Deep Learning

6/ Sequence Models

Francesca Galassi, MCF, Esir

francesca.galassi@irisa.fr
Lab Empenn Irisa-Inria

Sequences

What are Sequence Models ?

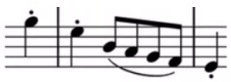
- Sequence models are designed for handling sequential data, i.e., where the order of elements matters.
- Sequence models process input data **sequentially**, considering the context of previous elements to generate outputs.
- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are specialized types of neural networks for processing sequential data.
- Common applications include **time series prediction**, and Natural Language Processing (**NLP**) tasks.

Applications of Sequence Models

- **Natural Language Processing (NLP) tasks :**
  - **Machine translation :** Translates between languages.
  - **Named-entity recognition :** Identifies people, countries, etc. in a sentence.
  - **Sentiment analysis :** Analyzes input phrases (x) to predict sentiment (y).
- **Speech recognition :** Maps audio (x) to text (y).

"The quick brown fox jumped over the lazy dog."

- **Music generation :** Creates music sequences.



- **DNA sequence analysis :** Identifies protein-coding segments.
- **Video activity recognition :** Recognizes activities in videos.

Notation

➟ **Named-Entity Recognition :** Identifying people in a given sentence.

Yesterday, Harry Potter met Hermione Granger. $\longrightarrow$ Yesterday, Harry Potter met Hermione Granger.

➟ **Input** x **:**

Harry	Potter	and	Hermione	Granger	invented	a	new	spell
$x^{<1>}$	$x^{<2>}$			$\dots$				$x^{<9>}$

➟ **Output** y **:**

1	1	0	1	1	0	0	0	0
$y^{<1>}$	$y^{<2>}$			$\dots$				$y^{<9>}$

➟ Notation :

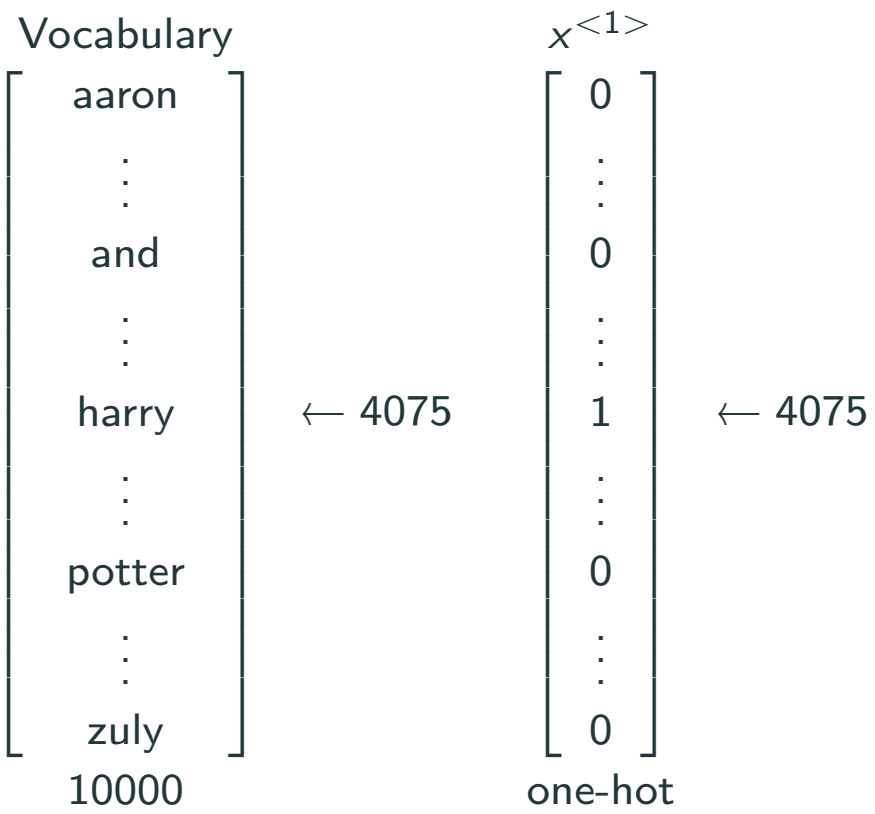
- $x^{(i)<t>}$ is the t-th word of the i-th training example
- $T_x^{(i)}$ is the length of the sequence
- $y^{(i)<t>}$ is the t-th entry in the output sequence of the i-th training example
- $T_y^{(i)}$ is the number of output entries

Representing Words

- Given a vocabulary, a word $x^{(i)<t>}$ in a sentence is represented as a one-hot vector.

"Harry Potter and Hermione Granger invented a new spell."

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$



Recurrent Neural Networks

RNNs vs Standard NNs

- **Why not use a standard NN ?**

$x^{<1>}$ $y^{<1>}$

$x^{<2>}$ $y^{<2>}$

$\vdots$ $\vdots$

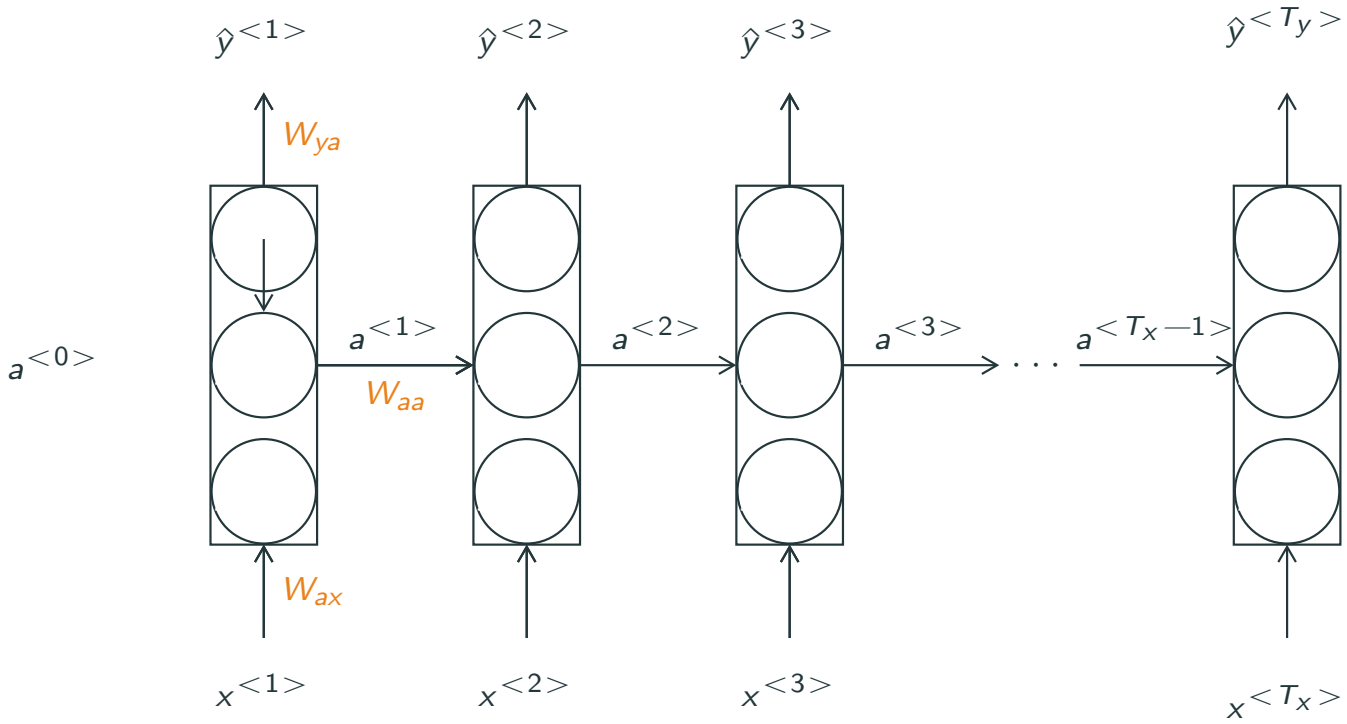
$x^{<T_x>}$ $y^{<T_y>}$

Problems :

- Inputs $x^{(i)<t>}$ (and outputs $y^{(i)<t>}$) can have different lengths in different examples i.
- Doesn't share features learned across different positions of text.
- Recurrent Neural Networks handle sequential data by maintaining a memory of past inputs.

Architecture

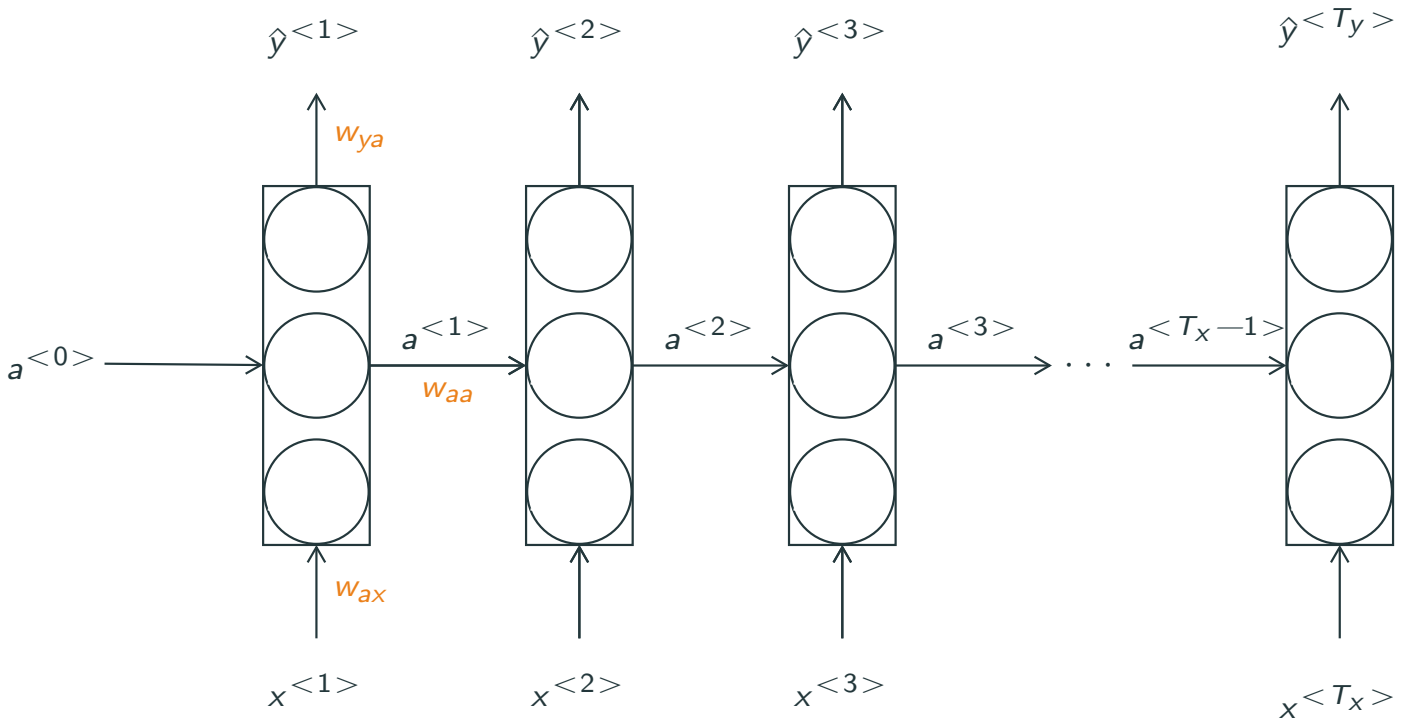
- RNN processes the sentence sequentially from left to right, word after word :



- Parameters for each time step are shared : W_{ax}, W_{aa}, W_{ay}
- Unidirectional : **only** information from earlier in the sequence used at a certain time ← Bidirectional RNN

  He said, "Teddy Roosevelt was a great President."
  He said, "Teddy bears are on sale !"

Computations : Forward Propagation



At each time step, t :

$$a^{<t>} = g\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right) \quad \leftarrow \tanh \mid \text{Relu}$$

$$\hat{y}^{<t>} = g\left(W_{ya}a^{<t>} + b_y\right) \quad \leftarrow \text{e. g. Sigmoid}$$

Computations : Forward Propagation

➟ Simplified RNN notation :

$$\boxed{\begin{aligned} a^{<t>} &= g\left(W_a[a^{<t-1>}, x^{<t>}] + b_a\right) \\ \hat{y}^{<t>} &= g\left(W_y a^{<t>} + b_y\right) \end{aligned}}$$

where W_a and W_y are weight matrices, $g(\cdot)$ is an activation function, and b_a and b_y are bias terms.

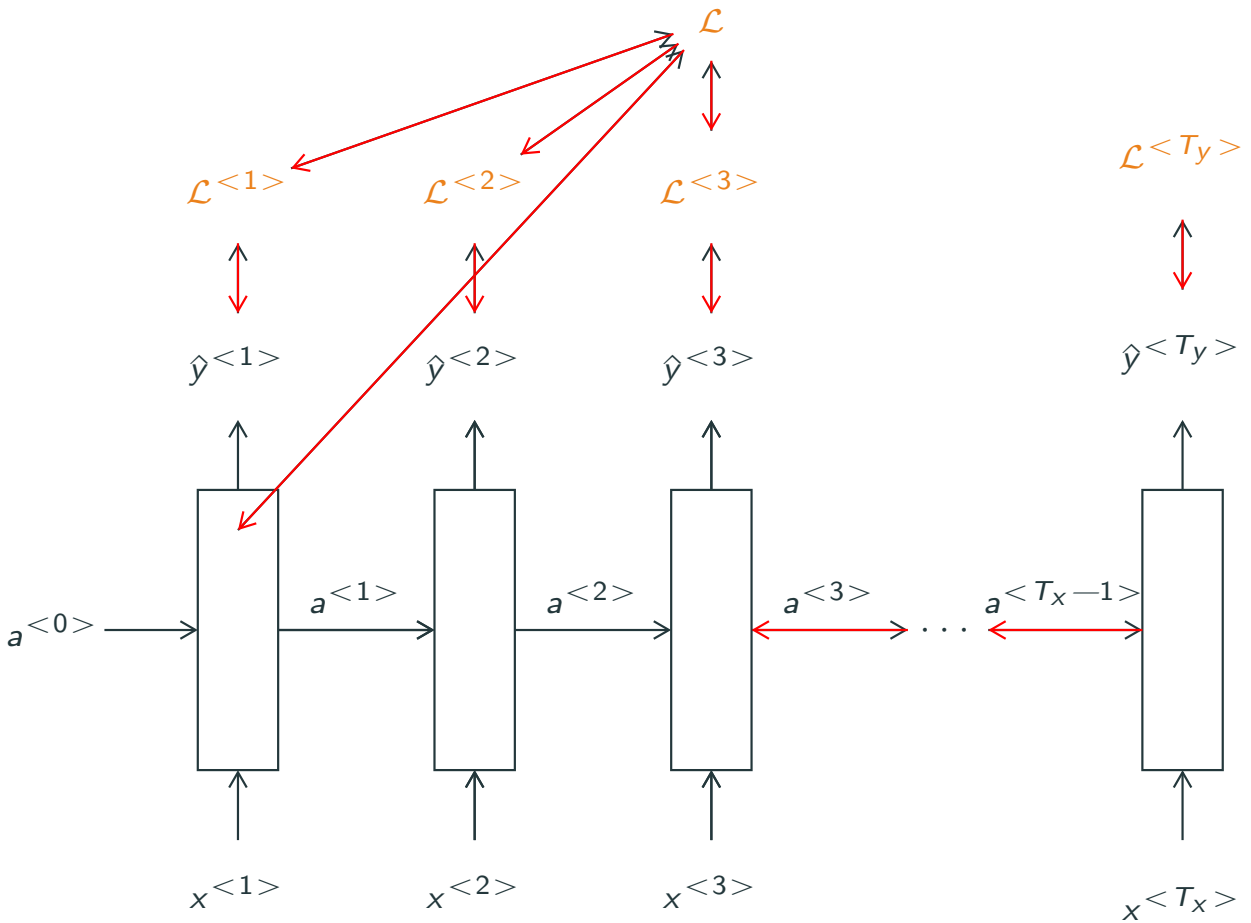
➟ Explanation :

$$a^{<t>} = g\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right) \quad \rightarrow \quad a^{<t>} = g\left(W_a[a^{<t-1>}, x^{<t>}] + b_a\right) *$$

$$\hat{y}^{<t>} = g\left(W_{ya}a^{<t>} + b_y\right)$$

$$* \; [W_{aa}, W_{ax}] = W_a \quad ; \quad [a^{<t-1>}; x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$$

Forward and Backward Propagation

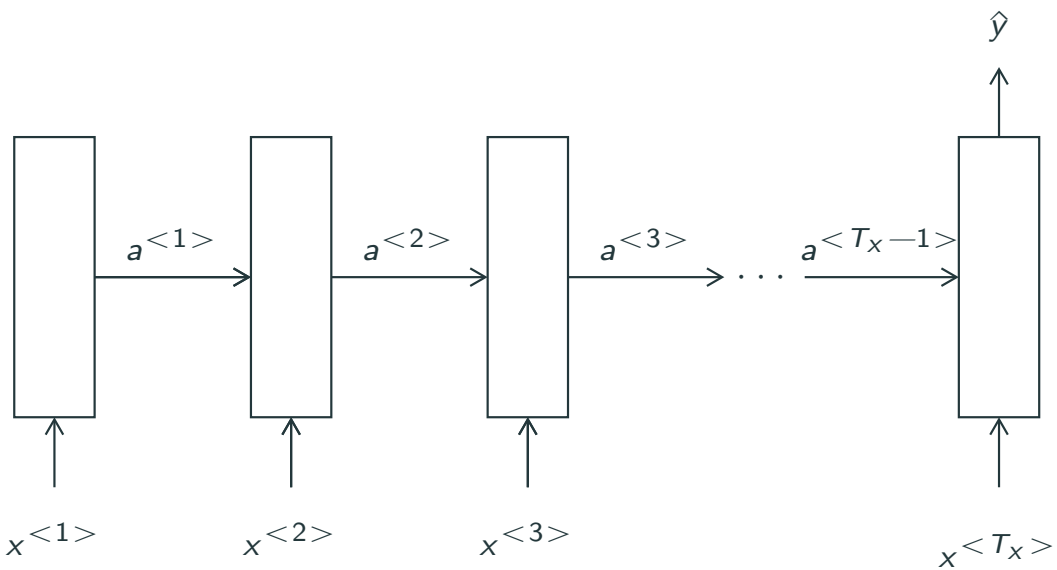


- Loss at each time step : $\mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right) = -y^{<t>} \log \hat{y}^{<t>} - \left(1 - y^{<t>}\right) \log \left(1 - \hat{y}^{<t>}\right)$
- Loss for the entire sequence : $\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T} \mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right)$
- Backpropagation through time (BPTT) : gradients are accumulated across both time steps and layers

Dirrerent types of RNNs

➟ **Many-to-one**, e.g., sentiment classification

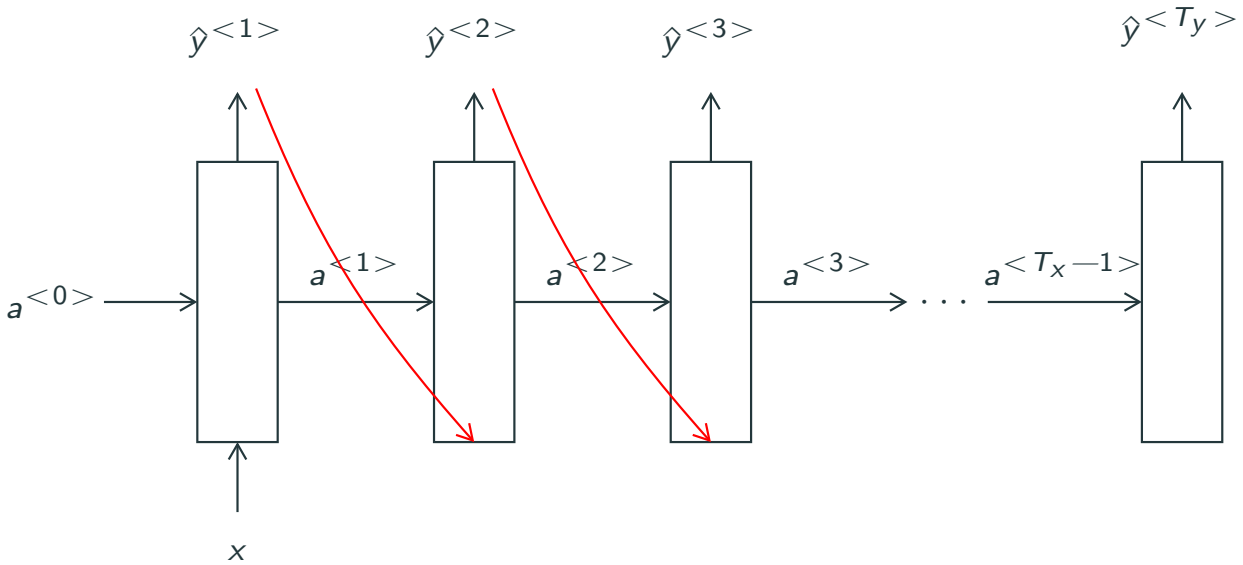
"There is nothing to like in this movie." ⟶ ★☆☆☆☆



Dirrerent types of RNNs

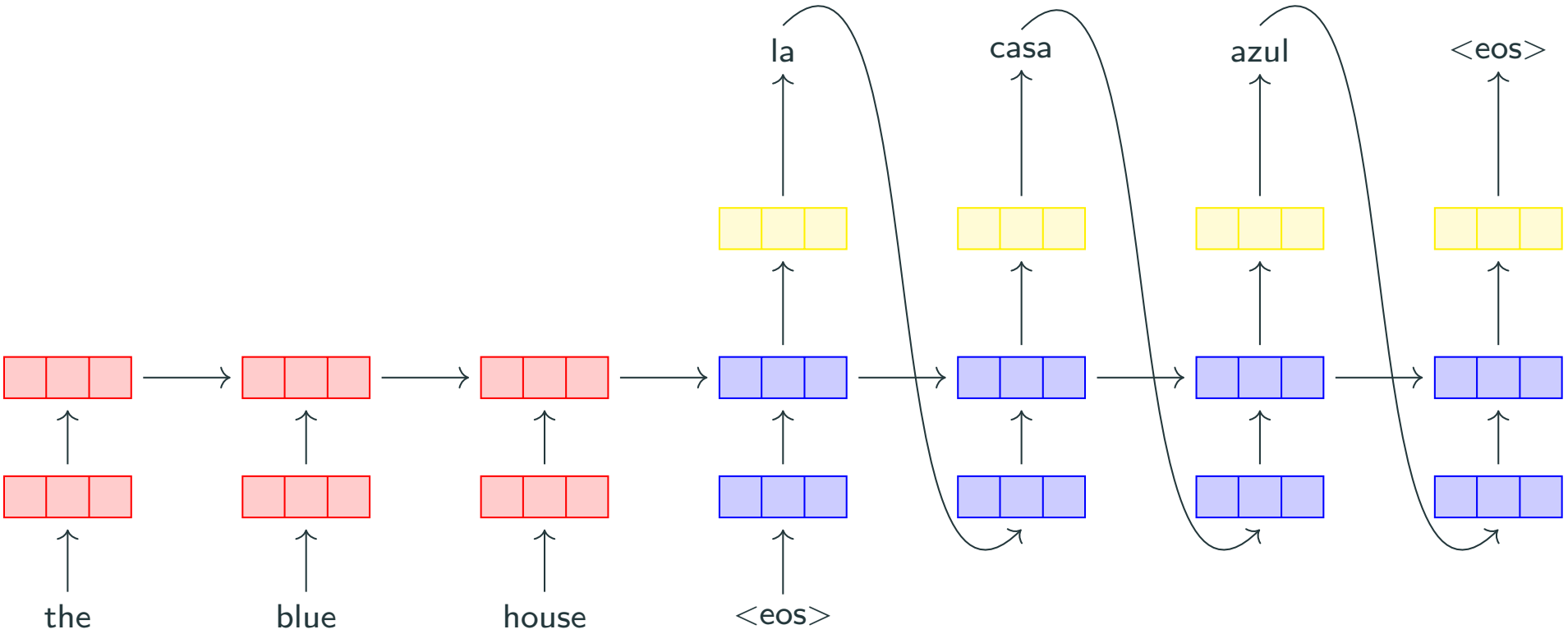
➟ **One-to-many**, e.g., music generation

$$x \rightarrow y^{<1>}\, y^{<2>} \ldots y^{<T_y>}$$



Dirrerent types of RNNs

- Many-to-many, e.g., machine translation



Sequence Generation

Language Modeling

- **Speech recognition** : leveraging a language model

  $P(\text{The apple and pair salad}) = 3.2 \times 10^{-13}$

  $P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$

- Given a sentence, a language model estimates $P(\text{sentence})$

  A sentence is represented as an output sequence : $y^{<1>}, y^{<2>}, \ldots, y^{<T_y>}$

  A language model estimates $P(y^{<1>}, y^{<2>}, \ldots, y^{<T_y>})$

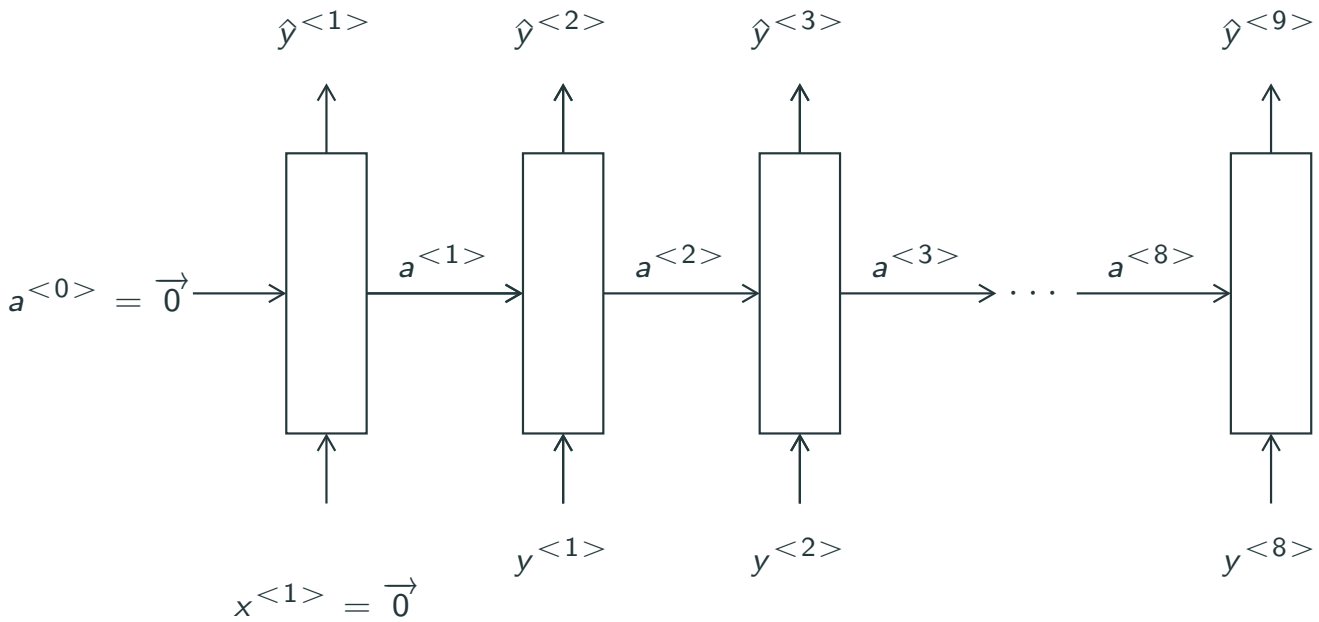
- Language modelling with an RNN : training on a vast **corpus of text**
- Tokenizing the text : mapping each word to an individual token (words) in a vocabulary

  Token $< UNK >$ for an unknown word

  Token $< EOS >$ for the end of a sentence

Language Modeling with an RNN

e.g., *Cats average 15 hours of sleep a day.* $<$ EOS $>$

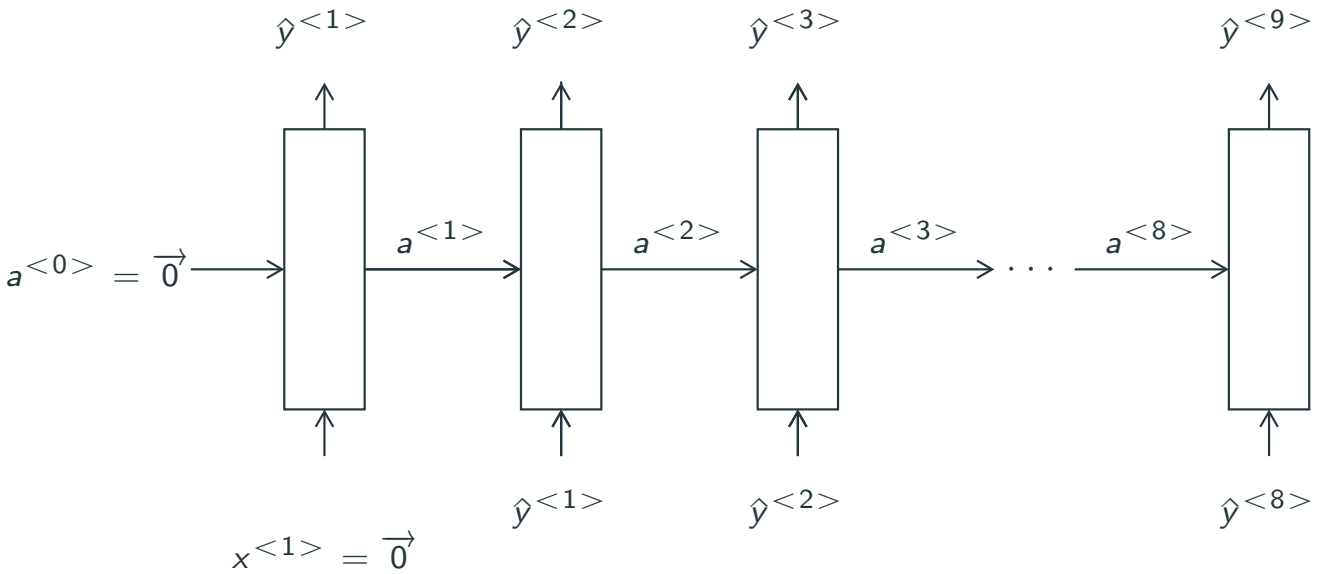


- Input $x^t = y^{<t-1>}$; Output $\hat{y}^{<t>} = P\left(y^{<t>} \mid y^{<1>}, \ldots, y^{<t-1>}\right)$
- Loss at a time t : $\mathcal{L}\left(\hat{y}^{<t>}, y^{<t>}\right) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$
- Overall loss : $\mathcal{L} = \sum_t \mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right)$

e.g., $P\left(y^{<1>}, y^{<2>}, y^{<3>}\right) = P\left(y^{<1>}\right) P\left(y^{<2>} \mid y^{<1>}\right) P\left(y^{<3>} \mid y^{<1>}, y^{<2>}\right)$

Sampling Novel Sequences

- Sampling (generating) a novel sequence of words from a trained RNN.



- Input $x^{(t)} = \hat{y}^{<(t-1)>}$; Output $\hat{y}^{<t>} = P\left(y^{<t>} \mid \hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>}\right)$
- Sampling : $word_t \sim P\left(word_t \mid word_1, word_2, \ldots, word_{t-1}\right)$

Sequence generation

- News

  President enrique peña nieto, announced sench's sulk former coming football langston paring.

  "I was not at all surprised," said hich langston.

  "Concussion epidemie", to be examined.

  The gray football the told some and this has on the uefa ison, should money as.

- Shakespeare

  The mortal moon hath her eclipse in love.

  And subject of this thou art another this fold.

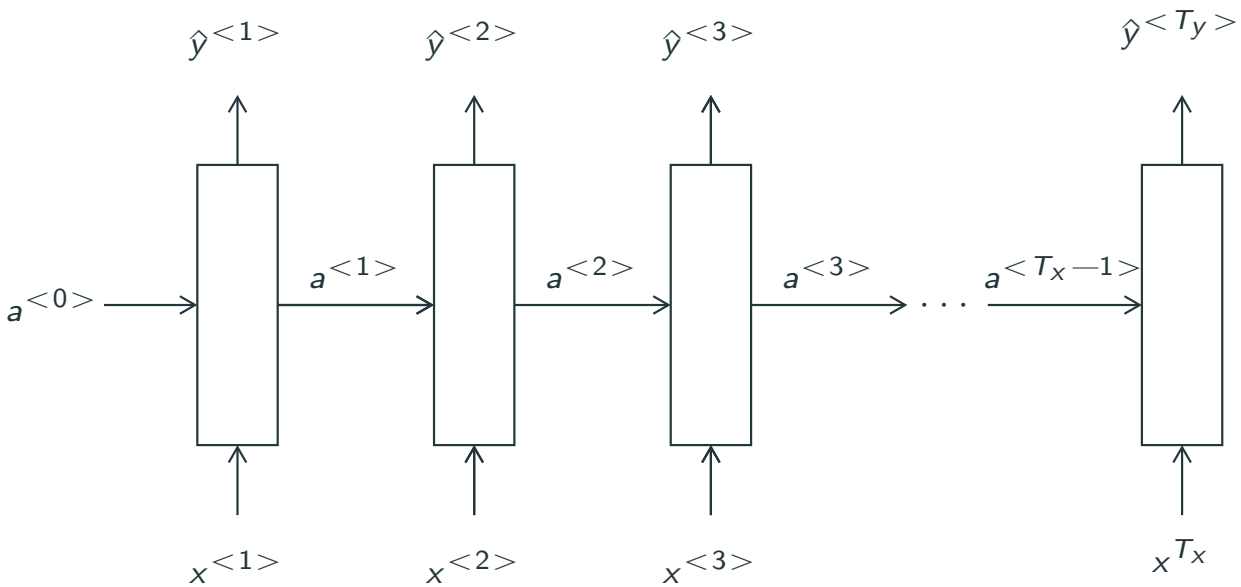
  When besser be my love to me see subl's.

  For whose are ruse of mine eyes heaved

More on RNNs

Vanishing Gradients with RNNs

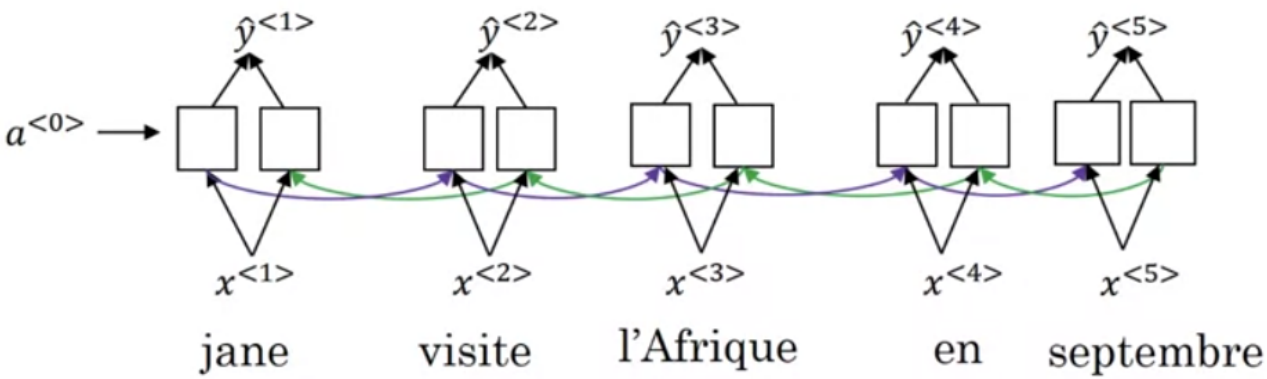
- **Vanishing Gradients :** Derivative may decrease exponentially as a function of the number of layers.
- **RNN :** Capturing long-term dependencies might be difficult.



- **Gated Recurrent Unit (GRU)**
- **Long Short Term Memory (LSTM)**

Bidirectional RNN

- BRNN : for better sequential understanding



- Prediction at time t : $\hat{y}^{<t>} = g\left(W_y\left[\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}\right] + b_y\right)$

  $\overrightarrow{a}^{<t>}$: Activation from the past

  $\overleftarrow{a}^{<t>}$: Activation from the future

Embedding

Featurized representation

- **One-hot** representation treats each word as a distinct entity

Apple (456) Orange (6257)

$$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- No inherent relationships between words
- Difficult to generalize across words

  "I want a glass of **orange juice**"
  "I want a glass of **apple . . .**"

- Featurized representation : each word represented by a set of features

Learning Word Embeddings

- Word embeddings : learned representations capturing semantic meanings and relationships.
- Featurized representations : High-dimensional vectors per word.
- Embedding : Translating word indices into fixed-size vectors, e.g. Word2Vec model.

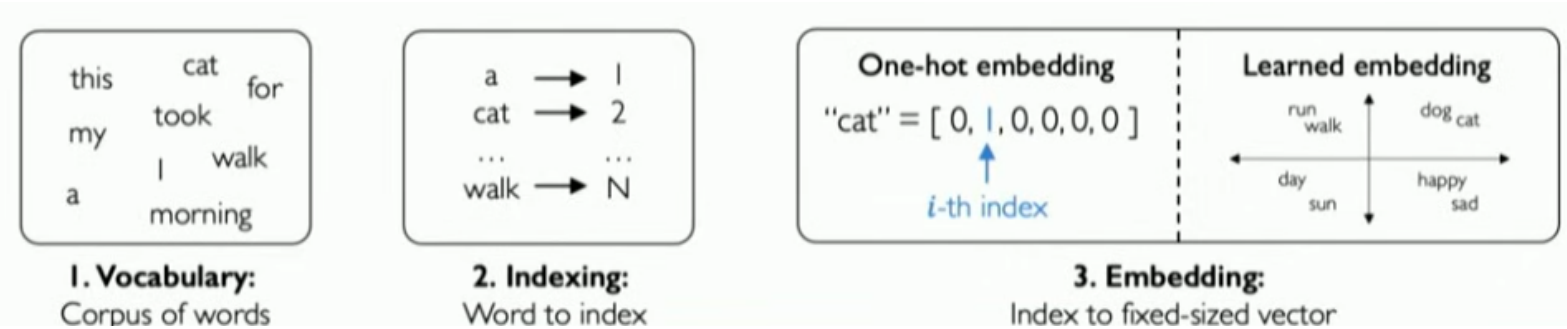


Image Source : MIT

- Visualizing word embeddings : **t-SNE** algorithm maps them to a lower-dimensional space.
- Note : similarity is measurable between word embeddings, e.g., cosine similarity.

Word Embeddings

- Transfer Learning with Word Embeddings : Applying knowledge gained from one task to another.
  1. Learn word embeddings from a large text corpus (1-100B words) or download pre-trained embeddings online.
  2. Transfer the learned embeddings to a new task with a smaller training set.
  3. Optionally, fine-tune the word embeddings with new data.

- Note : The terms *encoding* and *embedding* are similar.