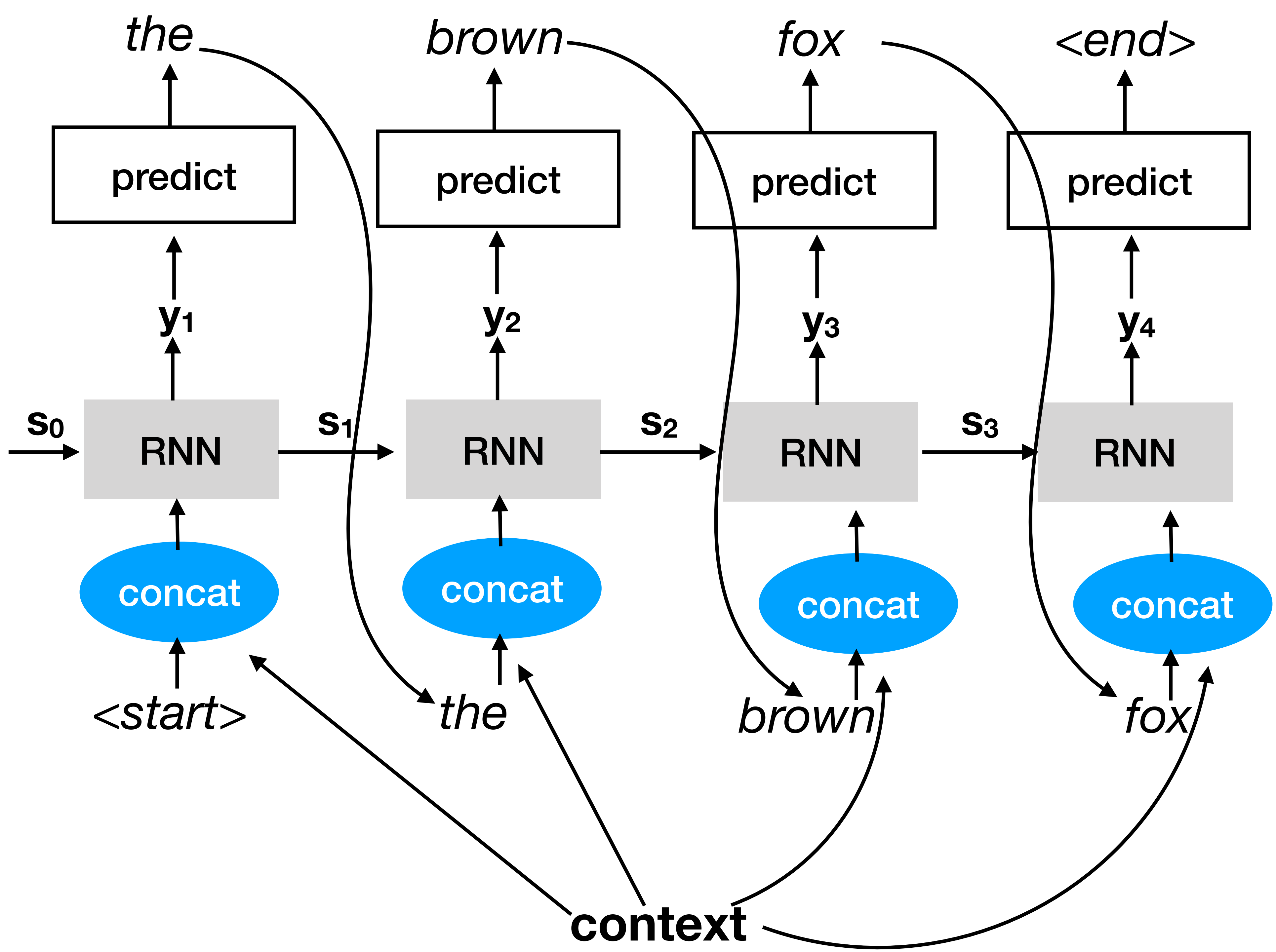
- **Transducer used for Generation**:



- Applications:
  language modeling

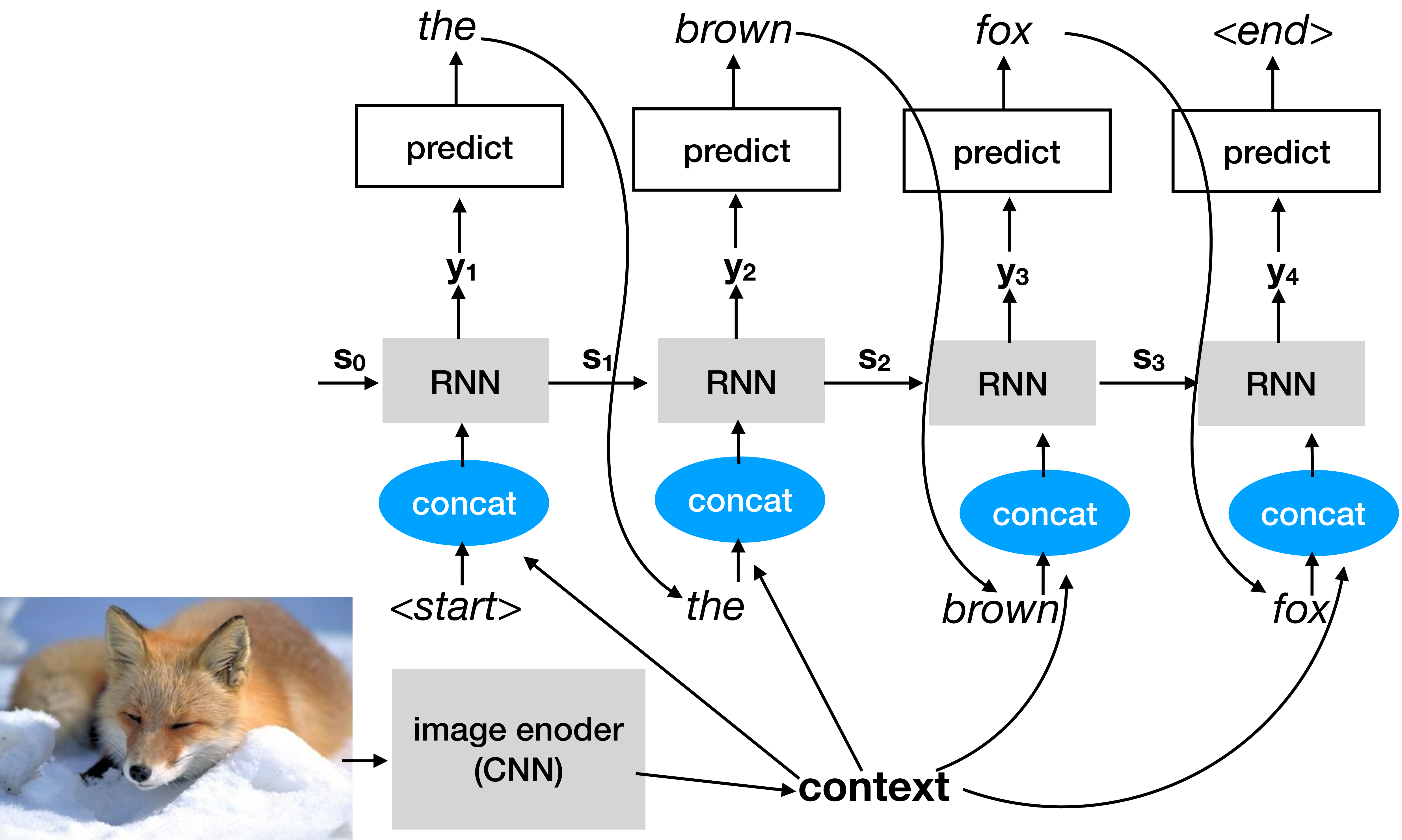
- Typically trained like a regular transducer.

Conditioned Generator



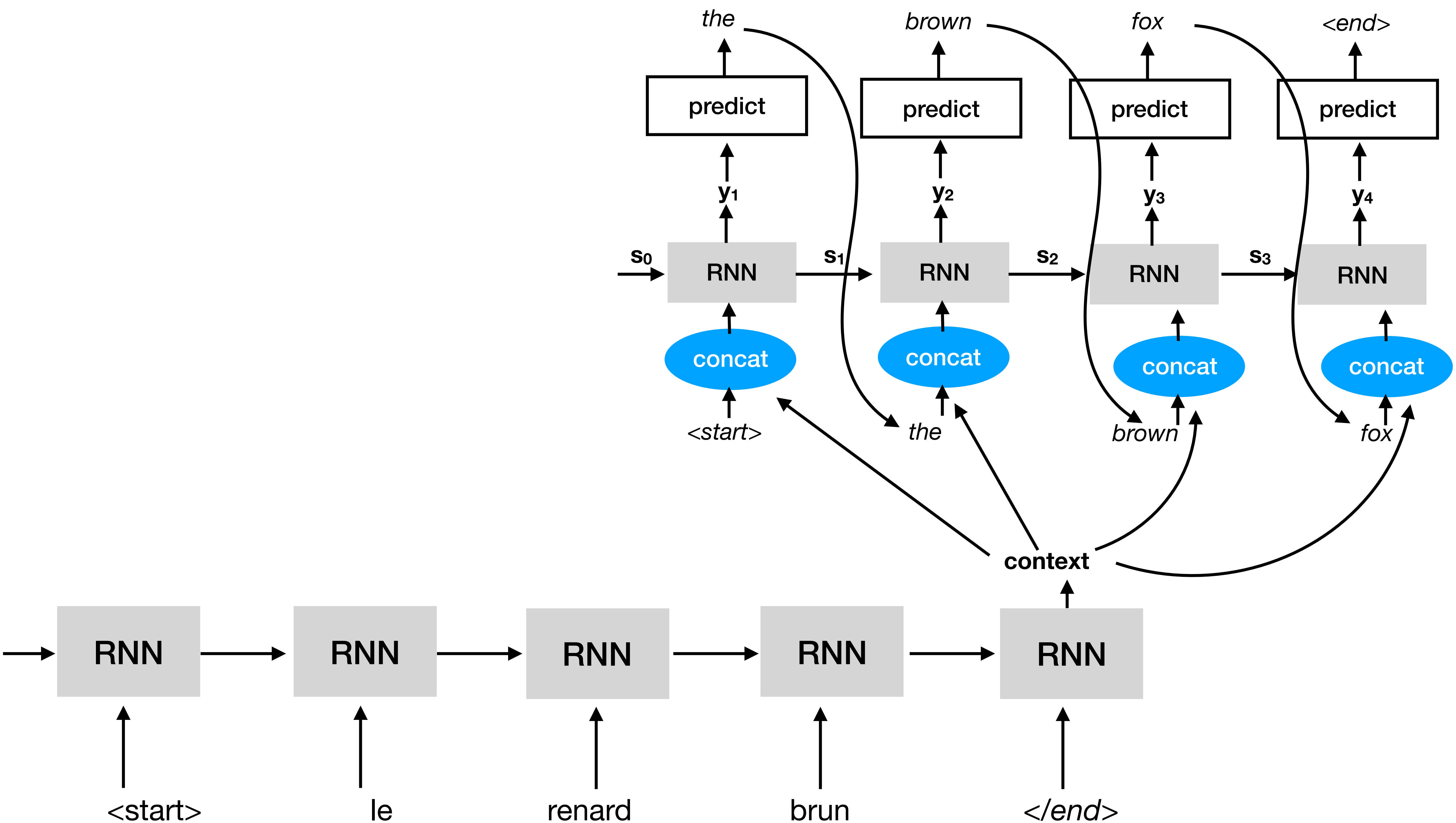
Conditioned Generator

- **Application: Image Captioning**



Sequence to Sequence

- **Application: Machine Translation**



Attention Mechanism

(Bahdanau et al., 2015, Luong et al., 2015)

- Problem with the simple encoder-decoder model:
    - For long phrases, fixed-length encoded representation becomes information bottleneck.
    - Not everything in the input sequence is equally important to predict each word in the decoder.
    - Can we integrate the idea of alignments into the encoder-decoder approach?

Attention Mechanism

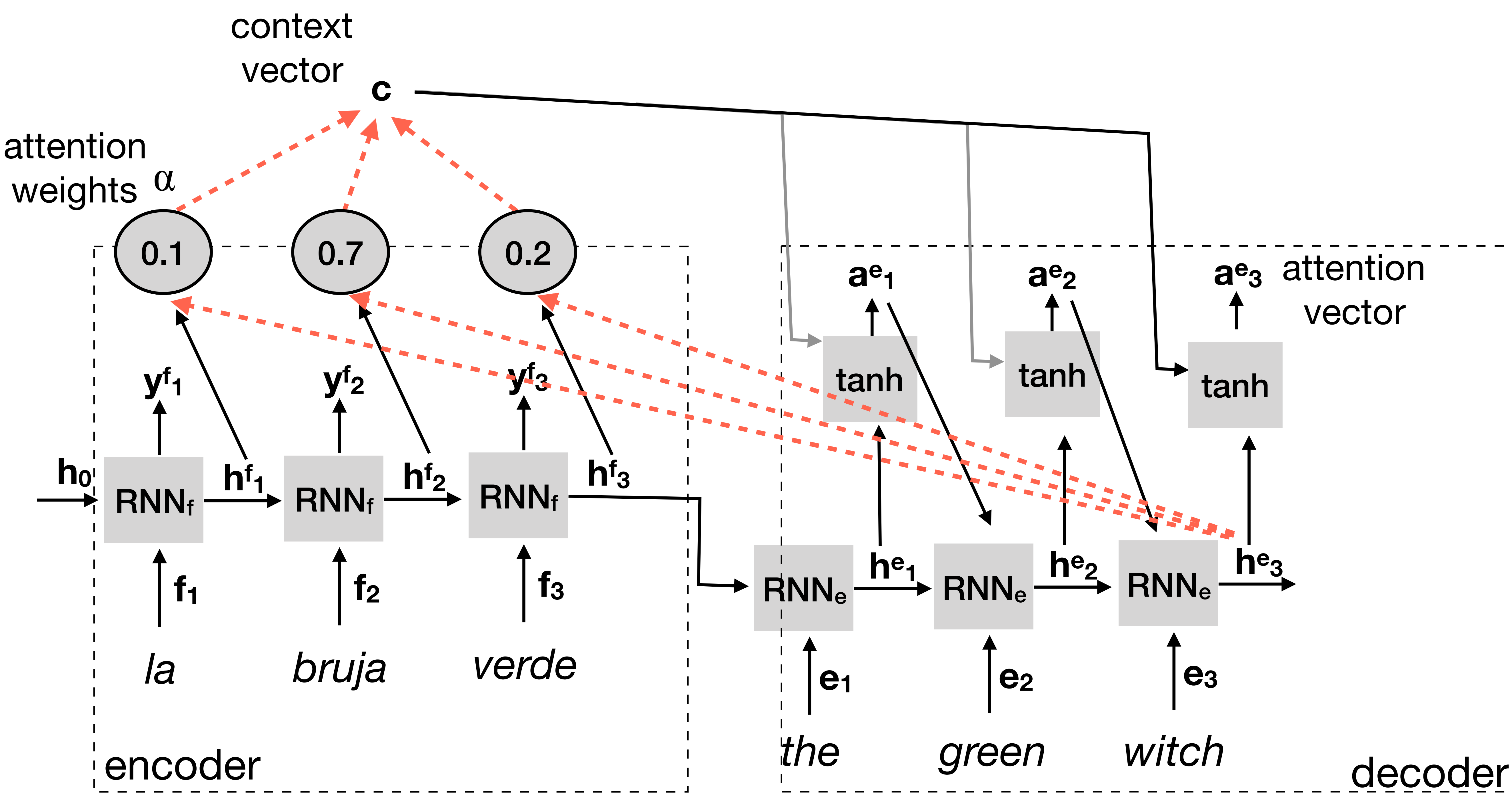
- Instead of throwing away hidden state vectors in the source sequence, maintain a sequence of these vectors.

- For each time step during decoding, select which positions in the source sentence contain the most relevant information.

    - Compute a **context vector** specific to each hidden representation.

    - Then combine the context vector with the current state to compute an **attention vector.**

Attention Mechanism



Attention Weights and Context Vector

- For each position in the target t, with hidden vector h_t
  - find **attention weights** for each context vector h_s

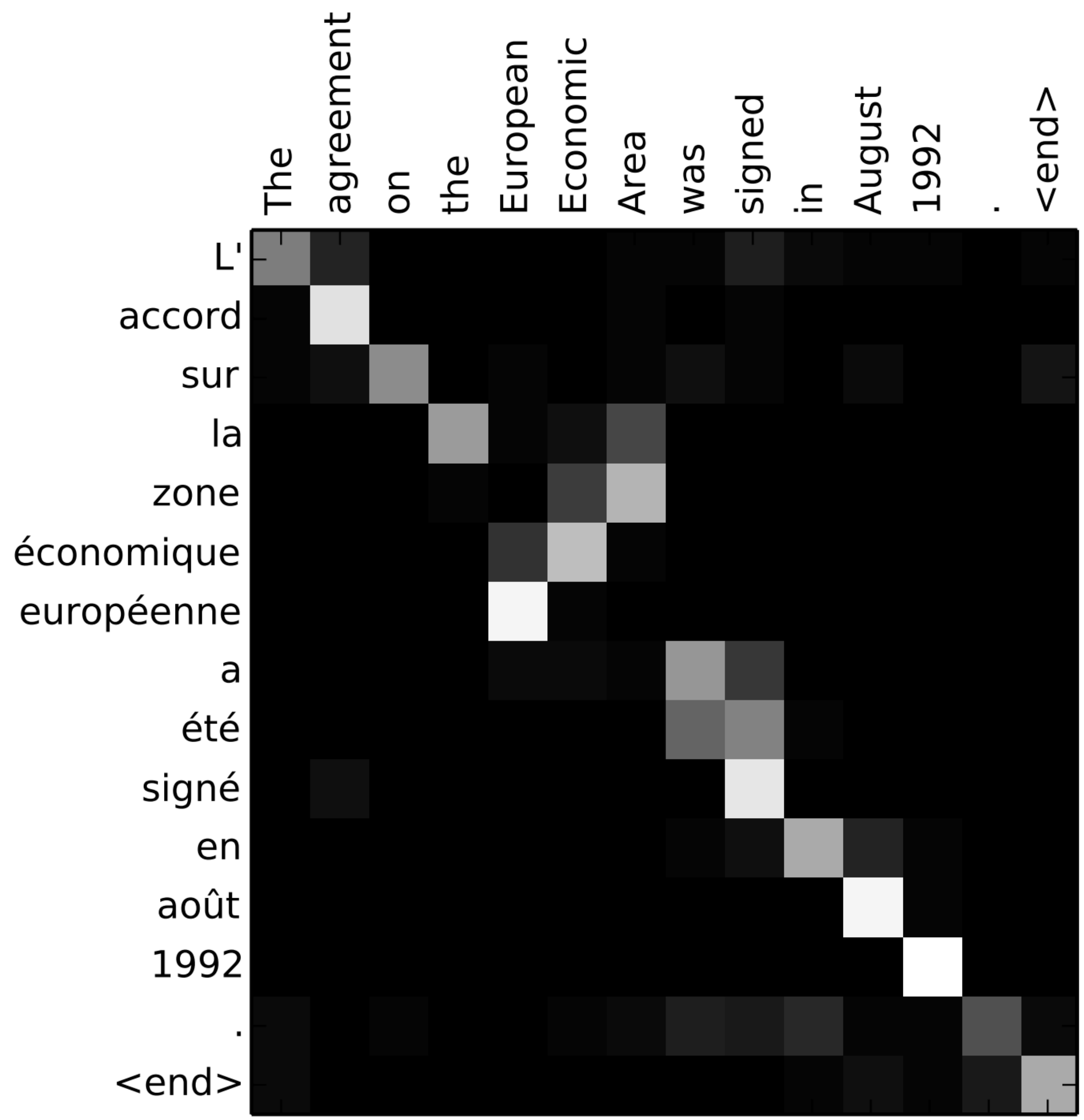
$$\alpha_{st} = \frac{exp(score(h_t, h_s))}{\sum_{s'} exp(score(h_t, h_{s'}))}$$

$$score(h_t, h_s) = h_t^\top W h_s$$ **(Luong et al., 2015)**

- Then compute a single **context vector** for t.

$$\mathbf{c_t} = \sum_s \alpha_{st} \mathbf{h_s}$$

Attention Weights and Alignments



Bahdanau et al., 2015

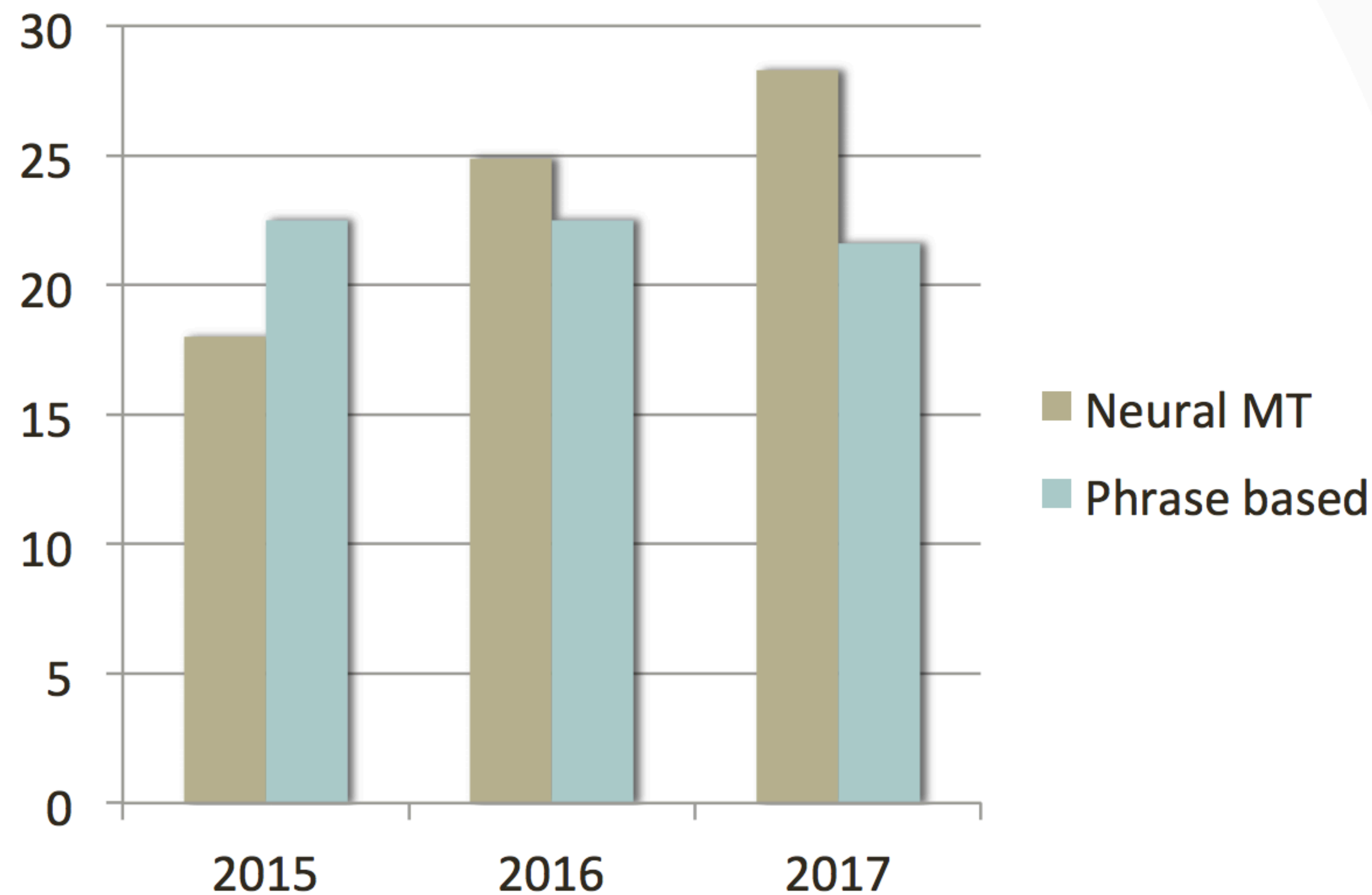
Attention Vector

- The context vector c_t and state h_t are combined to compute an attention vector.

$$\mathbf{a_t} = tanh(\mathbf{W_c}[\mathbf{c_t};\mathbf{h_t}])$$

- The attention vector is used to predict the next word and is also fed to the RNN in the next time step.

Phrase-based vs. Neural MT



Results from WMT (Workshop on Machine Translation)
German to English
2015: Montreal
2016 and 2017: Edinburgh

Slide from Kathy McKeown

See http://www.statmt.org/wmt17/ for latest results

Vanishing/Exploding Gradient Problem

- The function computed by the network looks like this:

$$\mathbf{s_3} = R(\mathbf{s_2}, \mathbf{x_3})$$

$$= R(R(\mathbf{s_1}, \mathbf{x_2}), \mathbf{x_3})$$

$$= R(R(R(\mathbf{s_0}, \mathbf{x_1}), \mathbf{x_2}), \mathbf{x_3})$$

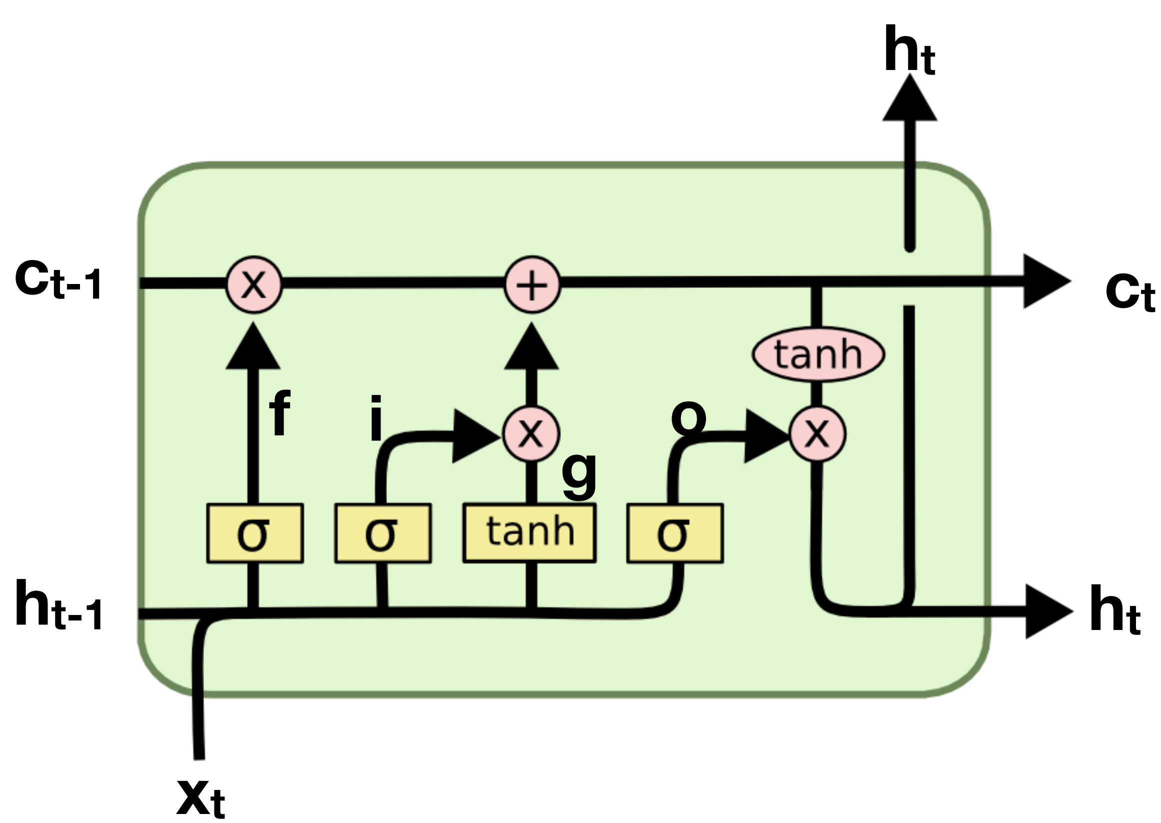
- The unfolded version of the network is a very deep feed-forward neural net.

- Because the gradients are multiplied as they propagate back through the network. Gradients < 1 become smaller and smaller. Gradients > 1 become larger and larger.

LSTM

Long Short-Term Memory [Hochreiter & Schmidhuber, 1997]

- Two states: "memory cell" **c** and working memory **h**.
- "gates" decide:
  - how much of the input to add to the memory
  - how much of current memory to forget



Acknowledgments

- Some material/examples from Yoav Goldberg, "Neural Network Methods for Natural Language Processing"

- Some slides from Kathy McKeown and Svetlana Lazebnik.