Recurrent Neural Networks

ML Instruction Team, Fall 2022

CE Department
Sharif University of Technology

Recurrent Neural Network

- A variant of the conventional feed-forward artificial neural networks to deal with sequential data
- Hold the knowledge about the past (Have memory!)

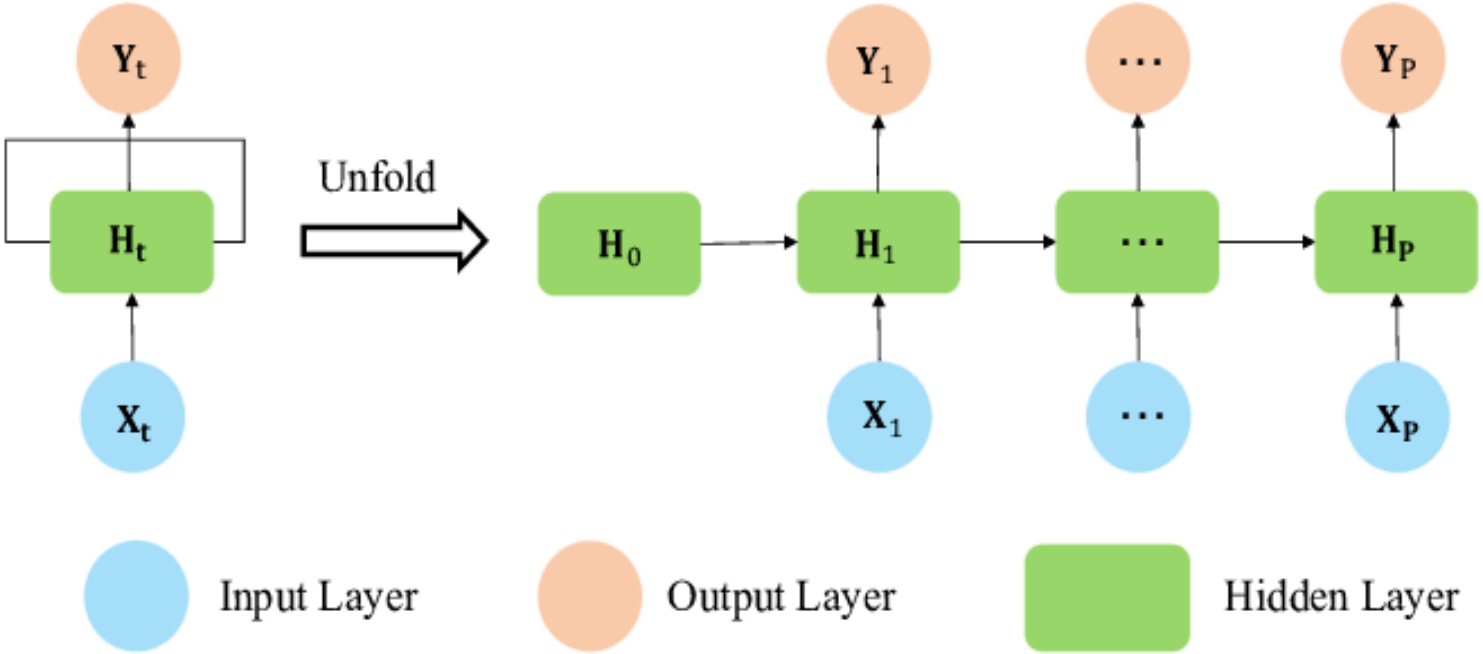


Figure: The folded and unfolded structure of recurrent neural networks, source

An Online Example!

Figure: SpeechBrain, source

- SpeechBrain: Automatic Speech Recognition
  - Model Input: Audio
  - Model Output: Text
  - An example (HelloWorld.mp3)

```
HELLO WORLD
```

- Interested? Test it yourself:
  https://huggingface.co/speechbrain/asr-crdnn-rnnlm-librispeech

Fake Wikipedia Page!

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

Figure: In case you were wondering, the yahoo url in the generated Wikipedia page doesn't actually exist, the model just hallucinated it.

Fake Algebraic Geometry Book!

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciely in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Figure: A sample of a recurrent network. The network is trained on the raw Latex source file of a book on algebraic geometry. Amazingly, the resulting sampled Latex almost compiles!

Fake C Code!

```c
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Figure: This time the network is trained on the linux source code. Notice the comments, pointer notation and brackets in the above code. What are the code errors?

The Effectiveness of Recurrent Neural Networks

- All previous examples were generated blindly by recurrent neural network with simple architectures.
- Interested? Take a look at the source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Modeling Series

- In many situations one must consider a series of inputs to produce an output.
  - Outputs too may be a series
- Examples...?

Example 1: Speech Recognition

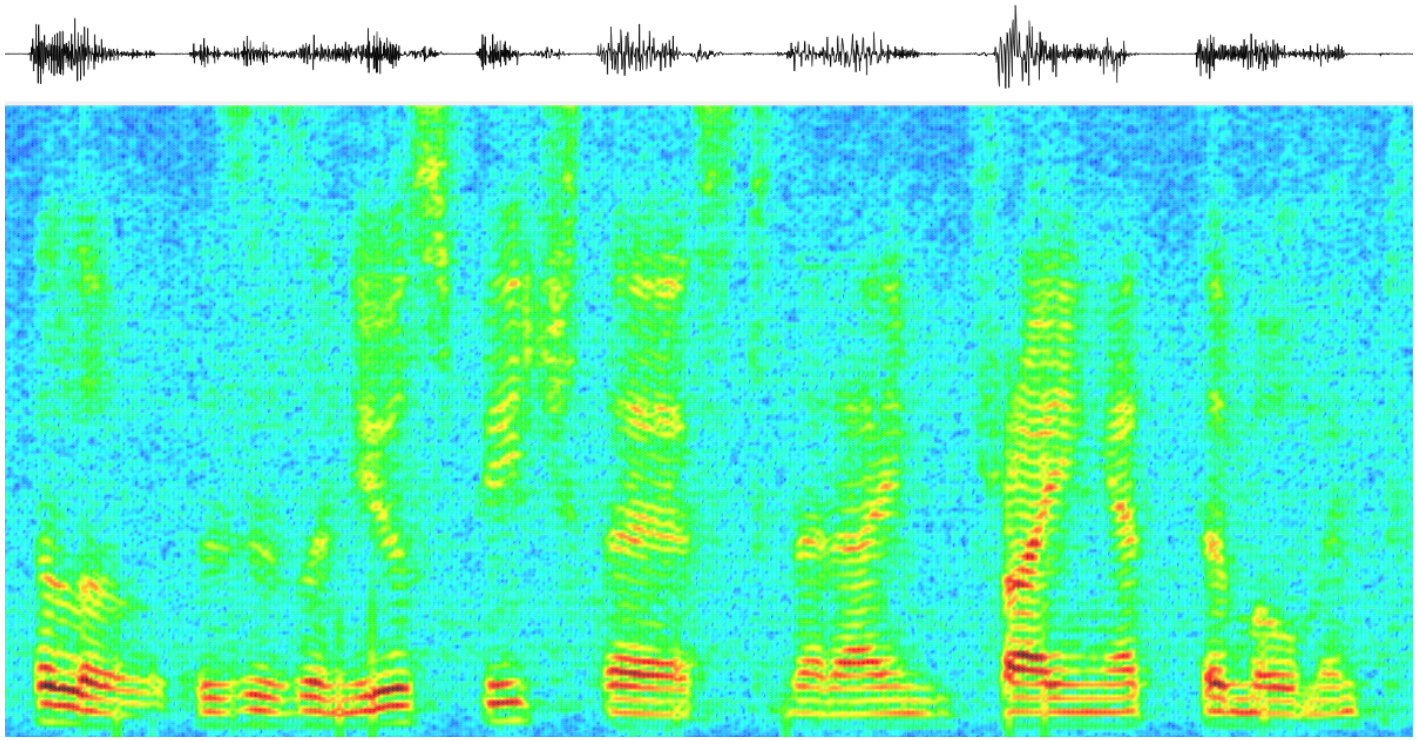


Figure: source

- Speech Recognition
  - Analyze a series of spectral vectors, determine what was said.
- Note: Inputs are sequences of vectors. Output is a classification result.

Example 2: Text Analysis

Stephen Curry scored 34 points and was named the NBA Finals MVP as the Warriors claimed the franchise's seventh championship overall. And this one completed a journey like none other, after a run of five consecutive finals, then a plummet to the bottom of the NBA, and now a return to greatness just two seasons after having the league's worst record.

- Football or Basketball?
- Text Analysis
  - E.g. analyze document, identify topic
    - Input series of words, output classification output
  - E.g. read English, output Persian
    - Input series of words, output series of words

Example 3: Stock Market Prediction

- Stock Market Prediction
  - Should I invest, vs. should I not invest in X?
  - Decision must be taken considering how things have fared over time.
- Note: Inputs are sequences of vectors. Output may be scalar or vector.

Long-Term Dependencies

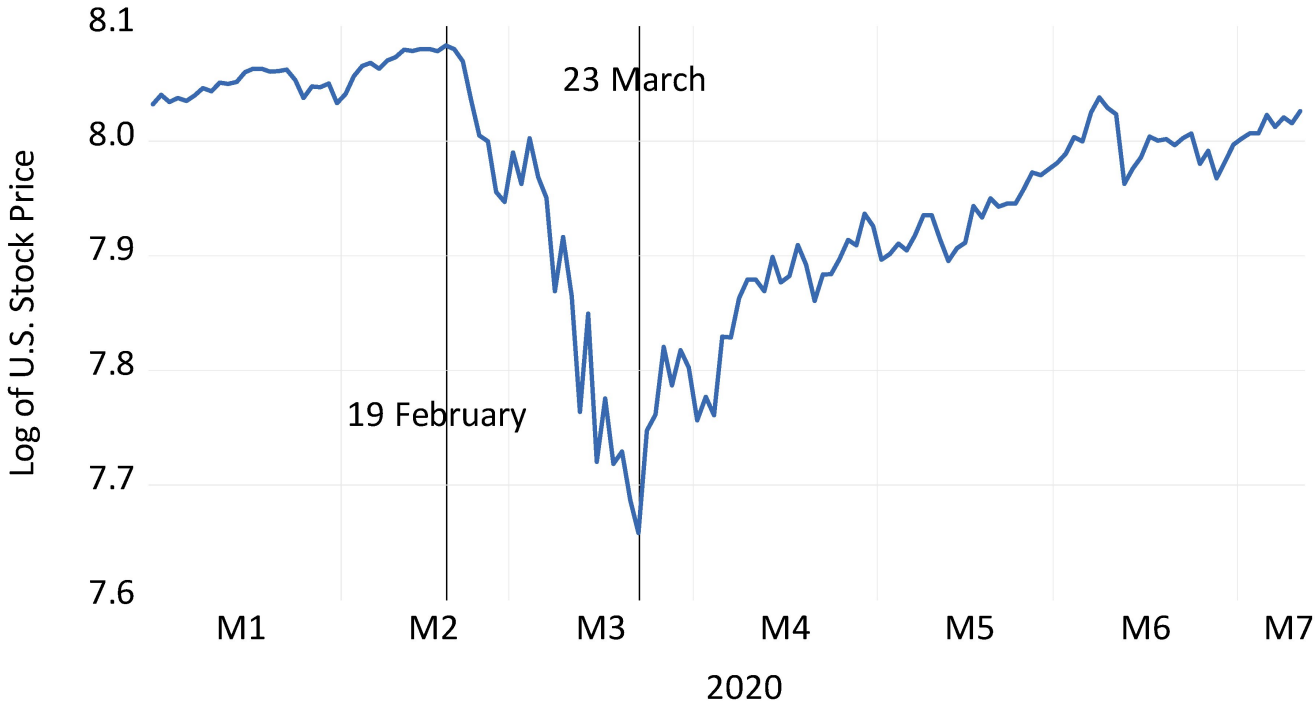


Figure: The Impact of the COVID-19 Pandemic on the U.S. Economy, source

- Systems often have long-term dependencies
  - Weekly/Monthly/Annual trends in the market
  - Though longer historic events tends to affect us less than more recent events
- Can you think of an example?

RNN, An Infinite Response System

- We can process a sequence of vectors x by applying a recurrence formula at every time step
  - $h_t = f(x_t, h_{t-1}), \qquad y_t = g(h_t)$
  - h_t is the state of the network
  - x_t is the input vector at t
  - y_t is the output at t
  - Need to define initial state h_{-1} for $t = 0$
  - An input x_0 at $t = 0$ produces h_0
  - h_0 produces h_1 which produces h_2 and so on...
  - h_t can be produced from h_{t-1} even if x_t is 0
  - A single input influences the output for the rest of time
- This is a fully recurrent neural network, or simply a recurrent neural network
- Don't worry, we will get back to this slide

Vanilla Neural Networks

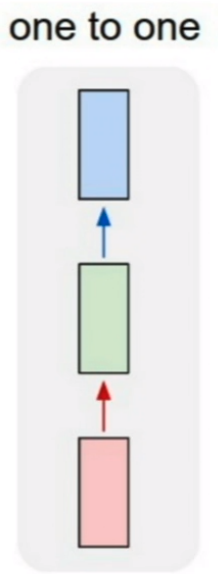


Figure: Types of Sequence Problems, source

- Vanilla Neural Networks
- Example: Image Classification
- Fixed-sized input and output

Sequence Output

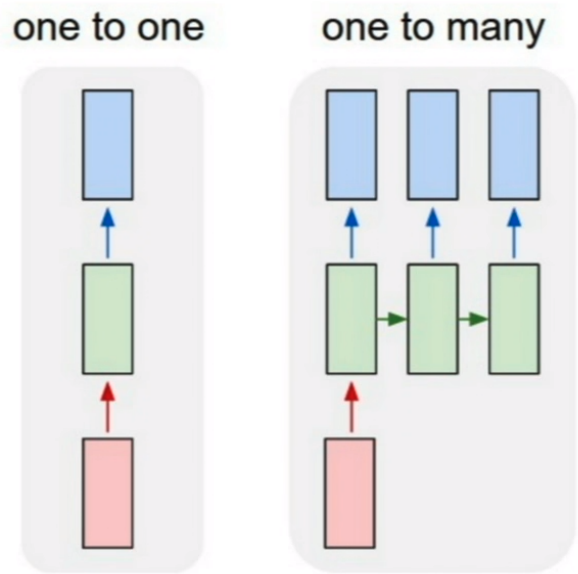


Figure: Types of Sequence Problems, source

- Sequence Output
- Example: Image Captioning
- image $\rightarrow$ sequence of words

Sequence Input

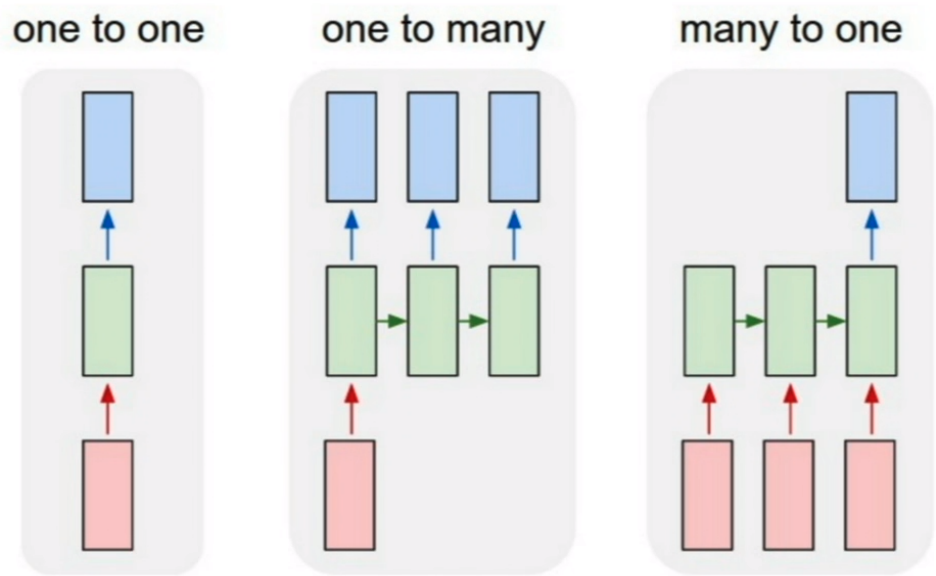


Figure: Types of Sequence Problems, source

- Sequence Input
- Example: Sentiment Analysis
- sequence of words $\rightarrow$ sentiment

Sequence Input And Sequence Output

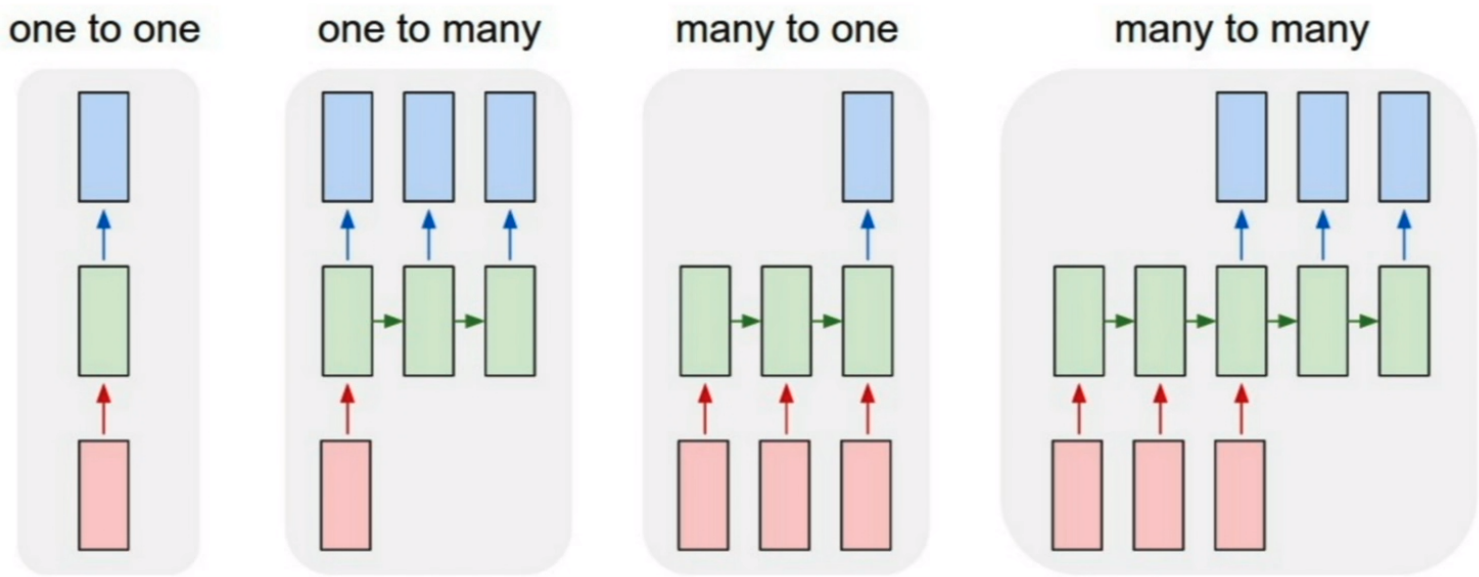


Figure: Types of Sequence Problems, source

- Sequence Input And Sequence Output
- Example: Machine Translation
- sequence of words in English $\rightarrow$ sequence of words in Persian

Synced Sequence Input And Output

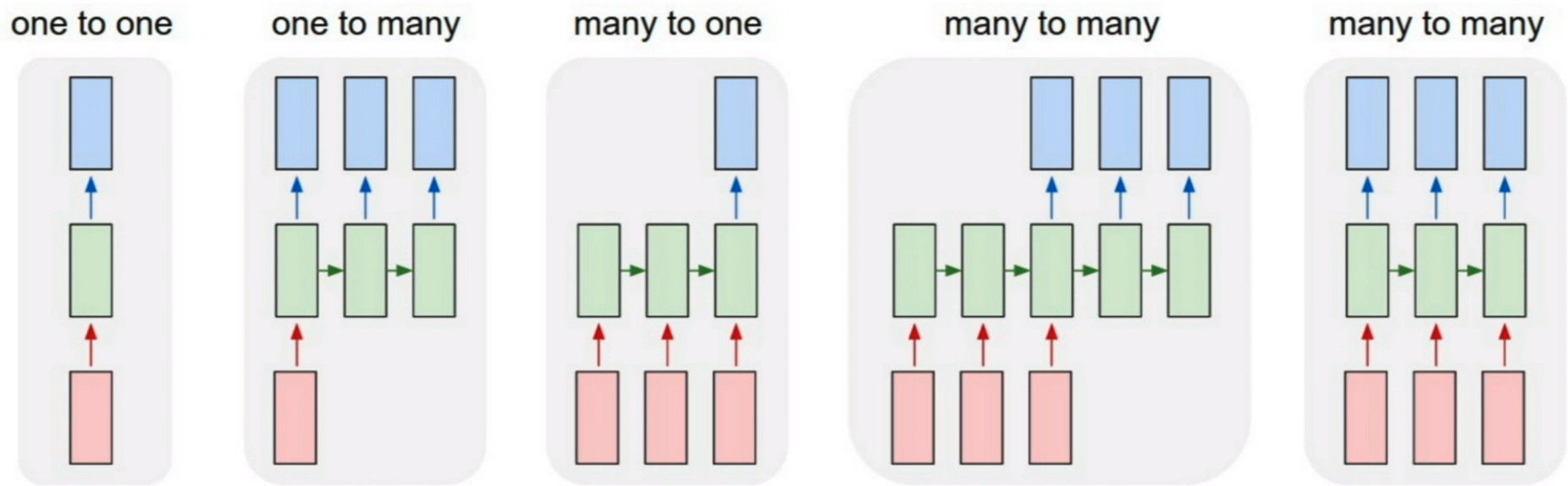


Figure: Types of Sequence Problems, source

- Synced Sequence Input And Output
- Example: Video Classification
- frames of the video $\rightarrow$ label of each frame

Latent Variable Model

- In n-grams for language modeling the conditional probability of token x_t at time step t only depends on the n previous tokens.
- If we want to incorporate the possible effect of tokens earlier than time step $t-n$ on x_t, we need to increase n
- By increasing n the number of model parameters would also increase exponentially with it
- Hence, rather than modeling $\mathbb{P}(x_t|x_{t-1},...,x_{t-n})$ it is preferable to use a latent variable model:

$$\mathbb{P}(x_t|x_{t-1},...,x_1) \approx \mathbb{P}(x_t|h_{t-1})$$

- h_{t-1} is a hidden state that stores the sequence information up to time step $t-1$
- In general, the hidden state at any time step t could be computed based on both the current input x_t and the previous hidden state h_{t-1} :

$$h_t = f(x_t, h_{t-1})$$

Recurrent Neural Network

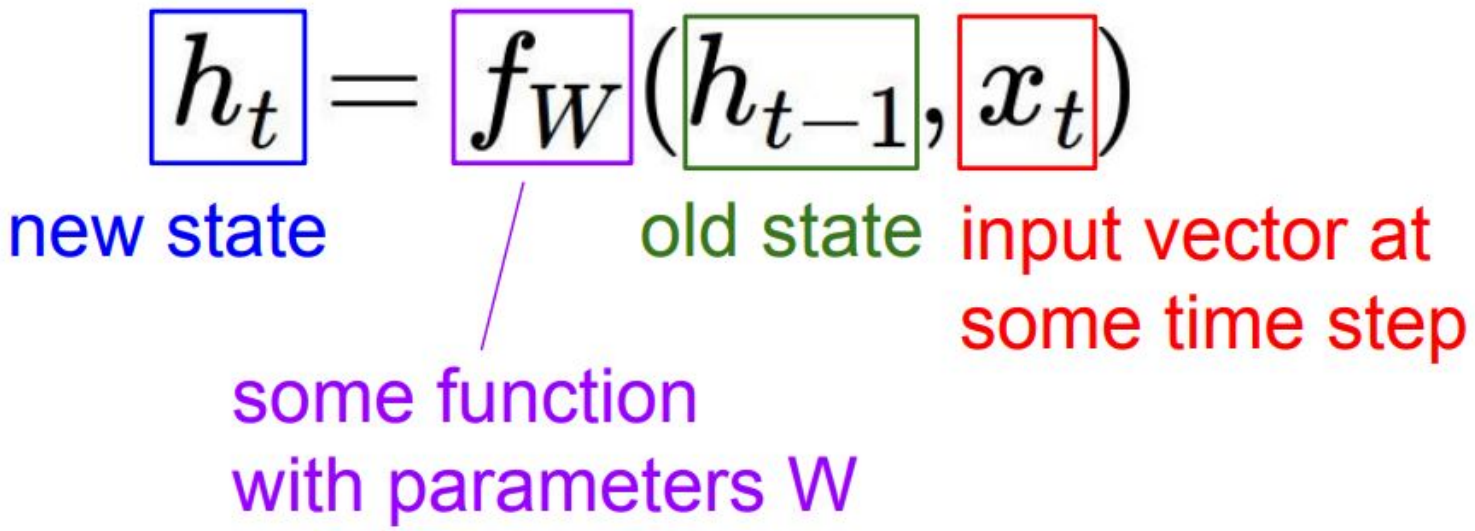
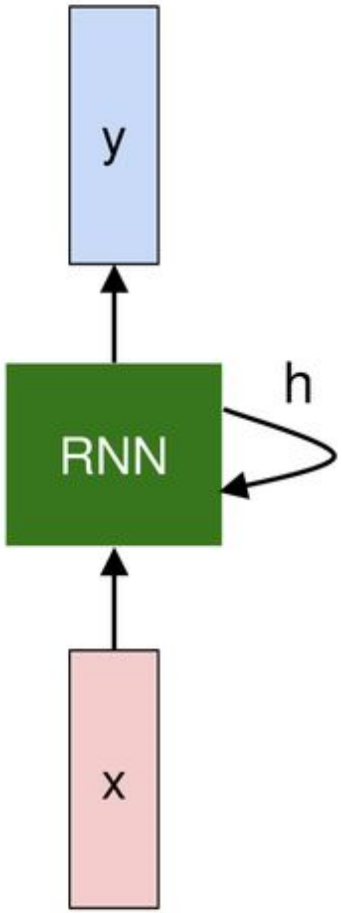


Figure: RNN formula, source

- We can process a sequence of vectors x by applying a recurrence formula at every time step
- The same function and the same set of parameters are used at every time step.

Vanilla RNN



$$h_t = f_W(h_{t-1}, x_t)$$
$$y_t = g_W(h_t)$$

$$\rightarrow \begin{cases} h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \\ y_t = W_{yh}h_t + b_y \end{cases}$$

RNN: Forward Pass

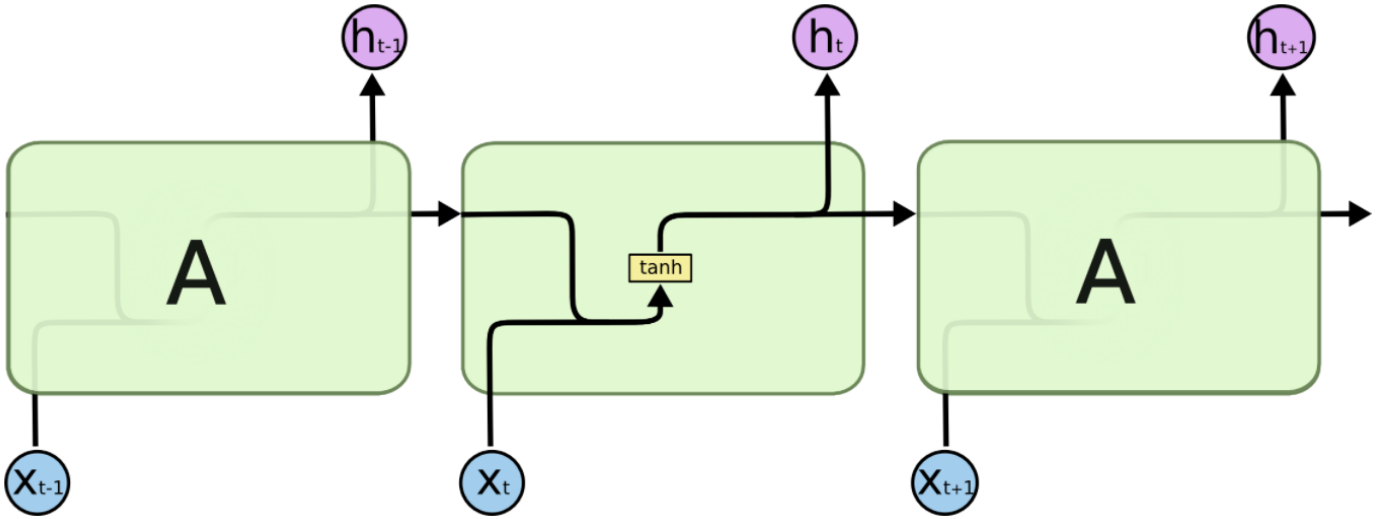


Figure: The repeating module in a standard RNN contains a single layer, source

$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \\
&= \tanh((W_{hh}W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h) \\
&= \tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h)
\end{aligned}
$$

RNN: Code Example

```
[24] 1 import tensorflow as tf
     2
     3 W_hh, h = tf.random.normal((4, 4)), tf.random.normal((4, 3))
     4 W_hx, x_t = tf.random.normal((4, 1)), tf.random.normal((1, 3))
     5 b_h = tf.random.normal((4, 3))
     6
     7 tf.matmul(W_hh, h) + tf.matmul(W_hx, x_t) + b_h
```

```
<tf.Tensor: shape=(4, 3), dtype=float32, numpy=
array([[ 9.386221 , -2.8952436, -9.169736 ],
       [ 1.3449984, -4.2384577, -1.438521 ],
       [ 1.0814373, -3.350898 , -4.3827868],
       [ 1.4290853, -0.9657709, -5.4122305]], dtype=float32)>
```

```
[25] 1 tf.matmul(tf.concat((W_hh, W_hx), axis=1), tf.concat((h, x_t), axis=0)) + b_h
```

```
<tf.Tensor: shape=(4, 3), dtype=float32, numpy=
array([[ 9.386221 , -2.8952441, -9.169736 ],
       [ 1.3449984, -4.2384577, -1.4385211],
       [ 1.0814373, -3.350898 , -4.3827868],
       [ 1.4290853, -0.9657709, -5.4122305]], dtype=float32)>
```

Figure: The calculation of $W_{hh}h_{t-1} + W_{hx}x_t$ for the hidden state is equivalent to matrix multiplication of concatenation of W_{hh} and W_{hx} and concatenation of h_{t-1} and x_t

RNN: Computational Graph

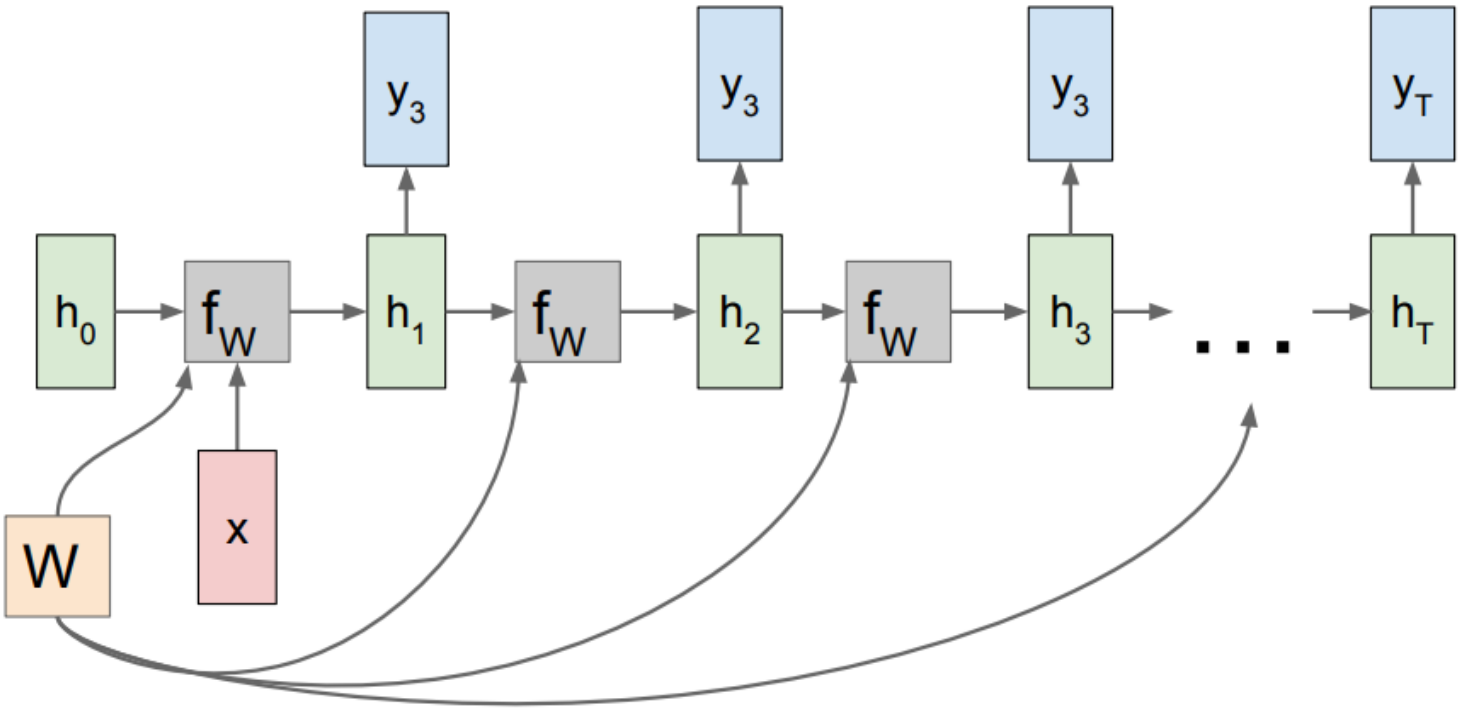


Figure: RNN One to Many Computational Graph, source

RNN: Computational Graph

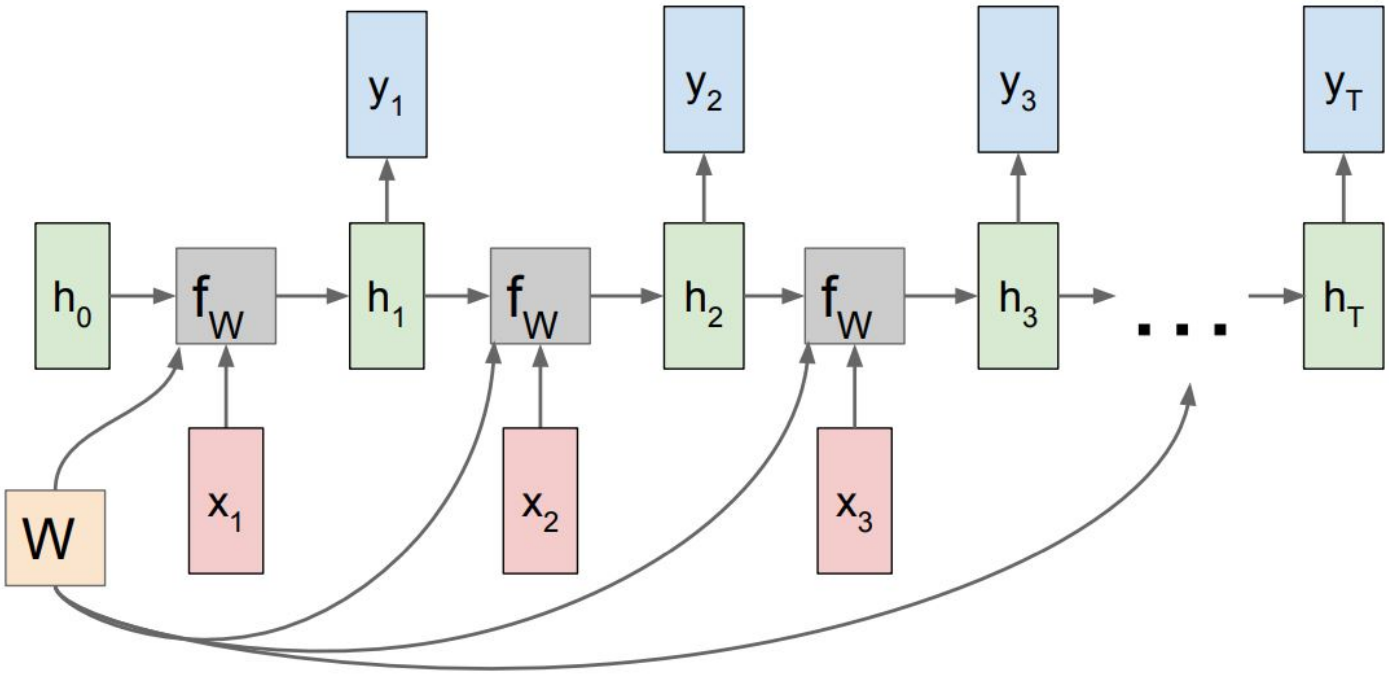


Figure: RNN Many to Many Computational Graph, source

Example: Character-Level Language Model

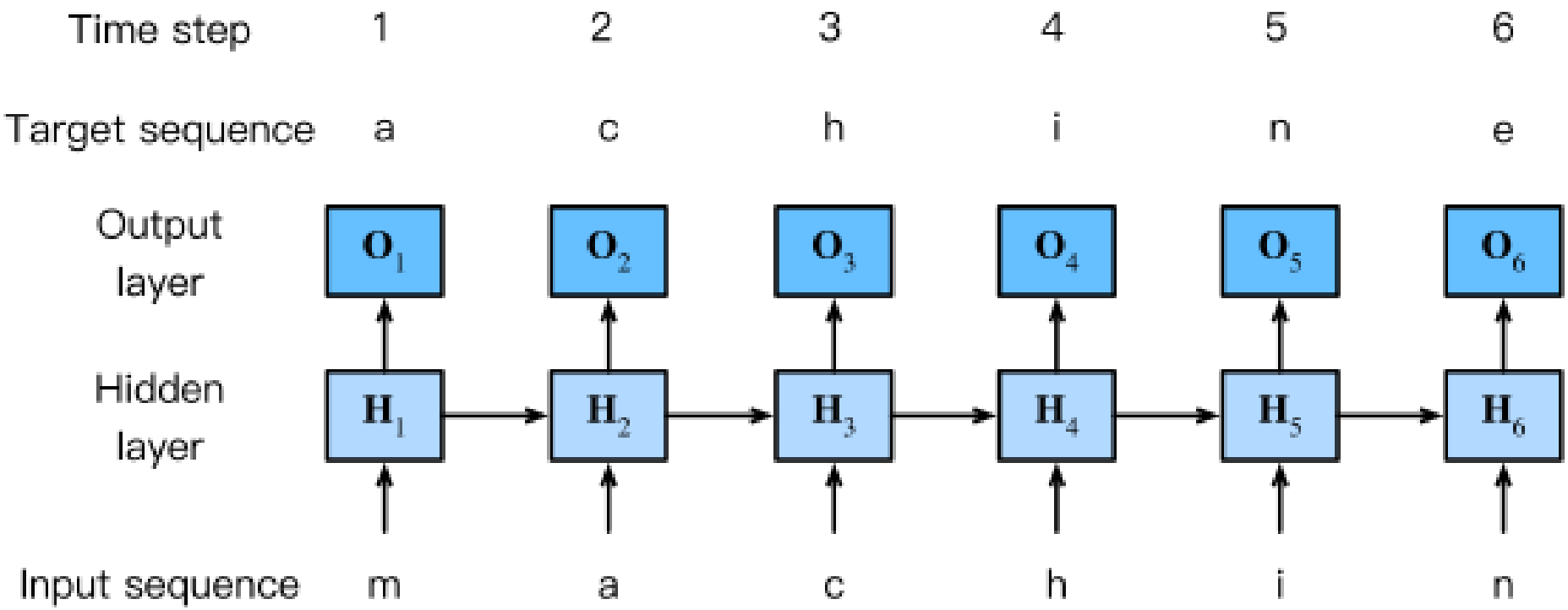
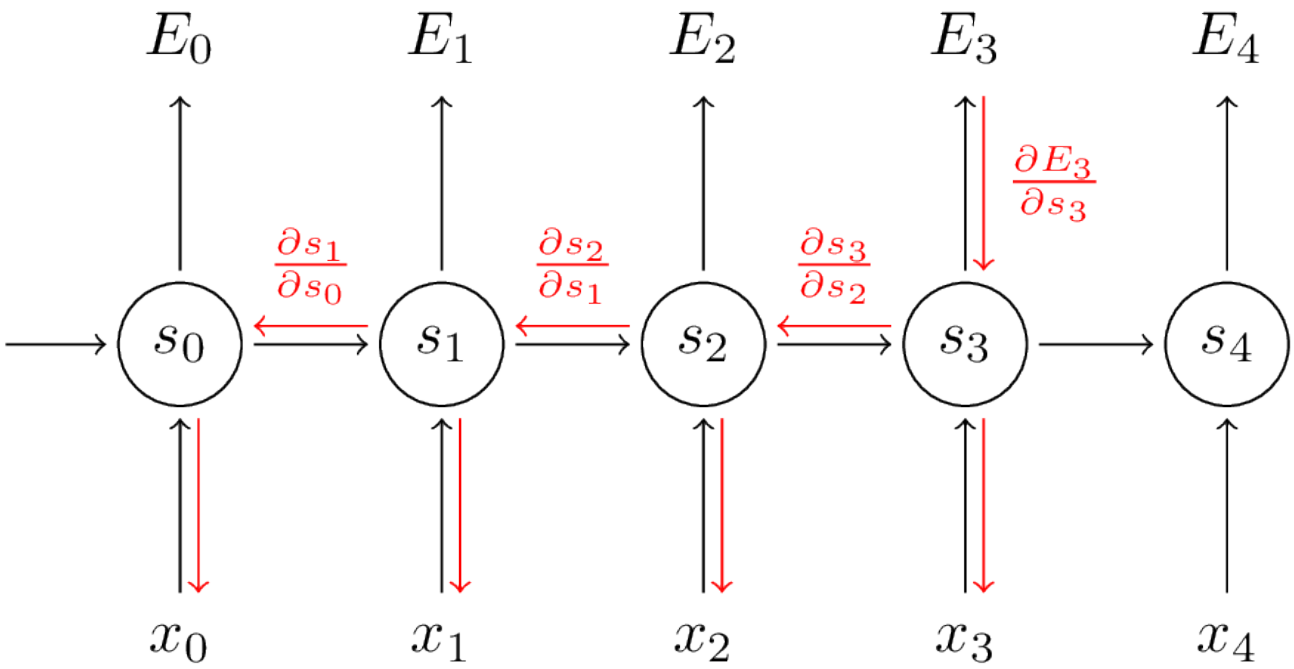


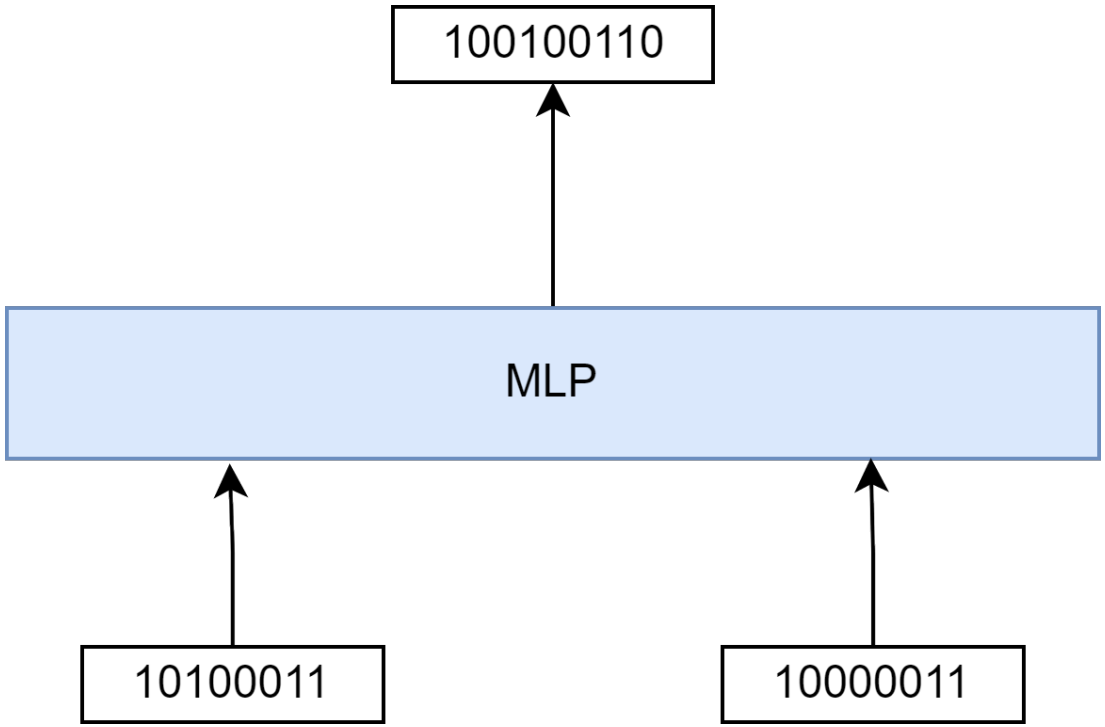
Figure: A character-level language model based on the RNN. The input and target sequences are "machin" and "achine", respectively, source

Training RNN: Backpropagation Through Time



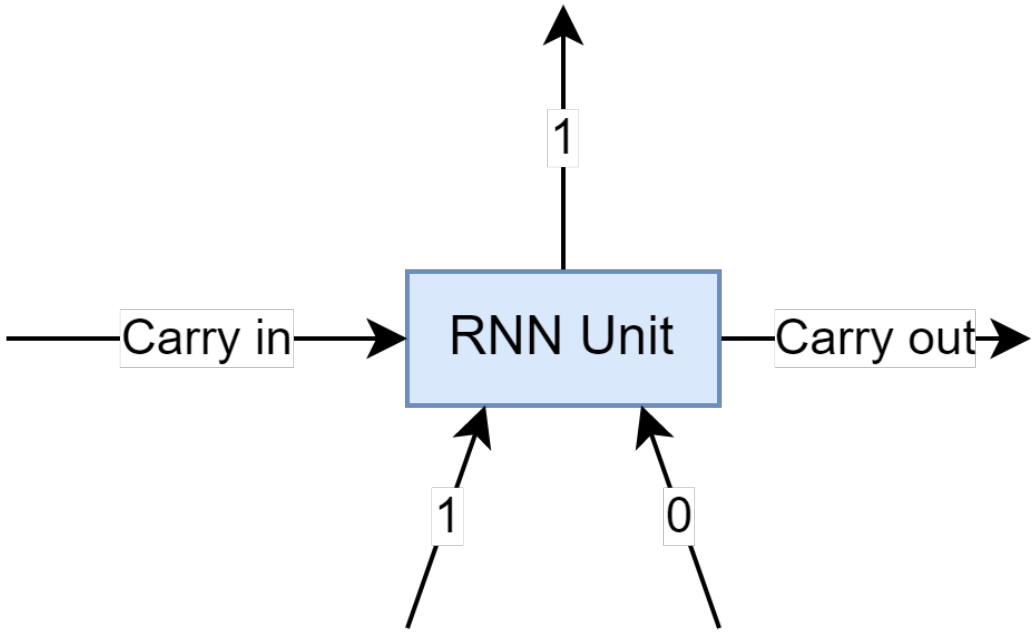
- We will explain BPTT fully in the next session

MLPs vs RNN: The Addition Problem



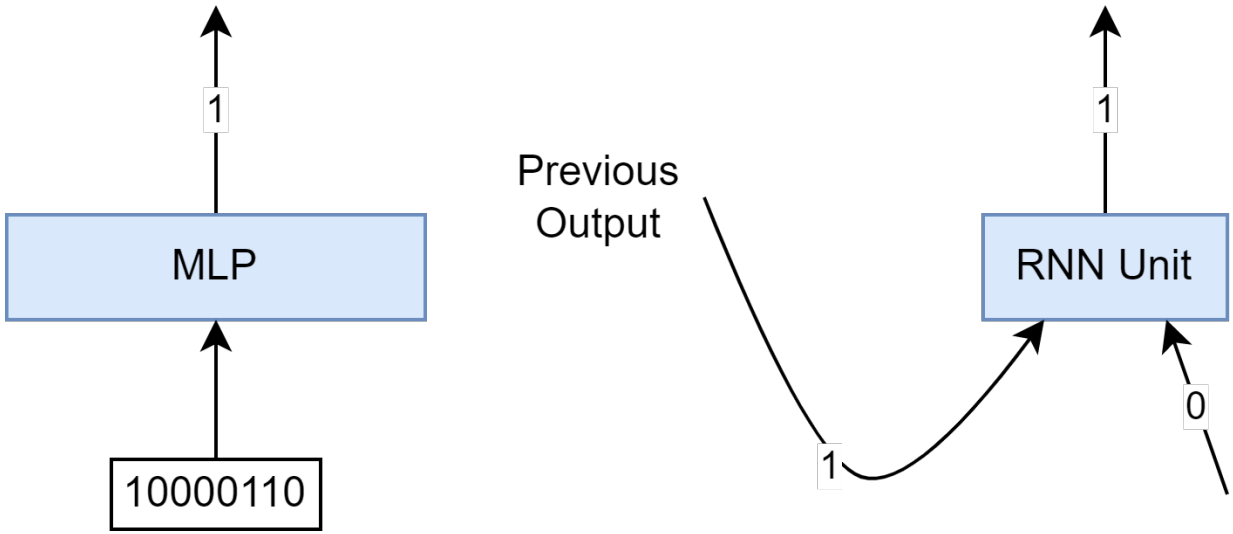
- The addition problem: Add two N-bit numbers to produce a N+1-bit number
  - Input is binary
  - MLP will require large number of training instances
  - Network trained for N-bit numbers will not work for N+1 bit numbers

MLPs vs RNN: The Addition Problem



- The addition problem: Add two N-bit numbers to produce a N+1-bit number
  - RNN solution: Very simple
  - Can add two numbers of any size
  - Needs very little training data

MLPs vs RNN: The Parity Problem



- Is the number of "ones" even or odd
  - MLP solution: XOR network, quite complex
  - RNN solution: Simple, generalizes to input of any size

Backpropagation Through Time (BPTT)

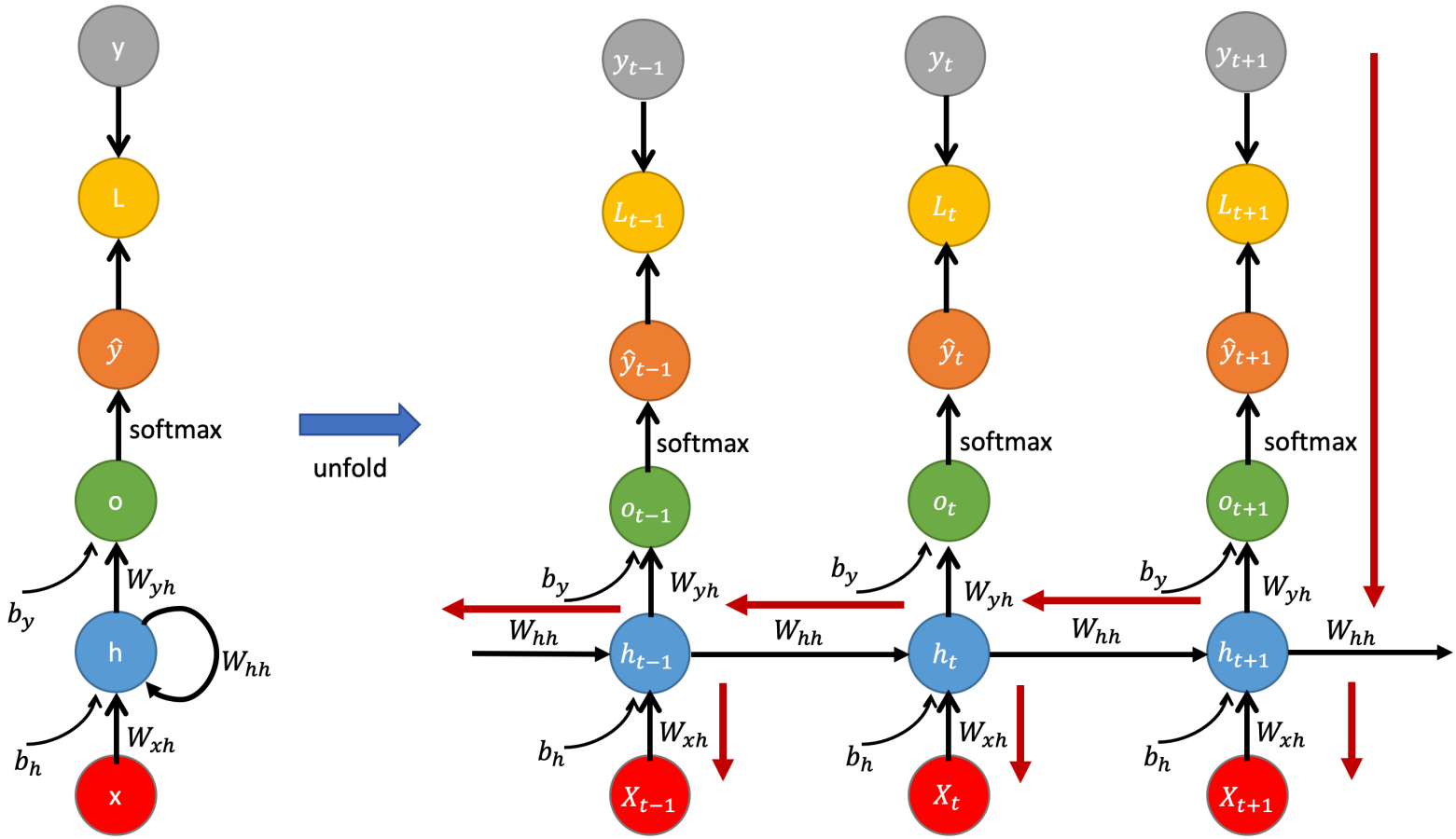


Figure: Simple RNN Computational Graph, Source

Backpropagation Through Time (BPTT)

- Suppose that in this example, we have

$$h_t = tanh(X_t.W_{xh} + h_{t-1}.W_{hh} + b_h)$$

$$o_t = h_t.W_{yh} + b_y$$

$$y_t = Softmax(o_t)$$

- And our loss function is Log Loss. So as you remember, we have

$$L(y, \hat{y}) = \sum_{t=1}^{T} L_t(y_t, \hat{y_t}) = -\sum_{t=1}^{T} y_t \, log \, \hat{y_t} = -\sum_{t=1}^{T} y_t \, log \, [Softmax(o_t))]$$

Backpropagation Through Time (BPTT)

- Now, we are going to calculate derivative of L w.r.t $W_{yh}, W_{hh}, W_{xh}, b_y, b_h$
  - Part I: The Straight Ones

$$\frac{\partial L}{\partial W_{yh}} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W_{yh}}$$

$$= \sum_{t=1}^{T} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial o_t} \frac{\partial o_t}{\partial W_{yh}}$$

$$= \sum_{t=1}^{T} (\hat{y}_t - y_t) \otimes h_t$$

$$\frac{\partial L}{\partial b_y} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial o_t} \frac{\partial o_t}{\partial b_y} = \sum_{t=1}^{T} (\hat{y}_t - y_t)$$

Backpropagation Through Time (BPTT)

- Cont.
  - Part II: The Tricky Ones

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_{hh}}$$

we know that h_t is a function of h_{t-1} and W_{hh}, h_{t-1} itself is a function of W_{hh} and h_{t-2}, and so on. Thus, we have

$$\frac{\partial h_t}{\partial W_{hh}} = \left(\frac{\partial h_t}{\partial W_{hh}}\right)_{h_{t-1}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_{hh}}$$

$$\frac{\partial h_{t-1}}{\partial W_{hh}} = \left(\frac{\partial h_{t-1}}{\partial W_{hh}}\right)_{h_{t-2}} + \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_{hh}}$$

In conclusion, the following equation holds (by substitution)

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \left(\sum_{k=1}^{t} \left(\prod_{j=k}^{t-1} \frac{\partial h_{j+1}}{\partial h_j} \right) \left(\frac{\partial h_k}{\partial W_{hh}} \right)_{h_{k-1}} \right)$$

Backpropagation Through Time (BPTT)

- Cont. It's also true that

$$\prod_{j=k}^{t-1} \frac{\partial h_{j+1}}{\partial h_j} = \frac{\partial h_t}{\partial h_k}$$

That leads us to

$$\frac{\partial L_t}{\partial W_{hh}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \left(\frac{\partial h_k}{\partial W_{hh}} \right)_{h_{k-1}}$$

and

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \left(\frac{\partial h_k}{\partial W_{hh}} \right)_{h_{k-1}}$$

Backpropagation Through Time (BPTT)

- Cont. Similar to the mentioned way, we could compute derivative of L w.r.t W_{xh}, b_h

$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^{T}\sum_{k=1}^{t} \frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\left(\frac{\partial h_k}{\partial W_{xh}}\right)_{h_{k-1}}$$

$$\frac{\partial L}{\partial b_h} = \sum_{t=1}^{T}\sum_{k=1}^{t} \frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\left(\frac{\partial h_k}{\partial b_h}\right)_{h_{k-1}}$$

Backpropagation Through Time (BPTT)

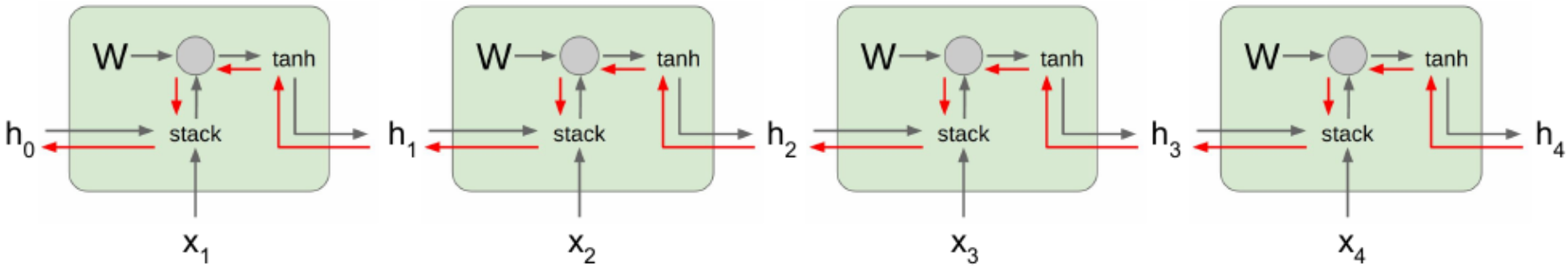


Figure: Vanilla RNN Gradient Flow, Source

- RNN Training Issues
  - Exploding Gradients
    What can we do to solve exploding?
  - Vanishing Gradients
    What can we do to solve vanishing?

RNN Unit

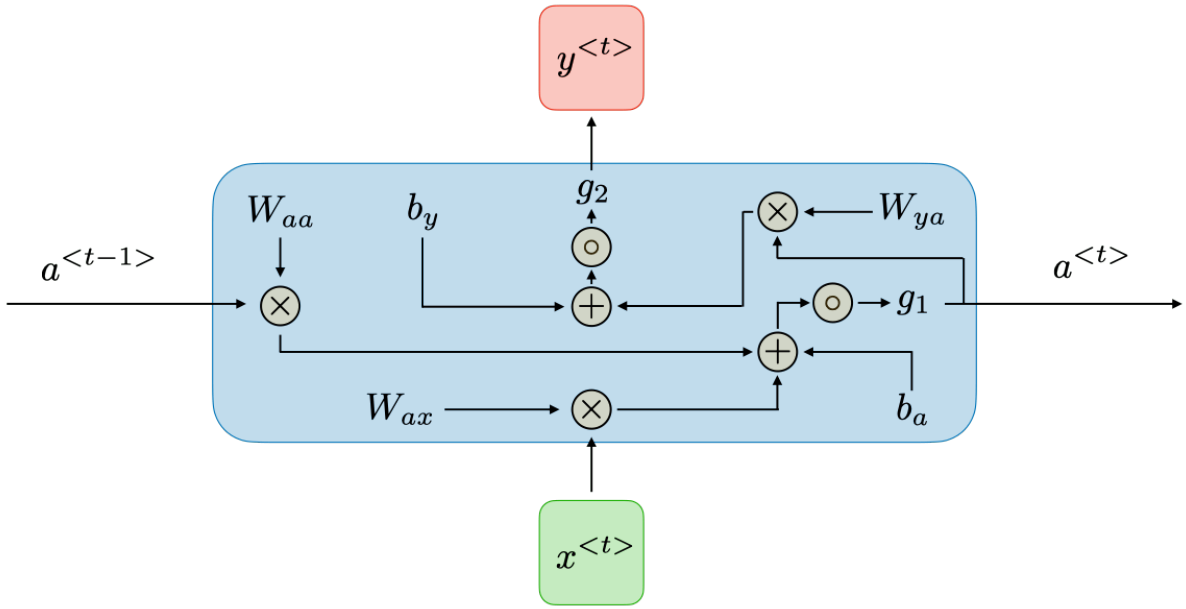


Figure: RNN Unit, Source

- For a simple RNN unit we had:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$y^{<t>} = g_2(W_{ga}a^{<t>} + b_y)$$

Gated Recurrent Unit (GRU)

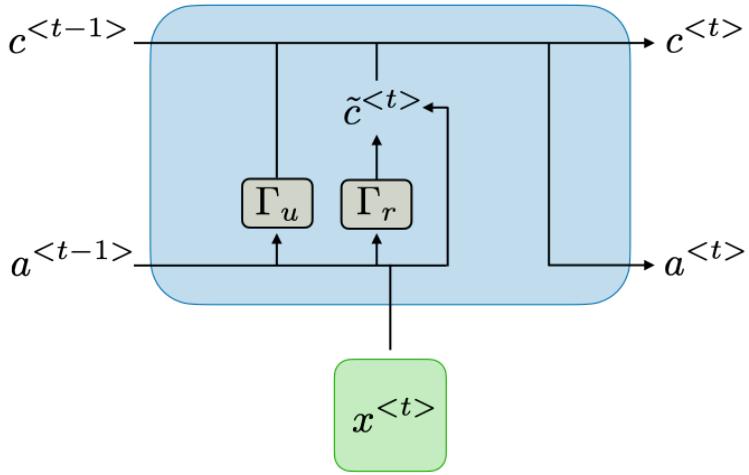


Figure: GRU Unit, Source

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_r = \sigma(W_r[a^{<t-1>}, x^{<t>}] + b_r)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$c^{<t>} = a^{<t>}$$

Long Short-Term Memory (LSTM)

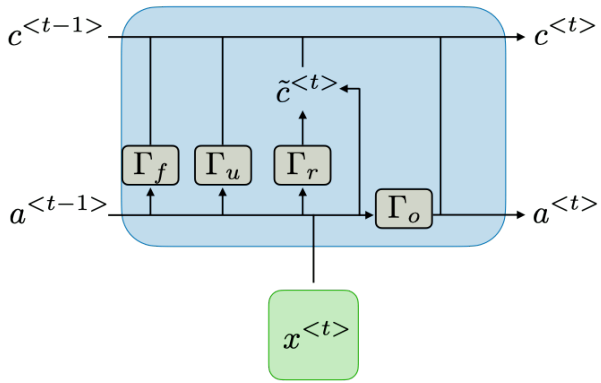


Figure: LSTM Unit, Source

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_r = \sigma(W_r[a^{<t-1>}, x^{<t>}] + b_r)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (\Gamma_f) * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$

Type of Gates

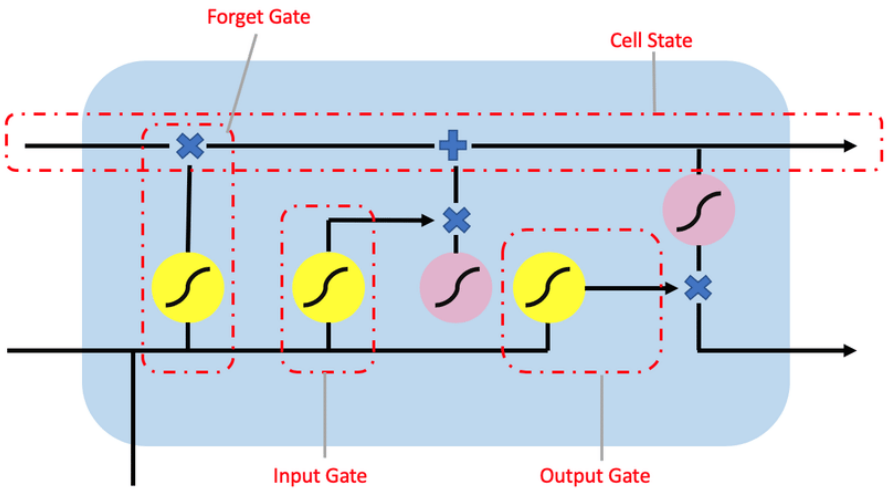


Figure: Gates, Source

- Forget gate Γ_f: Erase a cell or not
- Relevance Gate Γ_r: Drop previous information
- Update Gate Γ_u: How much past should matter
- Output gate Γ_o: How much to reveal of a cell

LSTM Walk Through

- The first step in our LSTM is to decide what information we're going to throw away from the cell state.

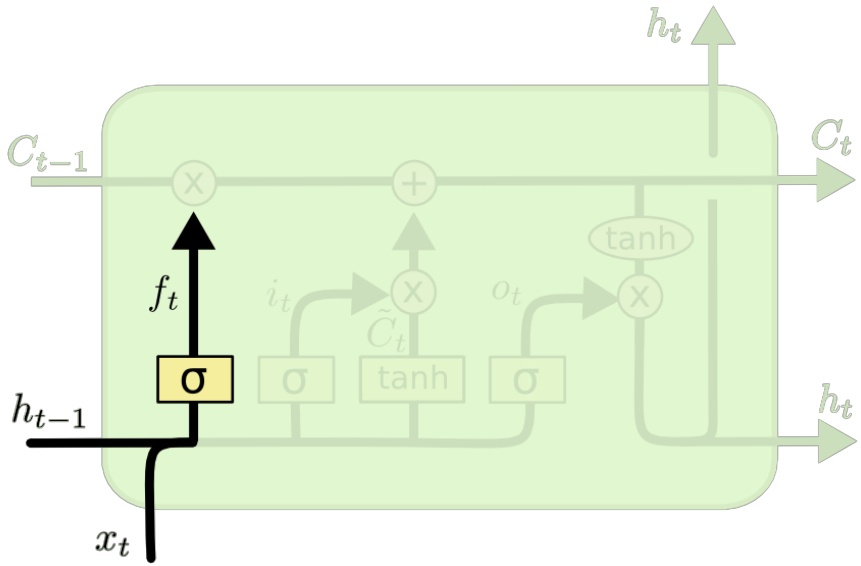


Figure: Forget Gate Layer, Source

LSTM Walk Through

- The next step is to decide what new information we're going to store in the cell state.

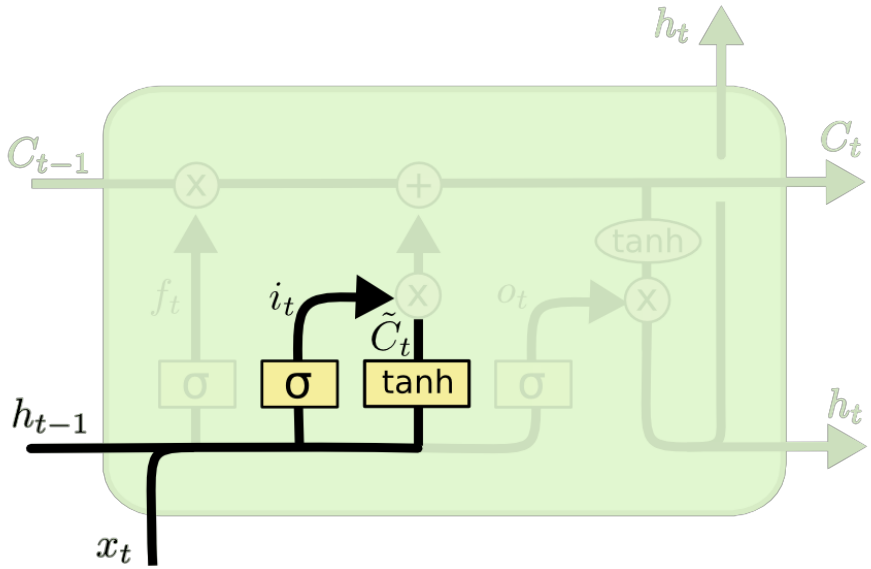


Figure: Input Gate Layer, Source

LSTM Walk Through

- It's now time to update the old cell state, Ct−1, into the new cell state Ct.

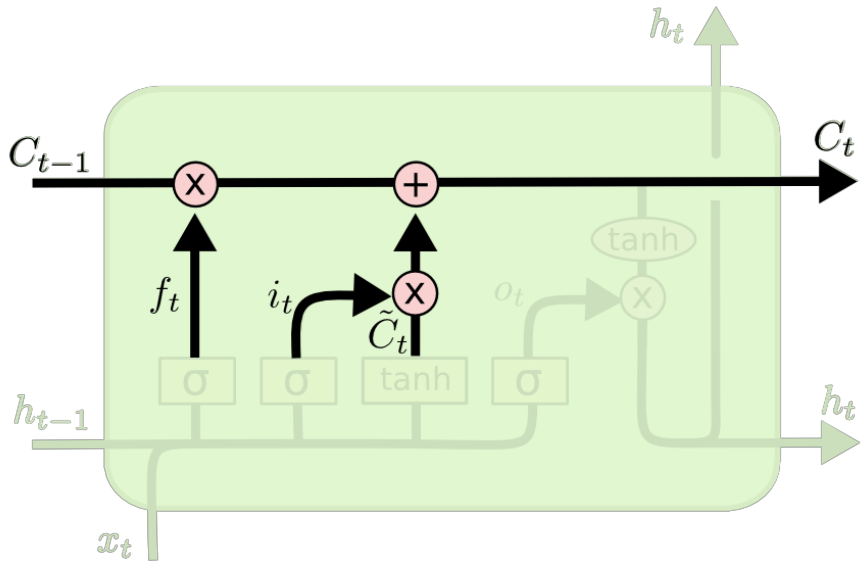


Figure: Update Cell State, Source

LSTM Walk Through

- Finally, we need to decide what we're going to output. This output will be based on our cell state

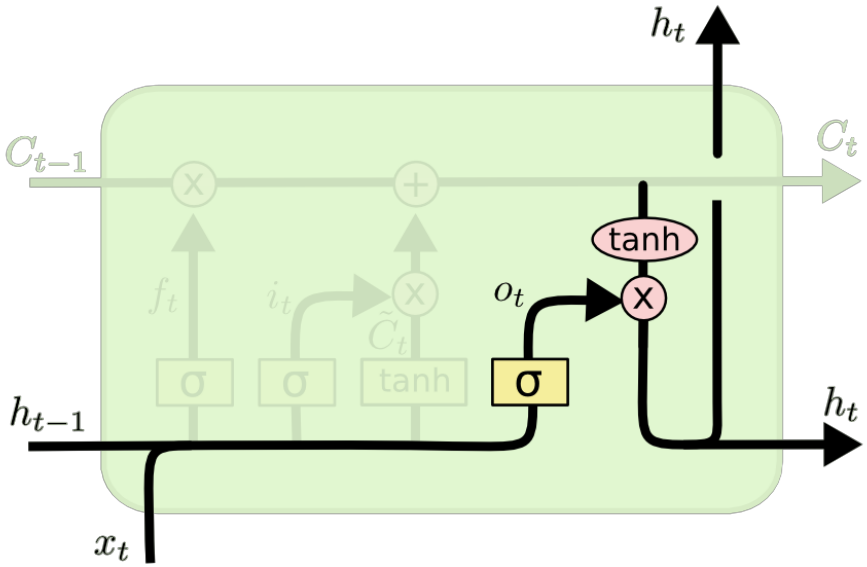


Figure: Output Gate Layer, Source

Why LSTMs?

- The LSTM does have the ability to remove and add information to the cell state.
- The gates in the previous slide let the information to pass through the units.
- The gates value are between zero and one and specify how much information should be let through .
- They also somehow solve the vanishing gradient .
- We can use more blocks of them so there will be more information to remember.

How LSTMs solve vanishing gradients

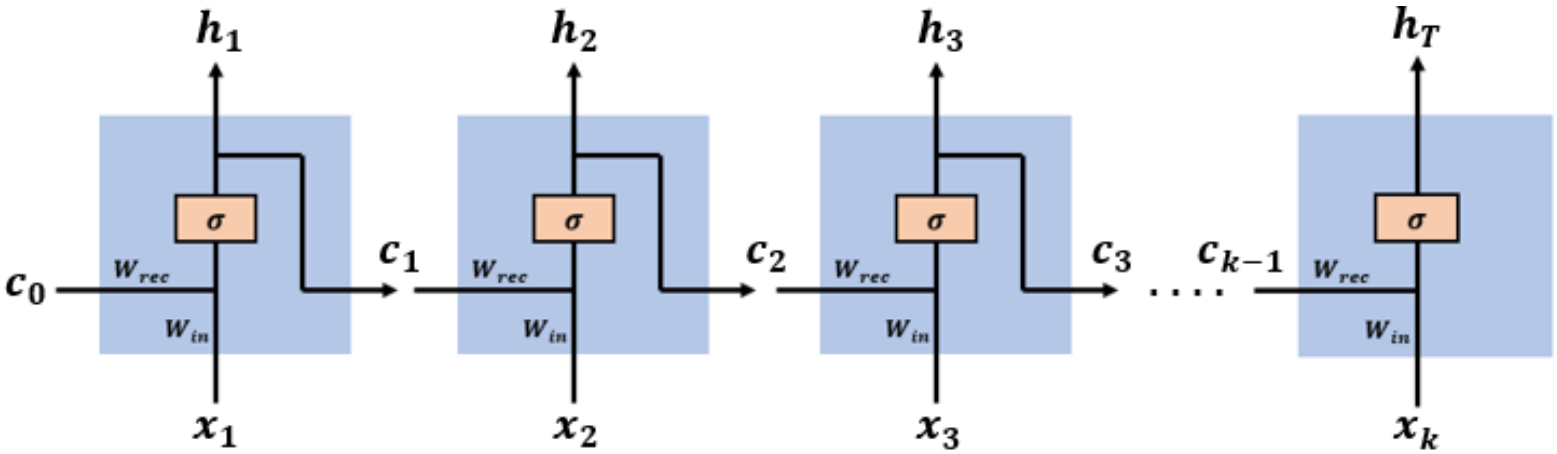


Figure: Simple Recurrent Neural Network, Source

How LSTMs solve vanishing gradients

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_k}{\partial W} = \frac{\partial L_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\cdots\frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W} = \frac{\partial L_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W}$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \sigma'(W_{rec}.c_{t-1} + W_{in}.x_t)W_{rec}$$

$$\frac{\partial L_k}{\partial W} = \frac{\partial L_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\sigma'(W_{rec}.c_{t-1} + W_{in}.x_t)W_{rec}\right)\frac{\partial c_1}{\partial W}$$

- For large K the gradient tends to vanish or if W_{rec} is large enough it cause exploding gradient which is solved by *Gradient Clipping*.

How LSTMs solve vanishing gradients

- In LSTM we also have

$$\frac{\partial L_k}{\partial W} = \frac{\partial L_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W}$$

- But

$$c^t = \Gamma_u * \tilde{c}^t + (\Gamma_f) * c^{t-1}$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial \Gamma_f}{\partial c_{t-1}}.c_{t-1} + \Gamma_f + \frac{\partial \Gamma_u}{\partial c_{t-1}}.\tilde{c_t} + \frac{\partial \tilde{c_t}}{\partial c_{t-1}}.\Gamma_u$$

- Consider that the *forget gate* is added to other terms and allows better control of gradient values. But it doesn't guarantee that there is no vanishing or exploding in gradient.

Bidirectional RNN

- How to get information from future?

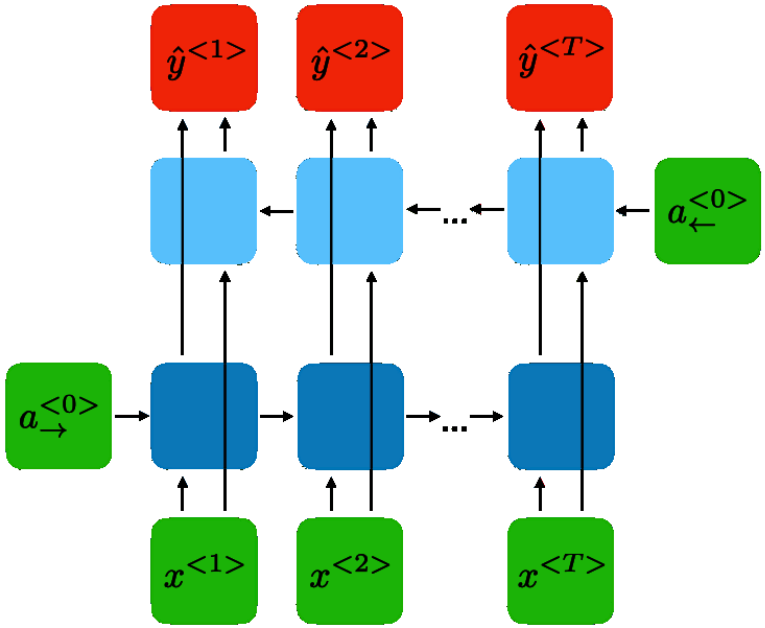


Figure: BRNN, Source

$$y^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<T-t>}] + b_y)$$

Bidirectional RNN

- They are usually used in natural language processing.
- They are powerful for modeling dependencies between words and phrases in both directions of the sequence because every component of an input sequence has information from both the past and present.

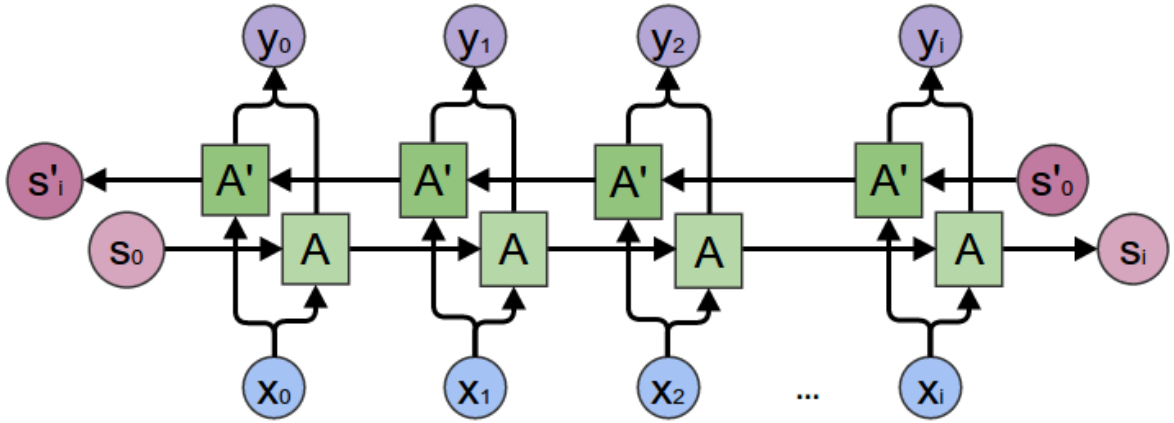


Figure: BRNN, Source

Back-propagation in BRNN

- It is exactly the same as simple RNN

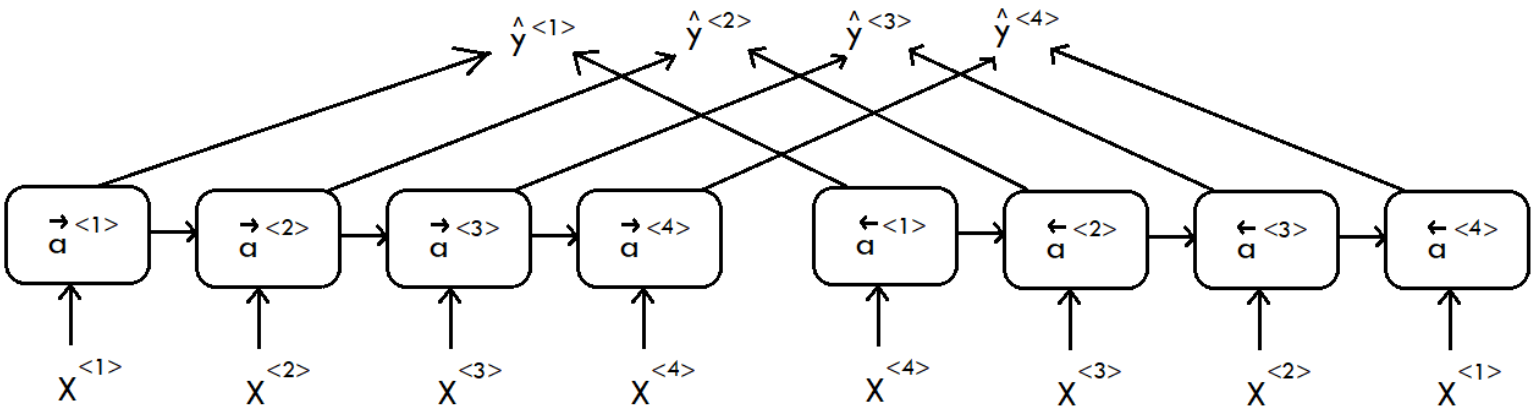


Figure: BRNN, Source

Encoder-Decoder

- If we want an output with a different length from the input (e.g. Machine Translation), what should we do?
- Which alignment method should we use?

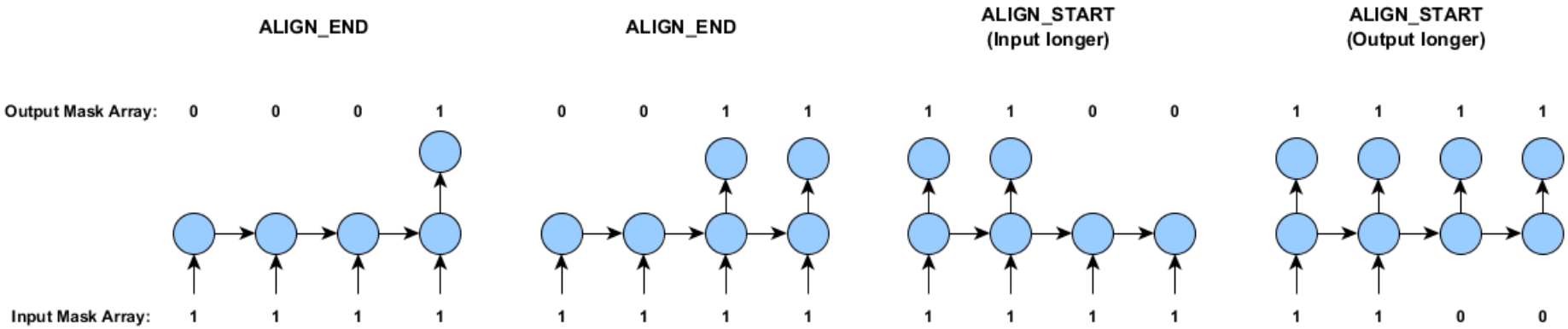


Figure: Problems in Sequence-to-Sequence Models, Source

Encoder-Decoder

Encoder-Decoder architecture could be a solution.

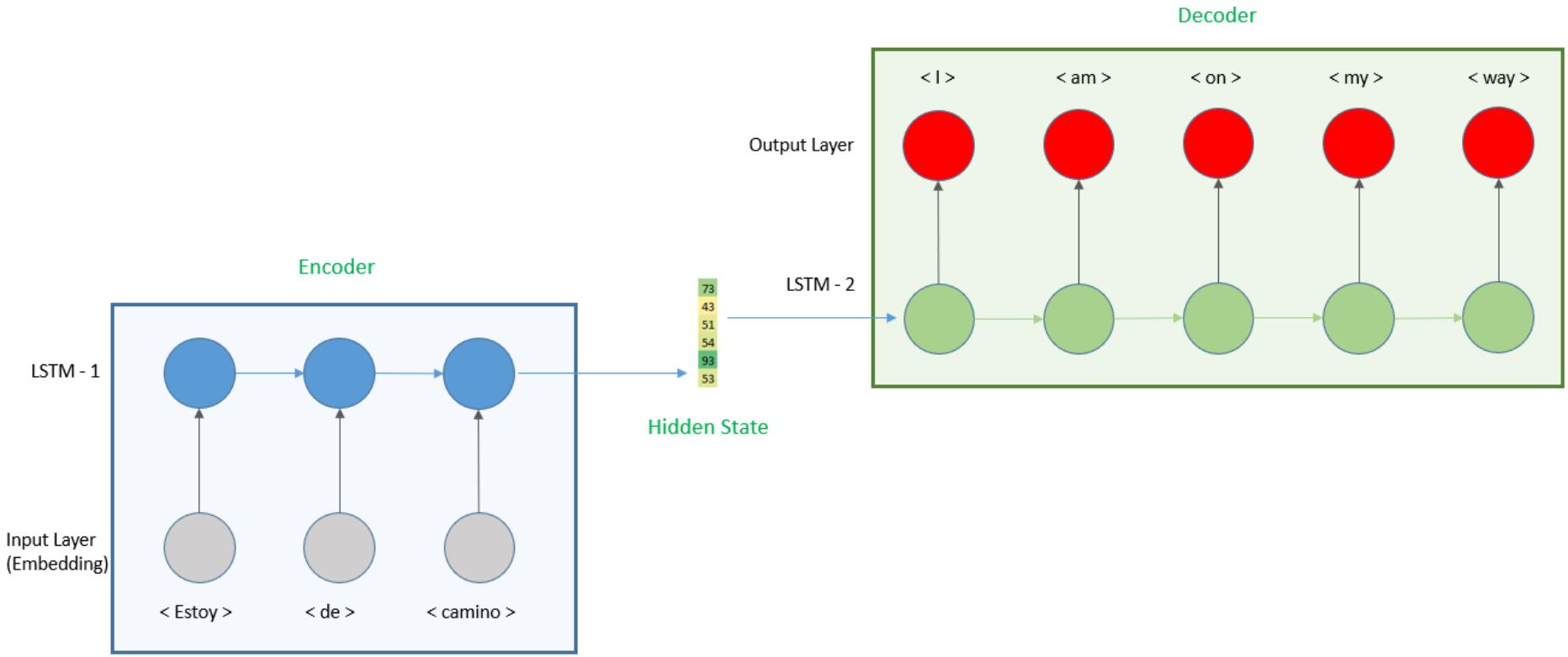


Figure: Encoder-Decoder, Source

Encoder-Decoder

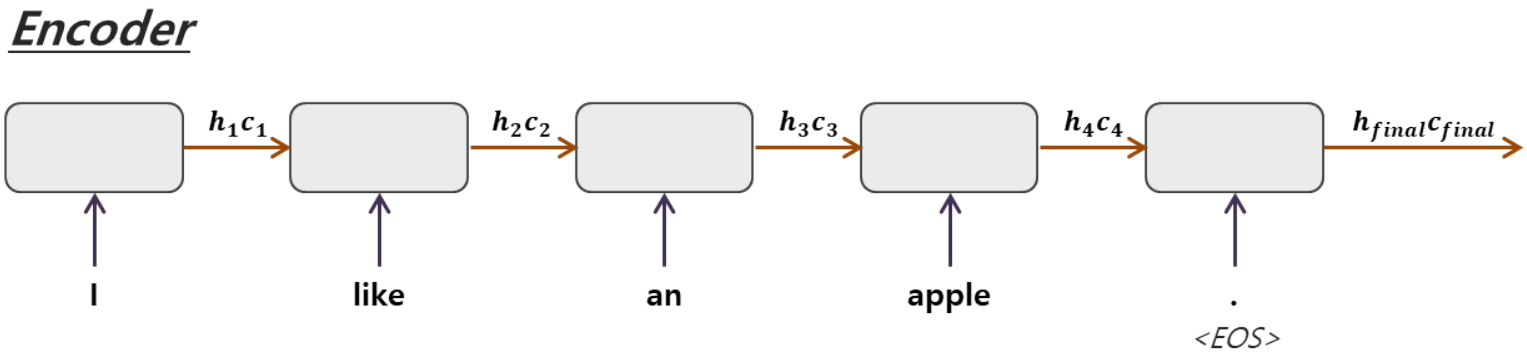


Figure: Encoder, Source

- **Encoder:** The encoder processes the input sequence and compresses the information into a fixed length vector, known as the context vector.

Encoder-Decoder

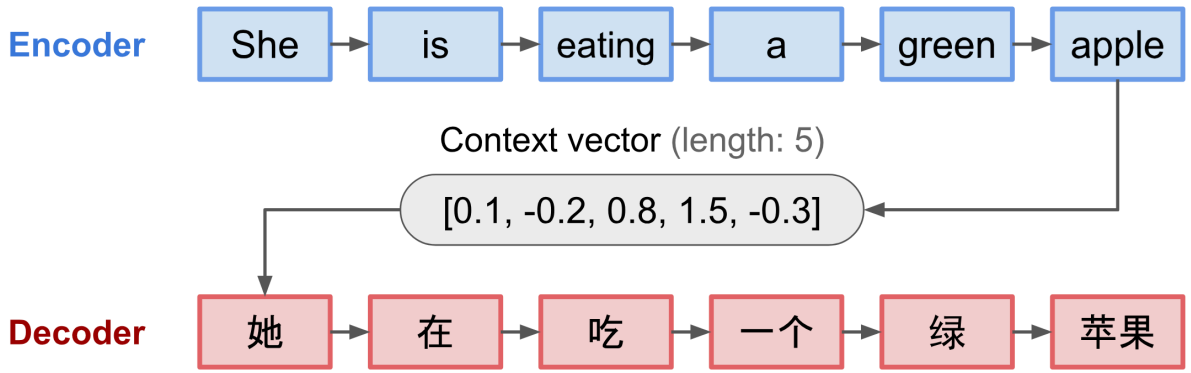


Figure: Context Vector, Source

- **Context Vector (Hidden State):** This vector is expected to convey the meaning of the whole source sequence from the encoder to the decoder.
  - The early work considered the last state of the encoder network as the context vector.

Encoder-Decoder

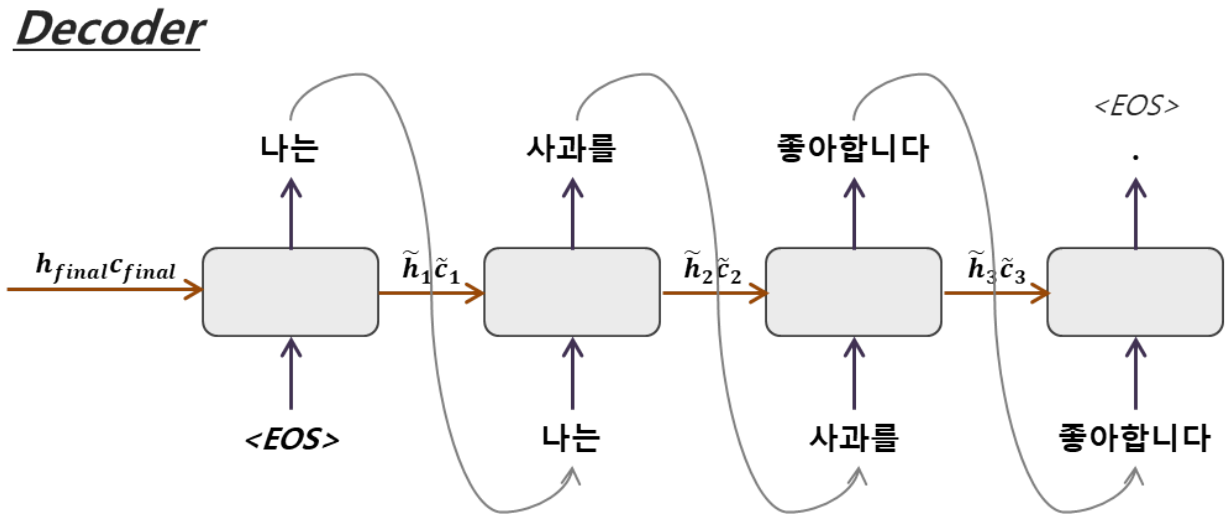


Figure: Decoder, Source

- **Decoder:** The decoder uses the context vector to output the desired target sequence.

Encoder-Decoder Limitations

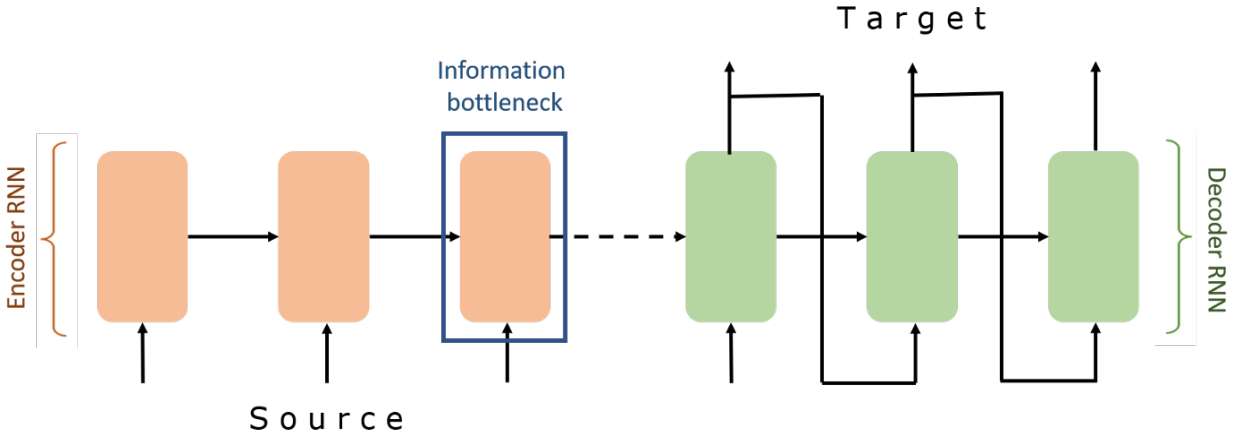


Figure: Bottleneck Phenomenon, Source

- The context vector is bottleneck . By increasing the length of the input sequence, the model captures the essential information roughly. How to solve it?

Encoder-Decoder Limitations

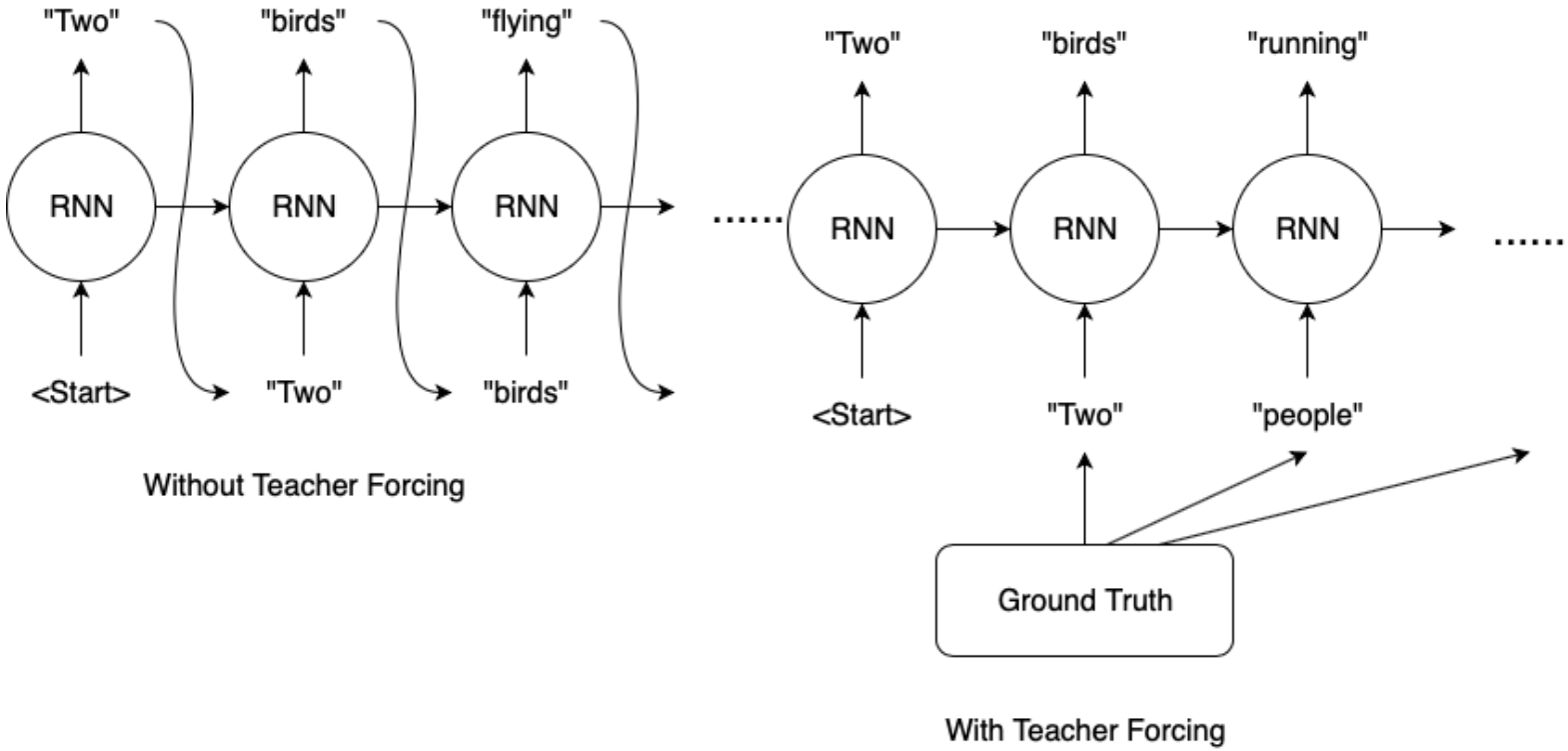


Figure: Teacher Forcing in Decoder Side, Source

Teacher Forcing

- Teacher forcing is a method for training recurrent neural networks more efficiently.
- Teacher forcing works by using the actual output at the current time step $y^{(t-1)}$ as input in the next time step, rather than the $o^{(t-1)}$ generated by the network.

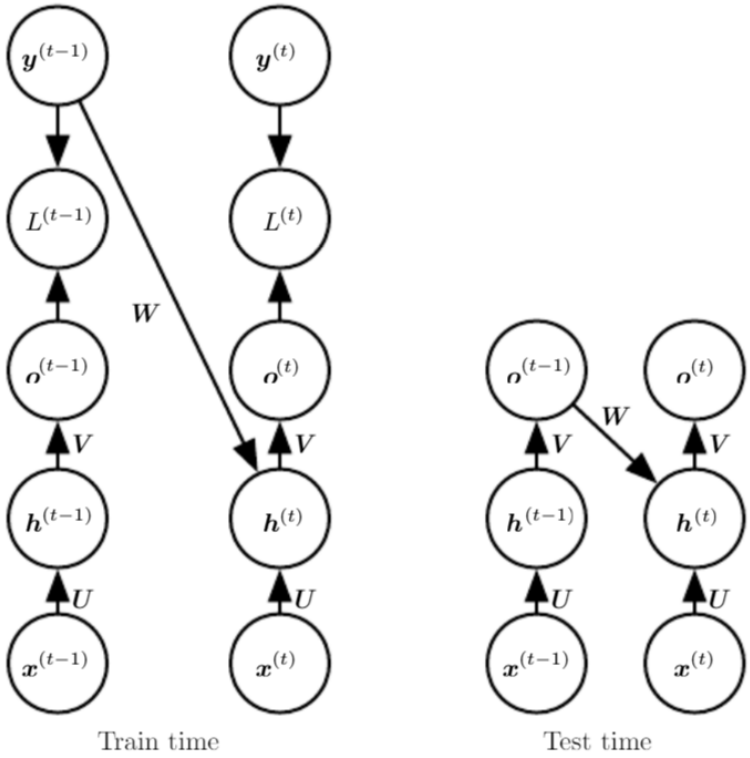


Figure: Teacher Forcing, Source

Teacher Forcing

- At the early stages of training, the predictions of the model are very bad.
- If we do not use Teacher Forcing, the hidden states of the model will be updated by a sequence of wrong predictions, errors will accumulate.
- It will be difficult for the model to learn from that.
- This technique allows us to prevent backpropagation through time which was complex and time-consuming. With teacher forcing the model will be trained faster .

Search Strategy

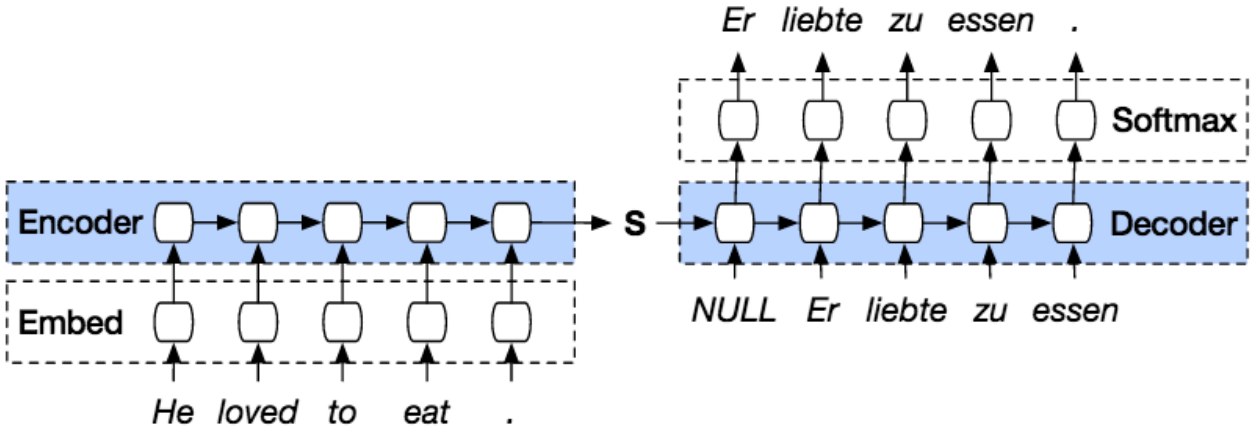


Figure: Simple Machine Translation Model, Source

- By now, we have discussed some important elements of sequence-to-sequence models (e.g. Machine Translation), including word embedding, LSTM, encoder, decoder, context vector, and softmax.
- But, there is a missing part! The search strategy for inference.

$$\hat{y}_T, \hat{y}_{T-1}, ..., \hat{y}_1 = \underset{\tilde{y}_T, \tilde{y}_{T-1}, ..., \tilde{y}_1}{argmax} P(\tilde{y}_T, \tilde{y}_{T-1}, ..., \tilde{y}_1 | S)$$

Notice: $\tilde{y}_t$ is the corresponding word to time step t.

Search Strategy

- **Exhaustive Search (Brute-force Search)**
  Iterate over all possible combinations of $\tilde{y}_t, \tilde{y}_{t-1}, \ldots, \tilde{y}_1$.
  - T: Total Time Step (the number of decoder units)
  - V: Vocabulary Size (the number of possible words)
  - Time Complexity: $O(V^T)$
  - So, it's not feasible.

Search Strategy

Greedy Search

$$max\, P(y_T, y_{T-1}, ..., y_1|S) = max \prod_{t=1}^{T} P(y_t|y_{t-1}, y_{t-2}, ..., y_1, S)$$

$$max \prod_{t=1}^{T} P(y_t|y_{t-1}, y_{t-2}, ..., y_1, S) \approx \prod_{t=1}^{T} max\, P(y_t|y_{t-1}, y_{t-2}, ..., y_1, S)$$

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

Figure: Greedy Search Example, Source

$$P((A, B, C, eos)|S) = P(A|(), S)P(B|(A), S)P(C|(A, B), S)P(eos|(A, B, C), S)$$

$$= 0.048$$

Search Strategy

Greedy Search

- Advantages
  - Time complexity: $O(TV)$
  - Sometimes it's a good approximation

- Disadvantages
  - It is somehow too naive.
  - It can be very inaccurate.

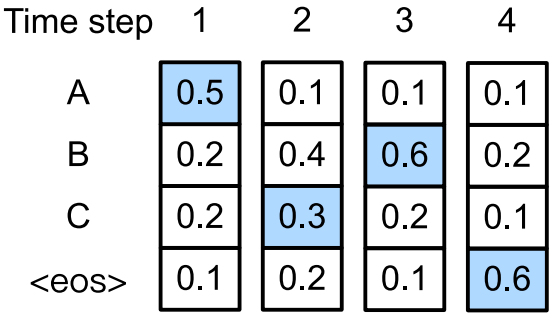


Figure: Greedy Search Example, Source

$$P((A,C,B,eos)|S) = P(A|(),S)P(C|(A),S)P(B|(A,C),S)P(eos|(A,C,B),S)$$

$$= 0.054$$

Beam Search

- **Beam Search**

  It's something between the two previous methods.

  - Keeping a number of candidates (beam width) instead of one in Greedy Search and all of the combinations in Exhaustive Search.

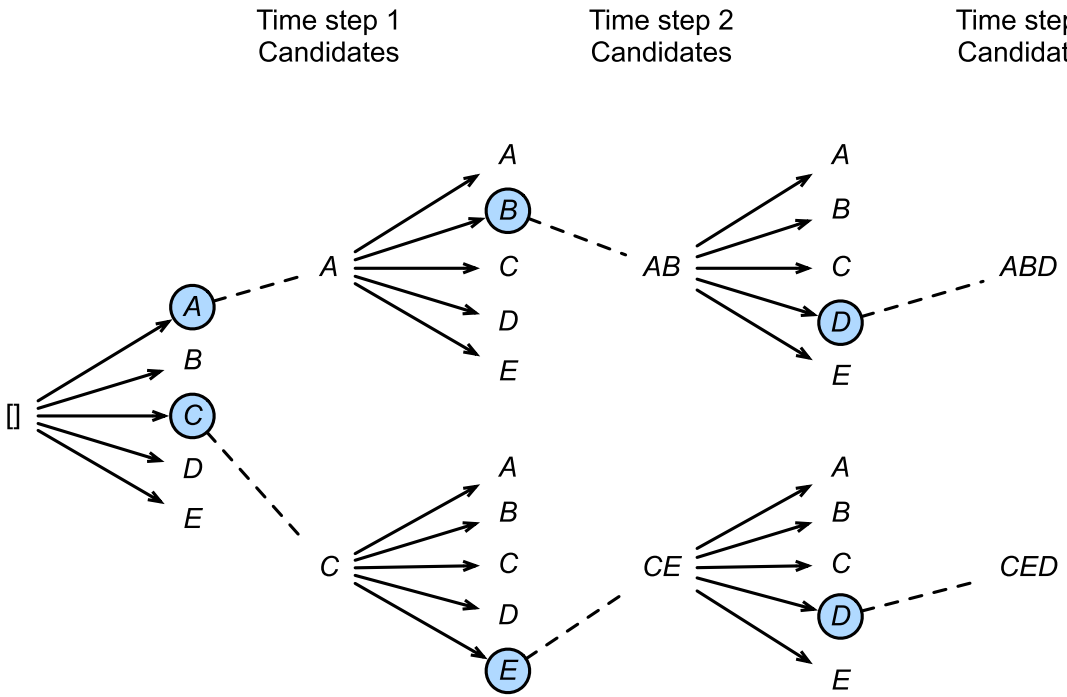


Figure: A Beam Search example in which beam width equals 2, Source

Beam Search

- **Beam Search**

  It's something between the two previous methods.

  - k: Beam Width
  - Time Complexity: $O(log(k)kVT)$ (By using max heap in each time step)

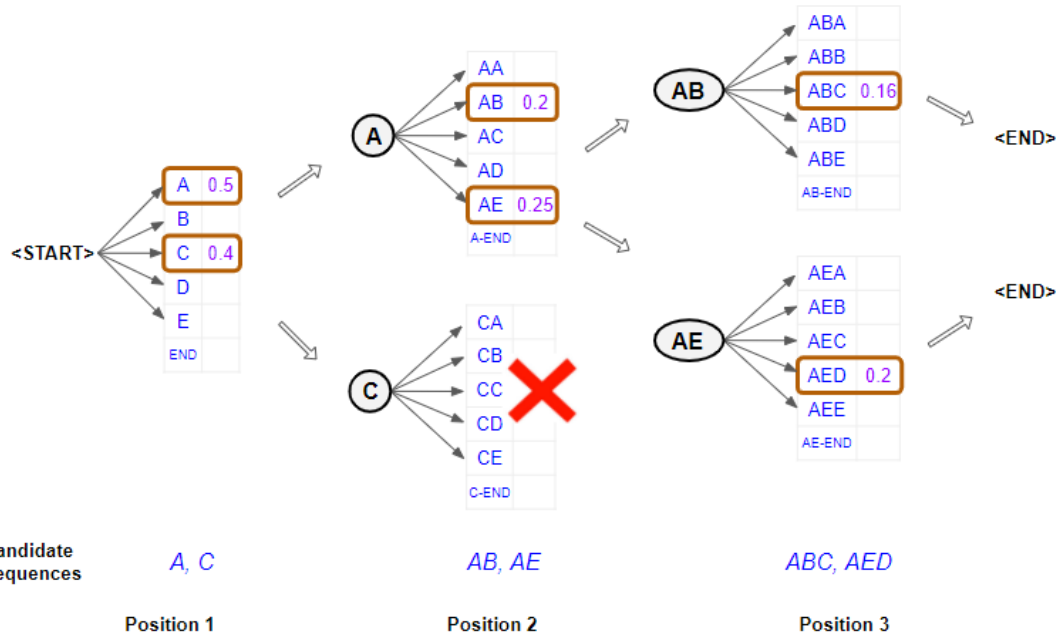


Figure: Another Beam Search example in which beam width equals 2, Source

References

Thank You!

Any Question?