Recurrent Neural Networks

Input to NLP

In other tasks, such as computer vision images can be scaled to a fixed size. However, in NLP, the input can be of variable length.

Figure 1: 50% center

"This is an example" => "Ti sa xml"

Sequence models

- Sequence models are deep learning models that are used for **time series** data or **sequential** data.
- Examples:
  - Speech recognition
  - Music generation
  - Sentiment classification
  - DNA sequence analysis

  - Machine translation
  - Video activity recognition
  - Name entity recognition
  - ...

Recurrent Neural Network model

- Notation:
  - x^t: Input at time t.
  - h^t: Hidden state at time t.
  - y^t: Output at time t.

$$h^t = f(h^{t-1}, x^t)$$

$$y^t = g(h^t)$$

Parameter sharing in RNNs

We can use the same weights for every time step.

$$h^t = f(h^{t-1}, x^t; W_f)$$

$$y^t = g(h^t; W_g)$$

Networks can be unrolled in time.

Simple RNN

$$y_i = f(W_{hy} h_i + b_y)$$

$$h_i = g(W_{hh} h_{i-1} + W_{xh} x_i + b_h)$$

Initially, h_0 is set to zero. f and g are non-linear activation functions.

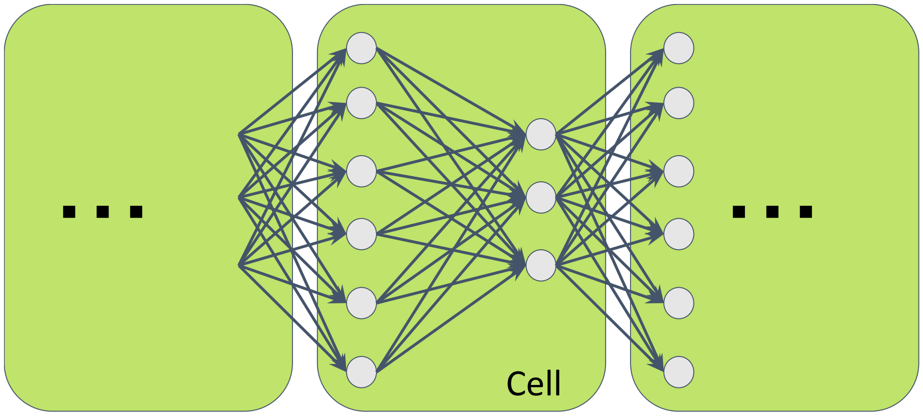


Figure 2: 50% center

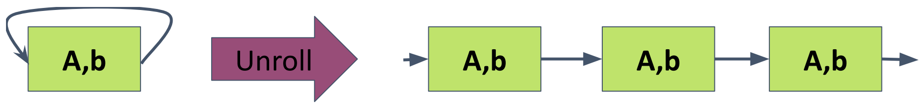


Figure 3: 50% center

RNNs for tagging

- Input: a sequence of words $x_1, \ldots, x_n$.
- Output: a sequence of tags $y_1, \ldots, y_n$.

The output can be computed directly using a softmax layer.

RNNs for language modeling

- Input: a sequence of words $x_1, \ldots, x_n$.
- Output: the probability of the next word x_{n+1}.

Again a softmax output layer can be used to compute the probability distribution.

RNNs for classification (acceptor)

Simple use of an RNN is as an acceptor. The final state of the RNN is used to classify the input sequence.

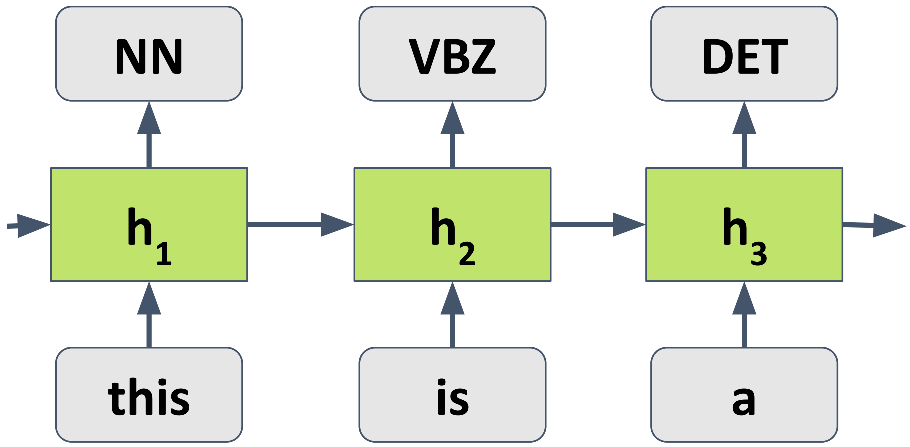


Figure 4: Tagging RNN

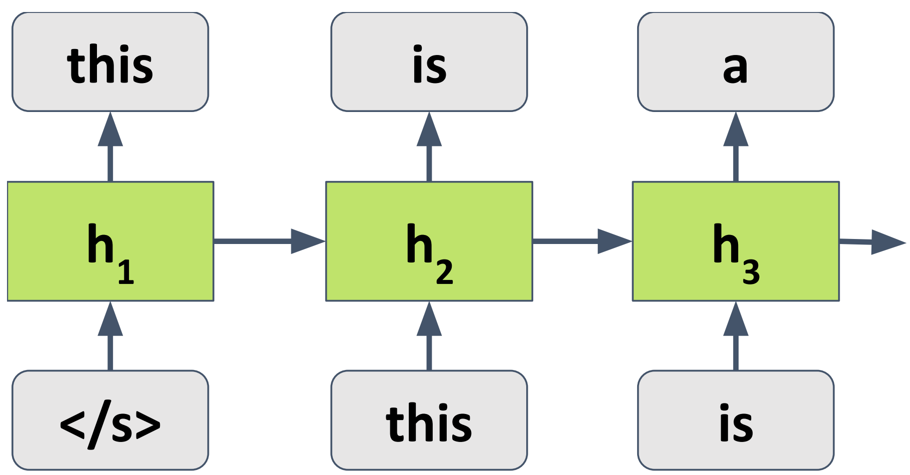


Figure 5: Language model RNN

$$y = f(W_{hy}h_n + b_y)$$

No prediction is made for other time steps.

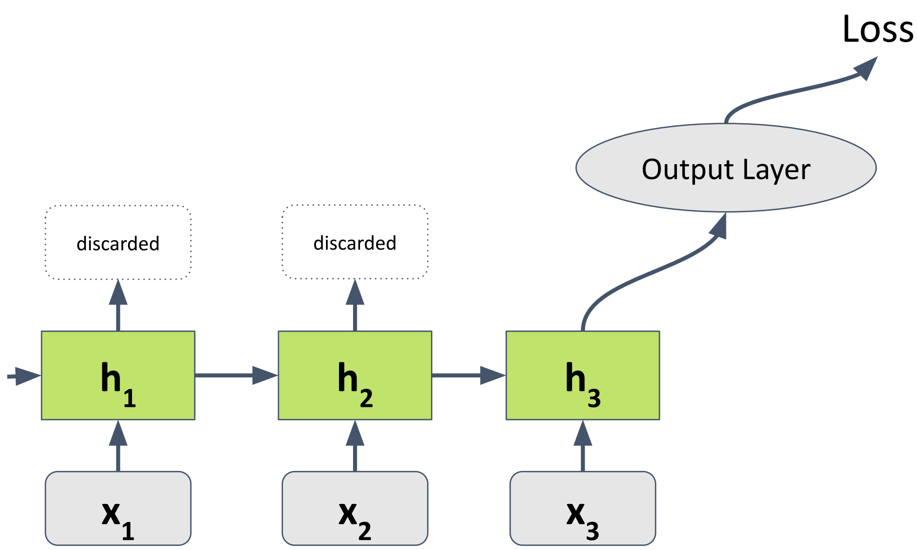


Figure 6: Acceptor RNN

RNNs for classification (attention)

Instead of discarding the hidden states, we can use them to compute a weighted sum of the hidden states.

$$y = f(\sum_{i=1}^{n} \alpha_i h_i)$$

$$\alpha_i = \frac{\exp(w_\alpha^T h_i)}{\sum_{j=1}^{n} \exp(w_\alpha^T h_j)}$$

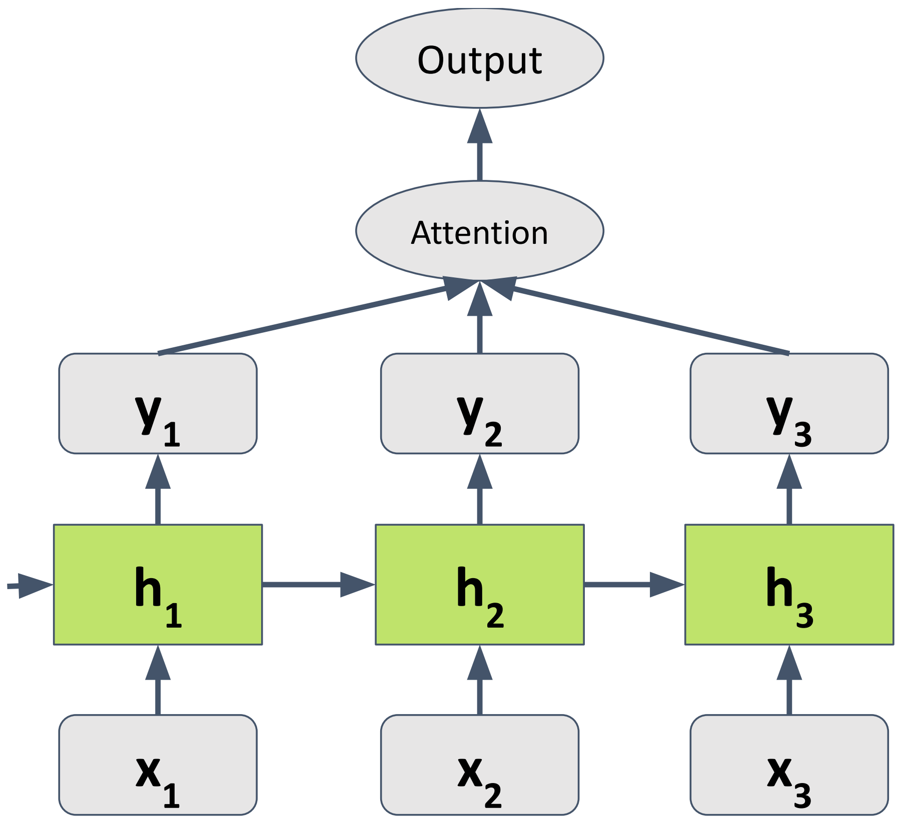


Figure 7: Attention RNN

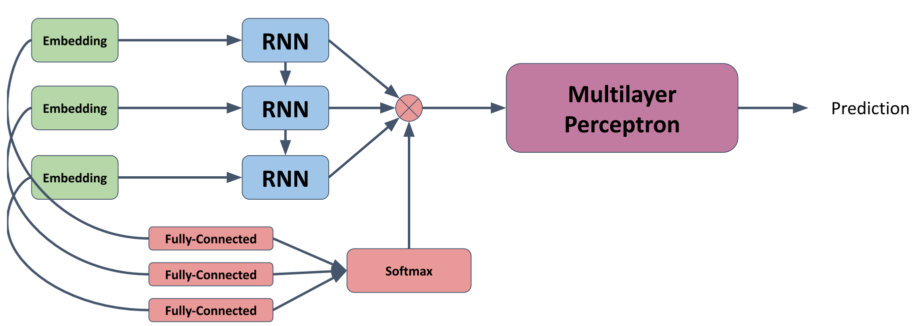


Figure 8: Embed, encode, attend, predict

Embed, encode, attend, predict

Read this article:

https://explosion.ai/blog/deep-learning-formula-nlp

Bidirectional RNNs

- In some cases, we want to use information from the future.
- We can use a bidirectional RNN to do this.
- The hidden state is computed from both the past and the future.

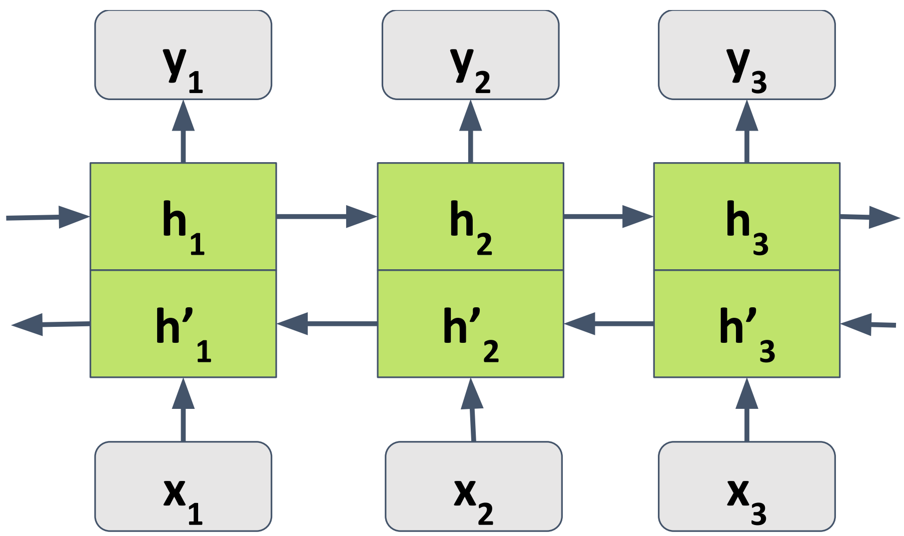


Figure 9: Bidirectional RNN

Bidirectional RNNs (formulae)

$$u_i = f(W_h u_{i-1} + W_x x_i + b_h)$$

$$v_i = f(V_h v_{i+1} + V_x x_i + c_h)$$

$$h_i = u_i + v_i$$

Backpropagation through time

- Gradients in a RNN are harder to compute than in a feedforward network.
- This is because the same weights are used at each time step.
- The gradients are summed over time steps.

Vanishing gradients

- The gradients can become very small.
- This is because the gradients are multiplied by the same weights at each time step.
- This is called the vanishing gradient problem.

Exploding gradients

- Alternatively, the gradients can become very large (exploding gradients).
- One way to deal with exploding gradients is to clip the gradients.
- If the gradient norm is larger than a threshold, the gradients are scaled down.

Gating

- Gating is a way to control the flow of information in a RNN.
- Gating can be used to deal with the vanishing gradient problem.

What is a gate?

- A gate is a function that takes two inputs and produces an output.
- The output is the pairwise product of the inputs.

$$g = x \odot y$$

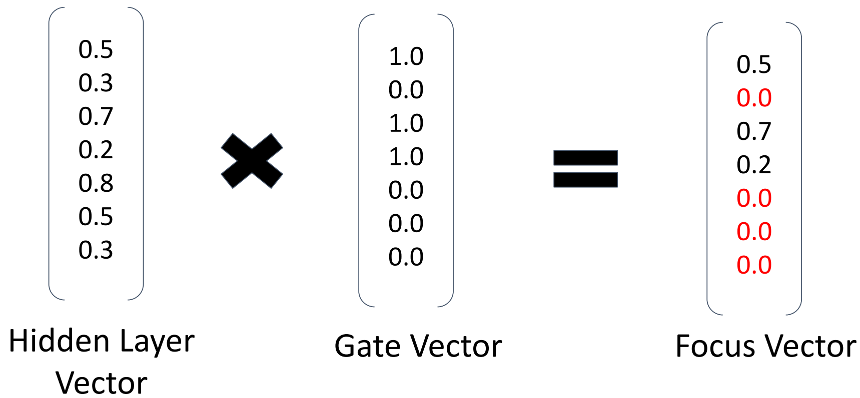


Figure 10: Gating

Gated Recurrent Unit (GRU)

- The GRU is a gated RNN.
- It has two gates: an update gate and a reset gate.
- The update gate controls how much of the previous state is kept.
- The reset gate controls how much of the previous state is forgotten.

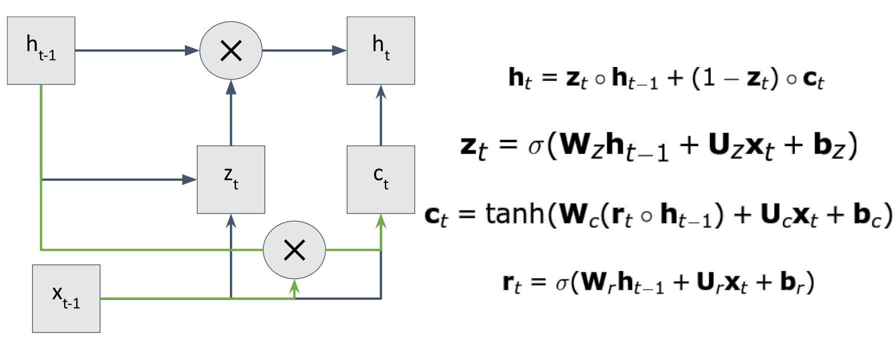


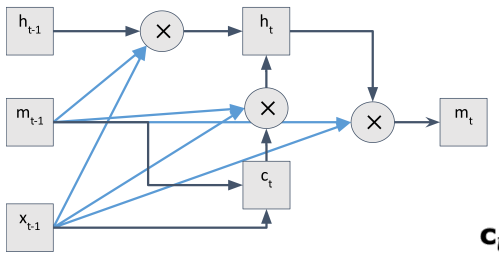
Figure 11: GRU

for reference only

Long Short-Term Memory (LSTM)

- The LSTM is a more complex gated RNN.
- It has three gates: an input gate, an output gate and a forget gate.
- The input gate controls how much of the input is kept.

- The output gate controls how much of the state is output.
- The forget gate controls how much of the state is forgotten.



$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{m}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{m}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{m}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{f}_t \circ \mathbf{h}_{t-1} + \mathbf{i}_t \circ \mathbf{c}_t$$
$$\mathbf{c}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{m}_{t-1} + \mathbf{b}_c)$$
$$\mathbf{m}_t = \mathbf{o}_t \circ \tanh(\mathbf{h}_t)$$

for reference only

GRUs and LSTMs

- In practice, GRUs and LSTMs perform similarly.
- GRUs are simpler and faster to train.
- LSTMs are more flexible and can be used in more complex situations.

Disadvantages of RNNs

- Training can be slow due to sequential processing.
  - They cannot be easily parallelized.
  - GPUs cannot be applied effectively
- Difficulty in capturing long-range dependencies.
- RNNs have been replaced by Transformer models in almost all applications.
  - Transformers do not naturally capture sequential information, but use positional encodings instead.

State Space Machines

- State Space Machines (SSMs) are a new type of sequence model that combine the benefits of RNNs and Transformers.
- SSMs can capture long-range dependencies and can be trained in parallel.
- SSMs have shown promising results in various NLP tasks.

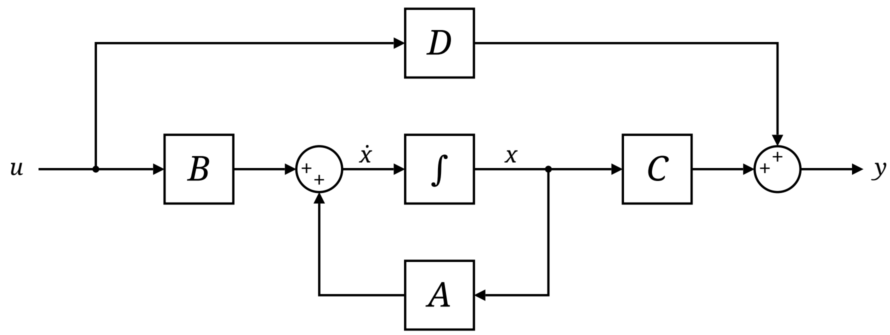


Figure 12: SSM

State Space Machines (SSMs)

$$\dot{x} = A(t)x(t) + B(t)u(t)$$

$$y(t) = C(t)x(t) + D(t)u(t)$$

- $x(t)$: hidden state
- $u(t)$: input
- $y(t)$: output
- Matrix A, B, C, D are time-dependent

Summary

- RNNs are sequence models that can handle variable-length input.
- RNNs can be used for various NLP tasks such as tagging, language modeling, and classification.
- Gating mechanisms such as GRUs and LSTMs help mitigate the vanishing gradient problem.
- RNNs have been largely replaced by Transformer models in NLP.
- State Space Machines (SSMs) are a new type of sequence model that combine the benefits of RNNs and Transformers.