COE548: LARGE LANGUAGE MODELS

Topic: Sequence-to-sequence models

الجامعة اللبنانية الأميركية
Lebanese American University

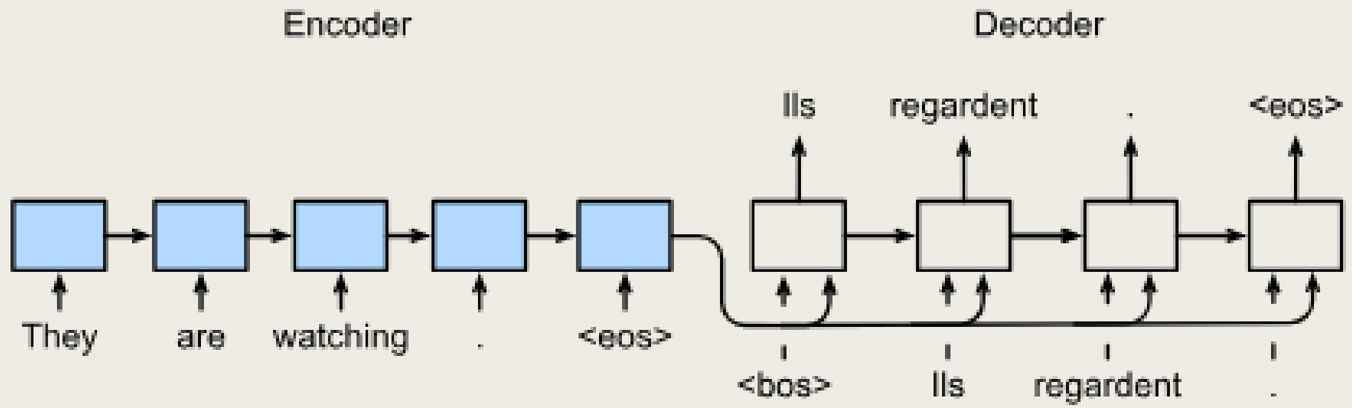
Outline

Sequence-to-sequence models

- Encoder/Decoder Motivation
- Encoder/Decoder Blocks
- RNNs as seq2seq Models
  - LSTMs

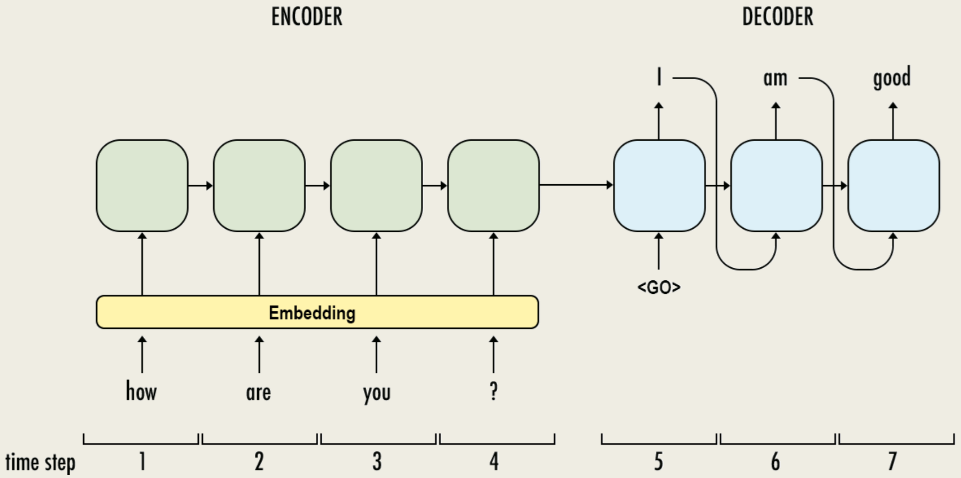
Sequence-to-sequence models

- Wikipedia definition: Sequence-to-sequence (seq2seq) models are a family of machine learning approaches used for NLP (language translation, image captioning, conversational models, text summarization, etc.). It uses sequence transformations: turns one sequence into another sequence.



Sequence-to-sequence models

- The seq2seq ML architecture consists of two fundamental components:
  - *An encoder*
    - The encoder processes the input sequence and transforms it into a fixed-size hidden representation.
  - *A decoder*
    - The decoder uses the hidden representation to generate output sequence.



Sequence-to-sequence models

- The encoder-decoder structure allows them to handle input and output sequences of different lengths, making them capable to handle sequential data.

- Seq2Seq models are trained using a dataset of input-output pairs, where the input is a sequence of tokens, and the output is also a sequence of tokens.

- The model is trained to maximize the likelihood of the correct output sequence given the input sequence.

Encoder Block

- The main purpose of the encoder block is to process the input sequence and capture information in a fixed-size context vector.
- Architecture:
  - *The input sequence is put into the encoder.*
  - *The encoder processes each element of the input sequence using neural networks.*
  - *Throughout this process, the encoder keeps an internal state, and the ultimate hidden state functions as the context vector that encapsulates a compressed representation of the entire input sequence.*
    - This context vector captures the semantic meaning and important information of the input sequence.
- The final hidden state of the encoder is then passed as the context vector to the decoder.

Decoder Block

- The decoder block is similar to encoder block. The decoder processes the context vector from encoder to generate output sequence incrementally.

- Architecture:
  - *In the training phase, the decoder receives both the context vector and the desired target output sequence (ground truth).*
  - *During inference, the decoder relies on its own previously generated outputs as inputs for subsequent steps.*

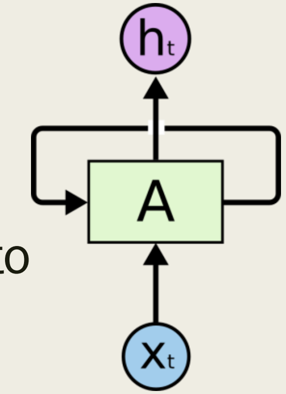
Decoder Block

- The decoder uses the context vector to comprehend the input sequence and create the corresponding output sequence.
- It engages in autoregressive generation, producing individual elements sequentially.
- At each time step, the decoder uses the current hidden state, the context vector, and the previous output token to generate a probability distribution over the possible next tokens.
- The token with the highest probability is then chosen as the output, and the process continues until the end of the output sequence is reached.

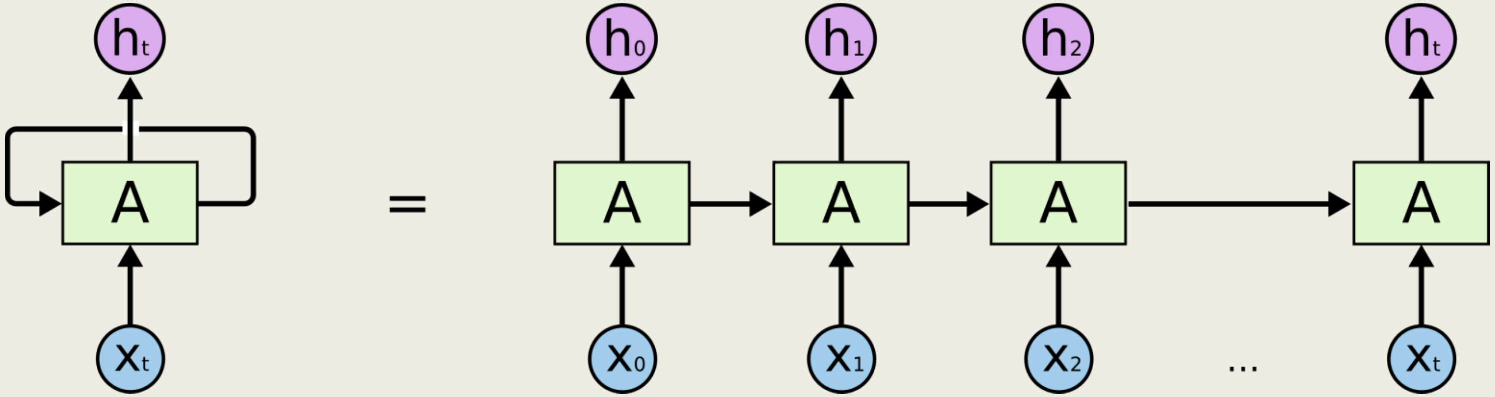
Seq2seq Models

- Before the arrival of Seq2Seq models, the machine translation systems relied on statistical methods and phrase-based approaches.
- That was not able to handle long-distance dependencies and capture global context.
- Seq2Seq models addressed the issues by leveraging the power of neural networks, especially recurrent neural networks (RNN).
- The concept of seq2seq model was introduced in the paper titled “Sequence to Sequence Learning with Neural Networks” by Google.
  - *Sutskever, I. "Sequence to Sequence Learning with Neural Networks." arXiv preprint arXiv:1409.3215 (2014).*
  - *https://jeremy-su1.github.io/images/2024-07-08-Seq2Seq-Learning/1409.3215v3.pdf*

Recurrent Neural Networks (RNNs) Recap



- Recurrent neural networks are designed to handle sequential data. They are networks with loops in them, allowing information to persist.
- The RNN looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next.
- A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.



Recurrent Neural Networks (RNNs) Recap

Mathematical breakdown of the RNN:

- The hidden layer, h_t, that is outputted at each time-step, t, from the RNN, given an input x_t, is given by the following equation

$$h_t = \varphi(W[x_t, h_{t-1}] + b)$$

Or better expanded

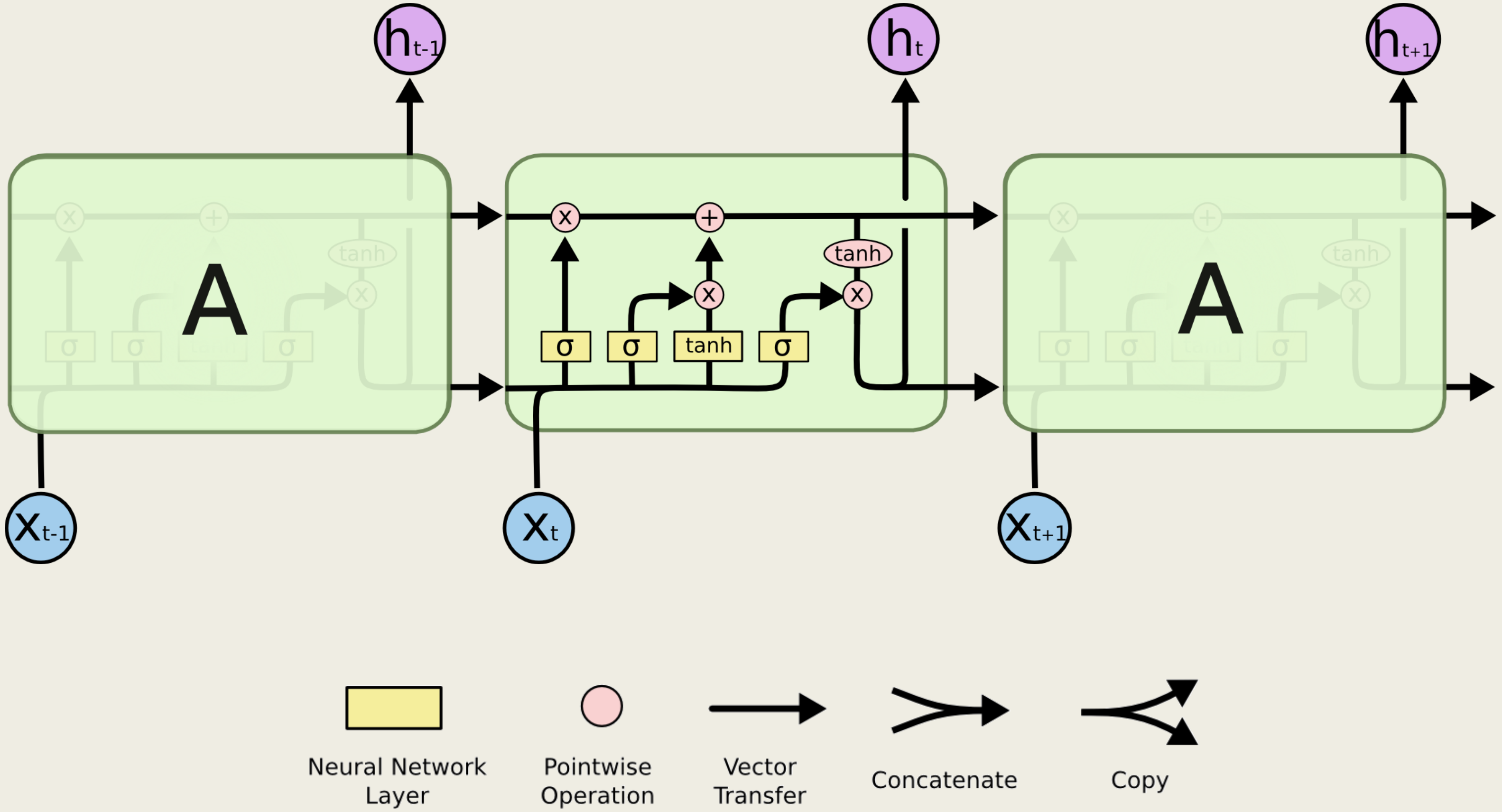
Can be thought of as the context vector.

$$h_t = \varphi(W_{hx}x_t + W_{hh}h_{t-1} + b)$$

- The hidden-state output can also be fed into a fully-connected NN layer to produce an output

$$y_t = \varphi\left(W_{hy}h_t + b\right)$$

Long Short-Term Memory (LSTM) Networks Recap

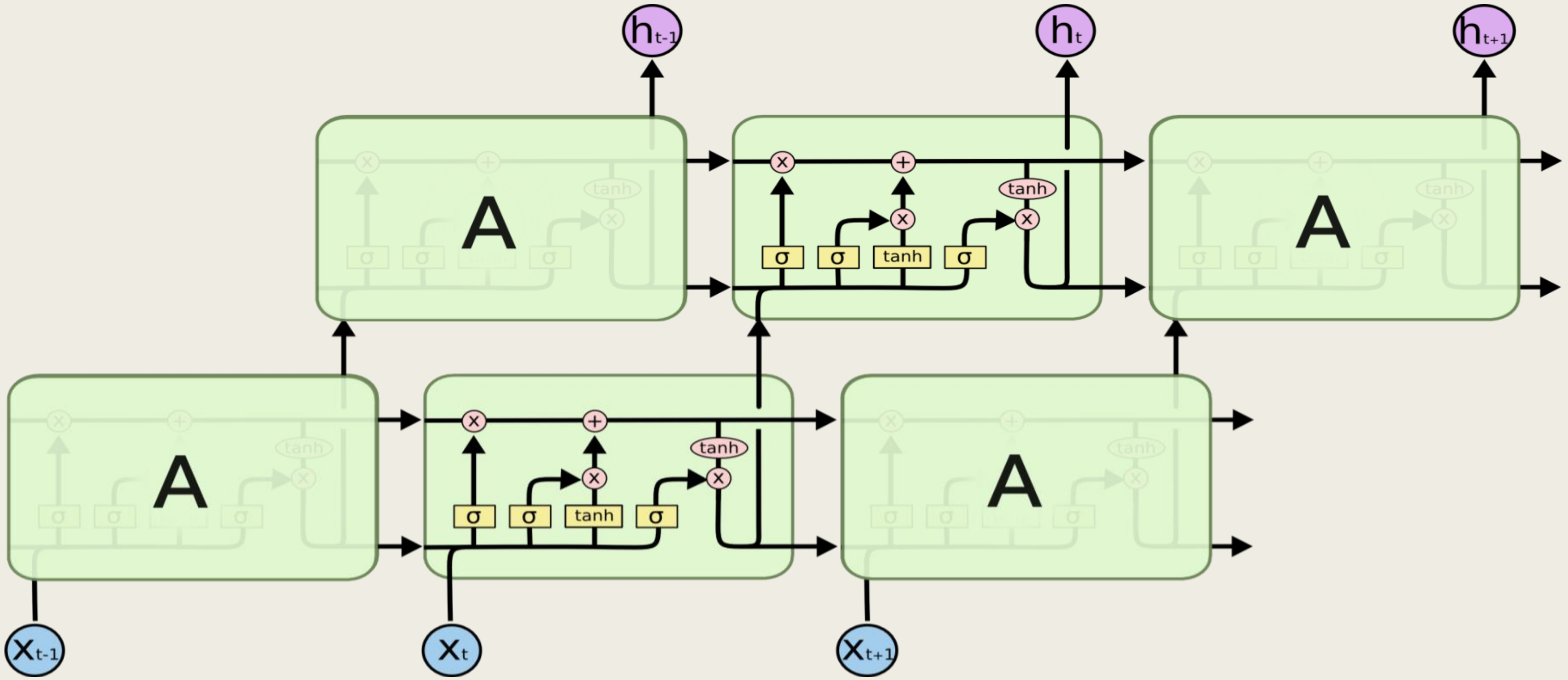


RNN Seq2seq Models

- Recurrent Neural Networks can easily map sequences to sequences when the alignment between the inputs and the outputs are known in advance.
- Although the vanilla version of RNN is rarely used, its more advanced version i.e. LSTM or GRU is used.
  - *This is because RNN suffers from the problem of vanishing gradient.*
- The tokens, and their respective word embeddings, are fed as input into the LSTM sequentially.
- At every time step, or every next token in the input sentence, the cell and hidden states of the LSTM of the previous token are also fed to the LSTM.

RNN Seq2seq Models

- The LSTM cells can be stacked for more complex representational capabilities. In the case of the stacked.



RNN Seq2seq Models: Encoder

- For stacked LSTM cells: the cell and hidden states of the LSTM in the first layer get fed as input to the LSTM in the second stacked layer.

- The last cell and hidden states of all the stacked LSTM layers are called the context vector.
  - *This is the encoded representational vector of the input sequence.*

- This concludes the “encoder” block of the RNN encoder-decoder seq2seq model.

RNN Seq2seq Models: Decoder

- The context vector from the encoder is used as the initial hidden state input in the decoder.
- Thus the decoder decodes the context vector by feeding it to a new set of LSTMs.
- The LSTMs of the encoder and decoder are different LSTMs (have their own separate weights and biases).

RNN Seq2seq Models: Decoder

- The first input into the decoder LSTMs also comes from an embedding layer (an embedding word vector).

- The first input should be a start-of-sentence <SOS> token (in original manuscript the actually used the <EOS> token to start the decoder).

- The output layer from the top stacked LSTM layer of the decoder model is fed to a fully connected NN layer, which helps transform them.

- The output size is the size of the vocabulary of the target sequence.

RNN Seq2seq Models: Decoder

- This output layer is fed through a softmax in order to select the output token.

- This selected output is what is used as input for the second sequence step in the decoder.

- This will keep happening until an <EOS> symbol is predicted by the decoder.

RNN Seq2seq Models: Training Phase

- During training, the true token is fed as input to the decoder after every token prediction.

- However during inference, the predicted output is used.

- Also, during training, the sequence is stopped whenever the true prediction is <EOS>, even if the decoder predicted a regular token.

RNN Seq2seq Models: Disadvantages

- Unrolling the LSTMS compresses the entire input sentence into a single context vector, which doesn’t work well for longer phrases (long-term dependencies) due to forgetting.

- Vanilla RNNs faced issues because they passed the short- and long-term memories through a single path.

- Even though LSTMs tried to circumvent this issue by introducing a separate path for long term memory, the issue of long-term forgetting, though kind of mitigated, is still an issue.