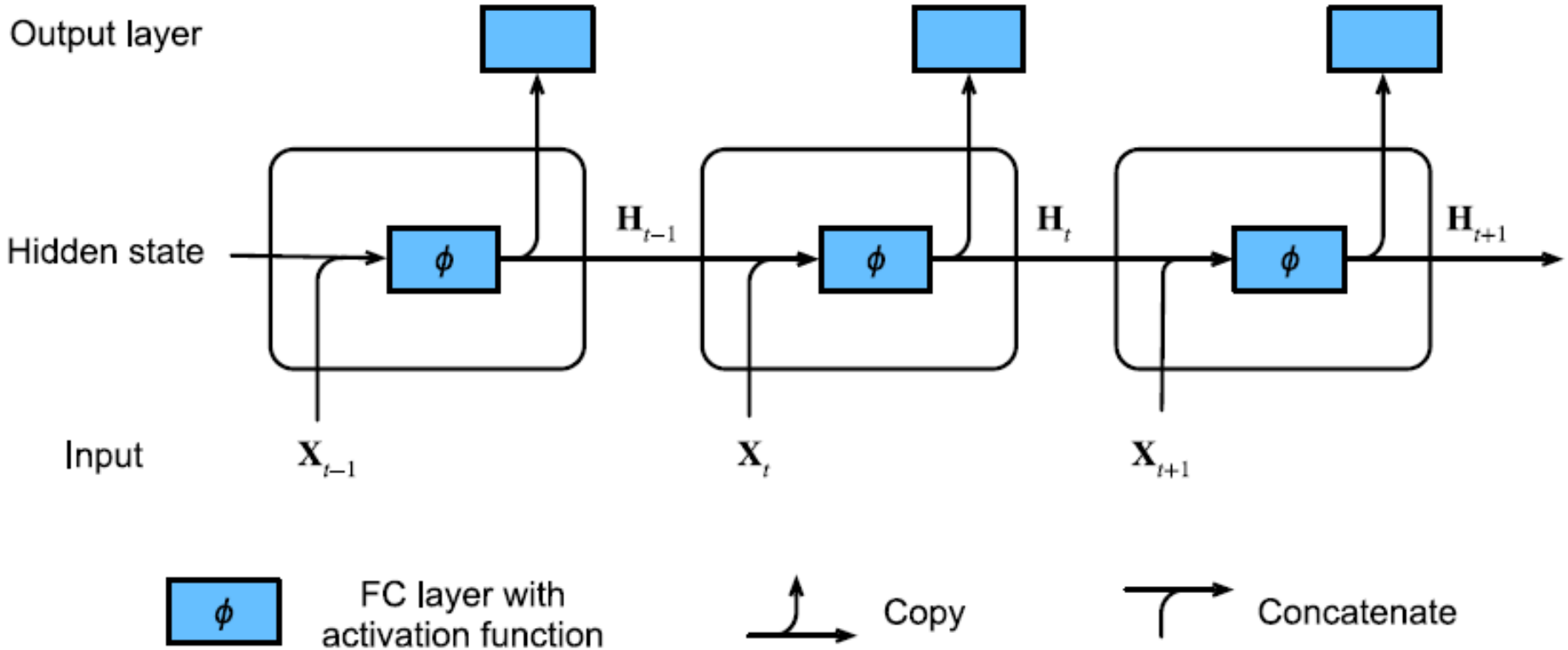
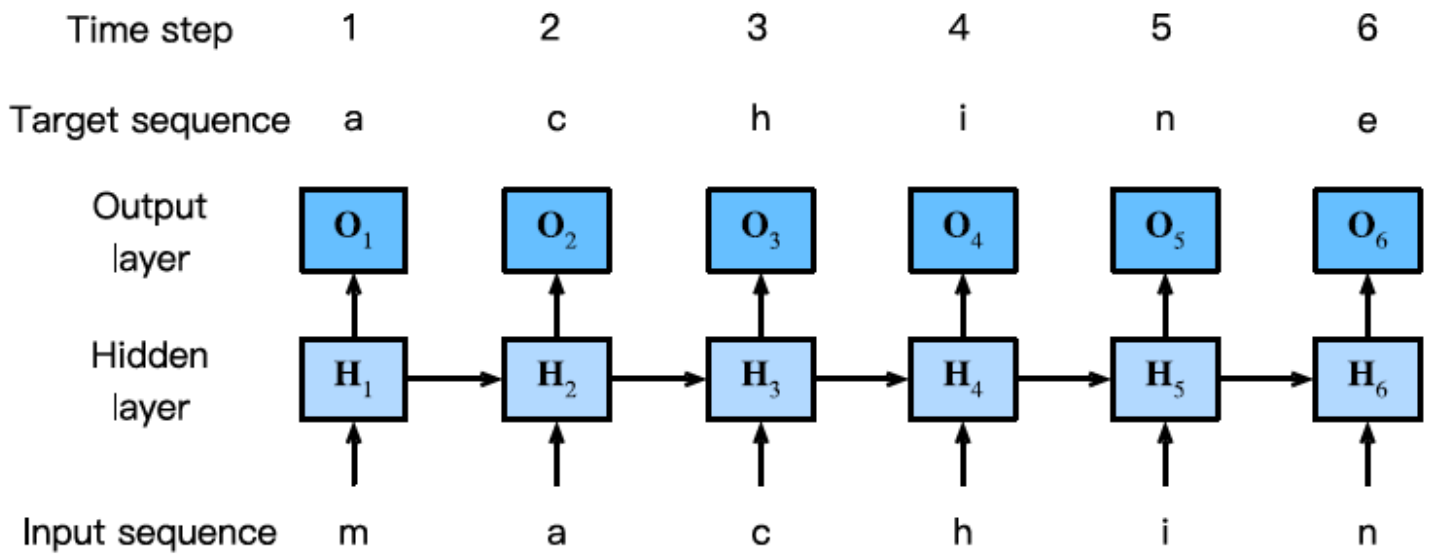
# RNN / Enc-Dec / Attention




An RNN with a hidden state.

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{\text{xh}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hh}} + \mathbf{b}_{\text{h}}).$$




A character-level language model based on the RNN. The input and target sequences are "machin" and "achine", respectively.

# Decoding / Generating Text

Once a language model has been learned, we can use it not only to predict the next token but to continue predicting each subsequent one, treating the previously predicted token as though it were the next in the input.

Sometimes we will just want to generate text as though we were starting at the beginning of a document.

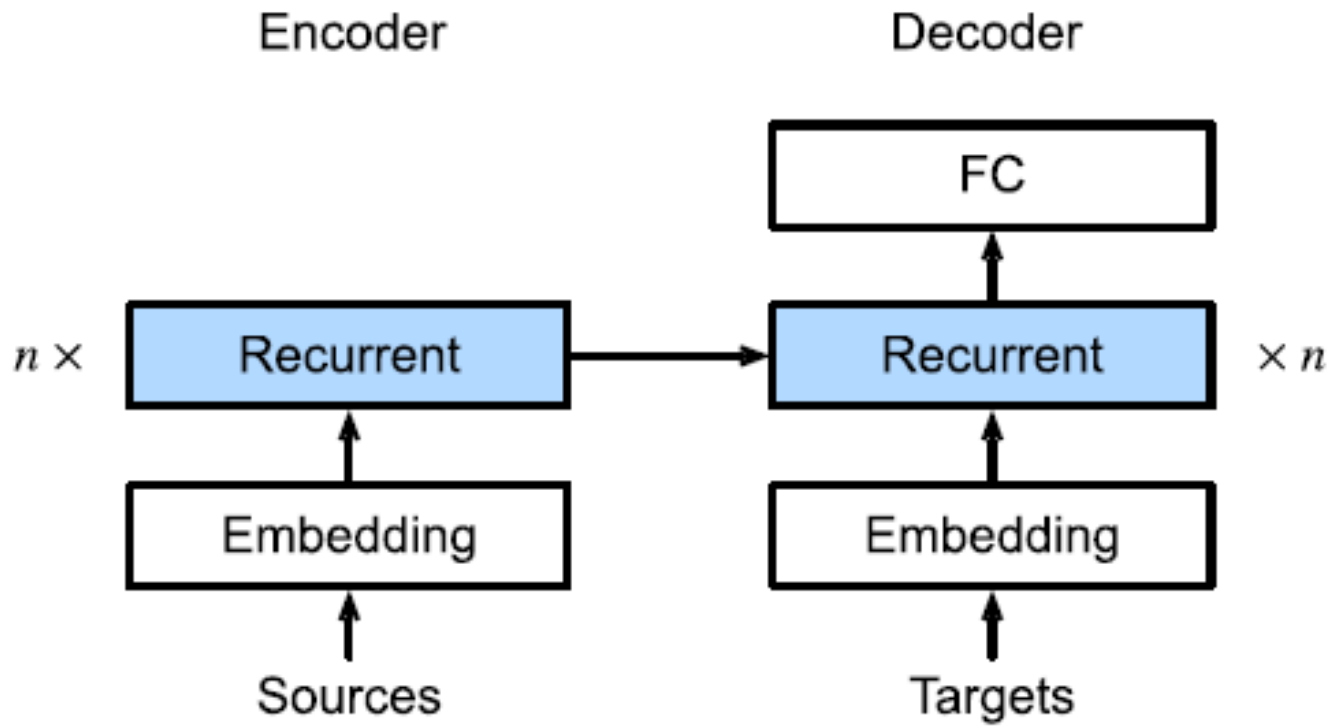
However, it is often useful to condition the language model on a user-supplied prefix.

For example, if we were developing an autocomplete feature for a search engine or to assist users in writing emails, we would want to feed in what they had written so far (the prefix), and then generate a likely continuation

# Encoder-Decoder Architecture: Sequence-to-Sequence Problems

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (pp. 3104–3112).

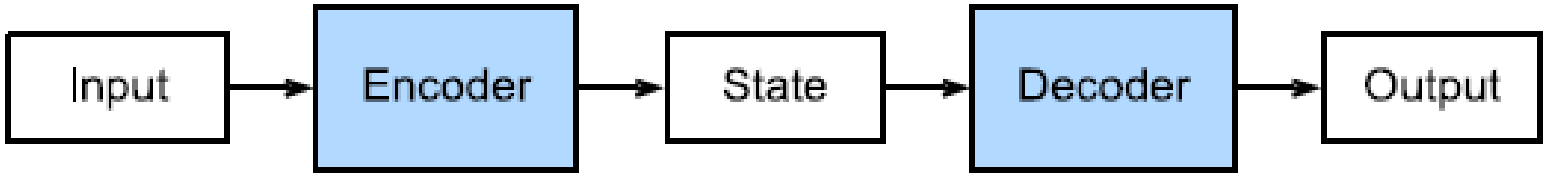
# Encoder–Decoder for Sequence-to-Sequence Learning




Layers in an RNN encoder–decoder model.

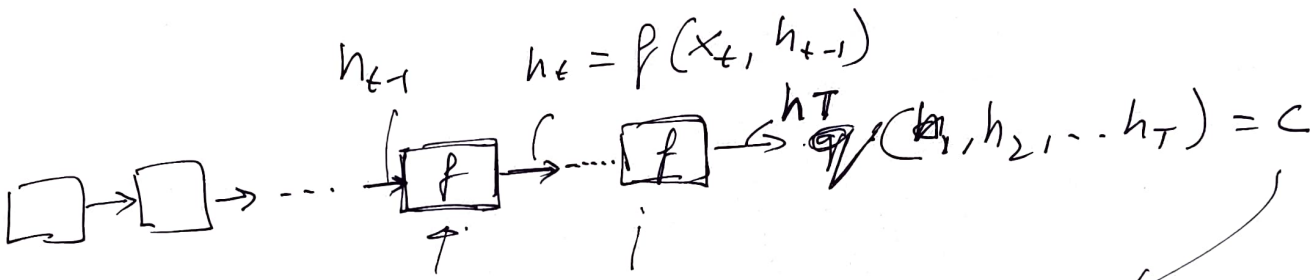
## Sequence-to-sequence problems like machine translation

- Inputs and outputs are of varying lengths that are unaligned
- Standard approach to handling this kind of mapping problems is to design an encoder–decoder architecture consisting of two major components
  - An encoder that takes a variable-length sequence as input
  - A decoder that acts as a conditional language mode
    - Takes in the encoded input and the leftwards context of the target sequence and predicting the subsequent token in the target sequence.



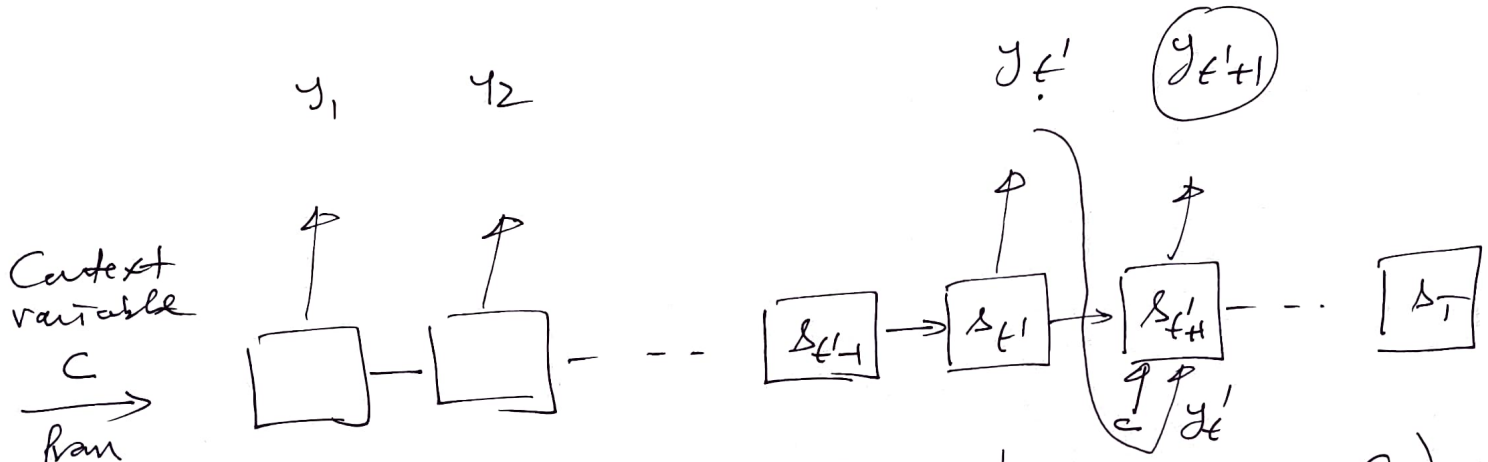

The encoder–decoder architecture.

# ENCODER




$$c = h_T \quad \text{specifically}$$

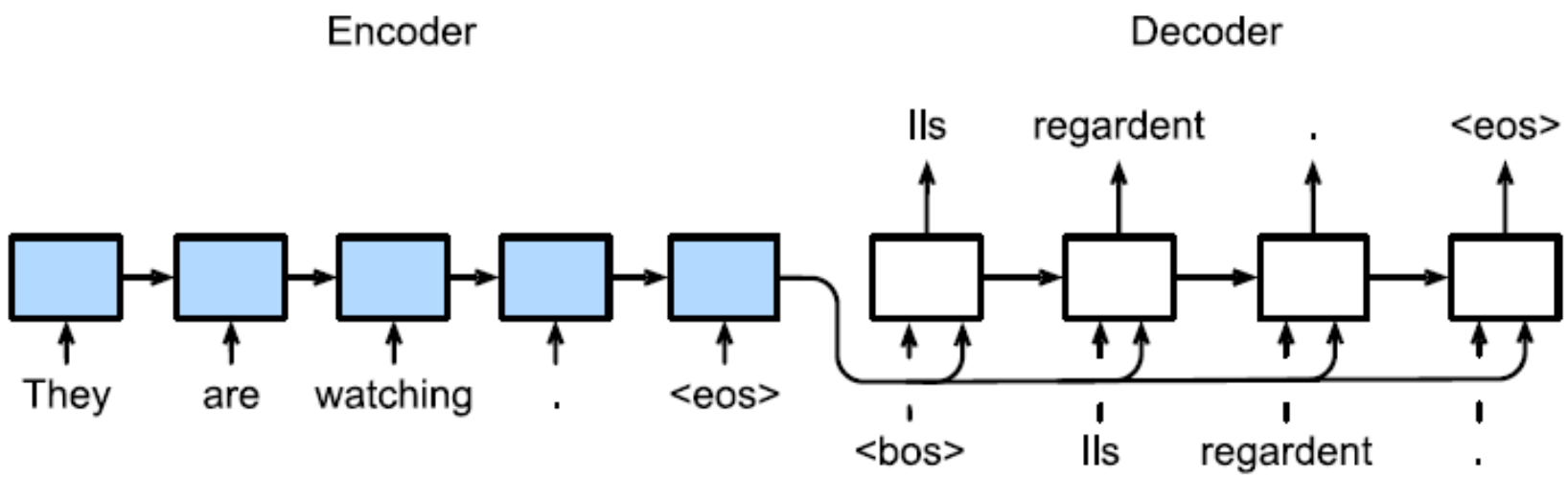
# DECODER



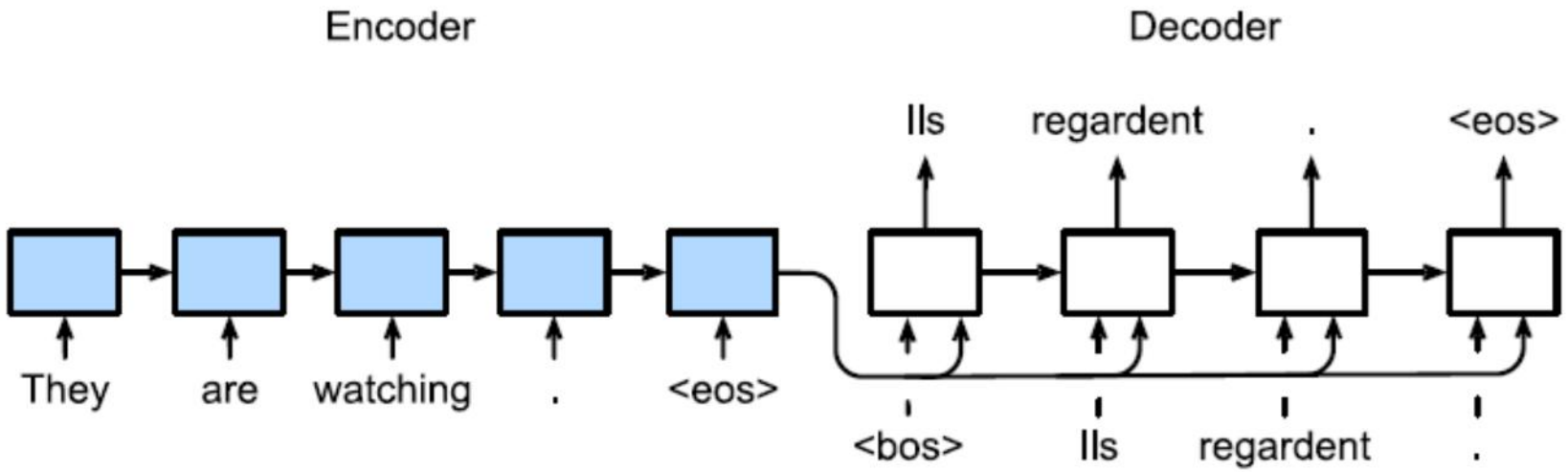

@ $t'+1$ : Need to get $P(y_{t'+1} \mid y_1, y_2, \cdots y_{t'}, C)$

$$s_{t'+1} = g(y_{t'}, C, s_{t'})$$

$s_{t'+1}$ → o/p layer → Softmax → $P(y_{t'+1} \mid y_1 y_2 \cdots y_{t'}, C)$



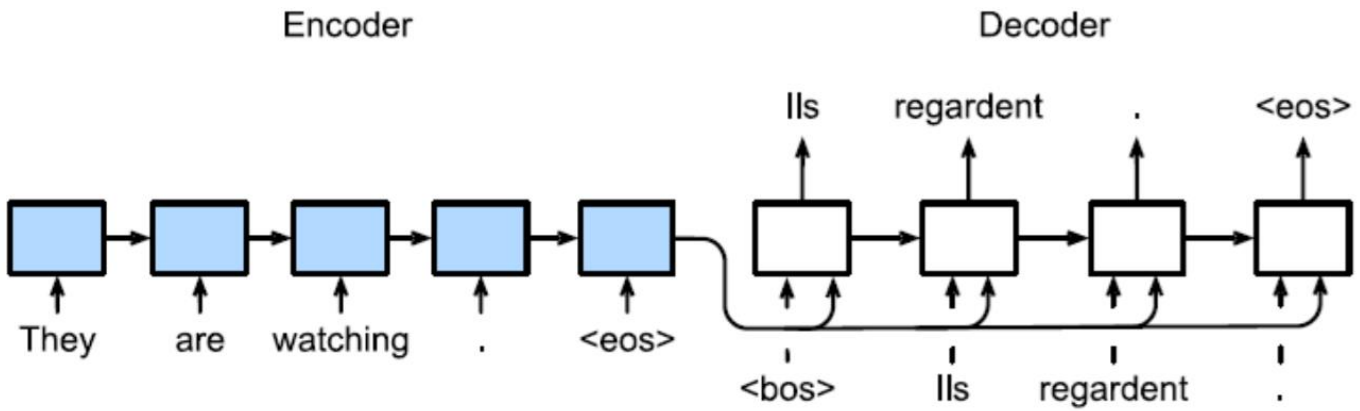

Sequence-to-sequence learning with an RNN encoder and an RNN decoder.




Sequence-to-sequence learning with an RNN encoder and an RNN decoder.

**Encoder**

- The encoder RNN will take a variable-length sequence as input and transform it into a **fixed-shape hidden state.**
- Later, we will see "attention" mechanisms, which allow us to access encoded inputs without having to compress the entire input into a single fixed-length representation.

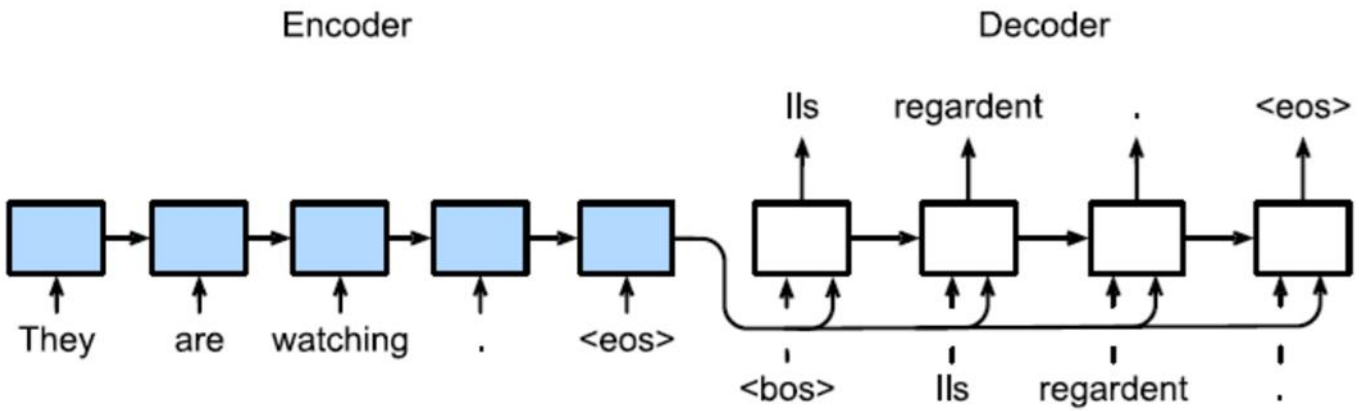



Sequence-to-sequence learning with an RNN encoder and an RNN decoder.

## Decoder

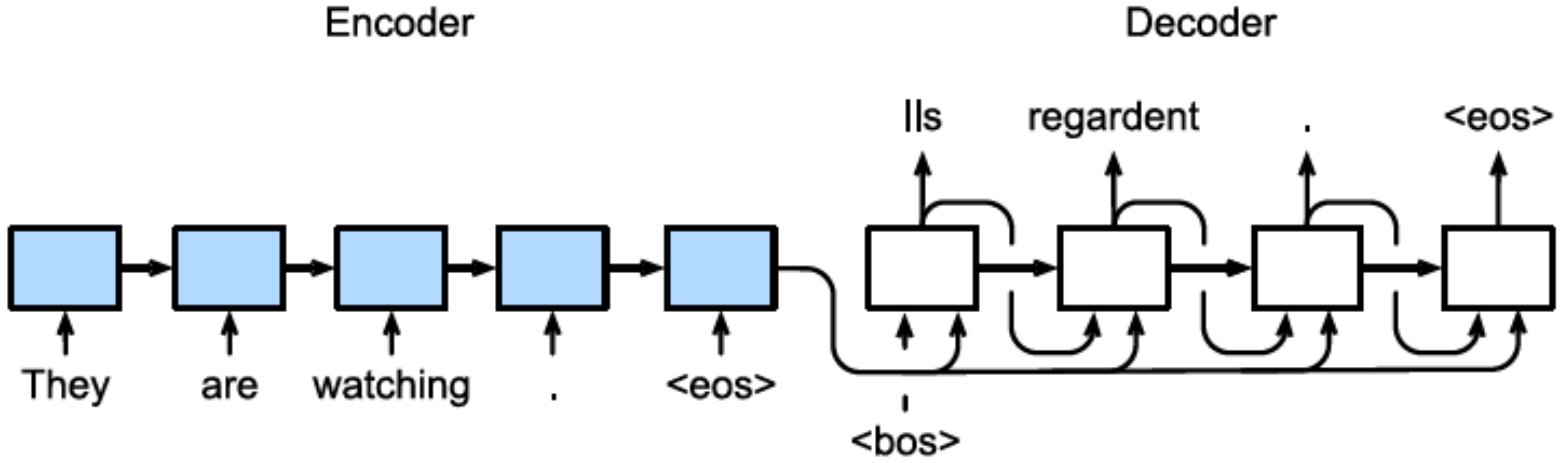
- A separate RNN || Generates the output sequence || one token at a time
- Predict each successive target token given both the input sequence and the preceding tokens in the output.
- Training: Decoder is conditioned upon the preceding tokens in the official “ground truth” label
- Test: Condition each output of the decoder on the tokens already predicted.
- Note: If we ignore the encoder, the decoder in a sequence-to sequence architecture behaves just like a normal language model

# TRAINING: TEACHER FORCING




Sequence-to-sequence learning with an RNN encoder and an RNN decoder.

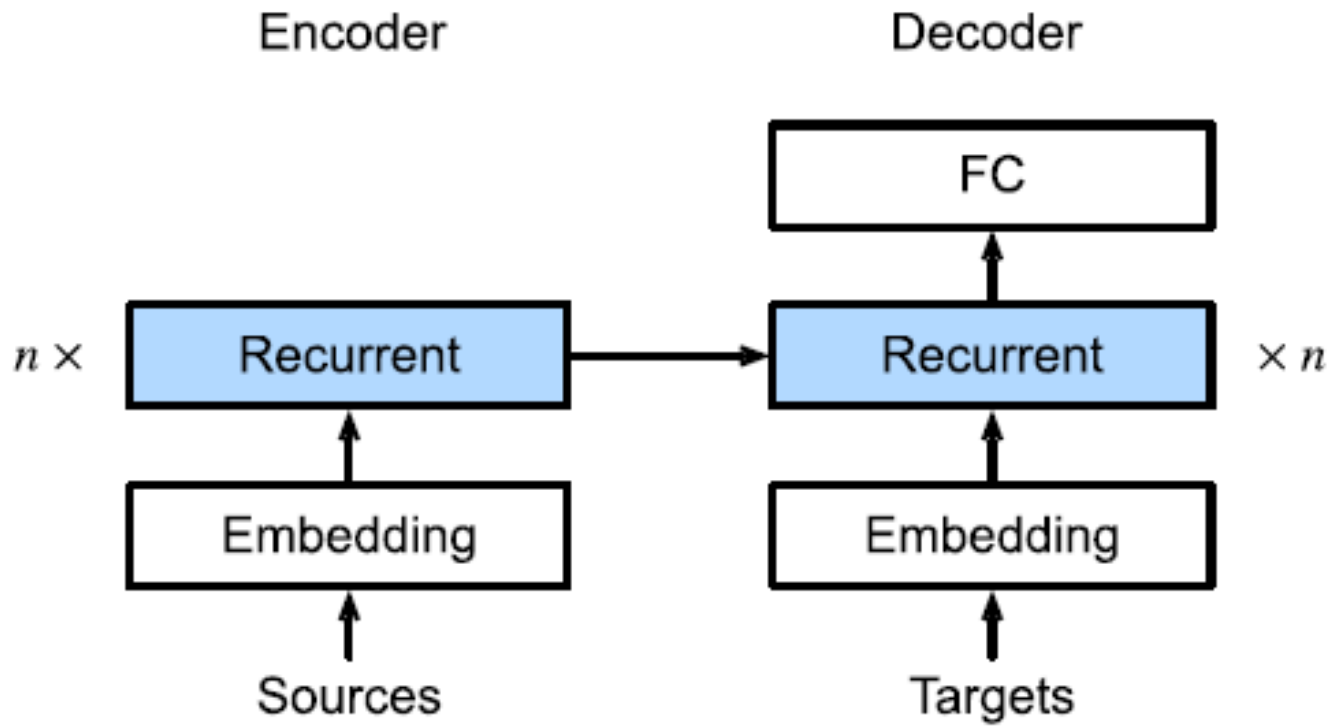
# TEST: PREDICTION




Predicting the output sequence token by token using an RNN encoder–decoder.

- Predict the output sequence at each step: Predicted token from the previous time step is fed into the decoder as an input.
- One simple strategy: Sample whichever token that has been assigned by the decoder the highest probability when predicting at each step.
- As in training, at the initial time step the beginning-of-sequence (“<bos>”) token is fed into the decoder.
- When the end-of-sequence (“<eos>”) token is predicted, the prediction of the output sequence is complete

# Encoder–Decoder for Sequence-to-Sequence Learning




Layers in an RNN encoder–decoder model.

Thank you !!