INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Recurrent Neural Network

IA Frameworks

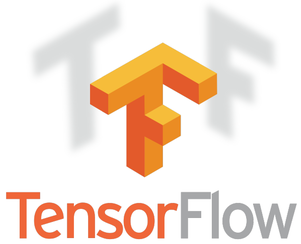
Brendan Guillouet - 16/11/2020

Institut National des Sciences Appliquées de Toulouse

Table of contents

Google cloud platform

ML Python Libraries



Python Environment

Viz' Python Libraries

Framework & Tool

Introduction

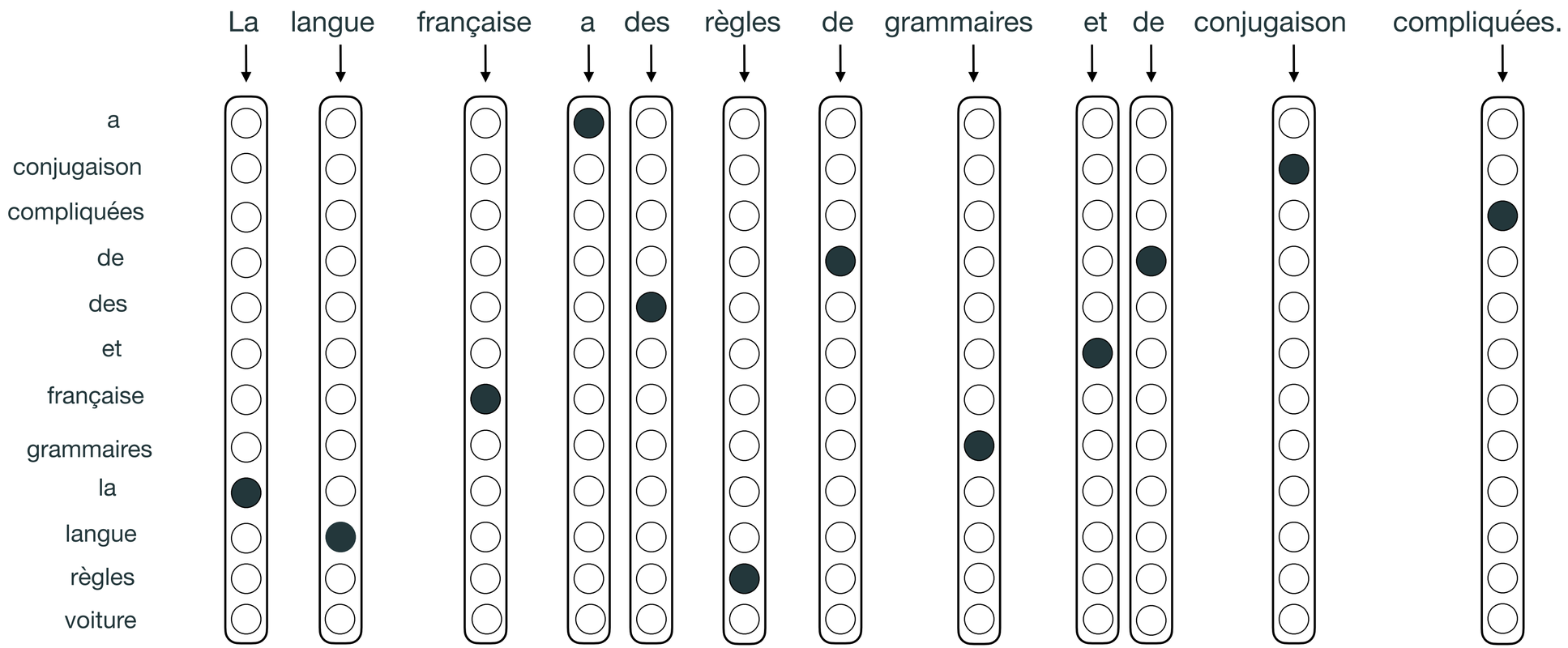
Motivations

Problem: Various text problem required to handle sequence data.

Type	X	Y
Text Generation	empty, scalar Ø,1,2	text sequence *this is a text generated*
Sentiment Classification	text sequence *It wasn't that good*	empty, scalar 2(/5)
Translation	text sequence *How are you?*	text sequence *Comment tu vas?*
Identity Recognition	text sequence *Harry potter is a wizard*	boolean sequence [1,1,0,0,0]

Motivations

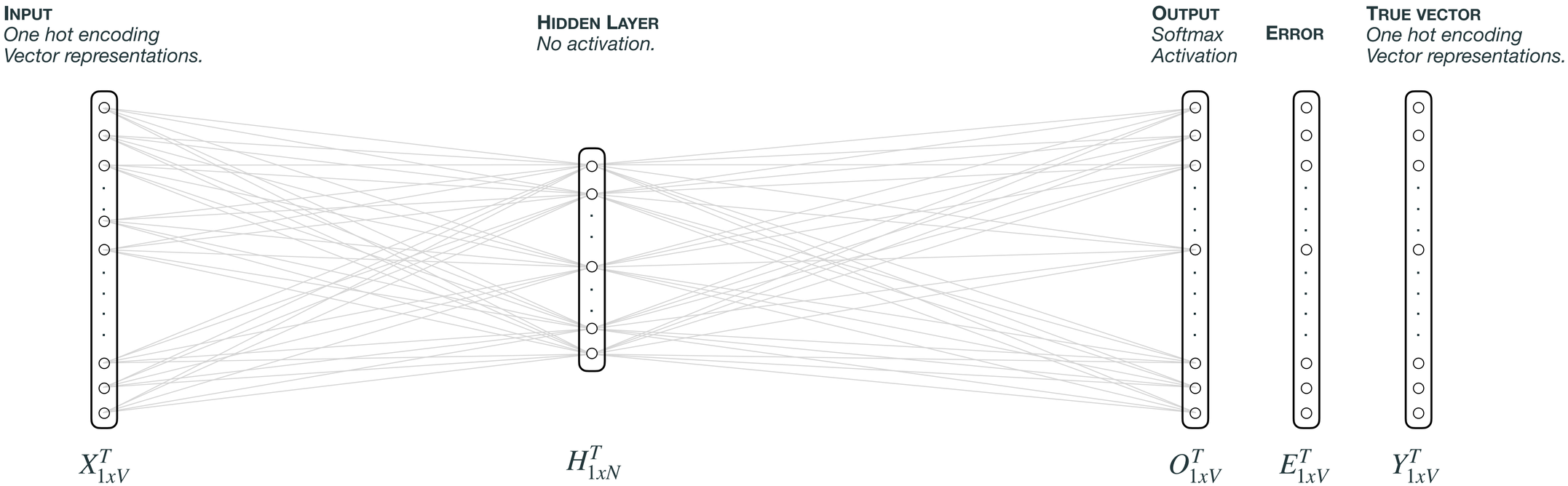
Machine learning and Deep learning algorithms are not adapted !



Dictionary size: V = 12

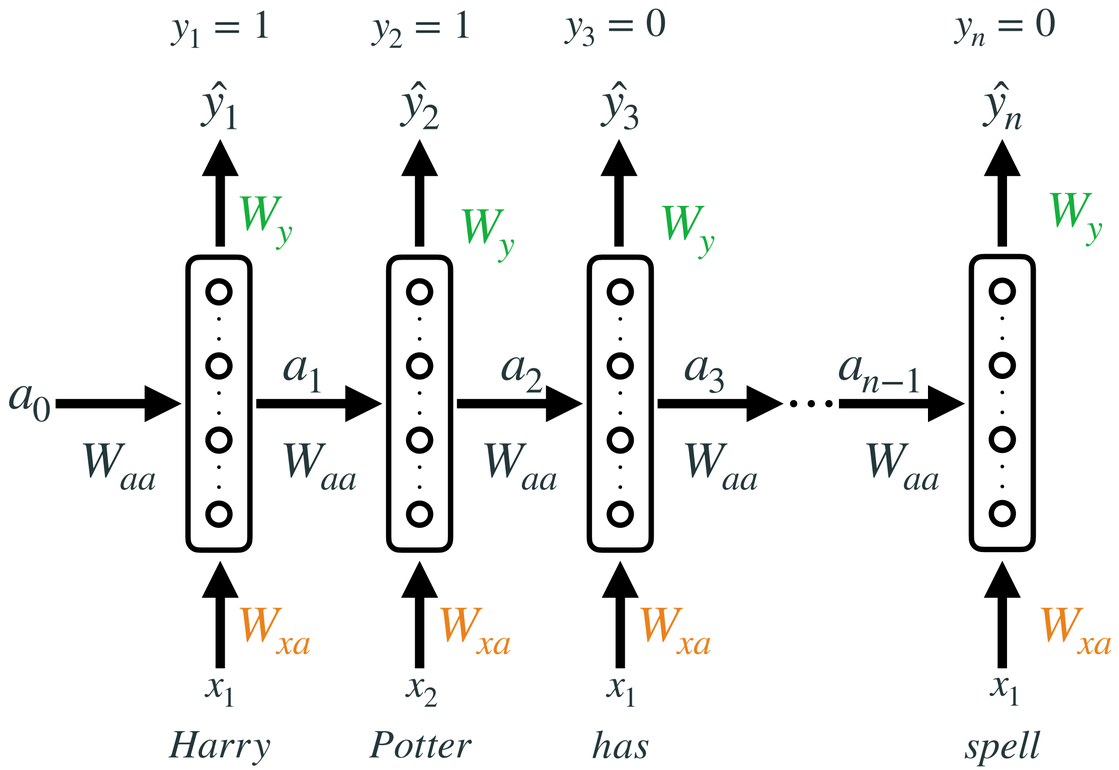
Motivations

- Size of sequences varies from one example to an other.
- Information between words like their position are not used.



RNN

RNN - Notation - Identity Recognition



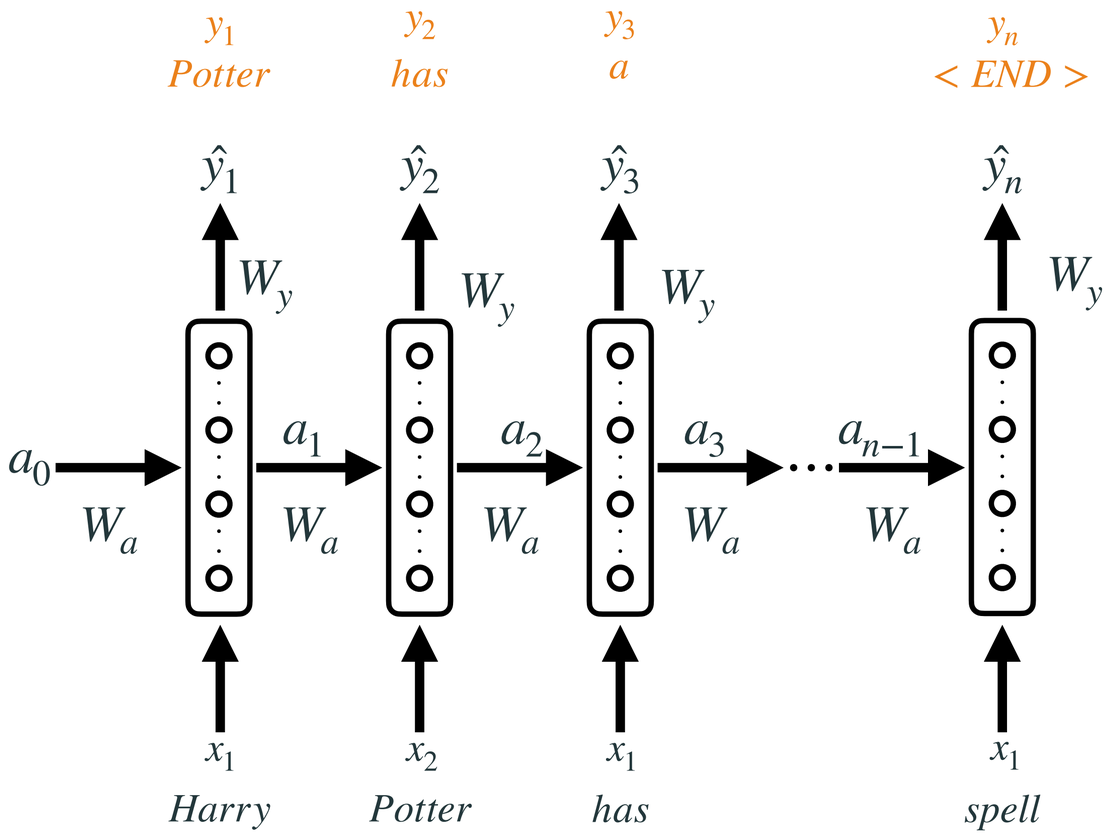
Activation Function

$$a_t = g_a(a_{t-1}W_{aa} + x_t W_{xa} + ba)$$
$$= g_a([a_{t-1}, x_t] \cdot [W_{aa} + W_{xa}]^T + ba)$$
$$= g_a([a_{t-1}, x_t]W_y + b_y)$$

$$\hat{y} = g_y(a_t \cdot W_{xa} + by)$$

- $x_t, (1 \times V)$: OHE representation.
- V : Dictionary's size.
- $y_t(1 \times 1)$: 0 or 1.
- $a_t : 1 \times H.$
- H : Number of neutrons.
- N : Number of timestep (Not layer!).
- g_a : Activation Function (**tanh/Relu**).
- g_y Activation Function (**sigmoid**).
- $W_{aa}, (H \times H, \forall t \in [1,...,n])$
- $W_{xa}, (V \times H, \forall t \in [1,...,n])$
- $W_y, (H \times 1, \forall t \in [1,...,n])$
- $W_a, (H + V \times H, \forall t \in [1,...,n])$

RNN - Notation - Text Generation

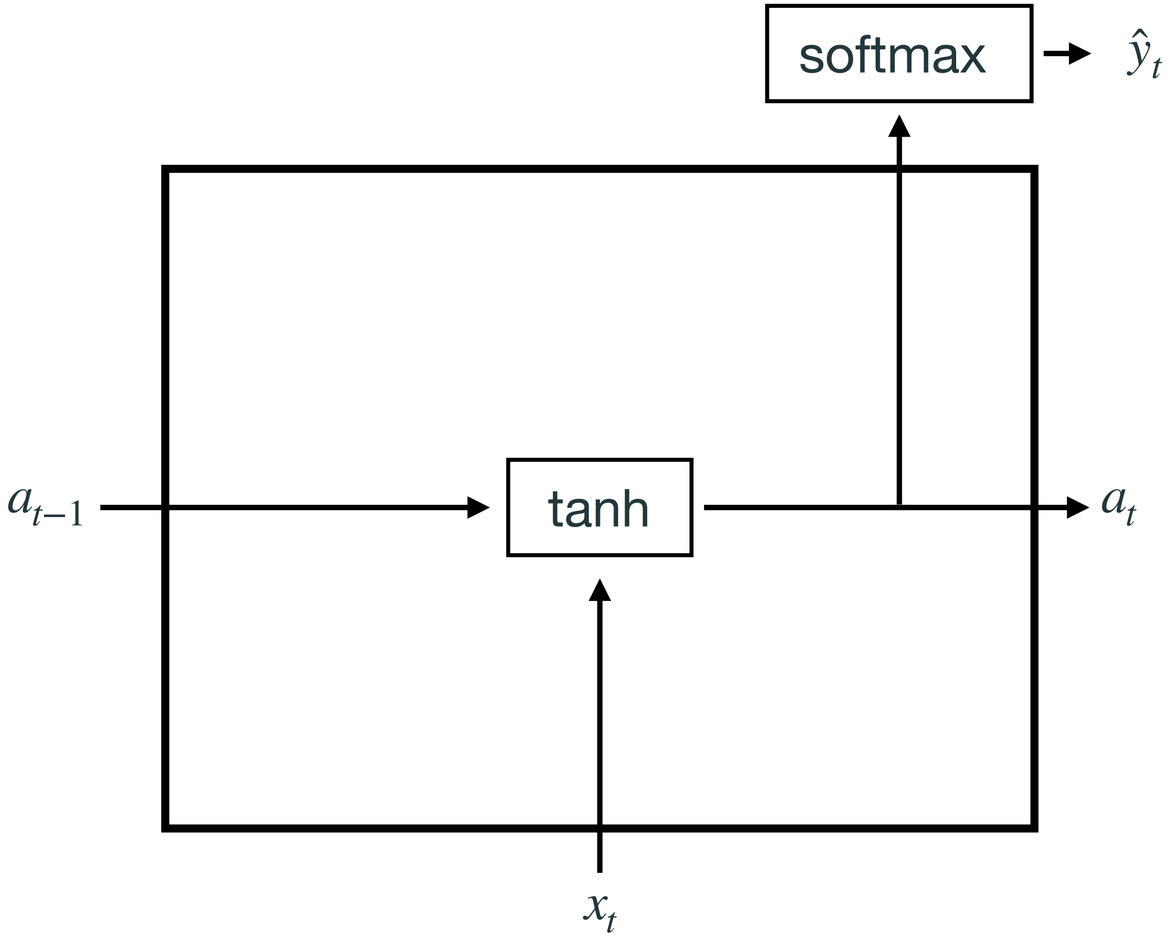


Activation Function

$$a_t = g_a([a_{t-1}, x_t] \cdot W_a + ba)$$
$$\hat{y} = g_y(a_t \cdot W_y + by)$$

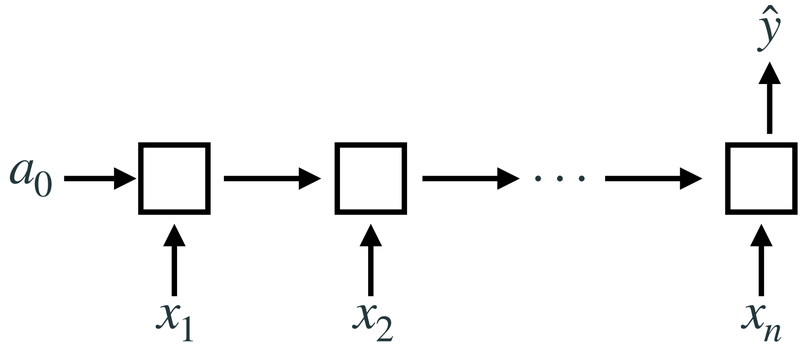
- $x_t, (1 \times V)$: OHE representation.
- V : Dictionary's size.
- $y_t, \hat{y}(1 \times V)$: OHE representation.
- $a_t : 1 \times H.$
- H : Number of neutrons.
- N : Number of timestep (Not layer!).
- g_a : Activation Function (**tanh/Relu**).
- g_y Activation Function (**Softmax**).
- $W_y, (H \times V, \forall t \in [1,...,n])$
- $W_a, (H + V \times H, \forall t \in [1,...,n])$

RNN Unit



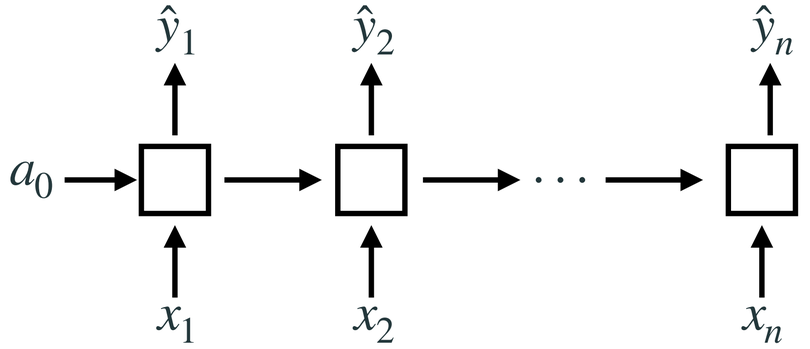
- $a_t = tanh([a_{t-1}, x_t]W_a + b_a)$.
- $\hat{y}_t = softmax(a_t W_y + b_y)$.

Different Type of RNN



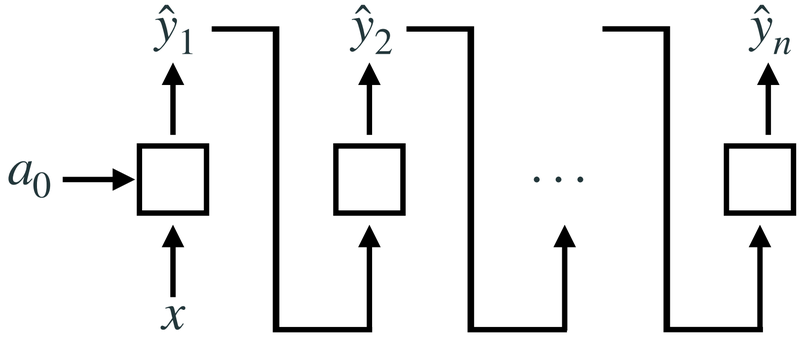
Many To One

Text classification



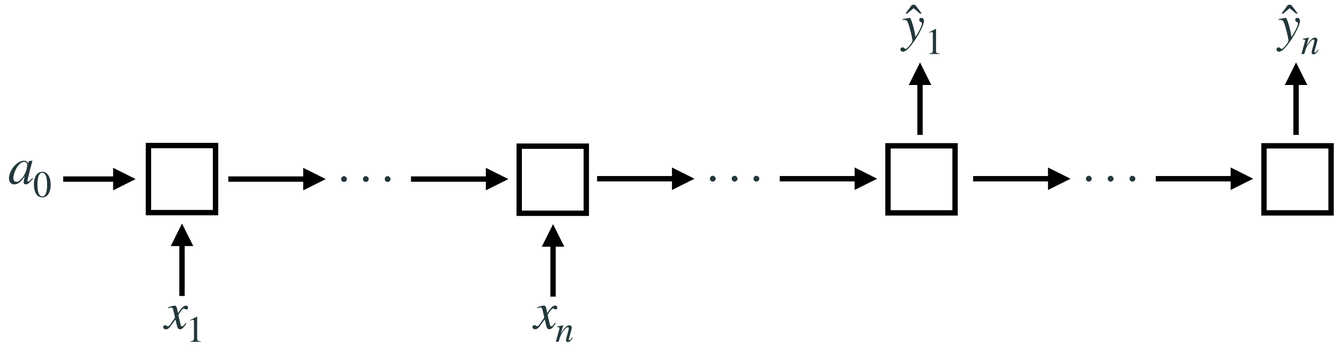
Many To Many

Identity recognition



One To Many

Text generation



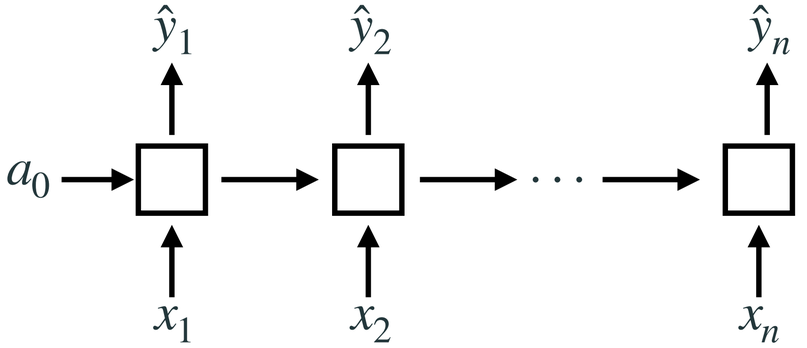
Many To Many

Automatic translation

Different Type of RNN

```
model = km.Sequential()
model.add(kl.SimpleRNN(units=10 ,activation="relu", input_shape=(3, 1)))
model.add(kl.Dense(1))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_2 (SimpleRNN)     (None, 10)                120
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

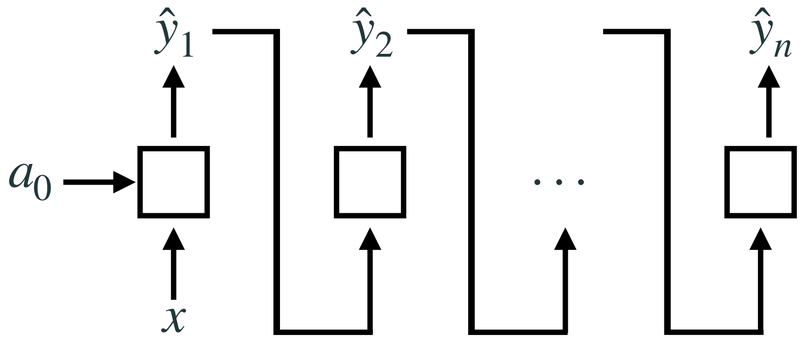


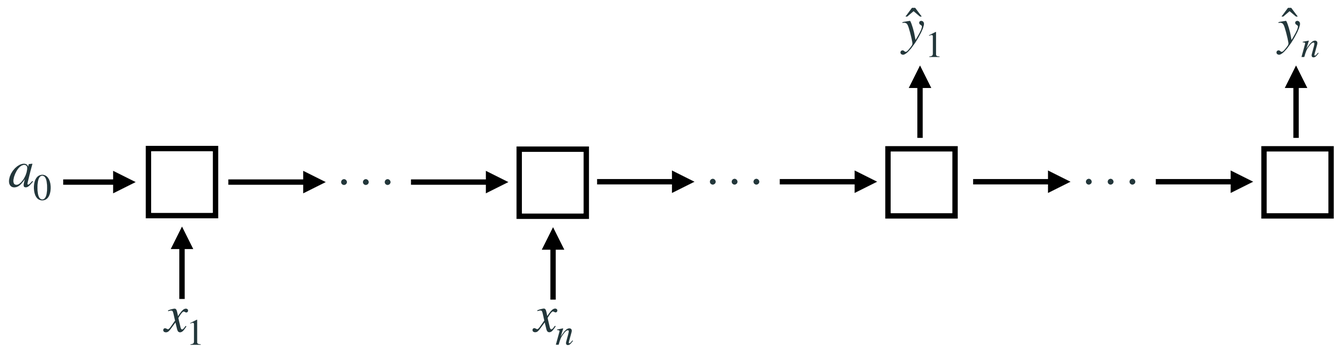
Many To One

Text classification

Many To Many

Identity recognition





One To Many

Text generation

Many To Many

Automatic translation

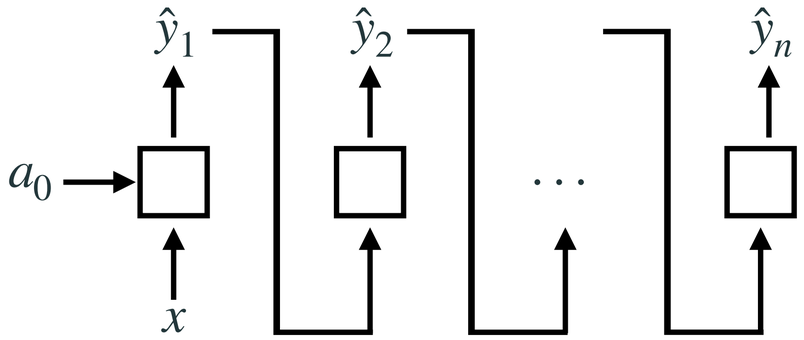
Different Type of RNN

```
model = km.Sequential()
model.add(kl.SimpleRNN(units=10 ,activation="relu", input_shape=(3, 1)))
model.add(kl.Dense(1))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_2 (SimpleRNN)     (None, 10)                120
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

Many To One

Text classification



One To Many

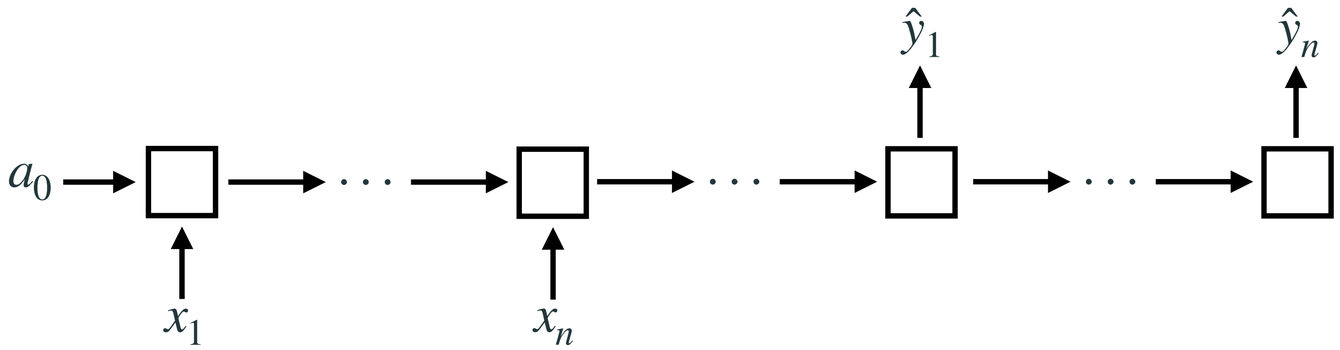
Text generation

```
model = km.Sequential()
model.add(kl.SimpleRNN(units=10 ,activation="relu", input_shape=(3, 1),
return_sequences=True))
model.add(kl.TimeDistributed(kl.Dense(1)))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_4 (SimpleRNN)     (None, 3, 10)             120
_________________________________________________________________
time_distributed (TimeDistri (None, 3, 1)              11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

Many To Many

Identity recognition



Many To Many

Automatic translation

Different Type of RNN

```
model = km.Sequential()
model.add(kl.SimpleRNN(units=10 ,activation="relu", input_shape=(3, 1)))
model.add(kl.Dense(1))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_2 (SimpleRNN)     (None, 10)                120
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

Many To One

Text classification

```
model = km.Sequential()
model.add(kl.SimpleRNN(units=10 ,activation="relu", input_shape=(3, 1),
return_sequences=True))
model.add(kl.TimeDistributed(kl.Dense(1)))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_4 (SimpleRNN)     (None, 3, 10)             120
_________________________________________________________________
time_distributed (TimeDistri (None, 3, 1)              11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

Many To Many

Identity recognition

```
model = km.Sequential()
model.add(kl.SimpleRNN(units=10 ,activation="relu", input_shape=(3, 1),
return_sequences=True))
model.add(kl.TimeDistributed(kl.Dense(1)))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_4 (SimpleRNN)     (None, 3, 10)             120
_________________________________________________________________
time_distributed (TimeDistri (None, 3, 1)              11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
_________________________________________________________________
```

One To Many

Text generation

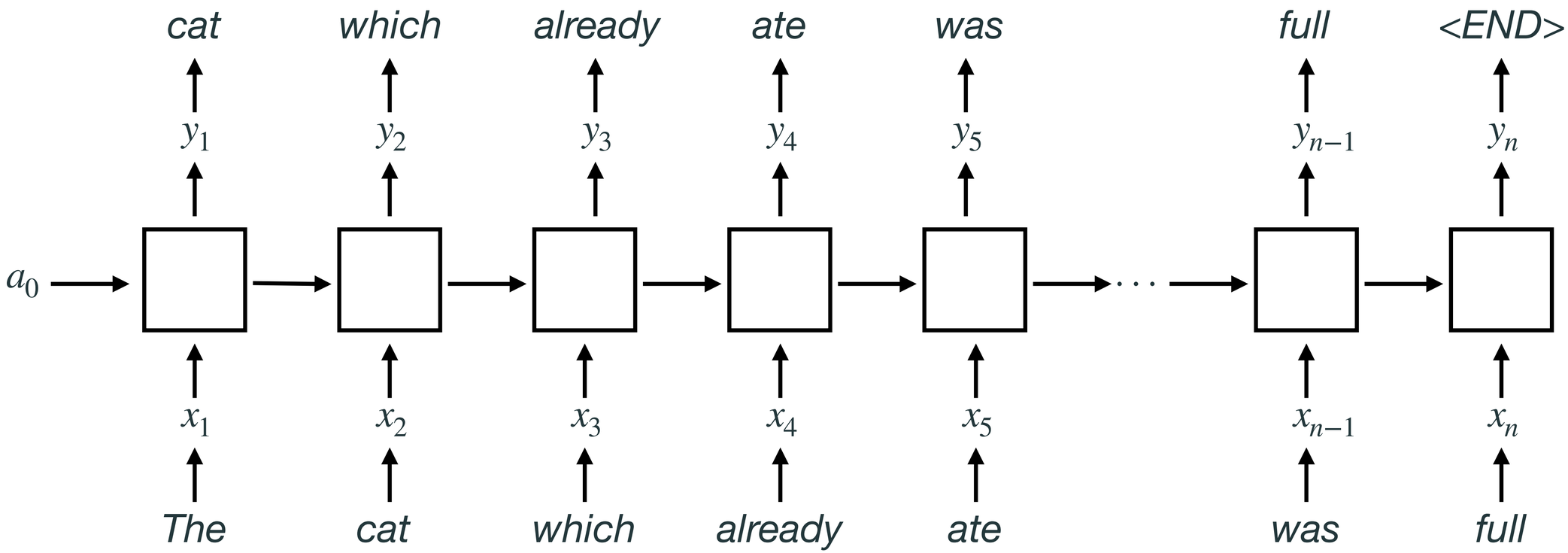
Many To Many

Automatic translation

Memory Cell

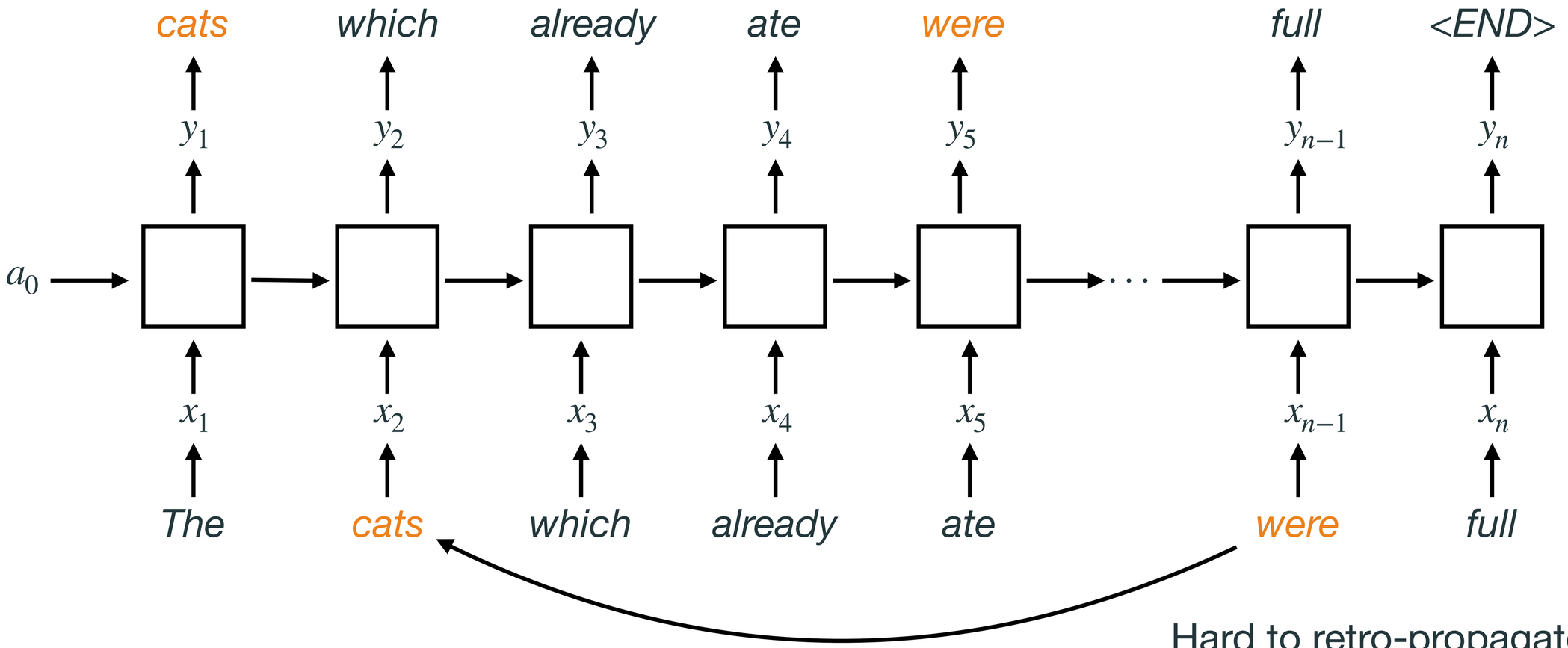
Vanishing Gradient

Problem : Long time dependency. *The cat, which already ate …, was full.*



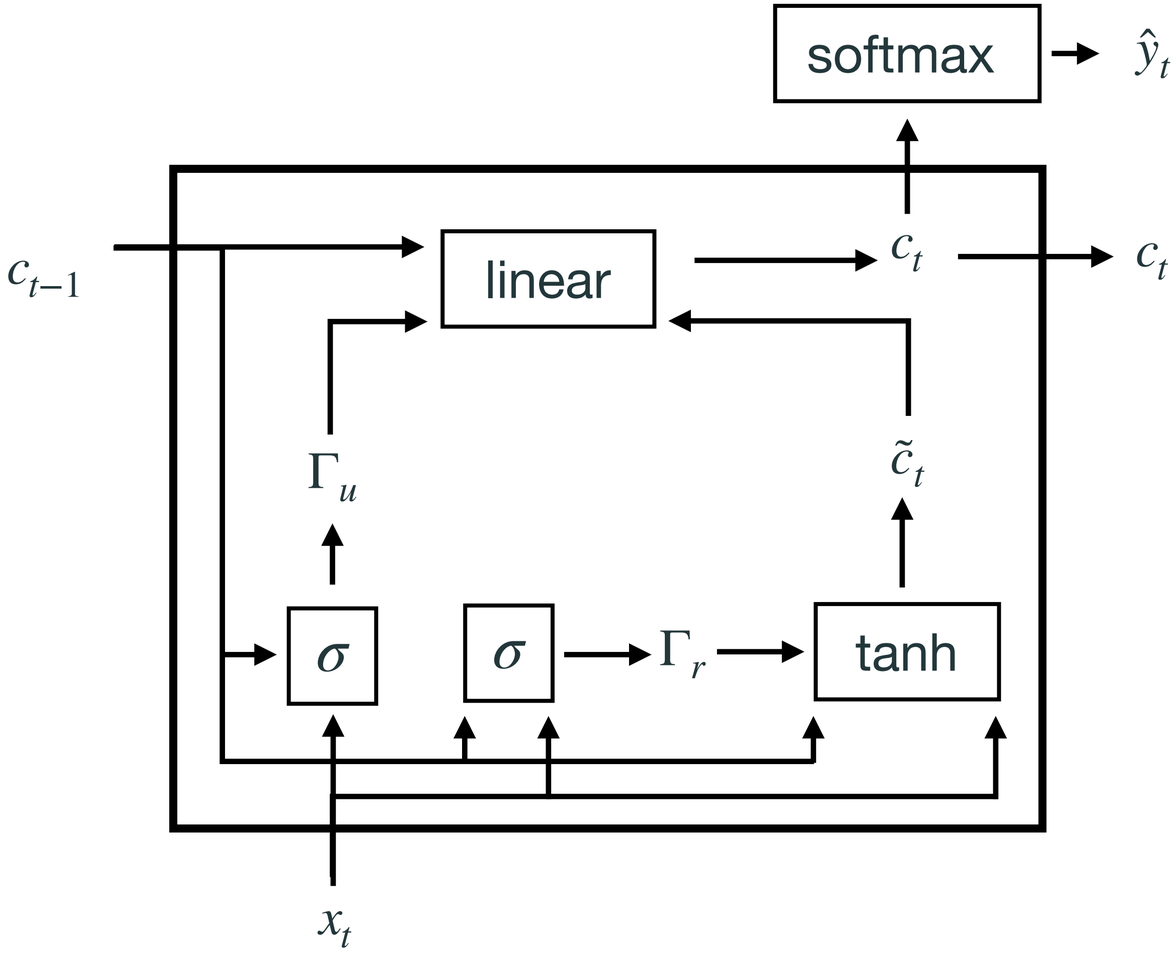
Vanishing Gradient

Problem : Long time dependency. *The cats, which already ate …, were full.*



Solution : Memory cell.

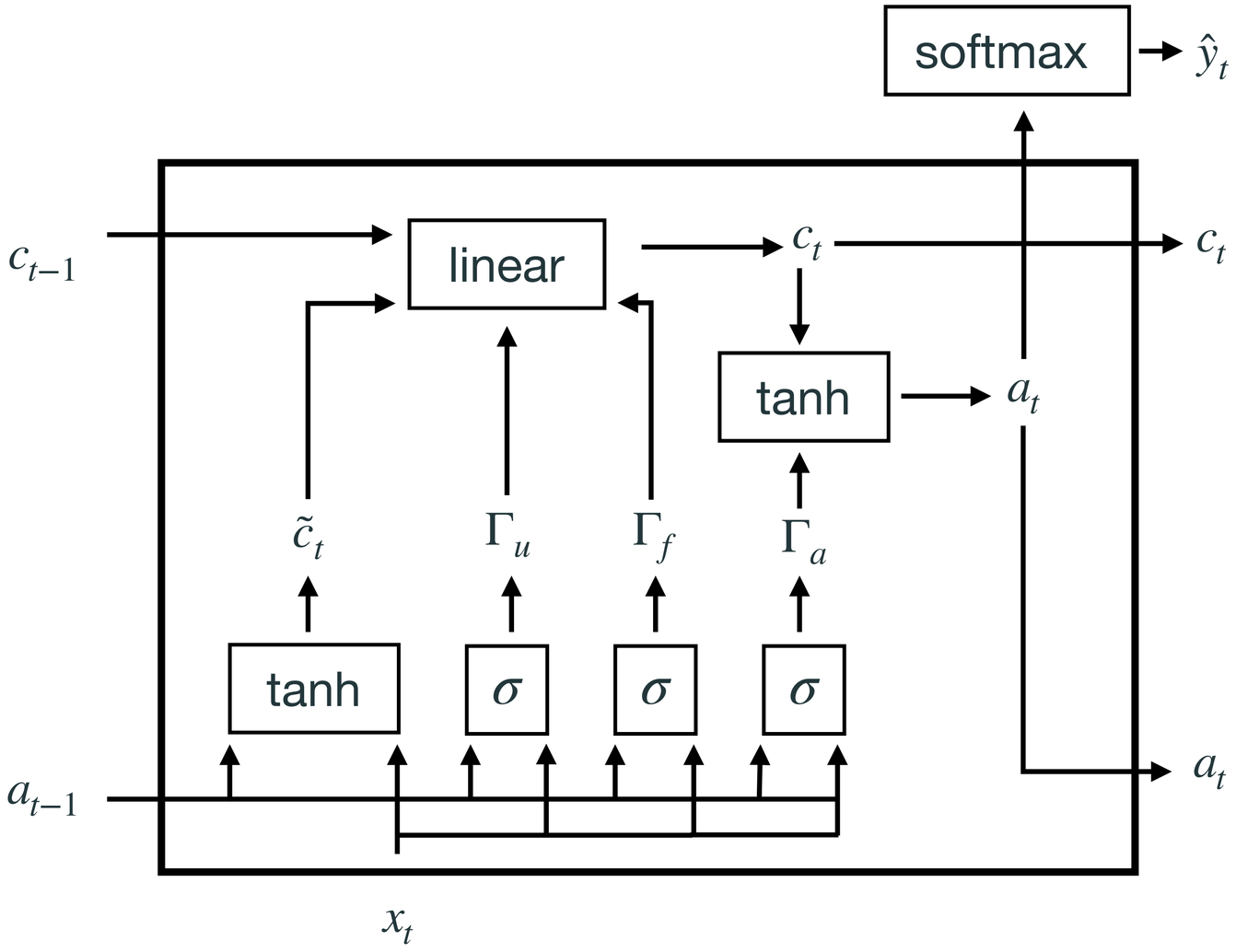
GRU Unit - Cho et al. [2014], Chung et al. [2014]



- c_t : memory cell.
- $c_t = a_t$.
- $\Gamma_r = \sigma([c_{t-1}, x_t]W_r + b_r)$.
- $\tilde{c}_t = tanh([\Gamma_r * c_{t-1}, x_t]W_c + b_c)$.
- $\Gamma_u = \sigma([c_{t-1}, x_t]W_u + b_u)$.
- $c_t = \Gamma_u * \tilde{c}_t + (1 - \Gamma_u) * c_{t-1}$.
- $\hat{y}_t = softmax(c_t W_y + b_y)$.

the	cats	which	already	ate	$\cdots$	were	full
$c_1^i = 0$	$c_2^i = 1$	$c_1^i = 1$	$c_1^i = 1$	$c_1^i = 1$	$\cdots$	$c_1^i = 0$	$c_1^i = 0$
$\Gamma_u^i = 0$	$\Gamma_u^i = 1$	$\Gamma_u^i = 0$	$\Gamma_u^i = 0$	$\Gamma_u^i = 0$	$\cdots$	$\Gamma_u^i = 1$	$\Gamma_u^i = 0$

LSTM Unit - Hochreiter and Schmidhuber [1997]



- c_t : memory cell.
- $c_t \neq a_t$.
- $\tilde{c} = tanh([a_{t-1}, x_t]W_a + b_a)$.
- $\Gamma_u = \sigma([a_{t-1}, x_t]W_u + b_u)$.
- $\Gamma_f = \sigma([a_{t-1}, x_t]W_u + b_f)$.
- $c_t = \Gamma_u * \tilde{c}_t + \Gamma_f * c_{t-1}$.
- $\Gamma_o = \sigma([a_{t-1}, x_t]W_r + b_o)$.
- $a_t = \Gamma_o * tanh(c_t)$.
- $\hat{y}_t = softmax(a_t W_y + b_y)$.

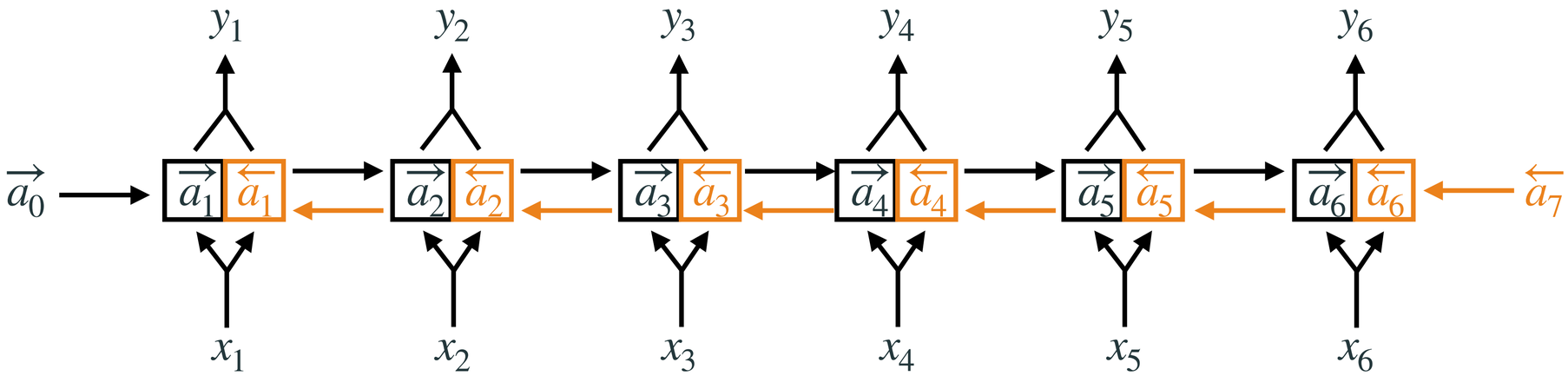
Cola's Blog

Other Properties

Bidirectional RNN (BRNN)

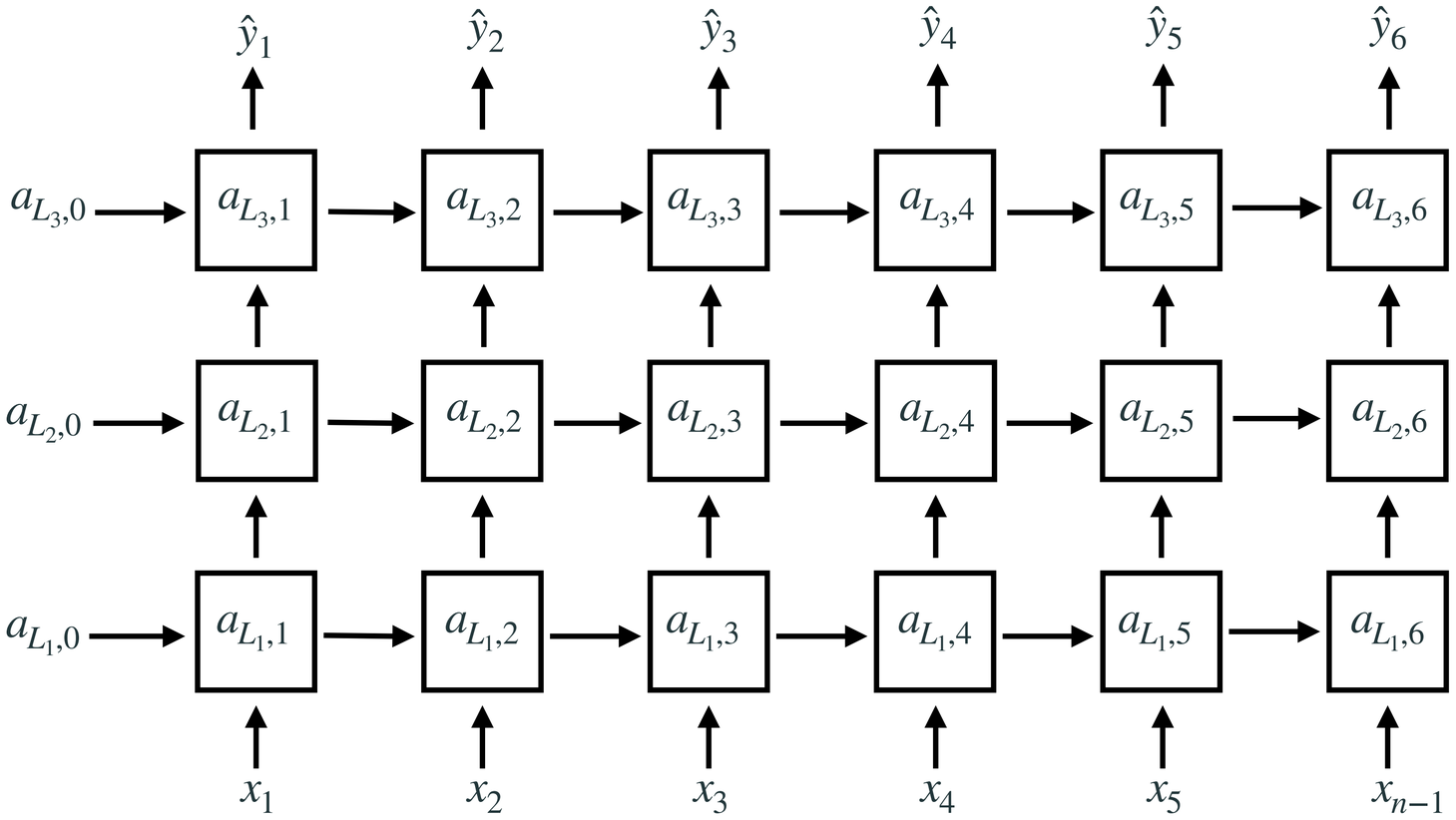
Problem : Long time dependency.

Example (Identity recognition): *Holland is a European's country*.
Holland is a European's president.



$$\overrightarrow{a_t} = tanh([\overrightarrow{a}_{t-1}, x_t]W_{\overrightarrow{a}} + b_{\overrightarrow{a}})$$
$$\overleftarrow{a_t} = tanh([\overleftarrow{a}_{t-1}, x_t]W_{\overleftarrow{a}} + b_{\overleftarrow{a}})$$
$$\hat{y} = softmax([\overrightarrow{a}_t, \overleftarrow{a}_t]W_y + b_y))$$

Deep RNN



$$a_{L_1,t} = tanh([a_{L_1,t-1}, x_t]W_{L_1,a} + b_{L_1,a})$$

$$a_{L_i,t} = tanh([a_{L_i,t-1}, a_{L_{i-1},t}]W_{L_i,a} + b_{L_i,a}), \forall i > 1$$

$$\hat{y} = softmax(a_{L3,t}W_y + b_y))$$

Deep Bidirectional RNN With Tensorflow

```
model = km.Sequential()
model.add(kl.LSTM(units=10 ,activation="relu", input_shape=(3, 1), return_sequences=True))
model.add(kl.Bidirectional(kl.GRU(units=10 ,activation="relu", return_sequences=True)))
model.add(kl.TimeDistributed(kl.Dense(1)))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 3, 10)             480
_________________________________________________________________
bidirectional (Bidirectional (None, 3, 20)             1320
_________________________________________________________________
time_distributed_2 (TimeDist (None, 3, 1)              21
=================================================================
Total params: 1,821
Trainable params: 1,821
Non-trainable params: 0
_________________________________________________________________
```

TP

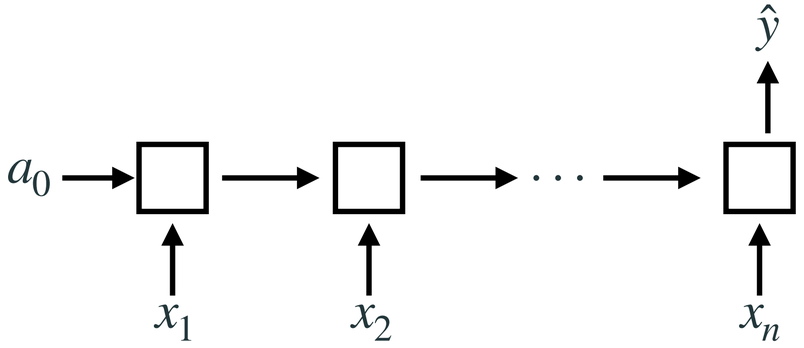
Example: Text Classification

$x =$ Pour apple iPhone 4: coque bumper silicone blanc - cet étui en silicone rigide..

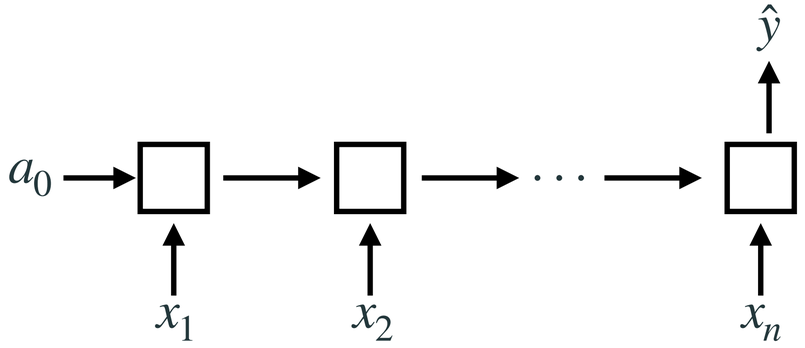
$x = [\ x_1\ ,\ x_2\ ,\ x_3\ ,\ x_4\ , \ldots ,\ x_n\ ]$

$x = [start, OHE(P), OHE(o), OHE(u), \ldots, End]$

$vocabulary = [a, \ldots, z, A, \ldots, Z, 0, ..., 9, ?, ;, ., \ldots, /, +\ ]$



Many To One
Learning



Many To One
Prediction

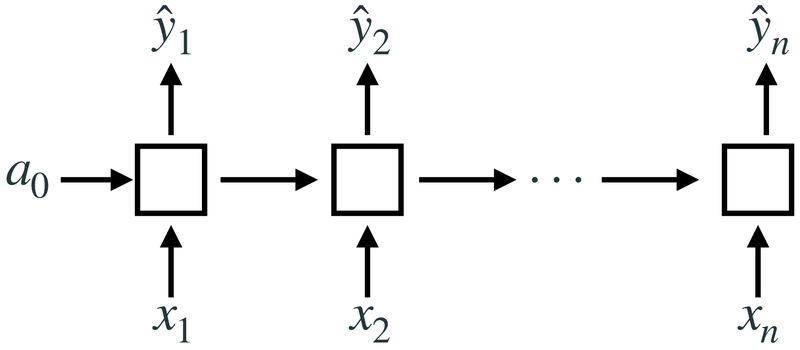
Example: Text Generation

$x =$ Pour apple iPhone 4: coque bumper silicone blanc - cet étui en silicone rigide..

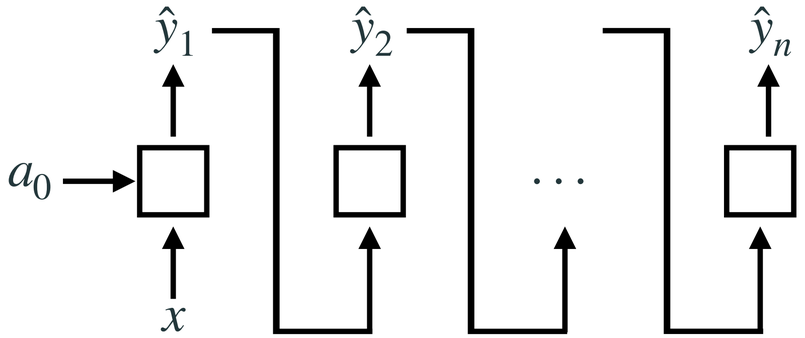
$x = [\ x_1\ ,\ x_2\ ,\ x_3\ ,\ x_4\ , \ldots ,\ x_n\ ]$

$x = [start, OHE(P), OHE(o), OHE(u), \ldots, End]$

$vocabulary = [a, \ldots, z, A, \ldots, Z, 0, ..., 9, ?, ;, ., \ldots, /, +\ ]$



Many To Many
Learning



One To Many
Generation