Sequence-to-sequence (encoder–decoder) networks

Statistical Methods in NLP 2
ISCL-BA-08

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2025

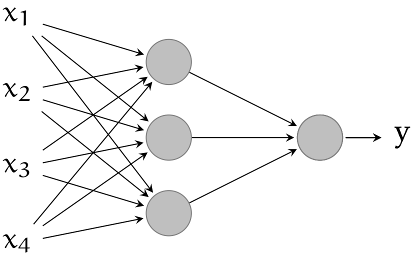
version: f1cc4d8 @2025-06-27

Encoder–decoder models

- All machine learning methods can be seen performing two tasks:
  - *encoding* the input into a useful representation
  - *decoding* the representation into the output
- In more traditional methods, encoding is ‘manual’, or external to the learning algorithm
- Modern deep learning methods include the encoder: they learn to build (multiple layers/hierarchies of) useful input representations

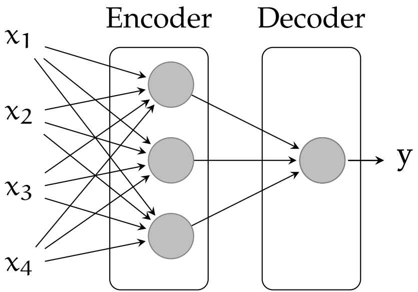
A simple encoder–decoder network

we can view any neural network as two parts



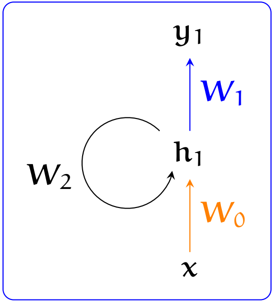
A simple encoder–decoder network

we can view any neural network as two parts

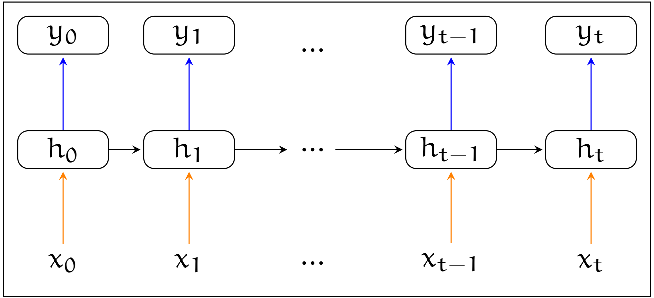


- The encoder encodes the input in hidden representations
- Decoder 'decodes' the encoded input to the output
- In computational linguistics, many tasks require encoding and decoding *sequences*

Recap: RNNs



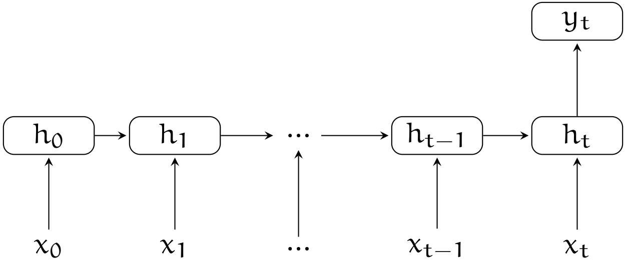
RNN



Unrolled RNN

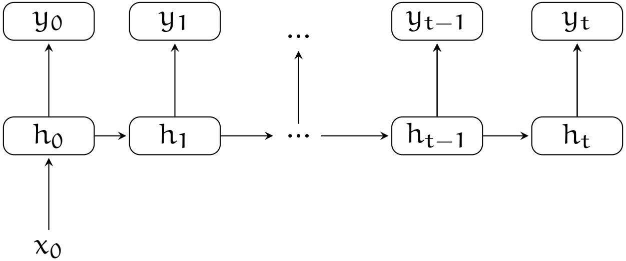
Uses of RNNs

Many-to-one (e.g., document classification)



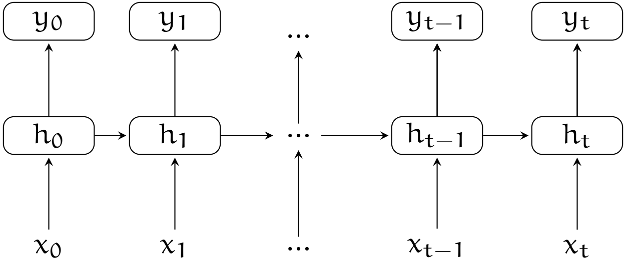
Uses of RNNs

One-to-many (e.g., caption generation)



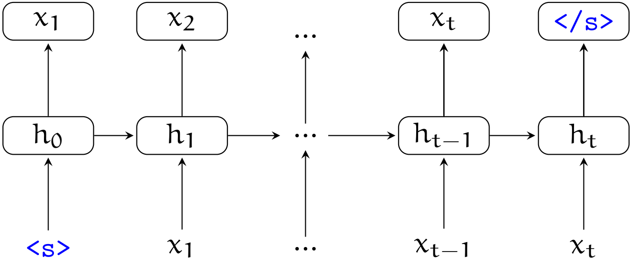
Uses of RNNs

Many-to-many (e.g., POS tagging, segmentation, ty recognition)



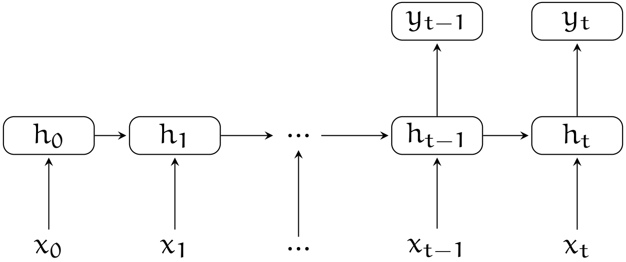
Uses of RNNs

Many-to-many – language models



Uses of RNNs

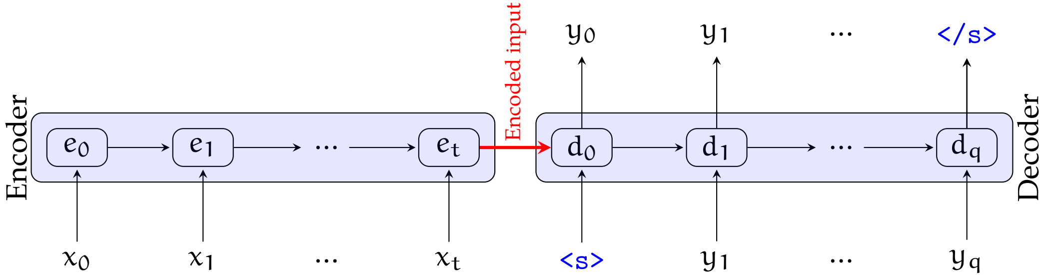
Many-to-many with a delay (e.g., machine translation, summarization, question answering, ...)



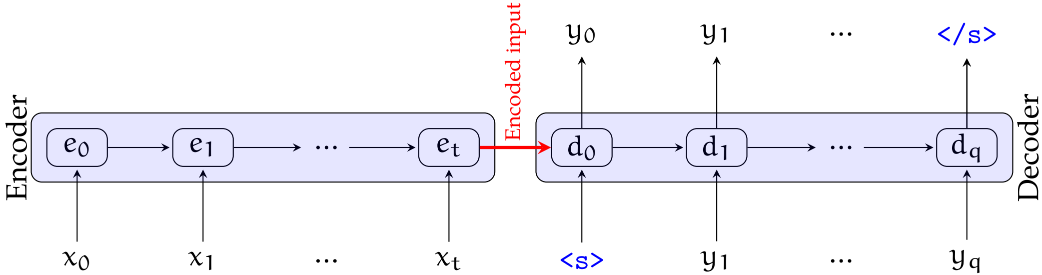
RNN problems, solutions, variations

- (Unfolded) RNNs can be deep (based on the length of the input), this results in
  - exploding gradients – solution: gradient clipping
  - vanishing gradients – solution: gated RNNs (to some extent)
- More generally, keeping relevant information over longer sequences are difficult – solution: attention (this lecture)
- RNNs condition the prediction in only one direction – solution: bidirectional RNNs
- It is also common to stack multiple layers of RNNs
- RNNs are inherently sequential, this prevents parallel processing

Sequence-to-sequence RNNs (seq2seq)



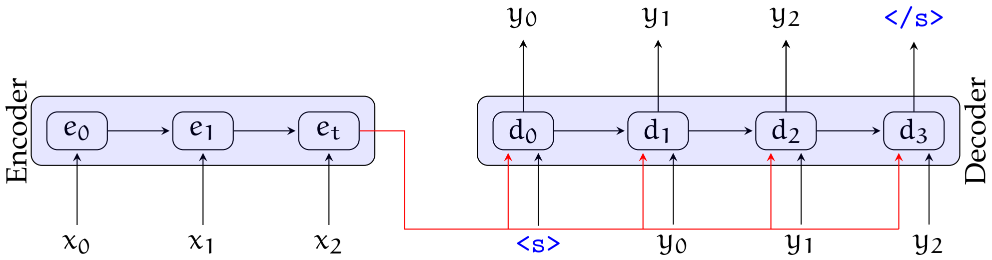
Sequence-to-sequence RNNs (seq2seq)



- Note that the decoder is a RNN language model
- Both input and output can be arbitrary length
- All information about the input is coded in a single encoding vector

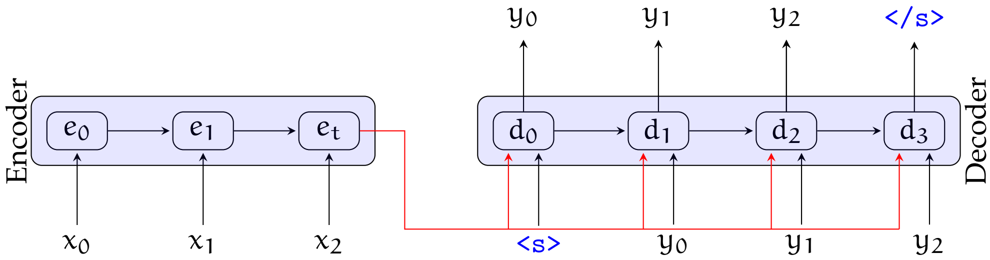
Sequence-to-sequence RNNs (seq2seq)

a simple improvement



Sequence-to-sequence RNNs (seq2seq)

a simple improvement



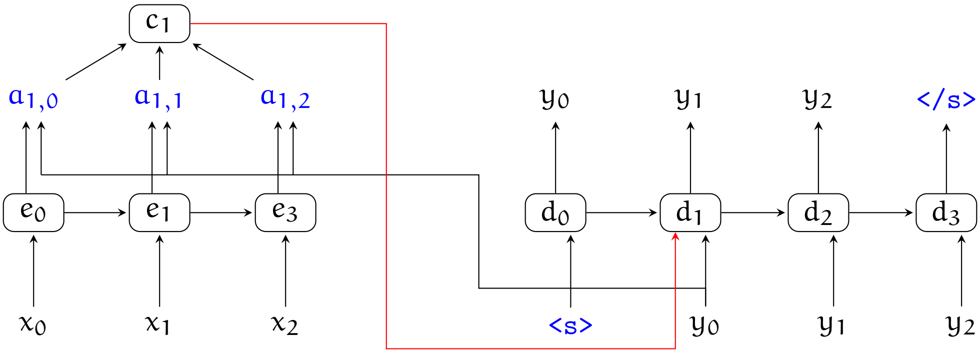
- Instead of passing the encoder output (context vector) to the first decoder state, pass it to all time steps
- Helps decoder to not to forget the encoder state, but early words in the encoder may not be represented well in the context vector

Attention: the general idea

- A naive solution could be passing all intermediate steps of the encoder to the decoder states (e.g., concatenate or average)
- However, in many problems, a part of the input is relevant for predicting the current output
- A common solution to the problem is *attention* mechanism
- Attention is about focusing into the relevant parts of the input
- In sequence-to-sequence problems, this corresponds to alignment

Sequence-to-sequence RNN with attention

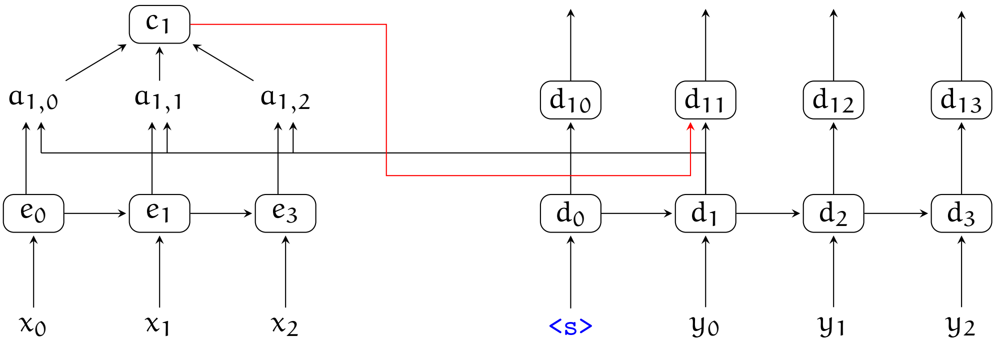
$$c_i = \sum_j a_{i,j} e_j$$



Sequence-to-sequence RNN with attention

another variation

$$c_i = \sum_j a_{i,j} e_j$$



Calculating attention weights

- The context vector is the sum of the encoder states, weighted by attention, $a_{i,j}$

$$\mathbf{c}_i = \sum_j a_{i,j} \mathbf{e}_j$$

- Typically the weights are normalized through *softmax* the result is an attention distribution

$$a_{i,j} = \frac{e^{f(\mathbf{d}_{i-1}, \mathbf{e}_j)}}{\sum_k e^{f(\mathbf{d}_{i-1}, \mathbf{e}_k)}}$$

- The attention function, $f(\cdot)$ computes the relevance of encoder state $\mathbf{e}_j$ to the decoder state $\mathbf{d}_i$

Some (common) attention functions

- Dot product:

$$f(\mathbf{d}_i, \mathbf{e}_j) = \mathbf{d}_i^T \mathbf{e}_j$$

- Generalized dot product:

$$f(\mathbf{d}_i, \mathbf{e}_j) = \mathbf{d}_i^T W_a \mathbf{e}_j$$

- Scaled dot product:

$$f(\mathbf{d}_i, \mathbf{e}_j) = \frac{\mathbf{d}_i^T \mathbf{e}_j}{\sqrt{k}}$$

- Additive attention:

$$f(\mathbf{d}_i, \mathbf{e}_j) = v^T \tanh(W_a d_i + U_a e_j)$$

Hard and soft attention

- The mechanism we described is called *soft attention*: The attention mechanism allows attending to more than one input in a weighted manner
- The case where the attention weights form a one-hot vector is called *hard attention*
- Soft attention is common, both because of its flexibility and ease of training (continuous / differentiable functions)

Attention and content addressable memory

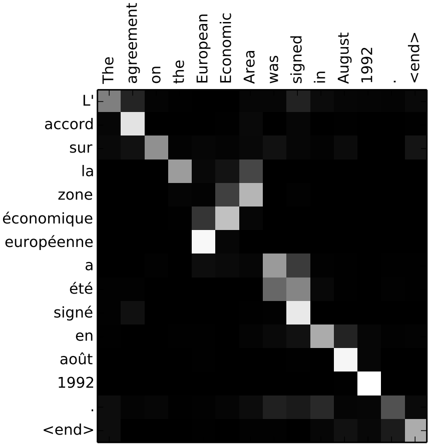
- In the literature, attention mechanism is often explained as a form of content-addressable (or associative) memory: decoder state is used to query the encoder states

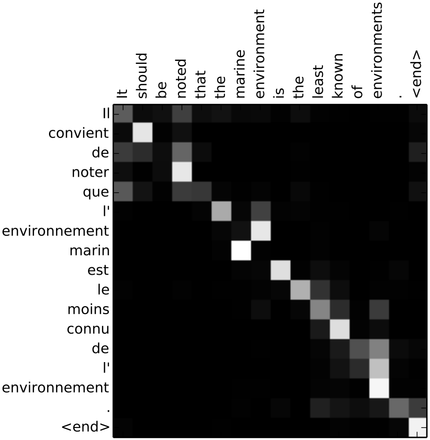
$$a_{i,j} = \frac{e^{f(\overset{\text{query}}{\mathbf{d}_{i-1}}, \overset{\text{key}}{\mathbf{e}_j})}}{\sum_k e^{f(\mathbf{d}_{i-1}, \mathbf{e}_k)}} \qquad \mathbf{c}_i = \sum_j a_{i,j} \overset{\text{value}}{\mathbf{e}_j}$$

- In this setting, *key* and *value* are the same
- In case of hard attention, the mechanism is equivalent to associative arrays (maps)

Example attention weights

machine translation (en–fr)





Example attention weights

caption generation

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Example attention weights

caption generation

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Summary

- Attention is a general mechanism to focus on certain parts of input
- It is also useful for generating ‘explanations’
- Attention is the basic mechanism behind the current state of the art models (transformers)
- Reading: Jurafsky and Martin (2025, Chapter 8)

Summary

- Attention is a general mechanism to focus on certain parts of input
- It is also useful for generating 'explanations'
- Attention is the basic mechanism behind the current state of the art models (transformers)
- Reading: Jurafsky and Martin (2025, Chapter 8)

Next:

- Self attention and transformers architecture (Reading: Jurafsky and Martin (2025, Chapter 9))

Additional reading, references, credits

- The translation example is from Bahdanau, Cho, and Bengio (2014)
- The image captioning examples are from Xu et al. (2015)

Additional reading, references, credits (cont.)

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Jurafsky, Daniel and James H. Martin (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released January 12, 2025. URL: `https://web.stanford.edu/~jurafsky/slp3/`.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). "Show, attend and tell: Neural image caption generation with visual attention". In: *International conference on machine learning*. PMLR, pp. 2048–2057.