

Cours 2 : Deep Learning pour les données séquentielles

François HU - 13/10/2020

Data Scientist au DataLab de la Société Générale Assurances

Doctorant à l'ENSAE-CREST

Les cours se trouvent ici : <https://curiousml.github.io/>

Sommaire

1. Introduction

2. Recurrent Neural Network (RNN)

- Modèles RNN « classique »
- D'autres architectures RNN
- Modèle Bidirectional-RNN (BRNN)
- Modèle Deep RNN

3. OPTION

- [option] LSTM / GRU
- [option] BERT

Programme

Introduction

Représentations vectorielles

Deep Learning pour NLP

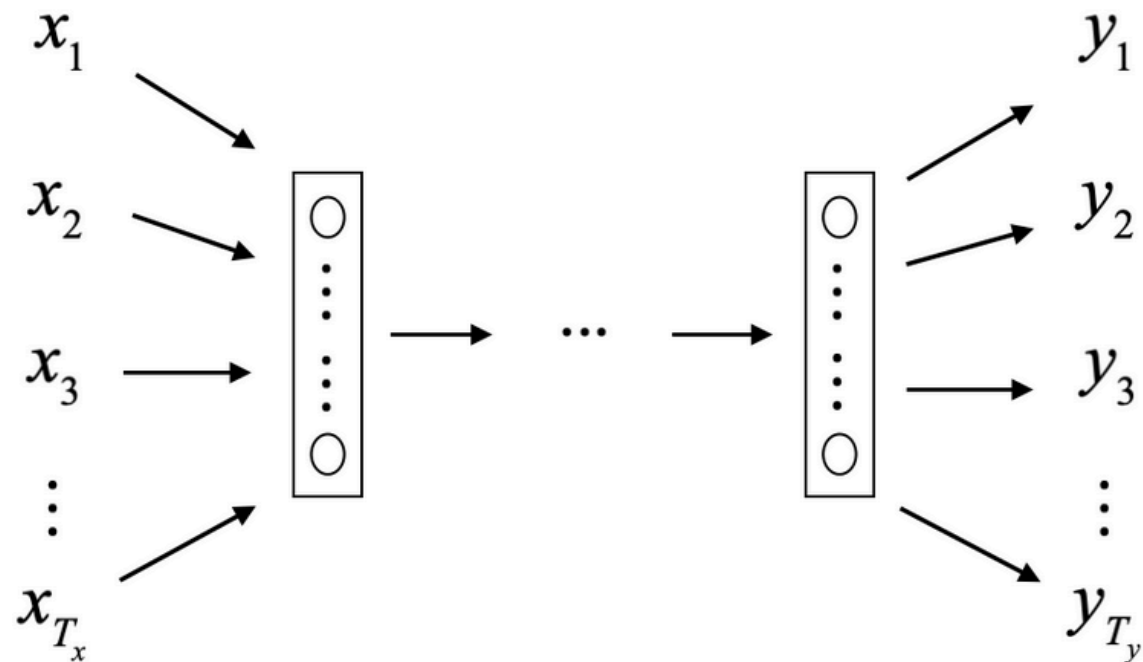
Active Learning

Introduction

Pourquoi les modèles séquentiels ?

- classification de textes / analyse de sentiment
- Named Entity Recognition (NER)
- génération de textes / de musiques
- traducteur de langue automatique
- ...

Pourquoi pas les réseaux de neurones « standard » ?

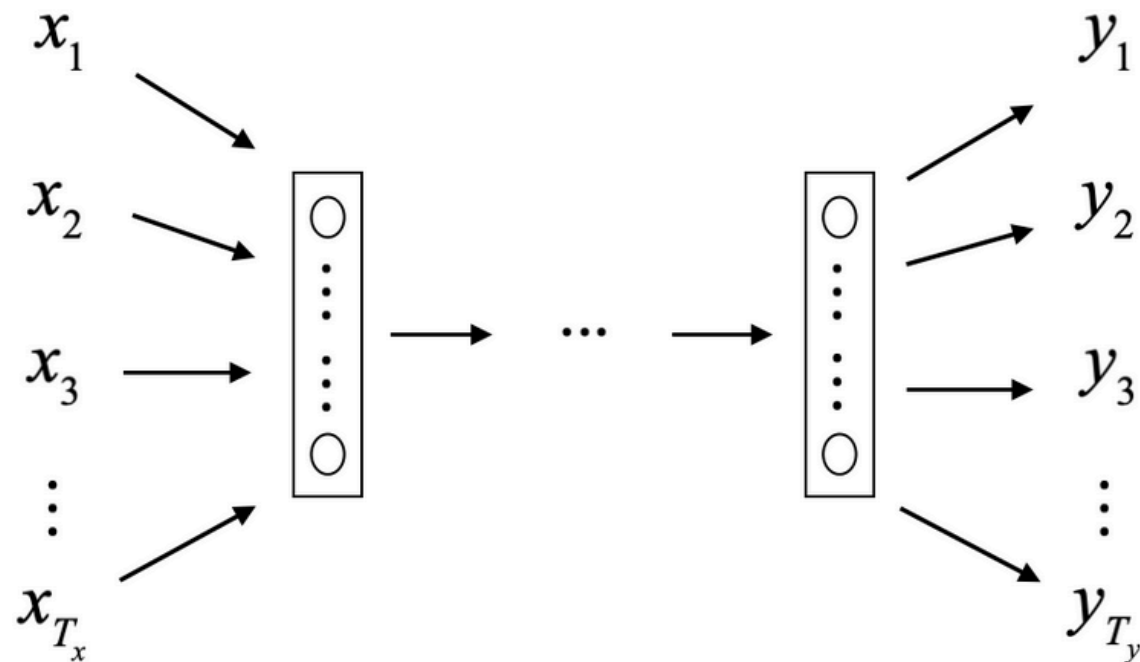


Introduction

Pourquoi les modèles séquentiels ?

- classification de textes / analyse de sentiment
- Named Entity Recognition (NER)
- génération de textes / de musiques
- traducteur de langue automatique
- ...

Pourquoi pas les réseaux de neurones « standard » ?

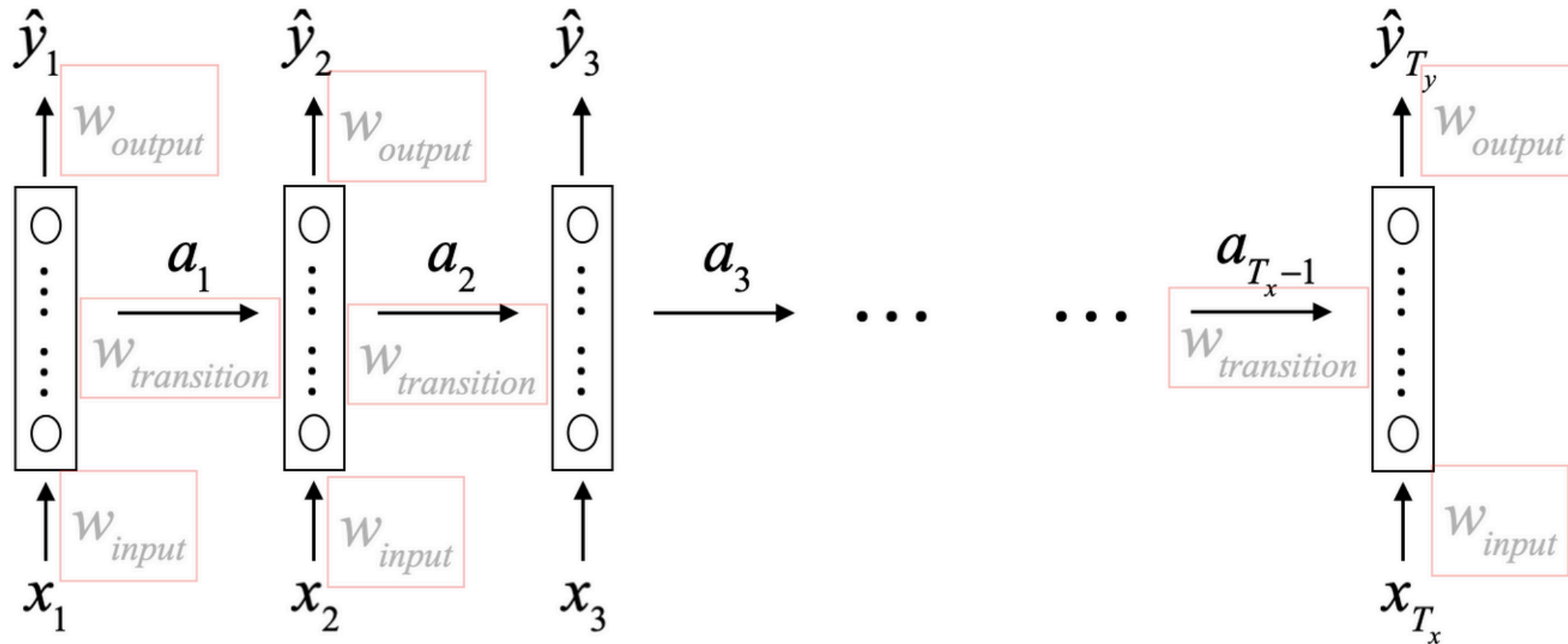


- inputs / outputs peuvent être de **tailles différentes**
- ne tient pas en compte des **différentes positions** des mots

2. Recurrent Neural Networks (RNN)

Modèle RNN « classique » (1/3)

Word embedding (plongement de mot en français) : vectorisation des mots de sorte que les mots apparaissant dans des contextes similaires ont des significations apparentées

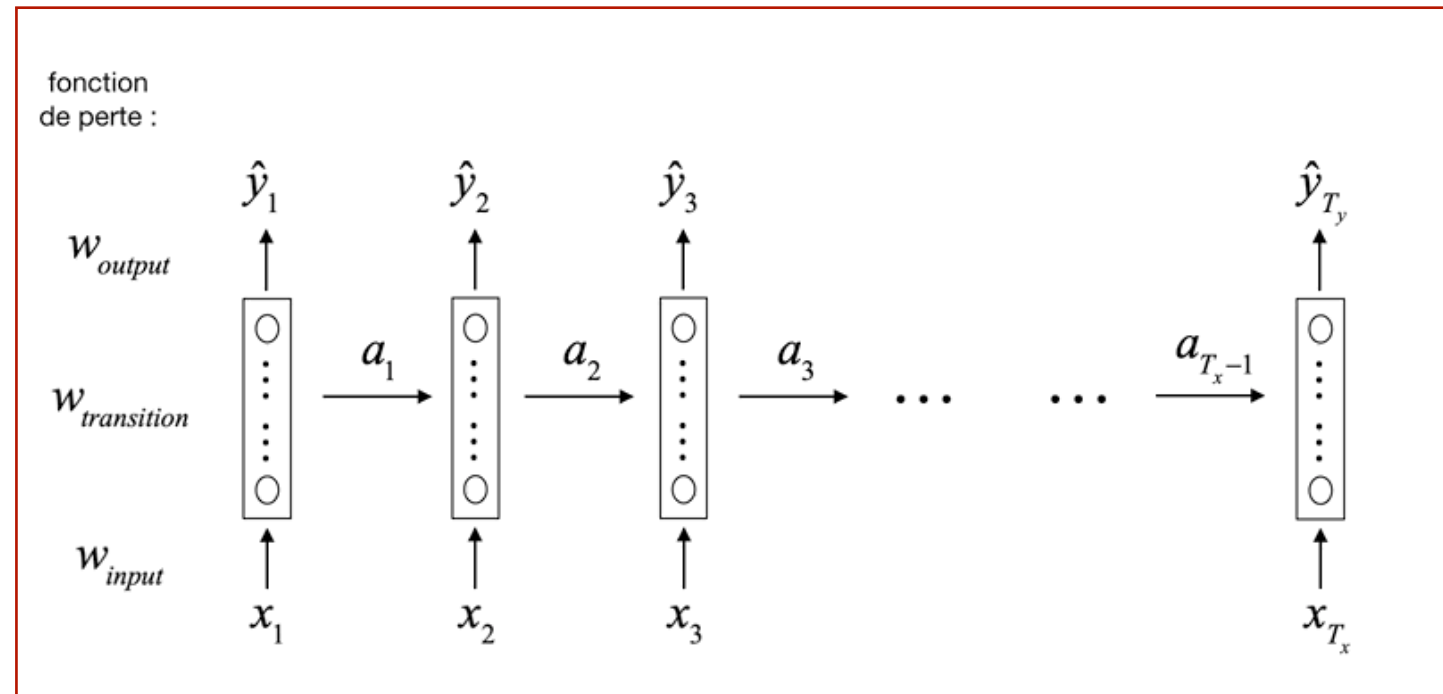


- possibilité de **manipuler des séquences** de taille variable
- tout calcul futur **tient compte des calculs passés**
- les poids / paramètres sont **partagés dans le temps**

Modèle RNN « classique » (2/3)

Propagation avant (forward propagation) d'un RNN :

Pour voir l'animation :
<https://curiousml.github.io/teaching/DSA/RNNforward.html>



étant donné les poids w_{input} , $w_{transition}$ et w_{output} , nous allons calculer a_t , \hat{y}_t et $l_t = l(y_t, \hat{y}_t)$

Au temps $t = 0$: $a_0 = \vec{0}$

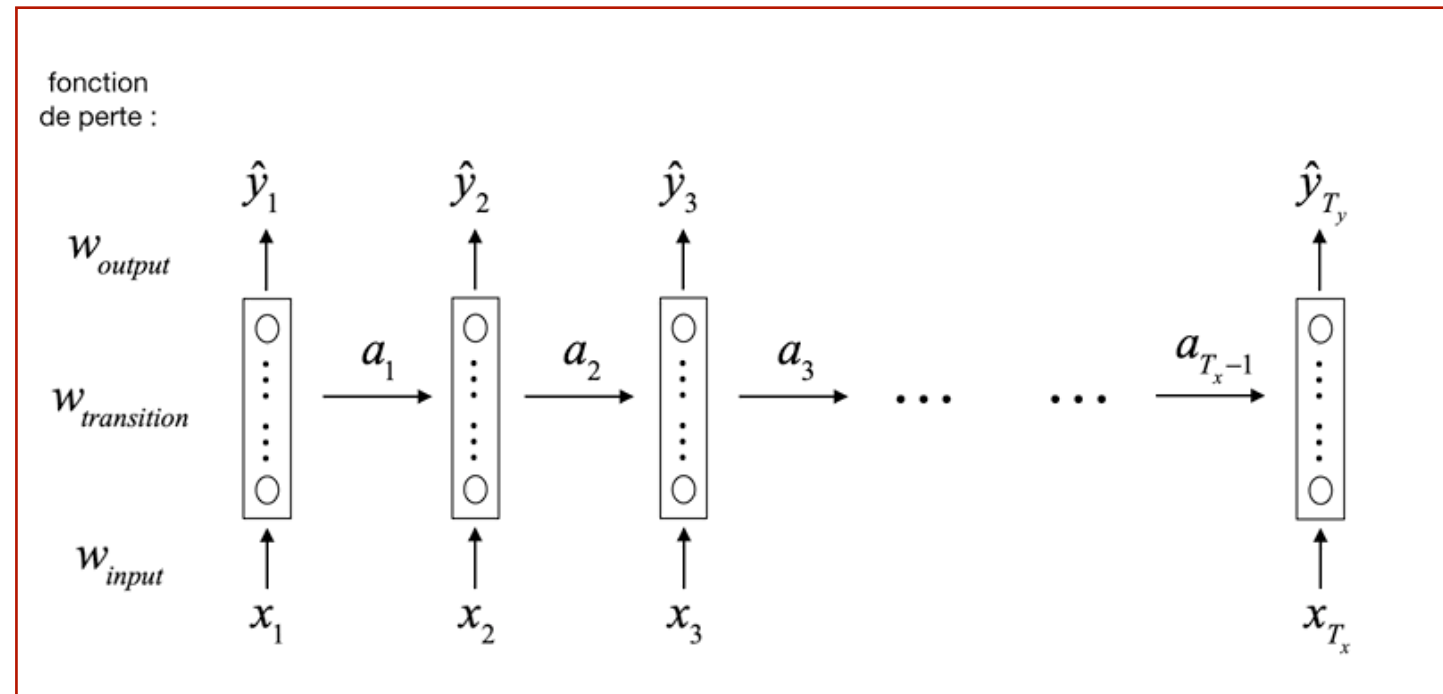
Au temps $t > 0$:

$$a_t = \sigma_{transition}(w_{transition} \cdot a_{t-1} + w_{input} \cdot x_t)$$
$$\hat{y}_t = \sigma_{output}(w_{output} \cdot a_t)$$

Modèle RNN « classique » (2/3)

Propagation avant (forward propagation) d'un RNN :

Pour voir l'animation :
<https://curiousml.github.io/teaching/DSA/RNNforward.html>



étant donné les poids w_{input} , $w_{transition}$ et w_{output} , nous allons calculer a_t , \hat{y}_t et $l_t = l(y_t, \hat{y}_t)$

Au temps $t = 0$: $a_0 = \vec{0}$

Au temps $t > 0$:

$$a_t = \sigma_{transition}(w_{transition} \cdot a_{t-1} + w_{input} \cdot x_t)$$
$$\hat{y}_t = \sigma_{output}(w_{output} \cdot a_t)$$

Exemple de
fonction de perte :
Cross-Entropy

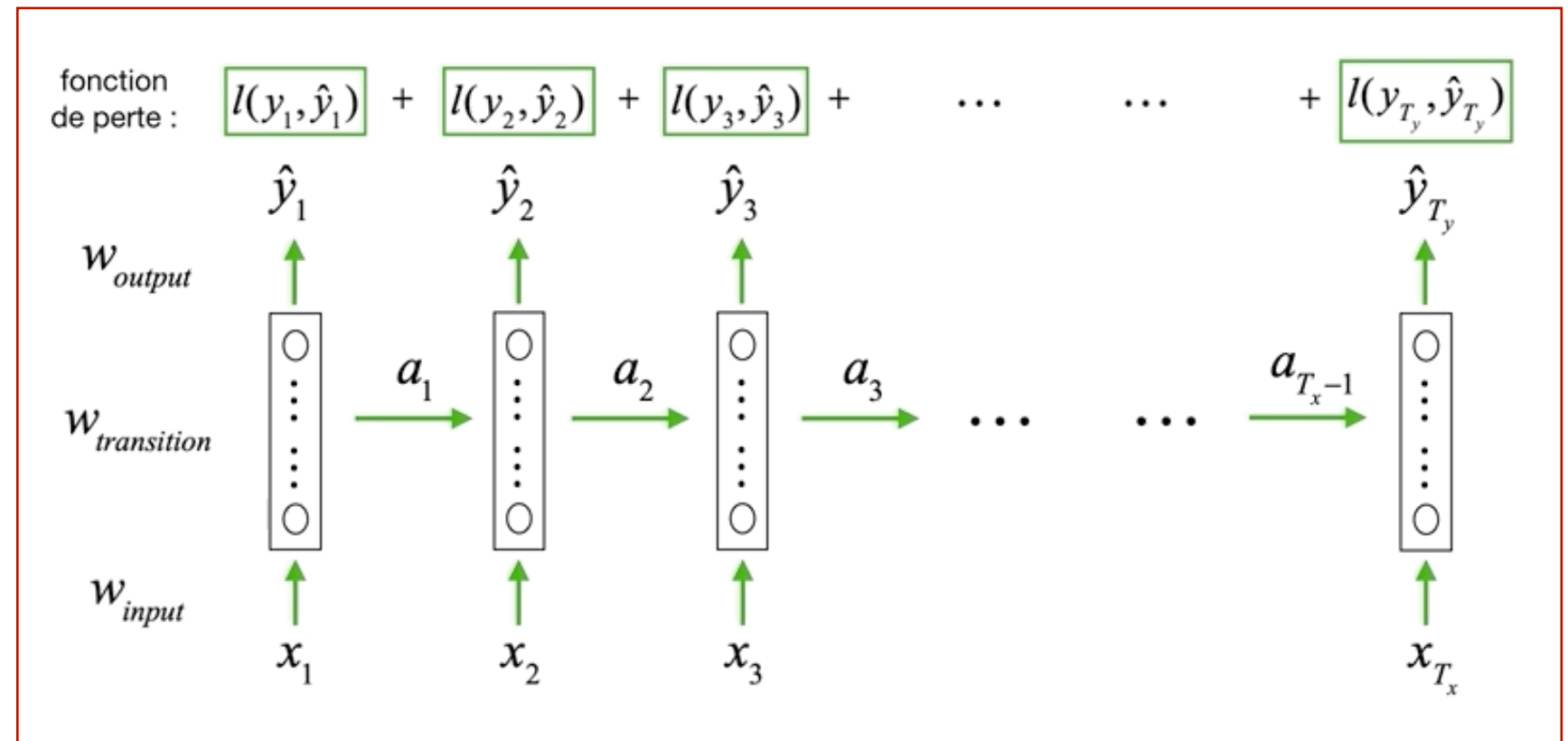
Exemple de fonction
d'activation :

$\sigma_{transition} \leftarrow \tanh$ ou ReLu

$\sigma_{output} \leftarrow \text{sigmoid}$ ou softmax

Modèle RNN « classique » (3/3)

Propagation arrière (back-propagation) d'un RNN :



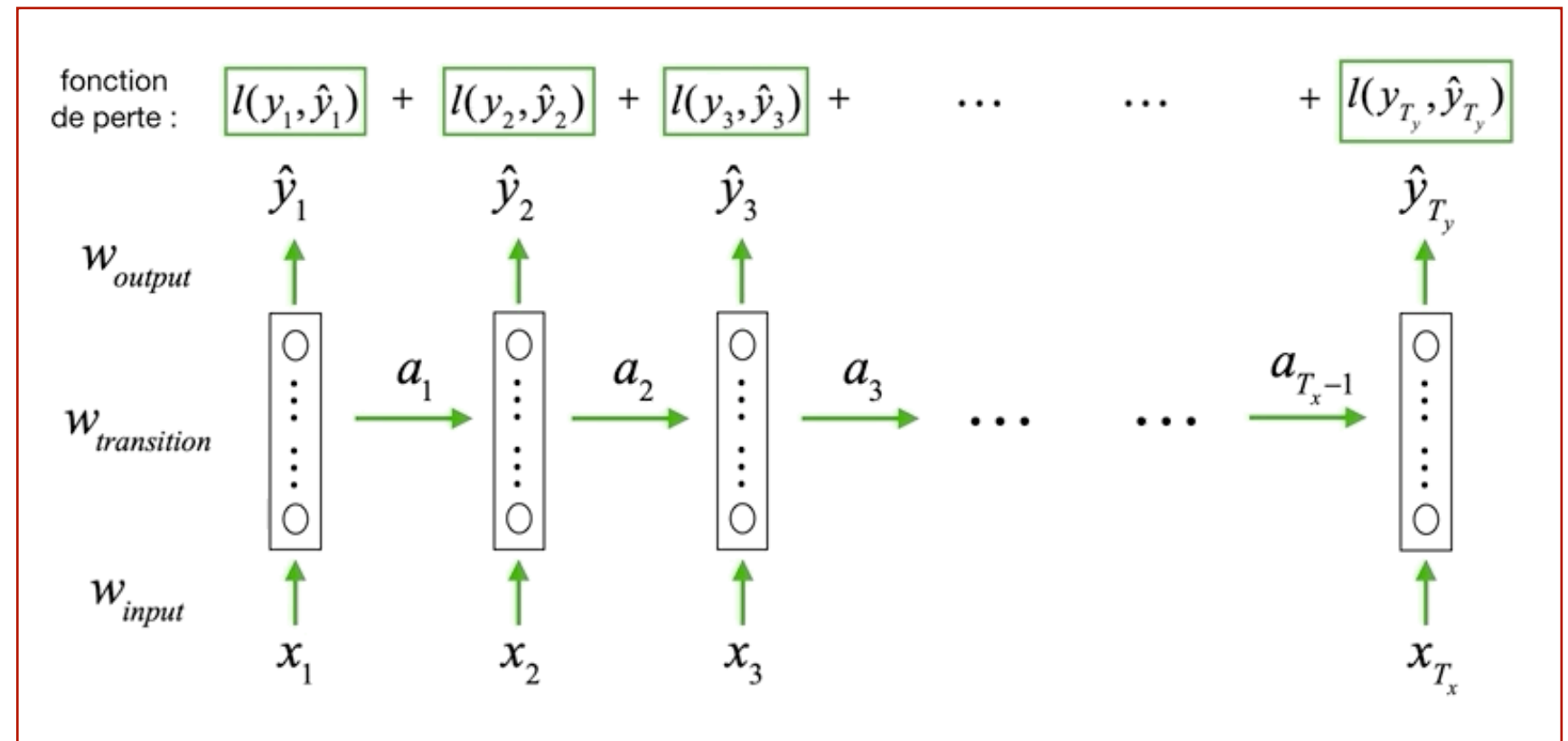
Pour voir l'animation :

<https://curiousml.github.io/teaching/DSA/RNNbackprop.html>

mettre à jour les poids w_{input} , $w_{transition}$ et w_{output} afin de minimiser les fonctions de pertes $l_t = l(y_t, \hat{y}_t)$

Modèle RNN « classique » (3/3)

Propagation arrière (back-propagation) d'un RNN :



Pour voir l'animation :

<https://curiousml.github.io/teaching/DSA/RNNbackprop.html>

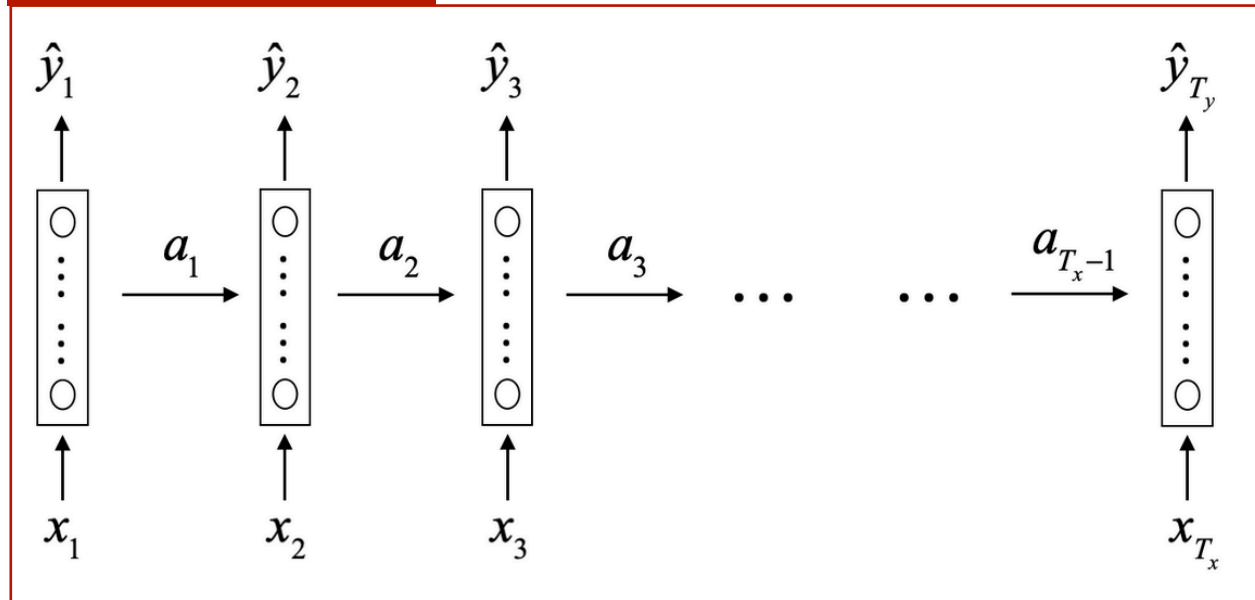
mettre à jour les poids w_{input} , $w_{transition}$ et w_{output} afin de minimiser les fonctions de pertes $l_t = l(y_t, \hat{y}_t)$

Par descente de gradient

D'autres architectures RNN

Architecture N-N, N-1, 1-N, N-M

Architecture N-N

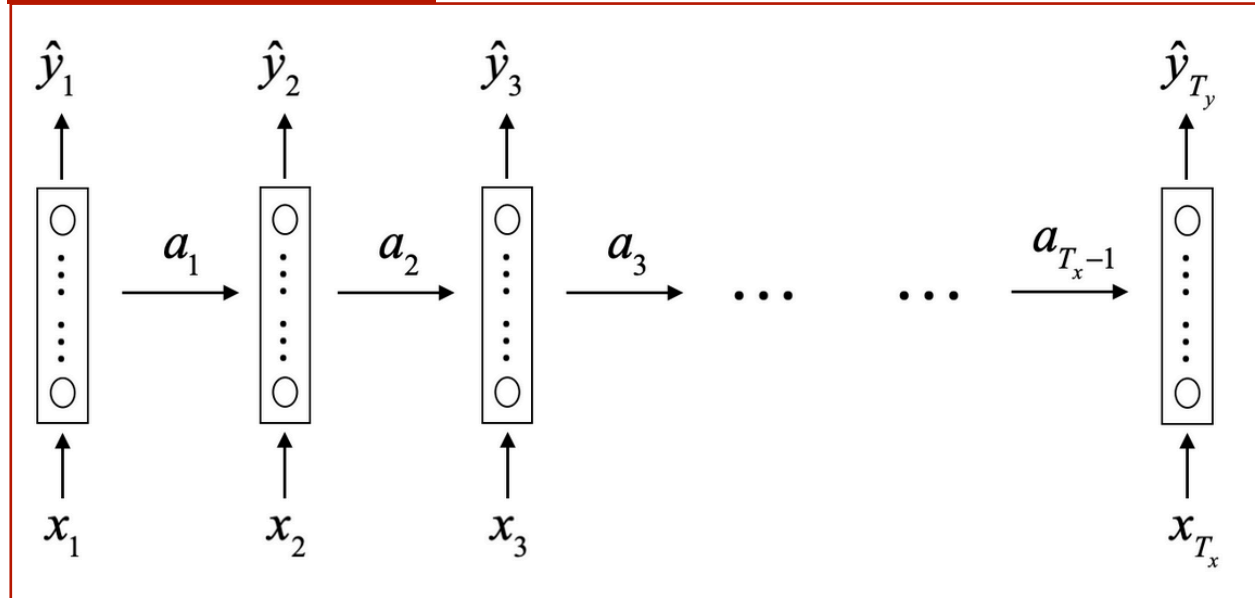


Named Entity Recognition (NER)

D'autres architectures RNN

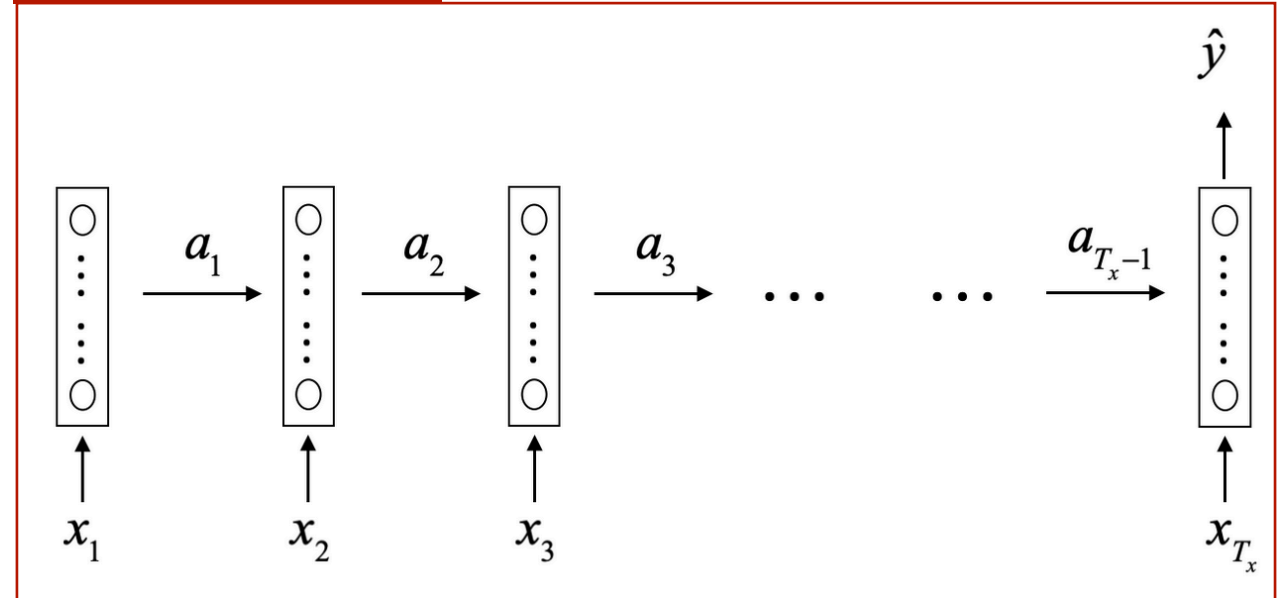
Architecture N-N, N-1, 1-N, N-M

Architecture N-N



↪ Named Entity Recognition (NER)

Architecture N-1

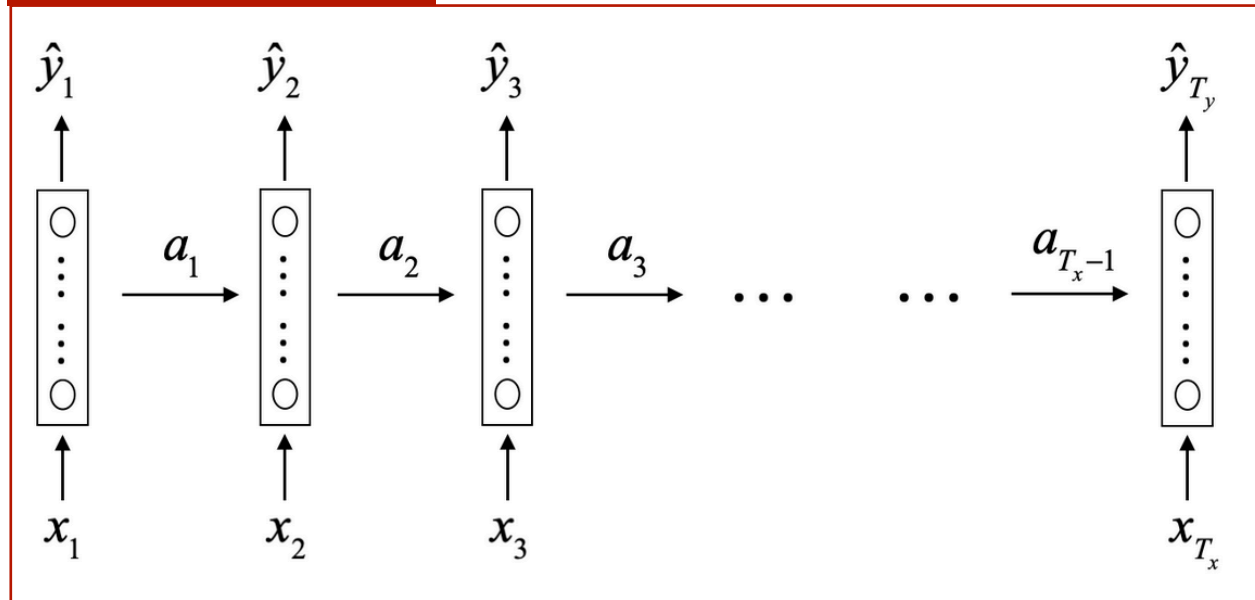


↪ Classification de textes

D'autres architectures RNN

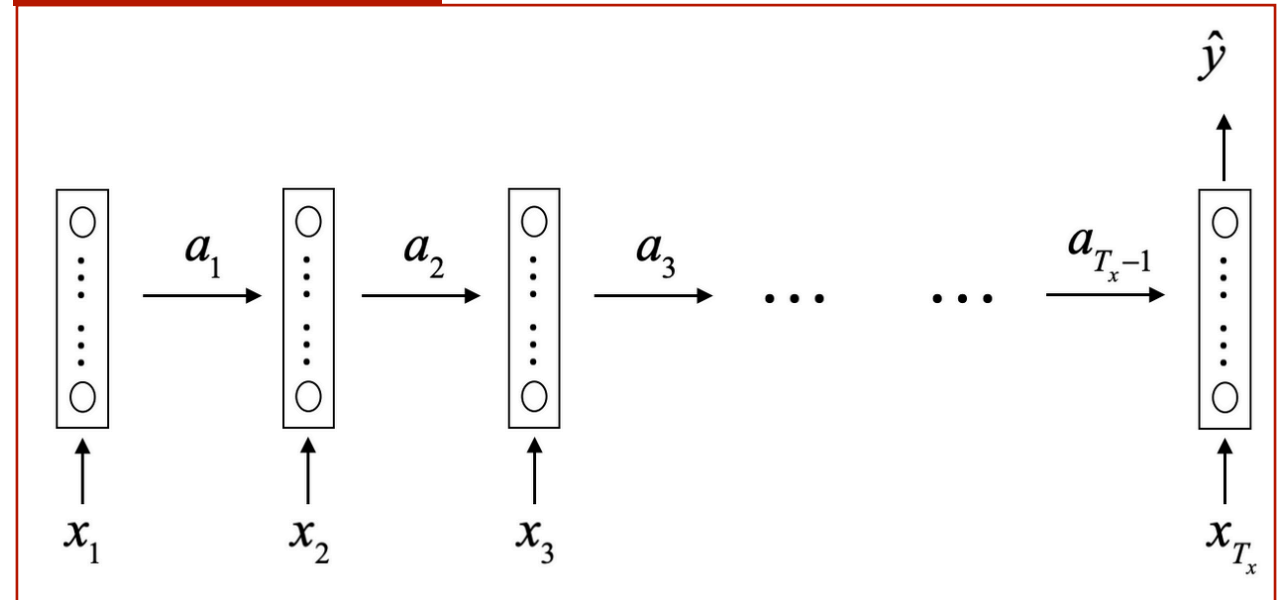
Architecture N-N, N-1, 1-N, N-M

Architecture N-N



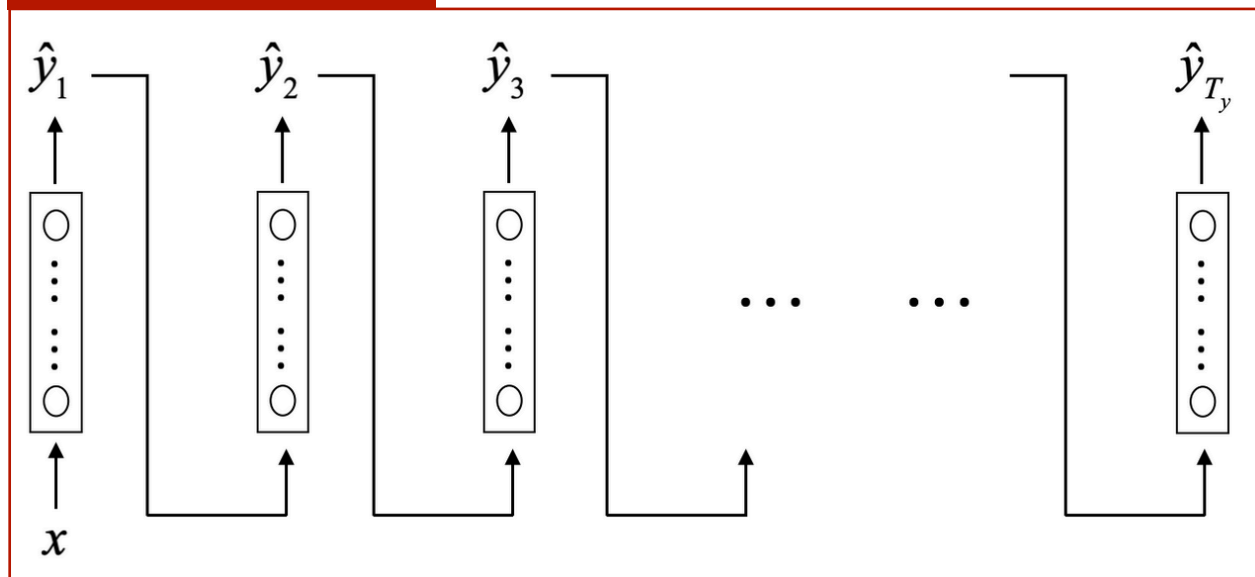
↪ Named Entity Recognition (NER)

Architecture N-1



↪ Classification de textes

Architecture 1-N

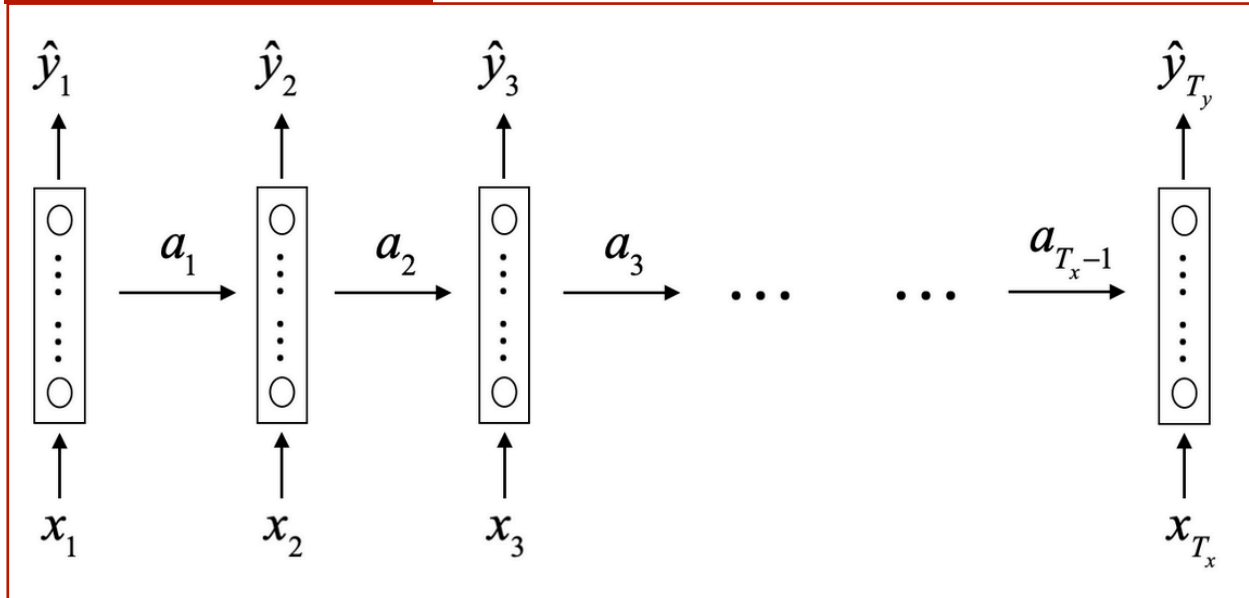


↪ Génération de séquences (textes, musique)

D'autres architectures RNN

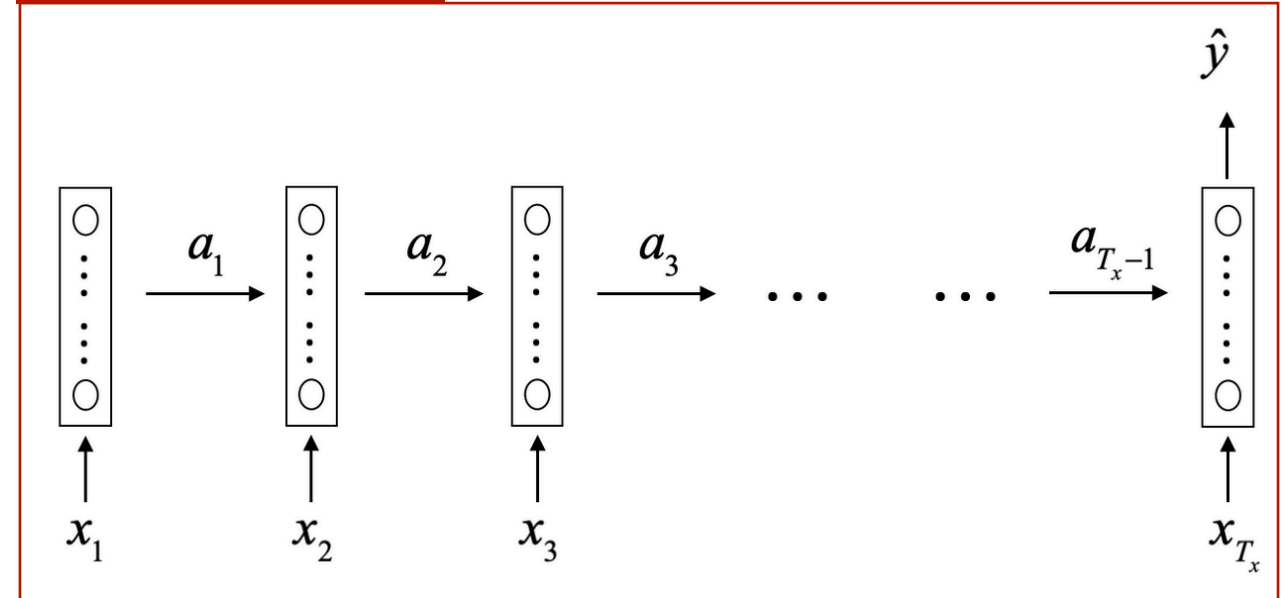
Architecture N-N, N-1, 1-N, N-M

Architecture N-N



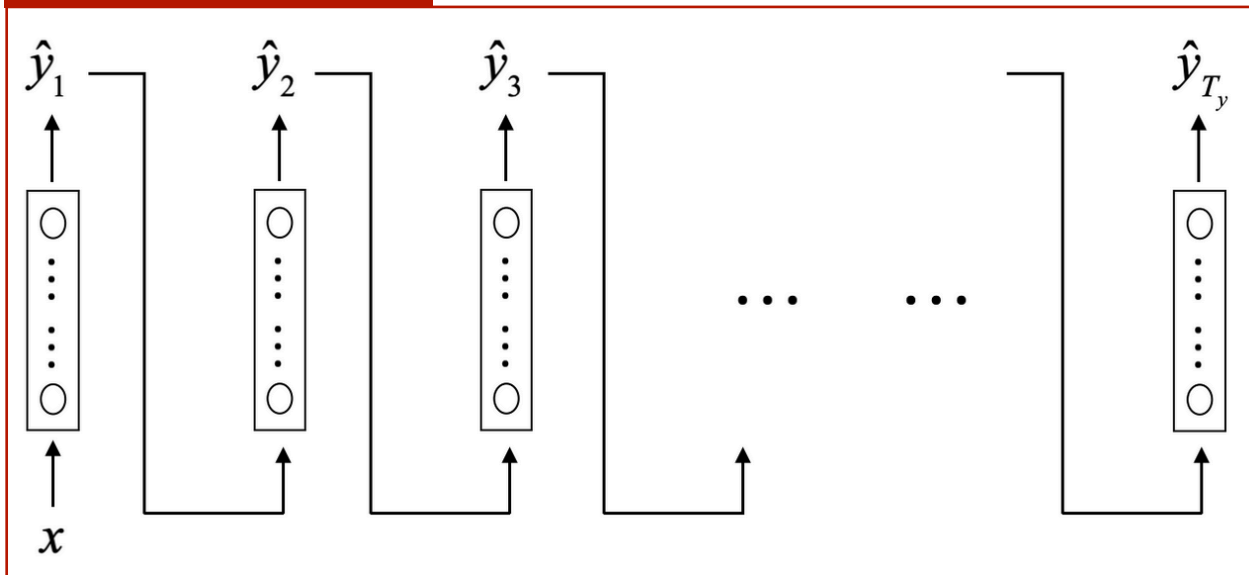
↪ Named Entity Recognition (NER)

Architecture N-1



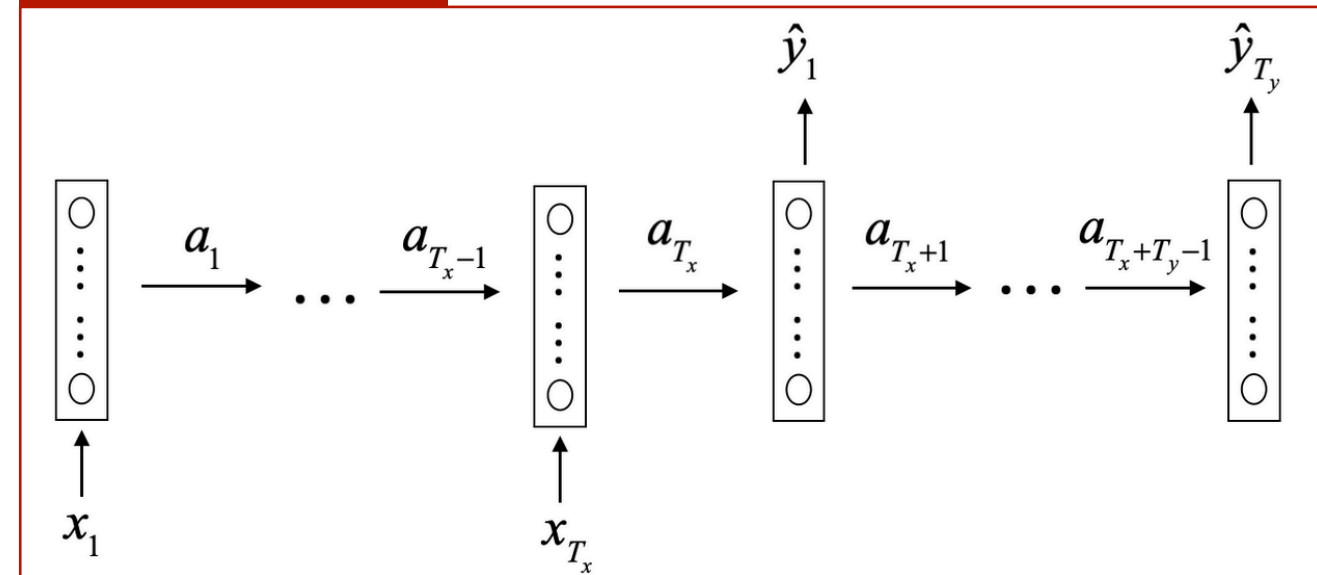
↪ Classification de textes

Architecture 1-N



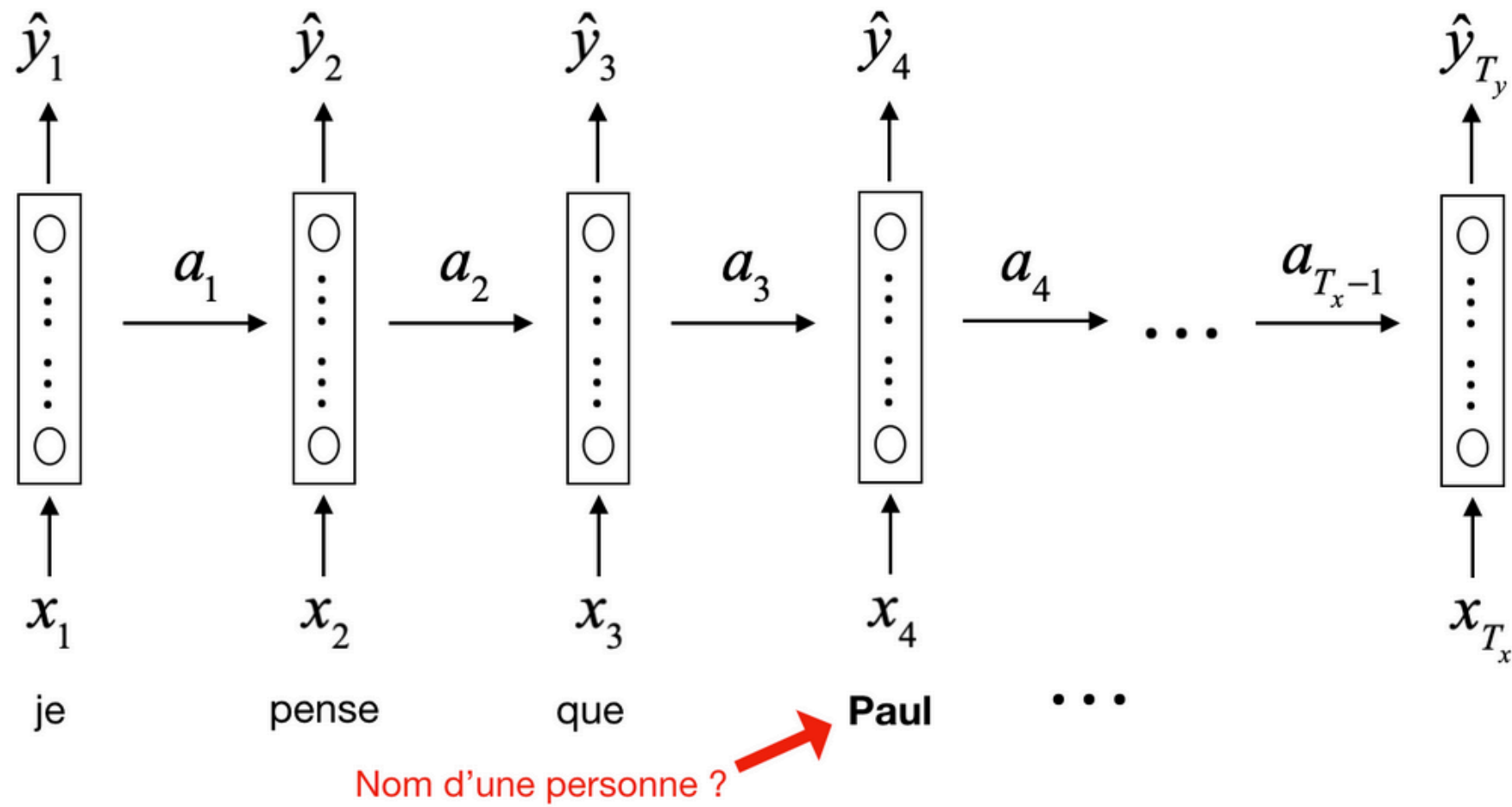
↪ Génération de séquences (textes, musique)

Architecture N-M

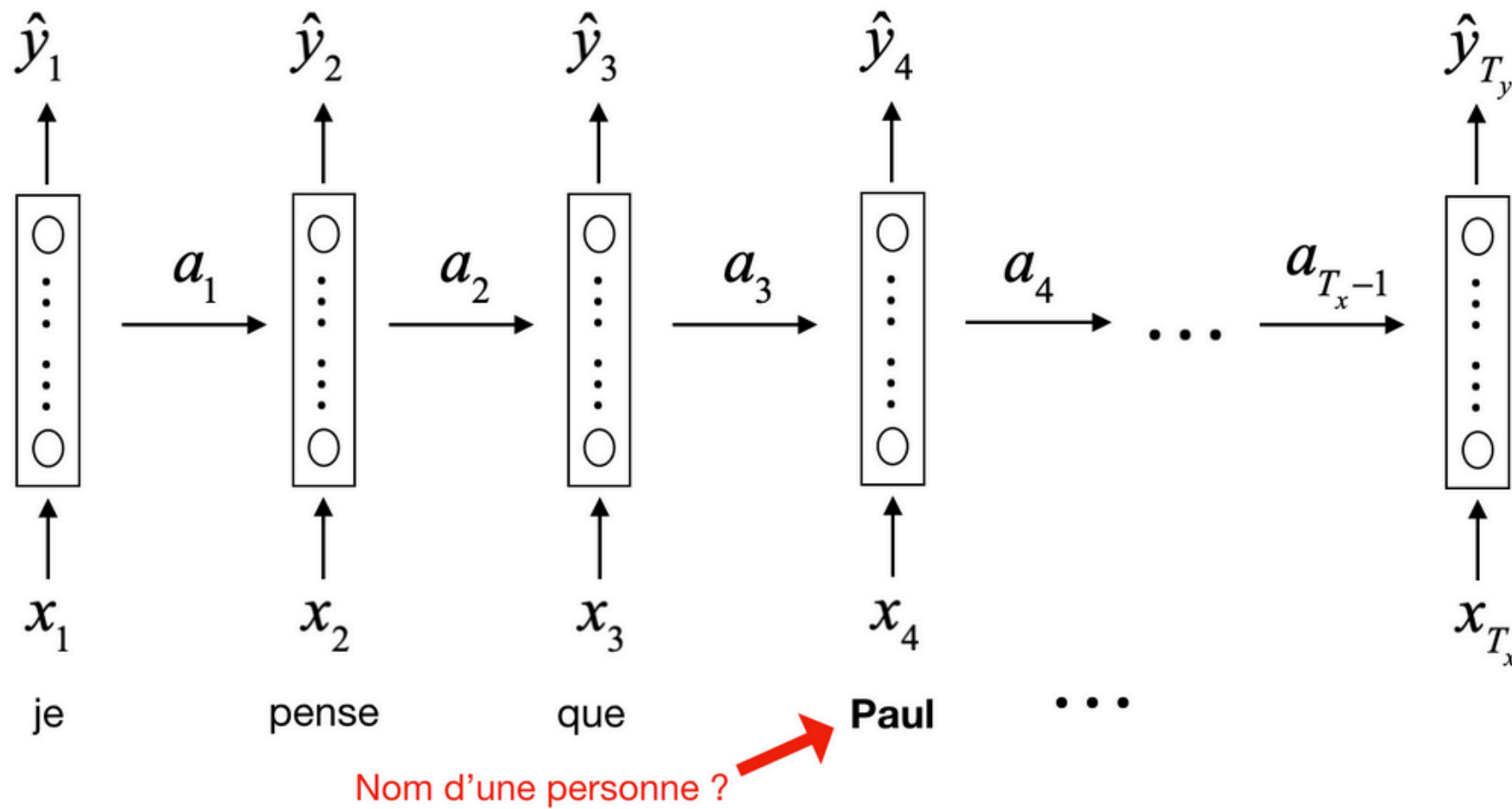


↪ Traducteur automatique

Modèle Bidirectional-RNN (1/2)

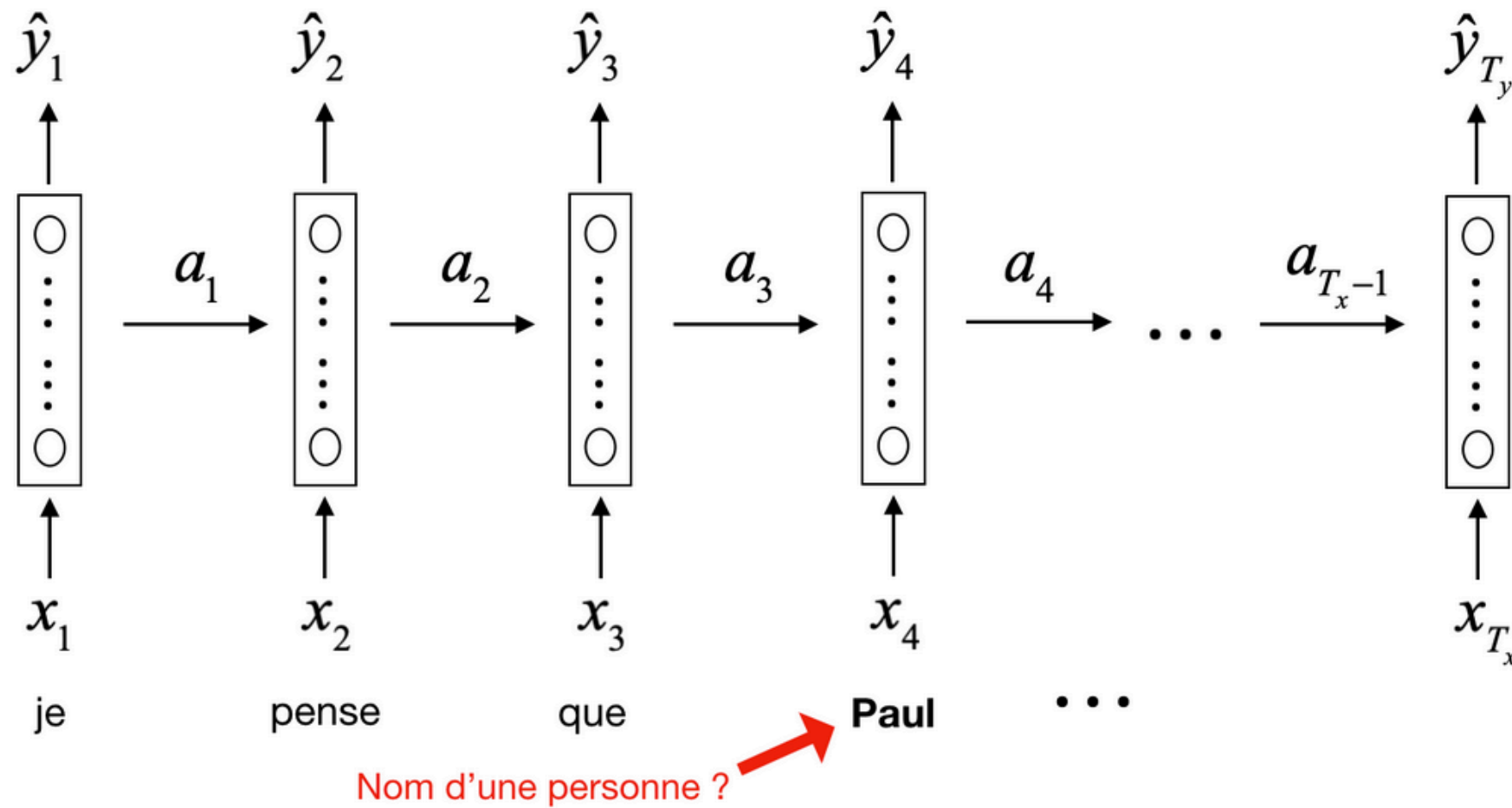


Modèle Bidirectional-RNN (1/2)



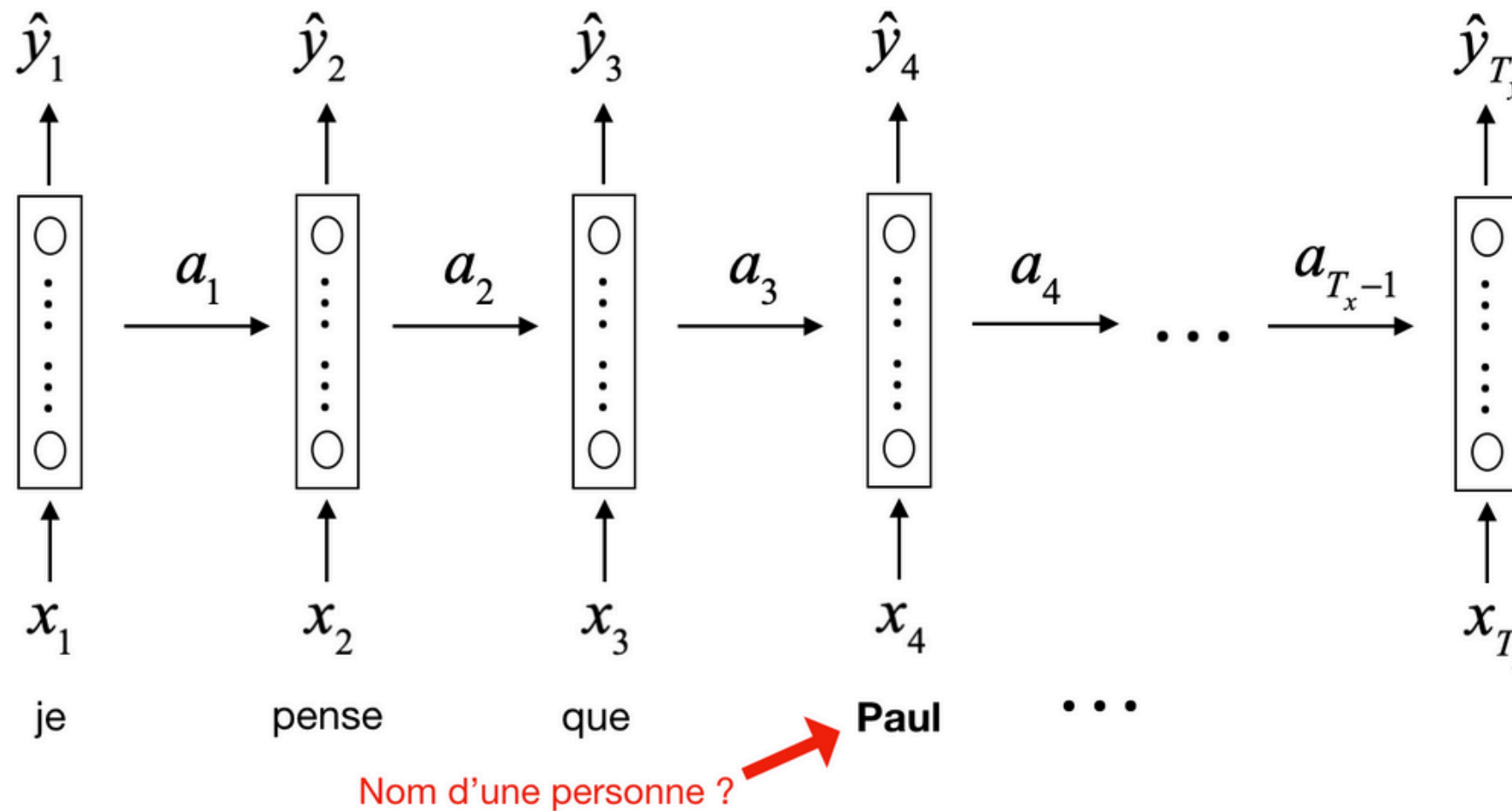
- oui, avec l'exemple : Je pense que **Paul** est né le 10 juillet 1989

Modèle Bidirectional-RNN (1/2)



- oui, avec l'exemple : Je pense que **Paul** est né le 10 juillet 1989
- **non**, avec l'exemple : Je pense que **Paul** est une boulangerie fondée en 1889

Modèle Bidirectional-RNN (1/2)



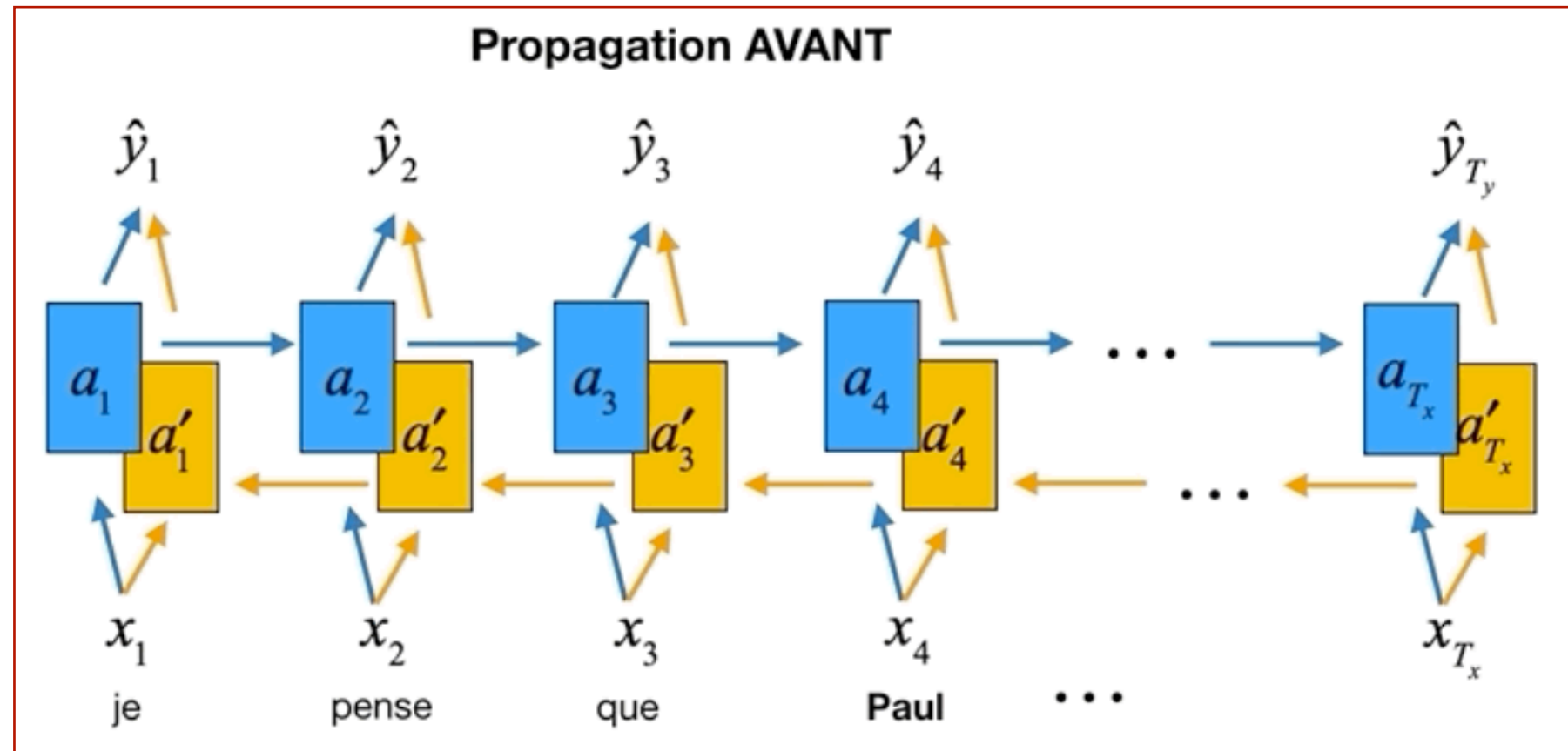
- oui, avec l'exemple : Je pense que **Paul** est né le 10 juillet 1989
- **non**, avec l'exemple : Je pense que **Paul** est une boulangerie fondée en 1889

Souvent, la prédiction d'une instance doit dépendre des **instances futures**

Modèle Bidirectional-RNN (2/2)

Pour voir l'animation :

<https://curiousml.github.io/teaching/DSA/BRNNforward.html>

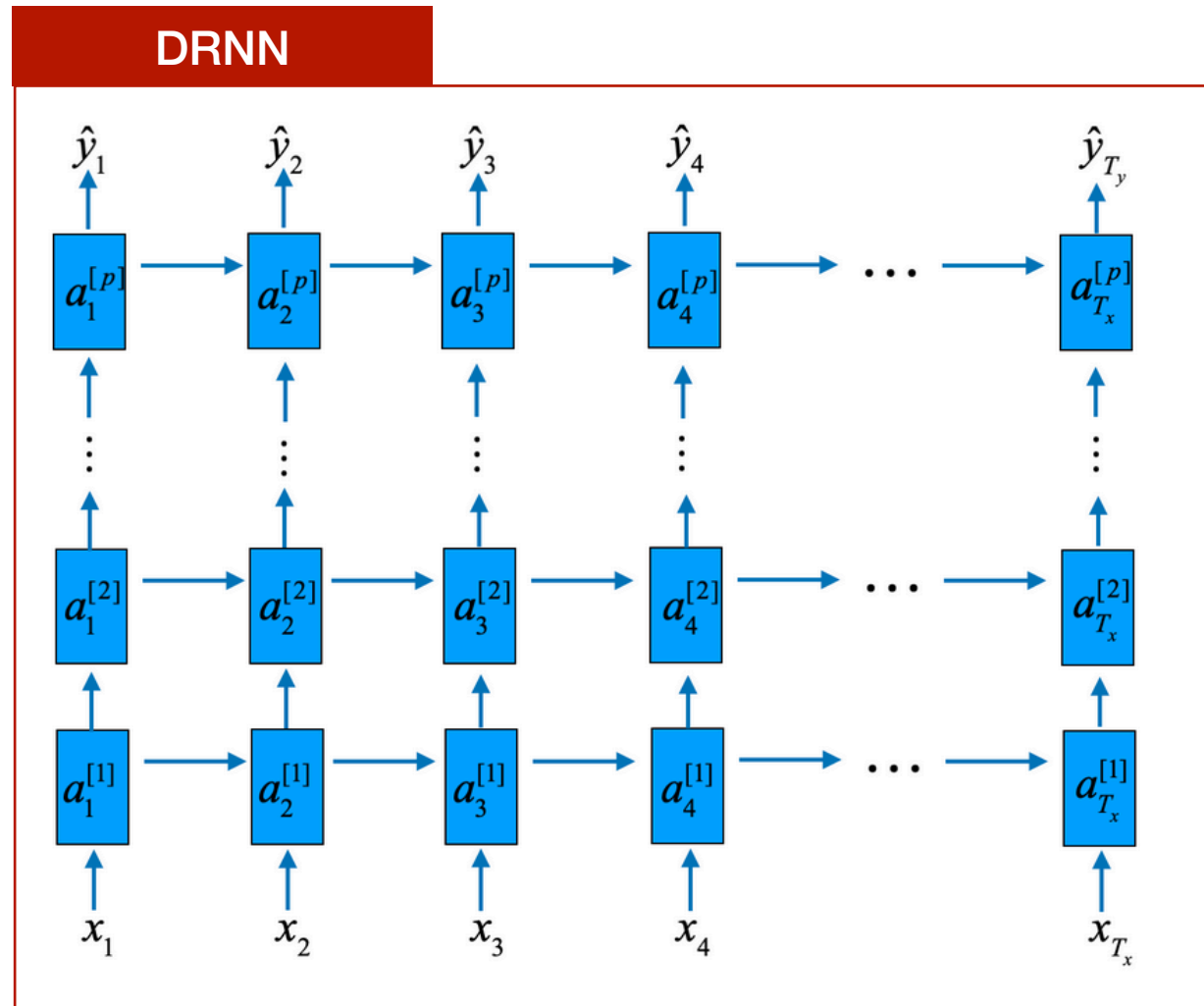


étant donné les poids w_{input} , w'_{input} , $w_{transition}$, $w'_{transition}$ et w_{output} , nous calculons :

- les activations a_t et les activation a'_t
- puis les prédictions $\hat{y}_t = \sigma(w_{output} \cdot [a_t, a'_t])$ et les fonctions de perte l_t

propagation arrière : mise à jour des poids par descente de gradient afin de minimiser les pertes

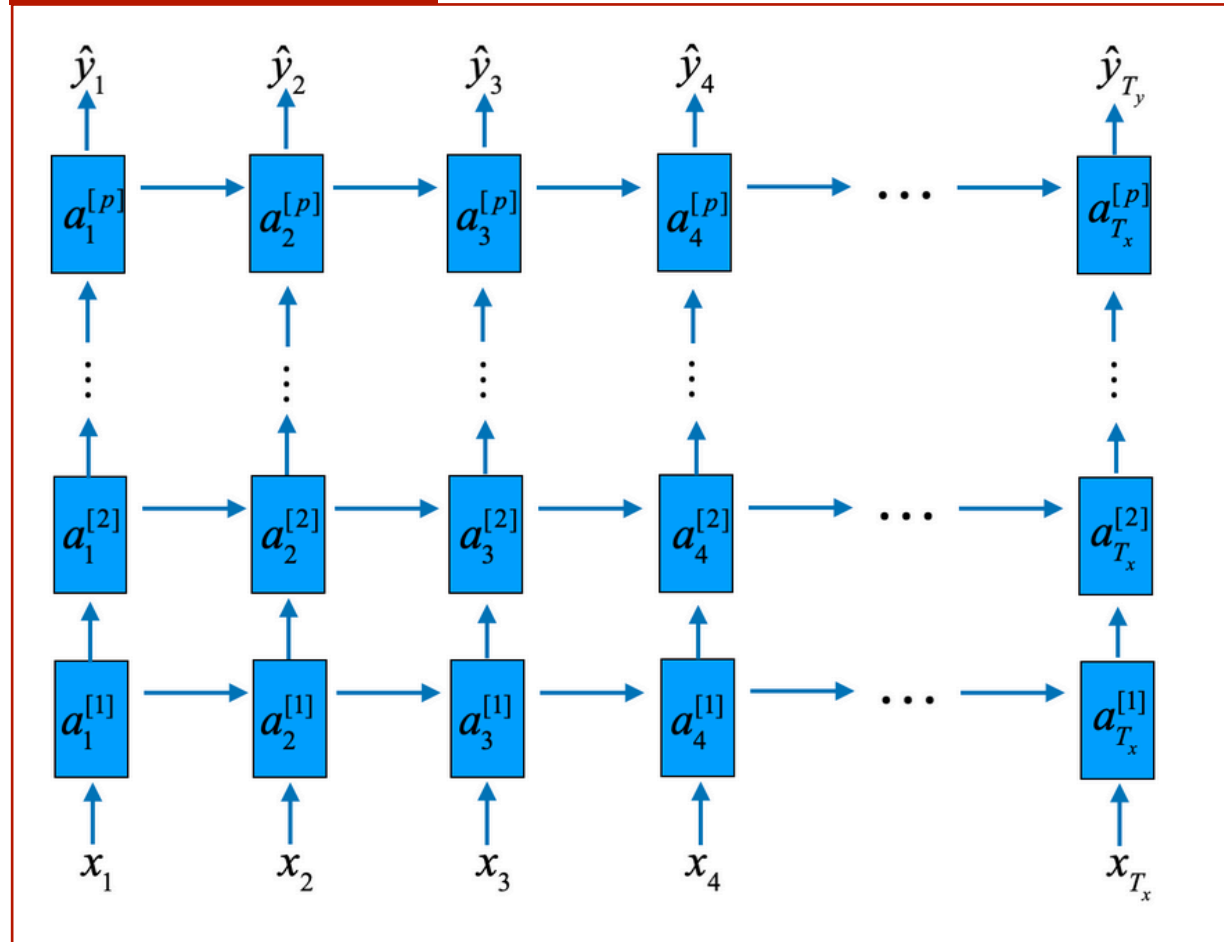
Modèle deep RNN



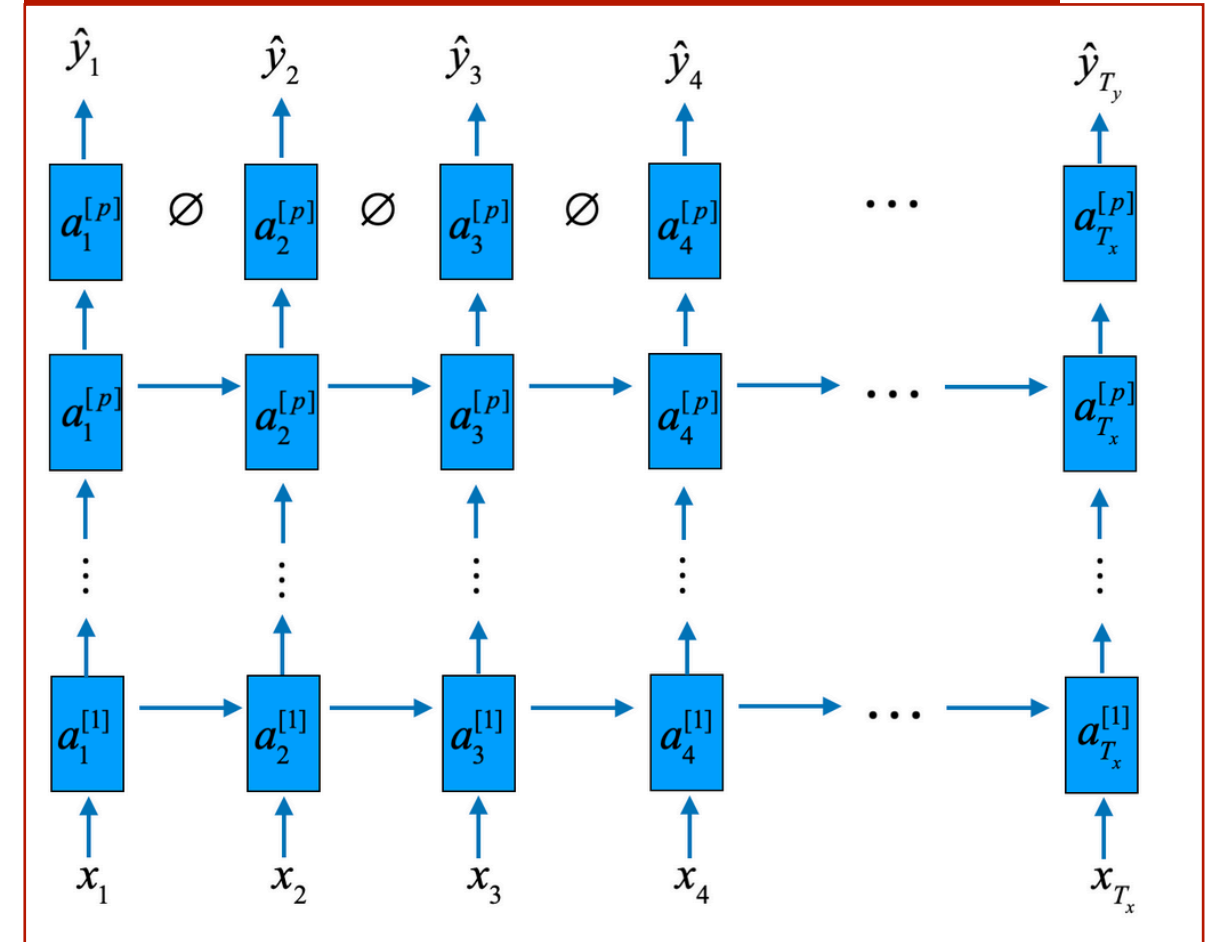
- Empiler les couches cachées
- Généralement entre 2 et 4 couches cachées (car grande complexité de calcul)

Modèle deep RNN

DRNN



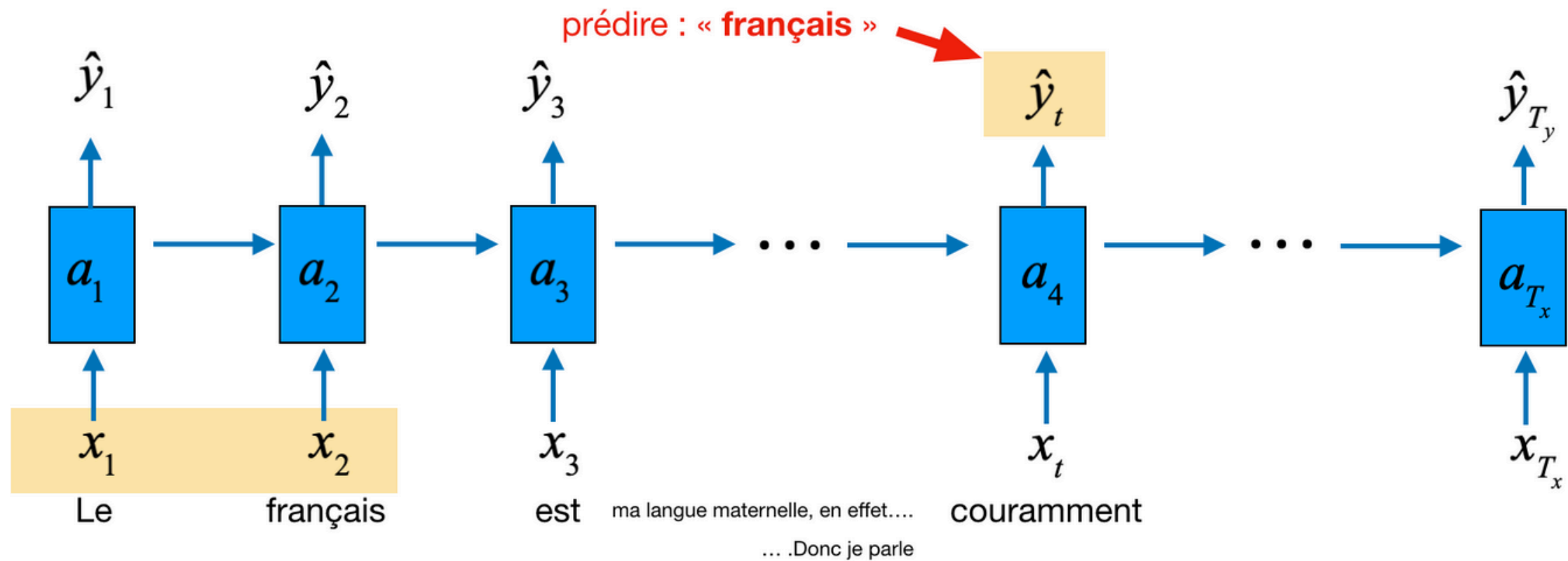
DRNN + couches NN supplémentaires



- Empiler les couches cachées
- Généralement entre 2 et 4 couches cachées (car grande complexité de calcul)

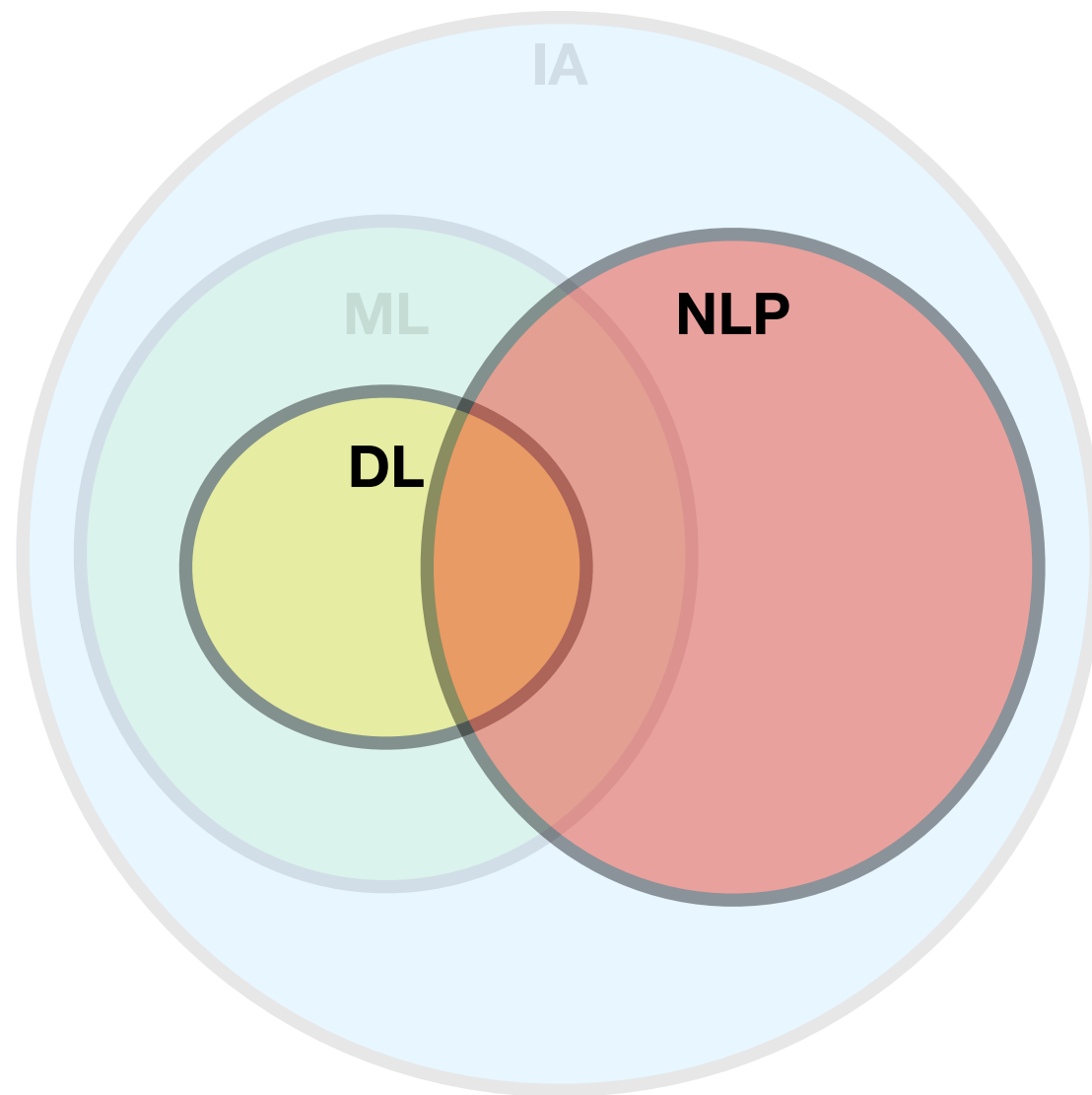
Ces couches supplémentaires n'ont pas de connections horizontales

Problème de la disparition du gradient



- les RNN classiques ne sont pas très bon pour les **dépendances à long terme** à cause de la disparition du gradient
- **solution** : LSTM / GRU (classique) / **BERT** (2018) sont des variantes des RNN

Résumé



Modèle RNN « classique »

D'autres architectures RNN

**Modèles Bidirectional-RNN
(BRNN)**

Modèle Deep RNN (DRNN)