Une Introduction aux Réseaux de Neurones Récurrents

Houcine Senoussi

2 février 2026

Table des matières

Table des matières

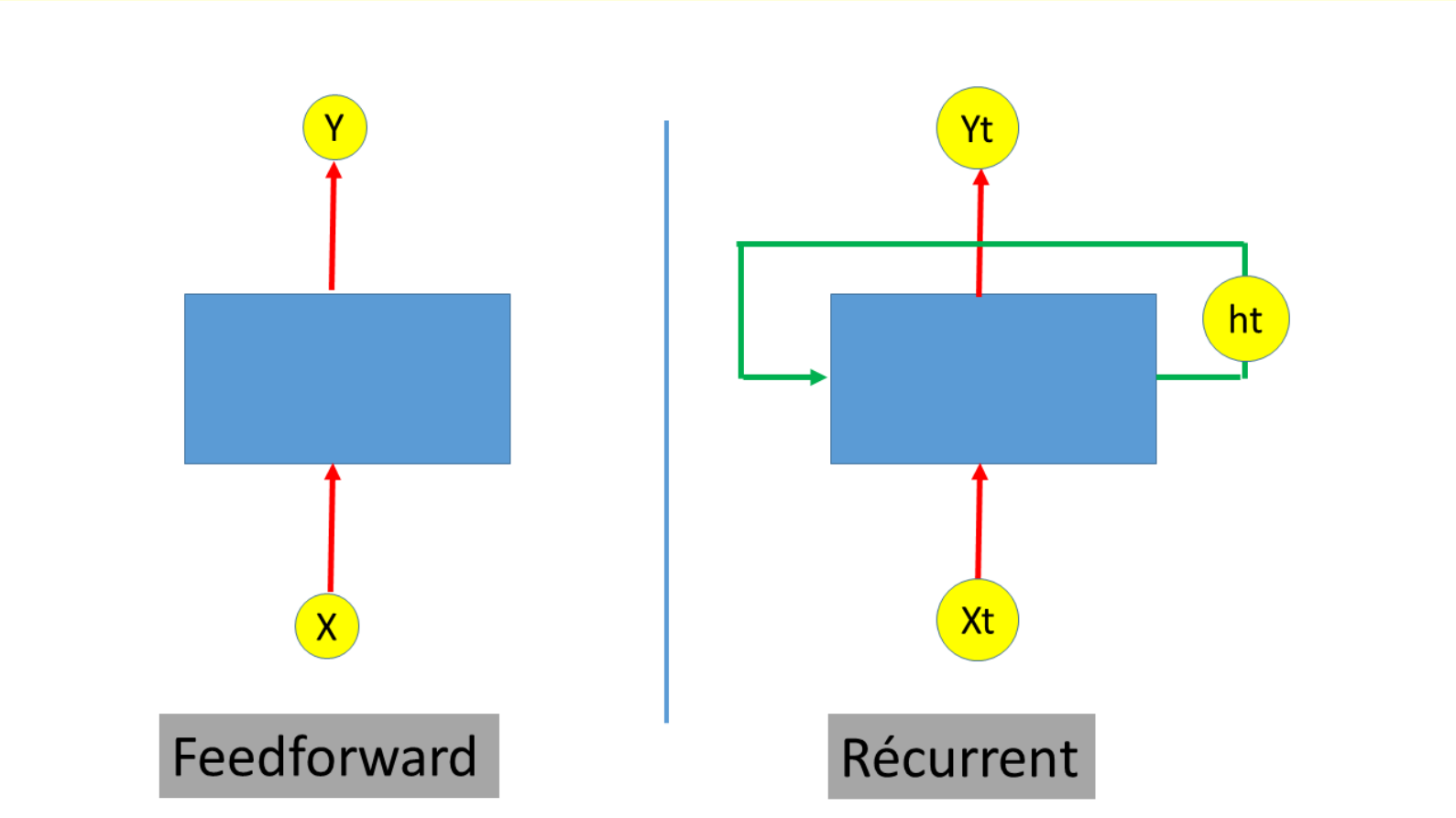
Introduction

- Les réseaux de neurones standards (totalement connectés et feedforward) ainsi les réseaux convolutifs (eux aussi feedforward) acceptent en entrée uniquement des données **taille fixe** (par exemple les valeurs de variables explicatives ou une image). Cette taille est déterminée par un hyperparamètre important : le nombre de neurones dans la couche d'entrée.
- Ils produisent une sortie de taille fixe. Cette taille est déterminée par un autre hyperparamètre important : le nombre de neurones dans la couche de sortie.
- Ils utilisent un nombre fixe de couches (c'est un autre hyperparamètre).
- En raison de ces caractéristiques, il est difficile d'utiliser ces réseaux lorsque l'entrée et/ou la sortie est une **séquence** plutôt qu'un seul vecteur (par exemple une série chronologique de données boursières).

Introduction-2

- Une autre caractéristique importante commune à ces réseaux est qu'ils **n'autorisent pas les cycles** : ils font circuler l'information dans une seule direction (**forward**), de la couche d'entrée vers la couche de sortie, en passant par les couches cachées.
- Si nous relâchons la dernière condition, nous obtenons des réseaux de neurones **récurrents** (**RNN**). En d'autres termes, les RNN sont des réseaux avec des **boucles**, ce qui signifie que la sortie d'une couche, ou de plusieurs couches successives, ou de l'ensemble du réseau, peut être renvoyée vers son entrée afin de calculer la sortie suivante (voir figure ci-dessous).

Introduction-3

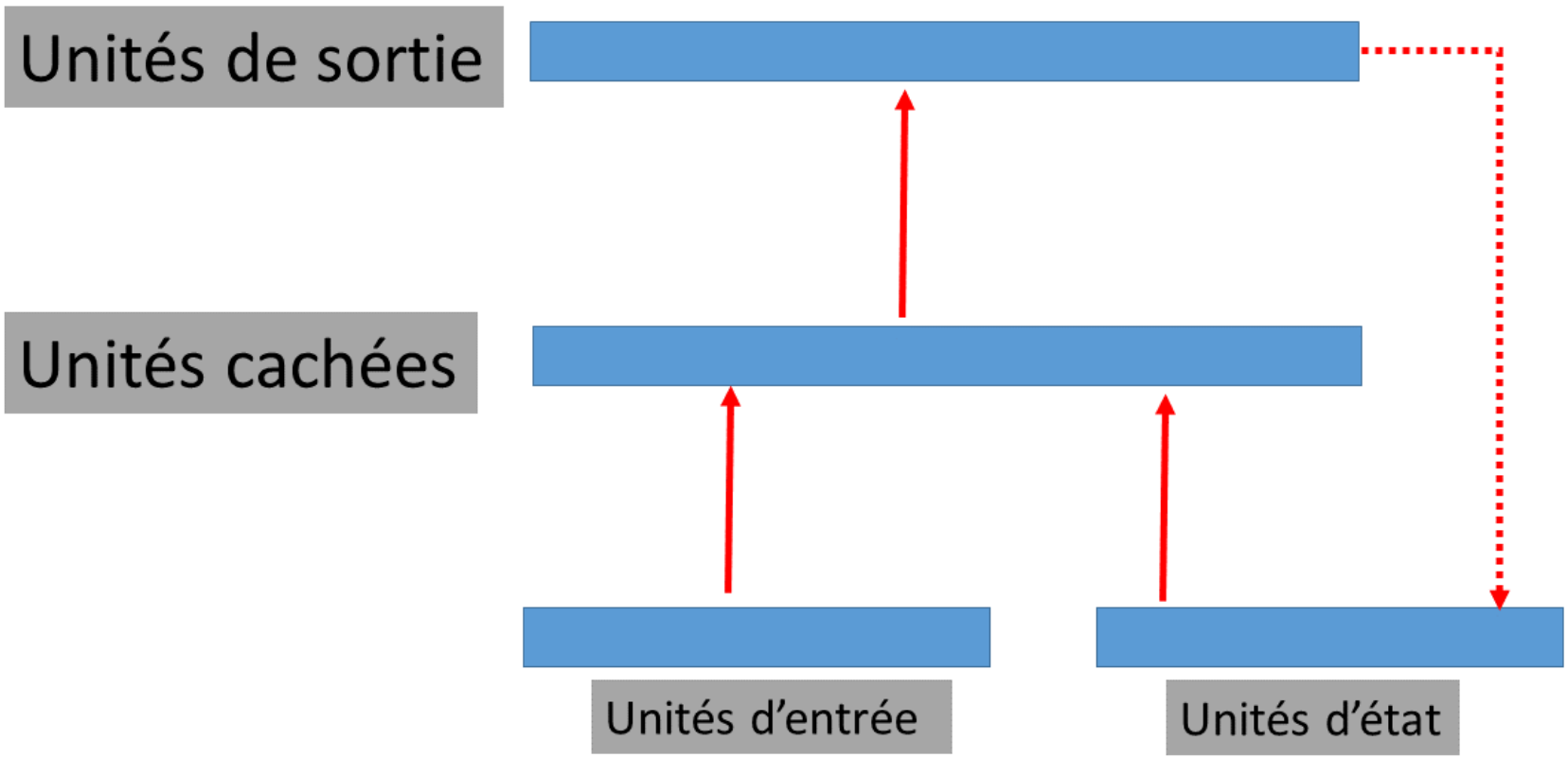


RNN-Un premier exemple : Le réseau de Jordan

- Introduit par M.I. Jordan en 1986.
- Il comporte trois couches. La couche d'entrée contient aussi des **unités d'état**.
- Ces couches sont connectées comme le montre la figure ci-dessous. Les lignes rouges continues représentent les connexions **entraînables**. La ligne rouge pointillée représente les connexions **un pour un** avec un **poids fixe** de $1,0$.
  - $H^t = \sigma_h(U.X^t + V.Y^{t-1} + B^h)$
  - $Y^t = \sigma_y(W.H^t + B^y)$

  où U, V et W sont des matrices de poids, B^h et B^y sont des vecteurs de biais, et σ_h et σ_y sont des fonctions d'activation.

Le réseau de Jordan-2



RNN-Un deuxième exemple : le réseau d'Elman

- Introduit par J.L. Elman en 1990.
- Il comporte trois couches. La couche d'entrée contient aussi des **unités contexte**.
- Ces couches sont connectées comme le montre la figure ci-dessous. Les lignes rouges continues représentent les connexions **entraînables**. La ligne rouge pointillée représente les connexions **un pour un** avec un **poids fixe** de 1.0.
  - $H^t = \sigma_h(U.X^t + V.H^{t-1} + B^h)$
  - $Y^t = \sigma_y(W.H^t + B^y)$

  où U, V et W sont des matrices de poids, B^h et B^y sont des vecteurs de biais, et σ_h et σ_y sont des fonctions d'activation.

Le réseau d'Elman-2

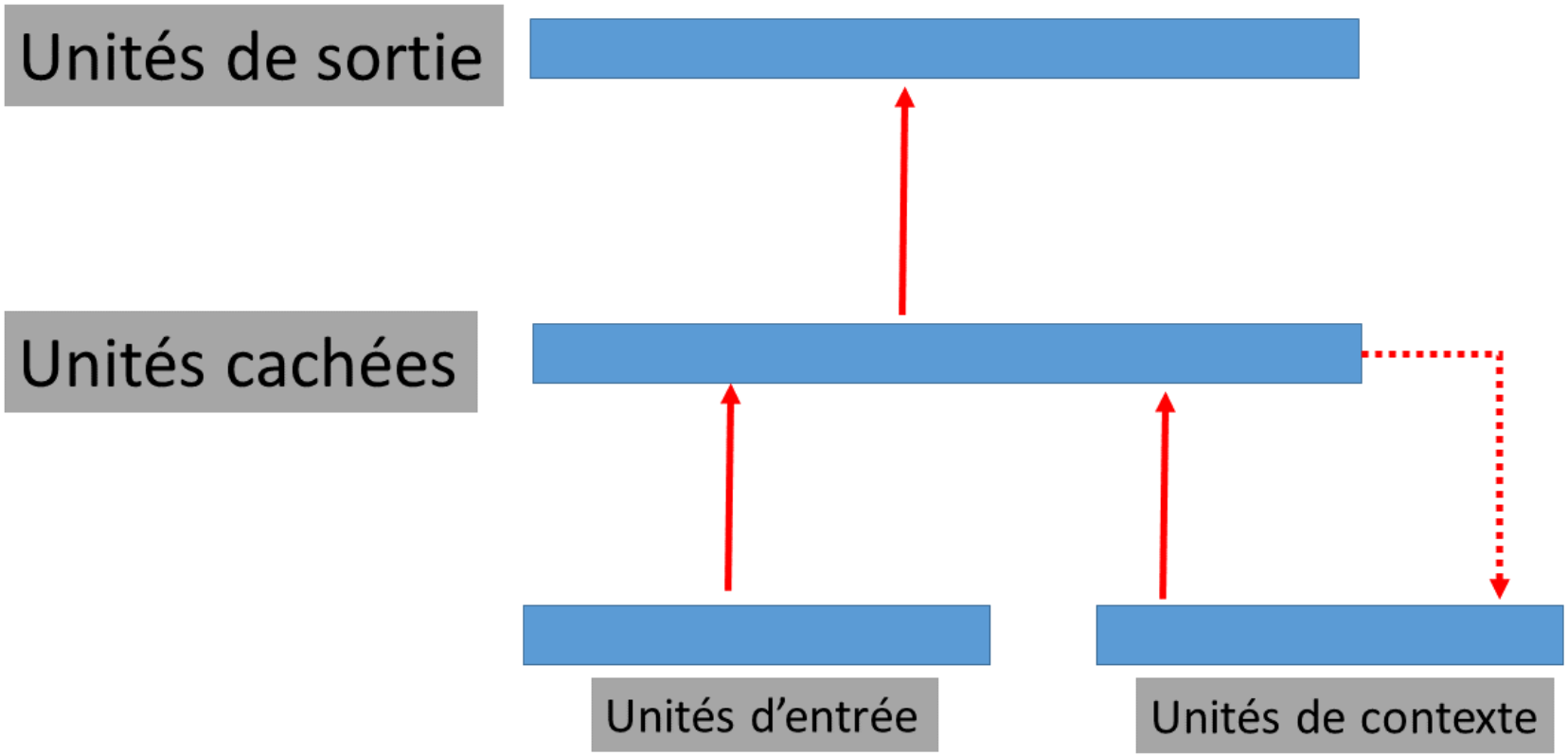


Table des matières

Définitions

- Comme nous l'avons dit plus haut, tout réseau de neurones ayant des cycles est un RNN. Mais le RNN **conventionnel** est défini comme suit :

Definition :

Un réseau de neurones récurrent (RNN) possède une couche d'entrée, une couche cachée et une couche de sortie. Son entrée est une séquence X_t, $t = 1, ..., T$ de vecteurs dont la longueur T est variable. Pour chaque valeur de t, la sortie H_t de la couche cachée, appelée **état caché (hidden state)**, et la sortie Y_t du réseau sont définies par :

- $H^t = \sigma_h(U.X^t + V.H^{t-1} + B^h)$
- $Y^t = \sigma_o(W.H^t + B^o)$

où U, V et W sont des matrices de poids, B^h et B^o sont des vecteurs de biais, et σ_h et σ_o sont des fonctions d'activation. Il est habituel d'utiliser la fonction sigmoïde ou la fonction tangente hyperbolique pour σ_h.

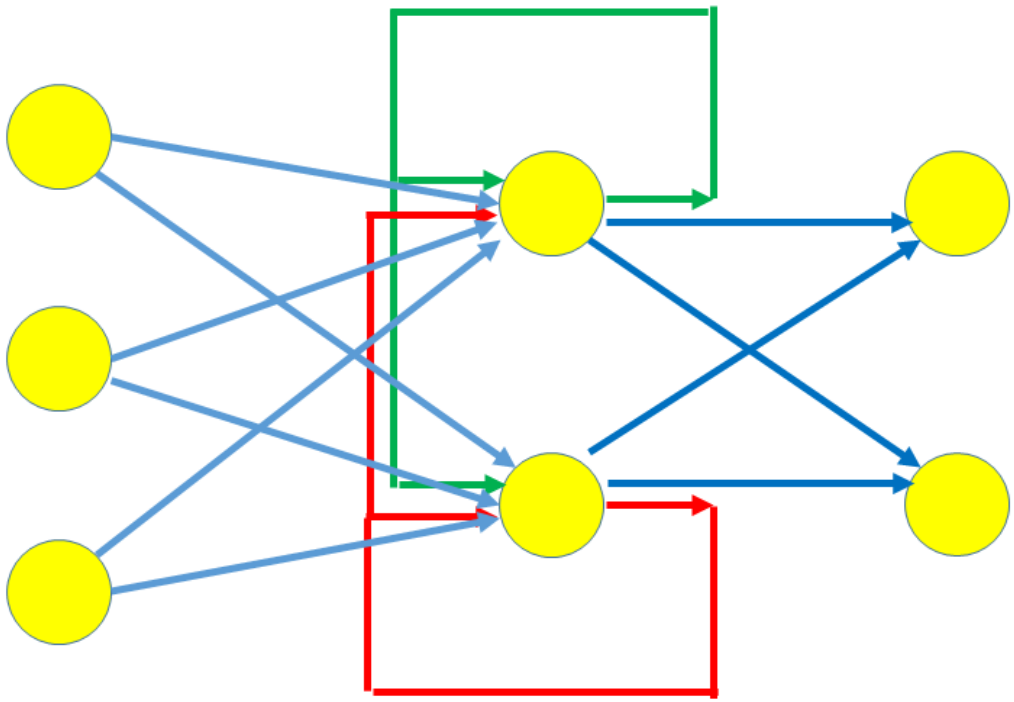


Figure – Un exemple de RNN

Définitions-3

- **Remarque** : Ce qui est véritablement défini ci-dessus est une **couche** récurrente (la couche cachée). Nous verrons qu'une telle couche peut être utilisée dans un réseau récurrent "**profond** (deep)" comportant plusieurs couches et/ou dans un réseau contenant différents types de couches.

Définitions-4

- Dans ce qui suit, nous donnons les équations définissant la sortie de chaque neurone des couches cachée et de sortie.
- Appelons n, m et p le nombre de neurones de la couche d'entrée, de la couche cachée et de la couche de sortie, respectivement.
- Pour chaque valeur de t, X^t est un vecteur de dimension n, H^t est un vecteur de dimension m et Y^t un vecteur de dimension p.
- Nous désignerons par x_i^t, h_j^t et y_k^t la i^{eme} entrée du réseau ($X^t[i]$), la sortie du j^{eme} neurone de la couche cachée ($H^t[j]$), et la sortie du k^{eme} neurone de la couche de sortie ($Y^t[k]$), respectivement.

Définitions-5

- U est une matrice de dimension (m, n). $U[j, i]$, que nous désignerons également par u_{ij} est le poids de la connexion entre le i^{eme} neurone de la couche d'entrée et le j^{eme} neurone de la couche cachée.
- V est une matrice de dimension (m, m). $V[j, j']$, que nous désignerons également par $V_{j'j}$ est le poids de la connexion joignant la sortie du j'^{eme} neurone de la couche cachée à l'entrée du j^{eme} neurone de la couche cachée.
- B^h est un vecteur de dimension m. $B^h[j]$, que nous désignerons également par b^h_j, est le biais du j^{eme} neurone de la couche cachée.

- W est une matrice de dimension (p, m). $W[k, j]$, que nous désignerons également par W_{jk} est le poids de la connexion entre le j^{eme} neurone de la couche cachée et le k^{eme} neurone de la couche de sortie.
- B^o est un vecteur de dimension p. $B^o[k]$, que nous désignerons également par b_k^o est le biais du k^{eme} neurone de la couche de sortie.

- En utilisant toutes ces notations et la définition des RNN donnée ci-dessus, nous déduisons les équations suivantes définissant les sorties des couches cachée et de sortie :

  For $t = 1, \cdots, T$

  - Pour $j = 1, \cdots, m$, $h_j^t = \sigma_h(\sum_{i=1}^{n} u_{ij} x_i^t + \sum_{j'=1}^{m} v_{j'j} h_{j'}^{t-1} + b_j^h)$
  - Pour $k = 1, \cdots, p$, $y_k^t = \sigma_o(\sum_{j=1}^{m} w_{jk} h_j^t + b_k^o)$

Paramètres et hyperparamètres d'un RNN

- Hyperparamètres :
  - Nombre de couches cachées, valeur par défaut 1.
  - Nombre de neurones dans chaque couche.
  - Fonction d'activation de la ou des couches cachées et de celle de la couche de sortie.
- Paramètres : poids et biais.
- Comme d'habitude, les hyperpamètres sont déterminés par le concepteur du réseau. Certaines « règles empiriques » peuvent être trouvées ici et là, mais il n'existe pas de véritables méthodes pour fixer les hyperpamètres.
- Les paramètres sont **appris** par une version spéciale de rétropropagation : **rétropropagation dans le temps** (voir ci-dessous).

Déplier/dérouler les RNN

- Pour voir explicitement le **flux d'information** dans un RNN, nous pouvons **le "déplier" dans le temps**, c'est-à-dire le considérer comme une séquence de copies, une copie pour chaque valeur de t, du même réseau. La figure ci-dessous montre le résultat d'un tel déroulement (seule la couche cachée a été représentée).

Déplier les RNN-2

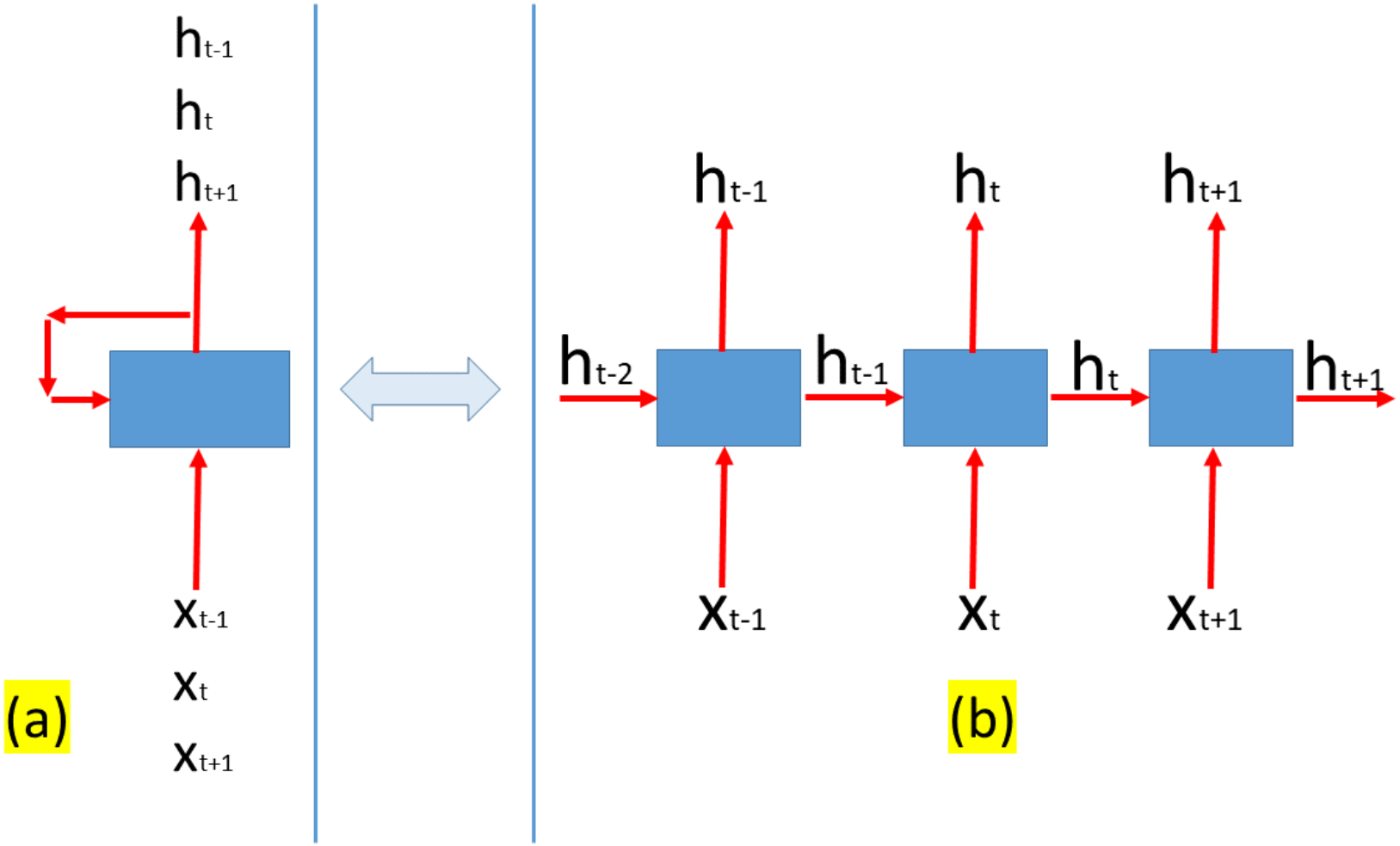


Table des matières

Traitement des données séquentielles

- La principale raison pour laquelle nous avons besoin de RNN est qu'il est "bon pour" le traitement des séquences (ce que les réseaux feedforward ne sont pas).
- Les séquences sont des collections de points de données avec un ordre défini, e.g. un ordre basé sur le temps.
- Exemples de données séquentielles :
  - Séries temporelles : e.g. données financières.
  - Documents : un document est une séquence de mots.
  - Images dans les vidéos.
- Grâce à ses boucles, RNN « garde en mémoire » une synthèse des éléments de données (la sous-séquence) déjà traités et l'utilise lors du traitement de l'élément suivant.

Traitement des données séquentielles-2

- Dans un problème **séquence à séquence** (*Seq2Seq*), une séquence source $X_1 \cdots X_t$ est mappée à une séquence cible $Y_1 \cdots Y_{T'}$. Dans le cas général, la taille T' de la séquence de sortie n'est pas nécessairement égale à la taille T de la séquence d'entrée.
- Chaque élément de la séquence d'entrée est représenté par un vecteur de dimension n, où n est le nombre de neurones de la couche d'entrée.
  - Par exemple, les mots peuvent être représentés en utilisant l'encodage **one-hot** ou une méthode **word embedding**.
- Un problème Seq2Seq bien connu dans lequel les RNN sont utilisés est la **traduction automatique**, dans laquelle une séquence d'entrée est un document dans une langue donnée et la séquence de sortie est un document dans une autre langue.

Traitement des données séquentielles-3

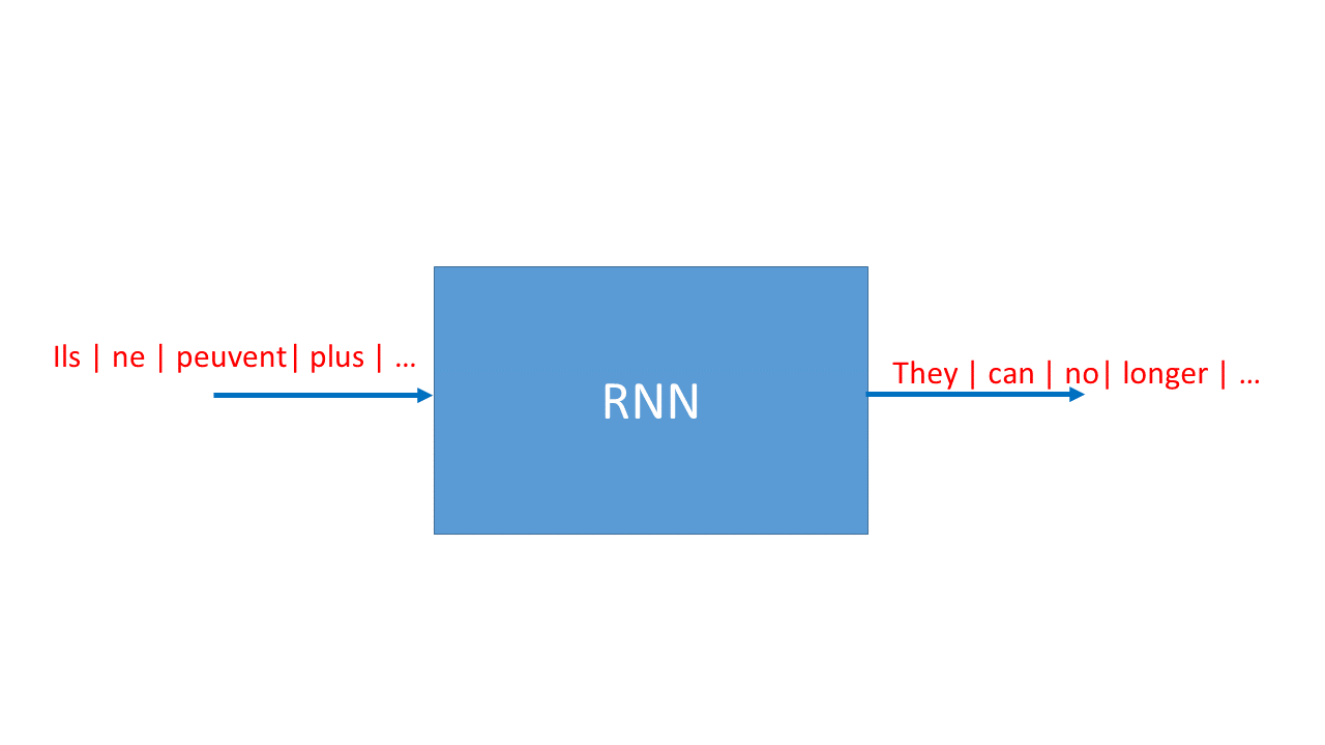


Figure – Traitement des données séquentielles.

Traitement des données séquentielles-4

- Intéressons-nous aux **entrées** et **sorties** d'un problème Seq2Seq. Trois types de problèmes Seq2seq peuvent être distingués en fonction de la taille T de la séquence d'entrée et de celle T' de la séquence de sortie :
  - Les problèmes **plusieurs-à-plusieurs** (**Many-to-many**) dans lesquels nous avons une séquence en entrée et une séquence en sortie. La **traduction** automatique appartient à cette catégorie.
  - Les problèmes **plusieurs-à-un** (**Many-to-one**) dans lesquels la sortie ne contient qu'un seul élément. Par exemple, dans l'**analyse des sentiments**, nous avons besoin que le RNN ne génère qu'une seule valeur (par exemple positive ou négative) résumant le document d'entrée.
  - Les problèmes **un à plusieurs** (**One-to-many**) : recevant une seule entrée, le RNN génère une séquence. L'image **captioning** est un exemple d'un tel problème : son entrée est une image et sa sortie est une séquence de mots.

Traitement des données séquentielles-Un exemple

- Supposons que nous ayons un vocabulaire $A = \{a_1, \cdots, a_{|A|}\}$ contenant $|A| = 10000$ mots, et que chaque mot est représenté par un vecteur réel de dimension n avec $n = 50$.
- Définissons un RNN capable de « deviner » le mot suivant dans une séquence de mots. Un tel RNN peut être défini comme suit :
  - Il a n neurones dans la couche d'entrée, 500 neurones dans la couche cachée et $|A|$ neurones dans la couche de sortie.
  - La fonction d'activation de la couche cachée est sigmoïde.
  - La couche de sortie a $|A|$ neurones et **softmax** comme fonction d'activation.

- Il s'ensuit que le vecteur de sortie Y^t est défini comme suit :

$$\text{Pour } k = 1, \cdots, |A| \; Y^t[k] = \frac{exp((W.H^t + B^o)[k])}{\sum_{k'=1}^{|A|} exp((W.H^t + B^o)[k'])}$$

- Pour chaque valeur de j, $Y_t[j]$ est la probabilité que le mot suivant après la séquence $X_1 \cdots X_t$ soit w_j. En d'autres termes, nous avons :

$$p(X_t = w_j | X_1, \cdots, X_t) = \frac{exp((W_o.H_t + B_o)[j])}{\sum_{i=1}^{|V|} exp((W_o.H_t + B_o)[i])}$$

- L'application décrite dans cet exemple constitue la base d'un sous-domaine important du traitement du langage naturel, la **modélisation du langage**, dans lequel le but est d'apprendre une distribution de probabilité sur des séquences de mots.

Table des matières

Apprentissage dans les RNNs

- Rappelons que dans un réseau de neurones, l'apprentissage est basé sur la minimisation de l'erreur, c'est-à-dire la différence entre les sorties prédites et cibles, et que cette erreur est calculée à l'aide d'une **fonction de perte** (par exemple l'erreur quadratique, l'entropie croisée binaire et l'entropie croisée catégorielle).
- L'erreur est une fonction $E(U, V, W, B^h, B^o)$ des paramètres du réseau, c'est-à-dire les poids et biais des couches cachées et de sortie. On note P l'ensemble de tous ces paramètres.
- On a $E(P) = \sum_t E^t(P)$, où $E^t(P)$ est l'erreur commise lors de l'étape t.
- Nous avons $E^t(P) = loss(\hat{Y}^t, Y^t)$, où $\hat{Y}^t$ est la sortie du réseau et Y^t la valeur cible.

Apprentissage dans les RNNs-2

- L'algorithme utilisé pour minimiser E est la **descente de gradient**. À chaque itération, pour **mettre à jour les poids et les biais**, cet algorithme doit d'abord calculer les dérivées partielles suivantes :
  - $\frac{\partial E}{\partial u_{ij}}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.
  - $\frac{\partial E}{\partial v_{j'j}}$ for $1 \leq j', j \leq m$.
  - $\frac{\partial E}{\partial b_j^h}$ for $1 \leq j \leq m$.
  - $\frac{\partial E}{\partial w_{jk}}$ for $1 \leq j \leq m$ and $1 \leq k \leq p$.
  - $\frac{\partial E}{\partial b_k^o}$ for $1 \leq k \leq p$.

Apprentissage dans les RNNs-3

- Ces calculs se font de la couche de sortie vers la couche cachée, et des valeurs t vers les valeurs $t-1$. C'est la **rétropropagation à travers le temps** (**Backpropagation Through Time**, **BPTT**).

Table des matières

Faiblesses des RNNs

- La principale faiblesse du RNN standard que nous venons de décrire est sa '**mémoire courte**'. Par exemple, dans le cas où les séquences à traiter sont des phrases, lorsque le RNN reçoit en entrée une phrase longue, qu'il doit donc traiter mot par mot, en arrivant à la fin de la phrase il a déjà 'oublié' les premiers mots. Cela peut fausser le traitement, lorsque, dans cette phrase il y a des dépendances longues.
  - Exemple : Elle m'a annoncé son intention de se présenter aux élections. Je n'y croyais pas, mais, par amitié, je lui ai écrit pour lui souhaiter bonne chance.
- Cela est dû au phénomène qu'on appelle **disparition du gradient/explosion du gradient** (**vanishing/exploding gradient**). Autrement dit, durant l'apprentissage, les valeurs du gradient tendent vers zéro, ou à l'inverse prennent des valeurs de plus en plus grandes, ce qui empêche l'apprentissage des **dépendances longues**.

LSTM

- Plusieurs solutions ont été trouvées à ce problème. la plus connue étant une variante de RNN appelée Long Short Term memory (LSTM)
- **LSTM** est composé de neurones plus **sophistiqués**, capables d'**oublier** certaines informations et d'en **renforcer** d'autres.

- Une façon d'introduire LSTM est de commencer par l'une des nouveautés qu'il contient : chaque **cellule** (neurone) de la couche cachée possède une **état** c_t (qui évolue donc avec le temps), qu'on appelle aussi **mémoire**. Cet état dépend à la fois de l'état à l'instant précédent (c_{t-1}) et une nouvelle valeur qu'on peut voir comme un **nouvel état candidat**, et que nous noterons $\tilde{c_t}$. Cette dépendance s'exprime de la manière suivante :
  - $c_t = f_t * c_{t-1} + i_t * \tilde{c_t}$
  - Aussi bien f_t que i_t sont comprises entre 0 et 1 (voir formules plus bas).
  - Lorsque f_t est proche de 0, tout se passe comme si la cellule **oublie** son ancien état. Lorsque f_t est proche de 1, tout se passe comme si elle garde intégralement de son ancien état.
  - Comme f_t détermine la part de l'**ancienne information** que la cellule conserve, i_t détermine la part de la **nouvelle information** que la cellule prend en compte.

- La sortie h_t de la cellule se calcule en fonction de l'état c_t de la manière suivante :
  - $h_t = o_t * tanh(c_t)$
  - o_t est aussi une valeur comprise entre 0 et 1, qui contrôle donc la part de l'état qui 'sortira' de la cellule.
- Les 3 valeurs f_t, i_t, et o_t correspondent à des éléments de la cellule qu'on appelle des **portes** (**gates**), dont le rôle est de moduler (contrôler) les parts d'information qui passent (ouverture de la porte, valeur proche de 1) ou qui sont bloquées (fermeture de la porte, valeur proche de 0).
- Ces 3 portes s'appellent respectivement la porte d'oubli (**forget**), d'entrée (**input**) et de sortie (**output**).

LSTM-Les formules

- Ci-dessous les valeurs de f_t, i_t, et o_t sous forme matricielle, c'est-à-dire pour l'ensemble des cellules de la couche plutôt que pour une cellule individuelle :
  - $I_t = \sigma(W_{xi}X_t + W_{hi}H_{t-1} + B_i)$
  - $F_t = \sigma(W_{xf}X_t + W_{hf}H_{t-1} + B_f)$
  - $O_t = \sigma(W_{xo}X_t + W_{ho}H_{t-1} + B_o)$
- Remarque : il existe plusieurs variantes de ces formules. Il en existe notamment une qui fait dépendre i_t, f_t, et o_t de c_{t-1} aussi.
- La valeur de $\tilde{c_t}$ est donnée par :
  - $\tilde{C}_t = tanh(W_{xc}X_t + W_{hc}H_{t-1} + B_c)$

Table des matières

Encodeur-Décodeur

- RNN et ses variantes peuvent être utilisés directement dans les problèmes Seq2seq, mais une façon plus efficace de les utiliser est d'en mettre deux, dans une structure appelée **encodeur-décodeur**(figure3).
- Dans un encodeur-décodeur (figure4), la séquence d'entrée (e.g. la phrase à traduire) est mise à l'entrée de l'**encodeur** (premier RNN) qui traite cette séquence et la **synthétise** sous la forme de ce qu'on appelle un **vecteur-contexte** qui sera envoyé à l'entrée du **décodeur** (deuxième RNN) qui à l'issue de son traitement produira la séquence de sortie (e.g. traduction de la phrase).

Encodeur-Décodeur-2

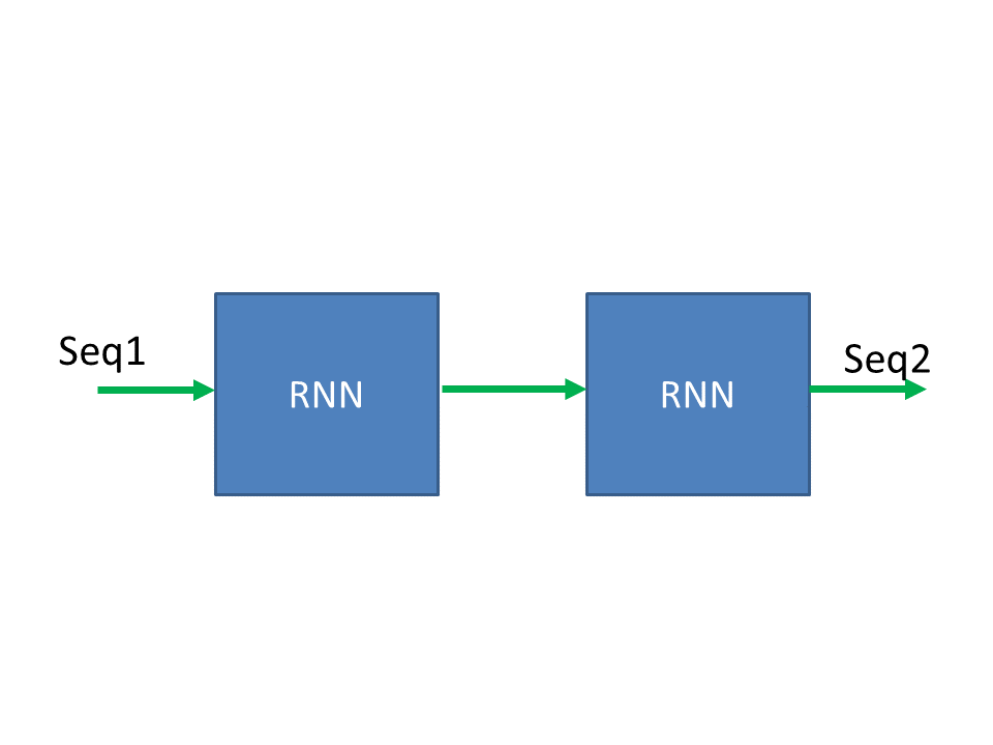


Figure – Encodeur-Décodeur (1).

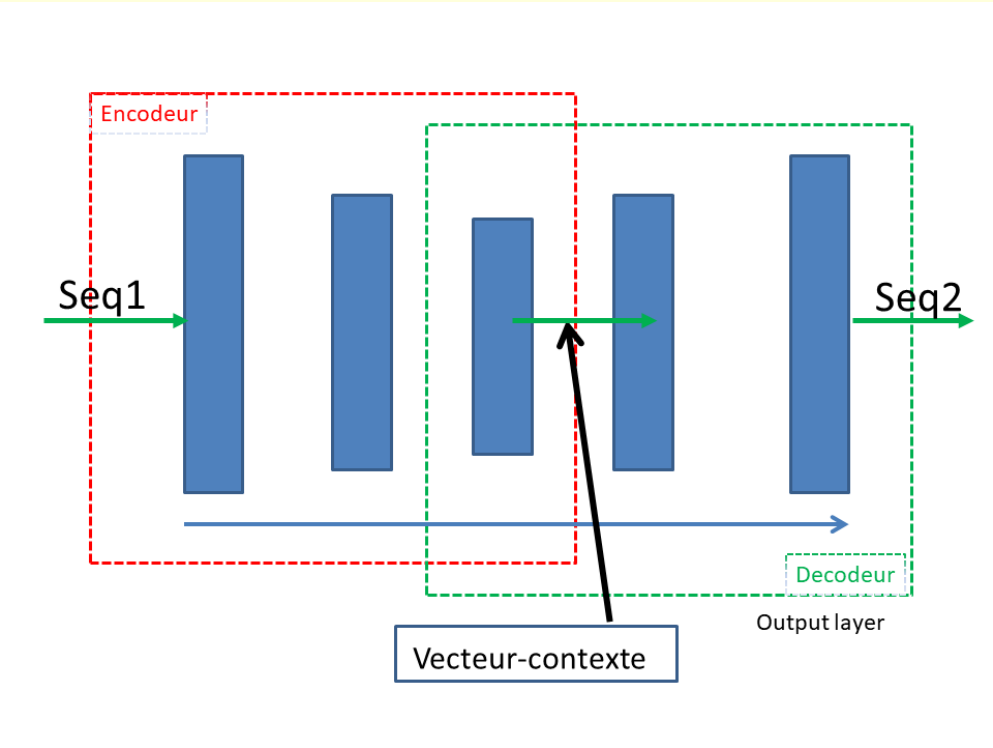


Figure – Encodeur-Décodeur (2).

Le mécanisme d'attention

- Le principe de l'**attention** est assez intuitif (et finalement assez simple) : pour traduire un mot dans une phrase, on n'a pas besoin de la **totalité** de la phrase mais de **quelques mots**. De même, pour reconnaître une personne dans une image, on n'a pas besoin de la totalité de l'image mais de certaines parties, par exemple celles qui semblent correspondre à des visages.
- L'encodeur-décodeur décrit plus haut s'appuie pour la traduction (plus généralement les problèmes Seq2seq) sur une **synthèse** de la phrase (plus généralement séquence) introduite dans l'encodeur.
- L'objet du mécanisme d'attention, ajouté à l'encodeur-décodeur, est donc de (1) ne pas se contenter d'une syntèse de la totalité de la phrase mais d'utiliser plus d'informations issues de l'encodeur, et (2) de **souligner**, à chaque moment du traitement (e.g. traduction) les parties de la phrase qui sont les plus importantes pour le traitement (e.g. pour la traduction du mot en cours).

Le mécanisme d'attention-2

- Ce mécanisme nécessite l'ajout à l'encodeur-décodeur d'un autre réseau de neurones (un RN standard, pas besoin d'un RNN).
- Il utilise les sorties successives de l'encodeur (une sortie correspond au traitement d'un mot).
- Il donne des **poids** différents, c'est-à-dire une **importance** différente, aux différentes sorties (donc aux mots correspondants).
- Comment ces poids sont-ils calculés ? par le nouveau réseau de neurones.
- Comment le nouveau RN calcule-t-il ces poids ? Il **apprend** à le faire en étant **entraîné** en même temps que le reste de l'encodeur-décodeur.

Le mécanisme d'attention-Exemple

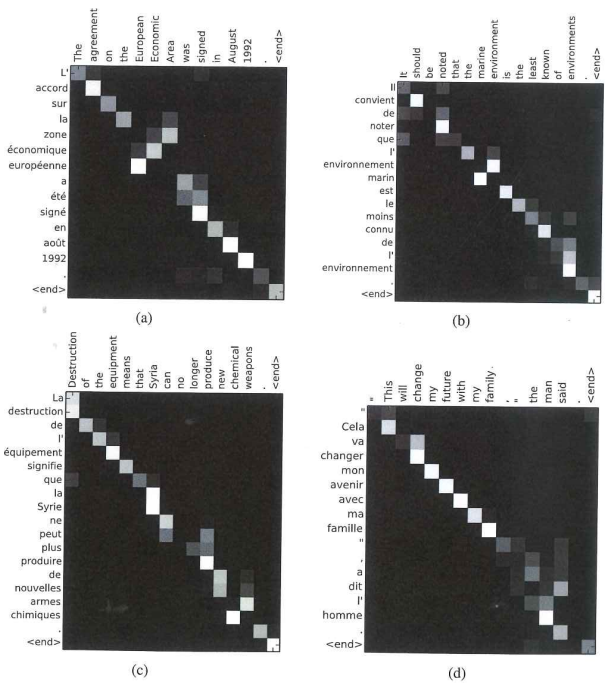


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

Le transformeur

- Le transformeur est une **architecture** (complexe) servant à traiter des données séquentielles.
- Il est basé sur le mécanisme d'**attention** qu'il utilise sans passer par les RNN.
- Plusieurs composantes du transformeur utilisent l'attention.
- Le transformeur est conçu pour le **passage à l'échelle**, autrement dit avoir plus de données le rend plus efficace. Ceci est notamment dû à son fonctionnement **parallèle**.

Table des matières

Conclusion

- Réseaux de neurones récurrents : définitions et utilisation.
- Traitement des données séquentielles.
- Des RNNs au transformeurs.