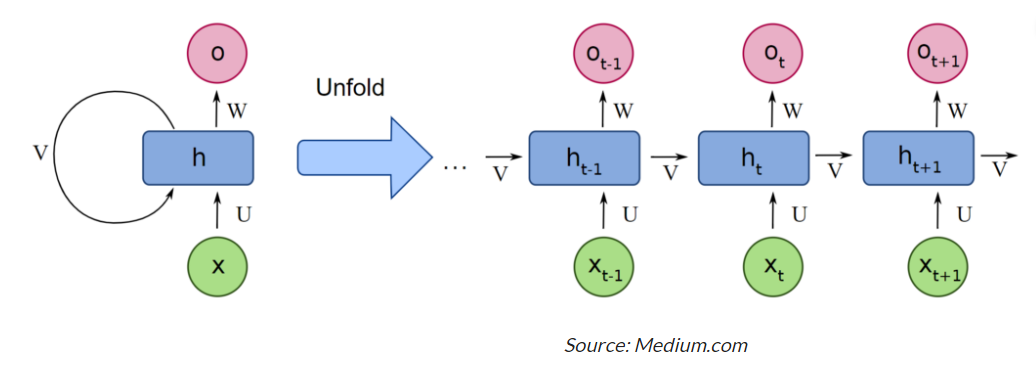
Un bref aperçu des réseaux de neurones récurrents (RNN)

Source : <https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/>

# Introduction sur les réseaux de neurones récurrents

Une approche d’apprentissage profond pour la modélisation de données séquentielles est Recurrent Neural Networks (RNN). Les RNN étaient la suggestion standard pour travailler avec des données séquentielles avant l’avènement des modèles d’attention. Des paramètres spécifiques pour chaque élément de la séquence peuvent être requis par un modèle d’avance en profondeur. Il peut également être incapable de généraliser aux séquences de longueur variable.



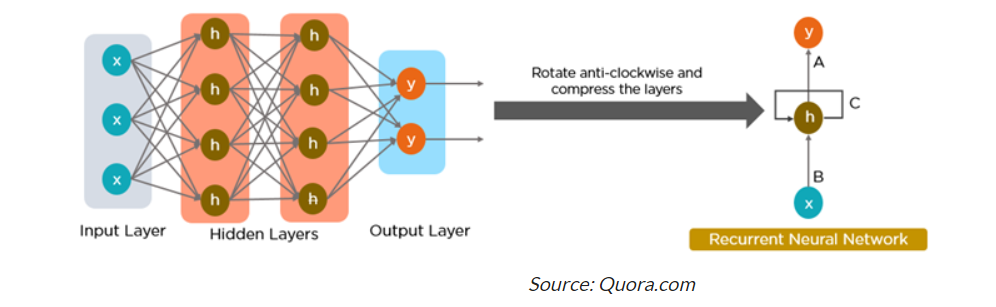
Les réseaux de neurones récurrents utilisent les mêmes poids pour chaque élément de la séquence, ce qui diminue le nombre de paramètres et permet au modèle de se généraliser à des séquences de longueurs variables. Les RNN se généralisent aux données structurées autres que les données séquentielles, telles que les données géographiques ou graphiques, en raison de leur conception.

Les réseaux de neurones récurrents, comme beaucoup d’autres techniques d’apprentissage profond, sont relativement anciens. Ils ont été développés pour la première fois dans les années 1980, mais nous n’avons pas apprécié leur plein potentiel jusqu’à récemment. L’avènement de la mémoire à long terme à court terme (LSTM) dans les années 1990, combiné à une augmentation de la puissance de calcul et aux grandes quantités de données que nous devons maintenant traiter, a vraiment poussé les RNN au premier plan.

# Qu’est-ce qu’un réseau neuronal récurrent (RNN) ?

Les réseaux de neurones imitent la fonction du cerveau humain dans les domaines de l’IA, de l’apprentissage automatique et de l’apprentissage profond, permettant aux programmes informatiques de reconnaître des modèles et de résoudre des problèmes courants.

Les RNN sont un type de réseau neuronal qui peut être utilisé pour modéliser des données de séquence. Les RNN, qui sont formés à partir de réseaux d’alimentation, sont similaires au cerveau humain dans leur comportement. En termes simples, les réseaux de neurones récurrents peuvent anticiper les données séquentielles d’une manière que d’autres algorithmes ne peuvent pas.

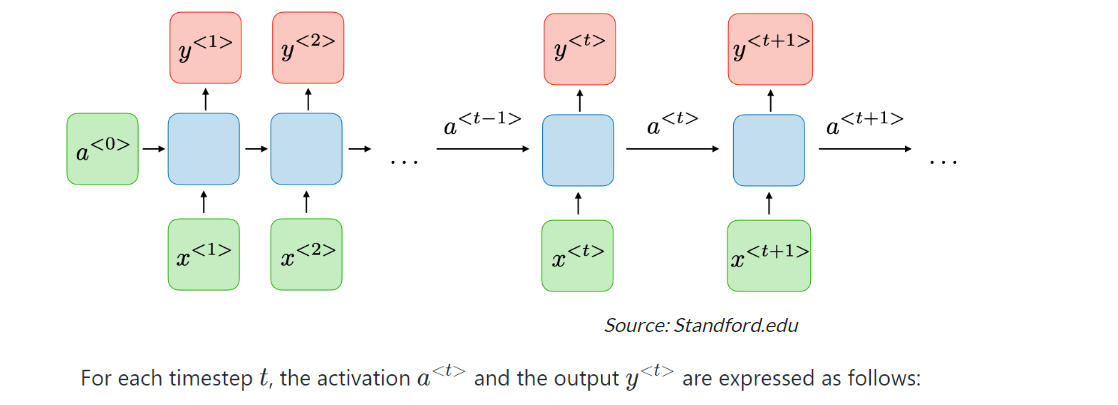


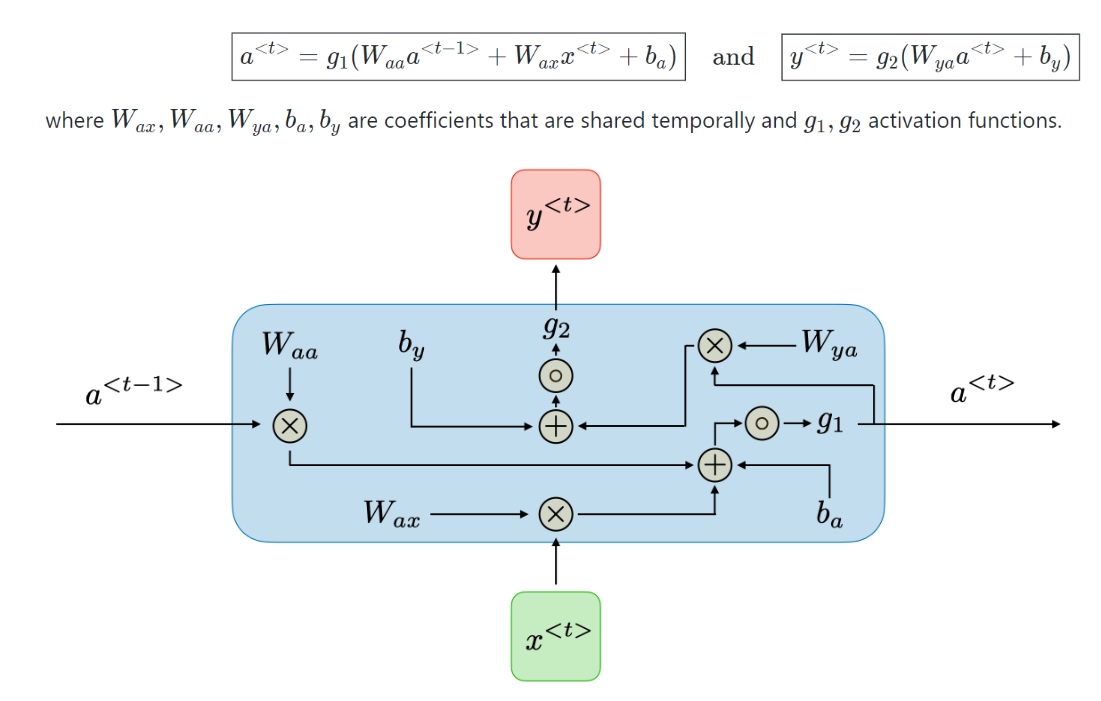
Toutes les entrées et sorties dans les réseaux de neurones standard sont indépendantes les unes des autres, mais dans certaines circonstances, comme lors de la prédiction du mot suivant d’une phrase, les mots précédents sont nécessaires, et donc les mots précédents doivent être mémorisés. En conséquence, RNN a été créé, qui a utilisé une couche cachée pour surmonter le problème. Le composant le plus important de RNN est l’état Caché, qui mémorise des informations spécifiques sur une séquence.

Les RNN ont une mémoire qui stocke toutes les informations sur les calculs. Il utilise les mêmes paramètres pour chaque entrée car il produit le même résultat en effectuant la même tâche sur toutes les entrées ou couches cachées.

# L’architecture d’un RNN traditionnel

Les RNN sont un type de réseau neuronal qui a des états cachés et permet d’utiliser les sorties passées comme entrées. Ils vont généralement comme ceci:





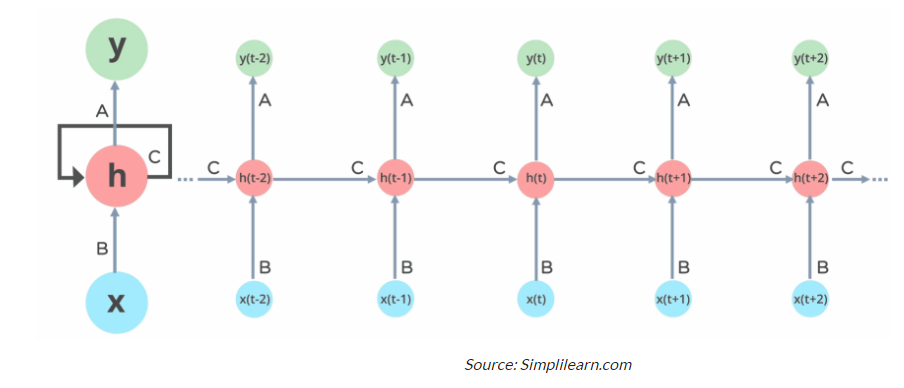
L’architecture RNN peut varier en fonction du problème que vous essayez de résoudre. De ceux qui ont une seule entrée et sortie à ceux qui en ont plusieurs (avec des variations entre les deux).

Vous trouverez ci-dessous quelques exemples d’architectures RNN qui peuvent vous aider à mieux comprendre cela.

* **Un à un:** Il n’y a qu’une seule paire ici. Une architecture one-to-one est utilisée dans les réseaux de neurones traditionnels.
* **Un à plusieurs :** Une seule entrée dans un réseau un-à-plusieurs peut entraîner de nombreuses sorties. Un réseau de trop est utilisé dans la production de musique, par exemple.
* **Plusieurs à un:** Dans ce scénario, une seule sortie est produite en combinant de nombreuses entrées à partir de pas de temps distincts. L’analyse des sentiments et l’identification des émotions utilisent de tels réseaux, dans lesquels l’étiquette de classe est déterminée par une séquence de mots.
* **Beaucoup à beaucoup:** Pour beaucoup à beaucoup, il existe de nombreuses options. Deux entrées produisent trois sorties. Les systèmes de traduction automatique, tels que l’anglais vers Français ou vice versa les systèmes de traduction, utilisent de nombreux réseaux.

# Comment fonctionnent les réseaux de neurones récurrents?

L’information dans les réseaux neuronaux récurrents passe par une boucle jusqu’à la couche cachée du milieu.



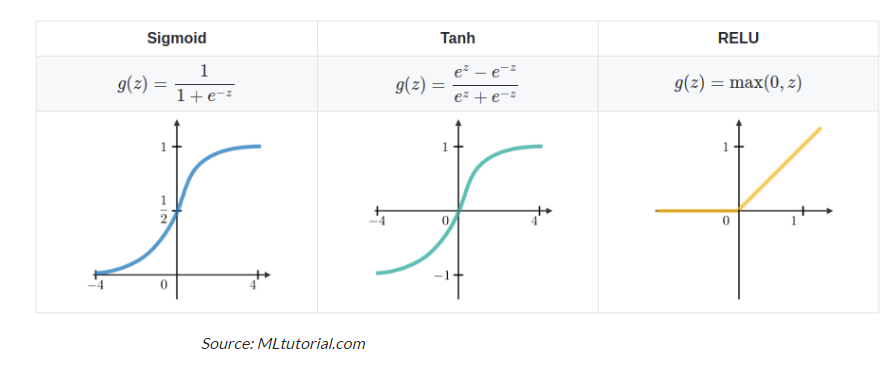
La couche d’entrée x reçoit et traite l’entrée du réseau neuronal avant de la transmettre à la couche intermédiaire.

Plusieurs couches cachées peuvent être trouvées dans la couche intermédiaire **h**, chacune avec ses propres fonctions d’activation, poids et biais. Vous pouvez utiliser un réseau neuronal récurrent si les différents paramètres des différentes couches cachées ne sont pas affectés par la couche précédente, c’est-à-dire qu’il n’y a pas de mémoire dans le réseau neuronal.

Les différentes fonctions d’activation, poids et biais seront normalisés par le réseau neuronal récurrent, garantissant que chaque couche cachée a les mêmes caractéristiques. Plutôt que de construire de nombreuses couches cachées, il n’en créera qu’une seule et la survolera en boucle autant de fois que nécessaire.

# Fonctions d’activation courantes

La fonction d’activation d’un neurone dicte s’il doit être activé ou désactivé. Les fonctions non linéaires transforment généralement la sortie d’un neurone en un nombre compris entre 0 et 1 ou -1 et 1.



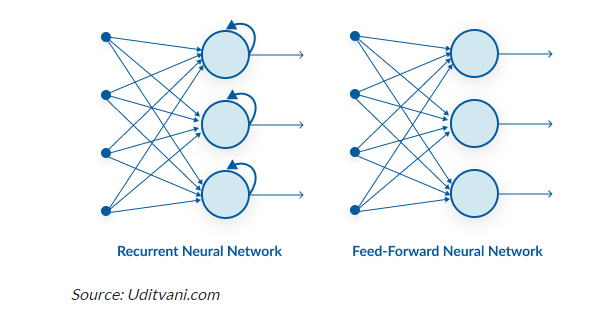
Voici quelques-unes des fonctions les plus couramment utilisées :

* **Sigmoïde :** La formule **g(z) = 1/(1 + e^-z)** est utilisée pour exprimer cela.
* **Tanh:** La formule **g(z) = (e^-z – e^-z)/(e^-z + e^-z)** est utilisée pour exprimer cela.
* **Relu :** La formule **g(z) = max(0 , z)** est utilisée pour exprimer cela.

# Réseau neuronal récurrent vs réseau neuronal Feedforward

Un réseau neuronal d’avance n’a qu’une seule voie de flux d’informations : de la couche d’entrée à la couche de sortie, en passant par les couches cachées. Les données circulent sur le réseau dans un itinéraire droit, sans jamais passer deux fois par le même nœud.

Le flux d’informations entre un RNN et un réseau neuronal d’alimentation directe est illustré dans les deux figures ci-dessous.



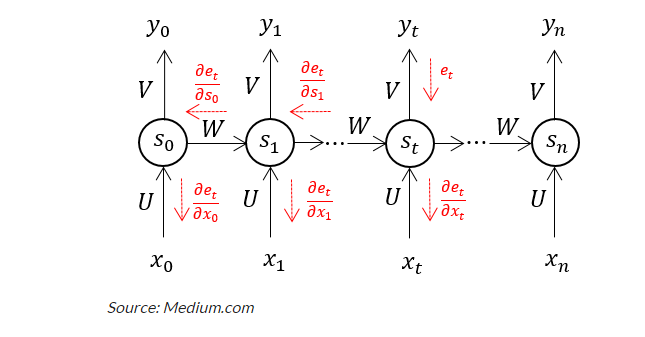
Les réseaux de neurones feed-forward sont de mauvaises prédictions de ce qui se passera ensuite parce qu’ils n’ont aucune mémoire des informations qu’ils reçoivent. Parce qu’il analyse simplement l’entrée actuelle, un réseau de feed-forward n’a aucune idée de l’ordre temporel. En dehors de sa formation, il n’a aucun souvenir de ce qui s’est passé dans le passé.

L’information est dans un cycle RNN via une boucle. Avant de porter un jugement, il évalue les entrées actuelles ainsi que ce qu’il a appris des entrées passées. Un réseau neuronal récurrent, en revanche, peut rappeler en raison de la mémoire interne. Il produit une sortie, la copie, puis la renvoie au réseau.

# Rétropropagation dans le temps (BPTT)

Lorsque nous appliquons un algorithme de rétropropagation à un réseau neuronal récurrent avec des données de séries chronologiques en entrée, nous l’appelons rétropropagation dans le temps.

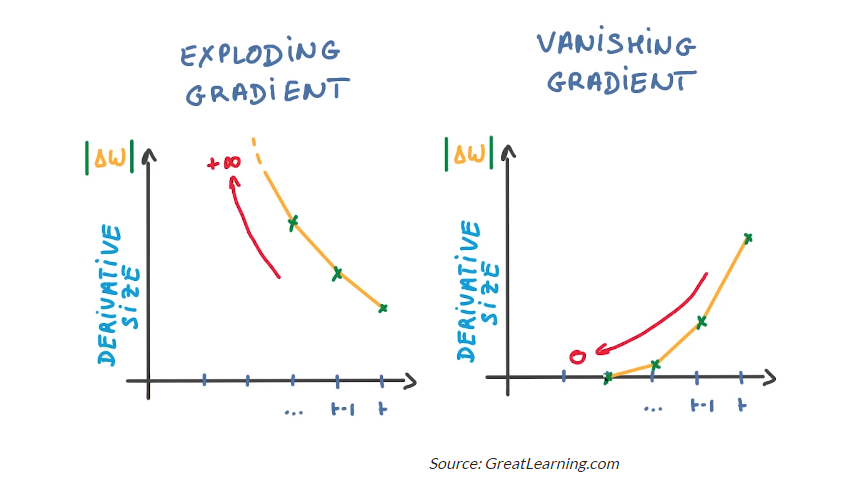
Une seule entrée est envoyée dans le réseau à la fois dans un RNN normal, et une seule sortie est obtenue. La rétropropagation, d’autre part, utilise à la fois les entrées actuelles et antérieures comme entrée. C’est ce qu’on appelle un pas de temps, et un pas de temps consistera en plusieurs points de données de série chronologique entrant dans le RNN en même temps.



La sortie du réseau neuronal est utilisée pour calculer et collecter les erreurs une fois qu’il s’est entraîné sur un ensemble de temps et vous a donné une sortie. Le réseau est ensuite remonté, et les poids sont recalculés et ajustés pour tenir compte des défauts.

# Deux numéros de RNN standard

Il y a deux défis clés que les RNN ont dû surmonter, mais pour les comprendre, il faut d’abord saisir ce qu’est un gradient.



En ce qui concerne ses entrées, un gradient est une dérivée partielle. Si vous n’êtes pas sûr de ce que cela implique, considérez ceci : un gradient quantifie à quel point la sortie d’une fonction varie lorsque les entrées sont légèrement modifiées.

La pente d’une fonction est également connue sous le nom de gradient. Plus la pente est raide, plus un modèle peut apprendre rapidement, plus la pente est élevée. Le modèle, en revanche, cessera d’apprendre si la pente est nulle. Un gradient est utilisé pour mesurer la variation de tous les poids par rapport à la variation de l’erreur.

* **Gradients d’explosion:** Les gradients d’explosion se produisent lorsque l’algorithme donne aux poids une priorité absurdement élevée sans raison apparente. Heureusement, tronquer ou écraser les dégradés est une solution simple à ce problème.
* **Dégradés de disparition :** les dégradés de disparition se produisent lorsque les valeurs de dégradé sont trop petites, ce qui entraîne l’arrêt de l’apprentissage du modèle ou prend beaucoup trop de temps. C’était un gros problème dans les années 1990, et il était beaucoup plus difficile à résoudre que les gradients qui explosent. Heureusement, le concept LSTM de Sepp Hochreiter et Juergen Schmidhuber a résolu le problème.

# RNN Applications

Les réseaux de neurones récurrents sont utilisés pour résoudre une variété de problèmes impliquant des données de séquence. Il existe de nombreux types de données de séquence, mais les suivants sont les plus courants: Audio, Texte, Vidéo, Séquences biologiques.

À l’aide de modèles RNN et de jeux de données de séquences, vous pouvez résoudre divers problèmes, notamment :

* Reconnaissance vocale
* Génération de musique
* Traductions automatiques
* Analyse de l’action vidéo
* Étude séquentielle du génome et de l’ADN

# Conclusion

* Les réseaux de neurones récurrents sont un outil polyvalent qui peut être utilisé dans une variété de situations. Ils sont utilisés dans une variété de méthodes pour la modélisation du langage et les générateurs de texte. Ils sont également utilisés dans la reconnaissance vocale.
* Ce type de réseau neuronal est utilisé pour créer des étiquettes pour les images qui ne sont pas marquées lorsqu’elles sont associées à des réseaux de neurones convolutifs. C’est incroyable à quel point cette combinaison fonctionne.
* Cependant, il y a un défaut avec les réseaux de neurones récurrents. Ils ont du mal à apprendre les dépendances à longue portée, ce qui signifie qu’ils ne comprennent pas les relations entre les données séparées par plusieurs étapes.
* Lorsque nous anticipons des mots, par exemple, nous pouvons avoir besoin de plus de contexte qu’un simple mot précédent. C’est ce qu’on appelle le problème du gradient de disparition, et il est résolu à l’aide d’un type spécial de réseau neuronal récurrent appelé réseaux de mémoire à long et à court terme (LSTM), qui est un sujet plus large qui sera discuté dans les prochains articles.

Code : [implémentation Python assignment\_internship/Basic (RNN avec Keras).ipynb à main · Hntam812/assignment\_internship (github.com)](https://github.com/Hntam812/assignment_internship/blob/main/Basic%20Python%20Implementation%20(RNN%20with%20Keras).ipynb)

LSTM vs GRU dans un réseau neuronal récurrent: une étude comparative

La mémoire à long terme en LSTM court est un type spécial de RNN capable d’apprendre des séquences à long terme. Ils ont été introduits par Schmidhuber et Hochreiter en 1997. Il est explicitement conçu pour éviter les problèmes de dépendance à long terme. Se souvenir des longues séquences pendant une longue période de temps est sa façon de travailler.

Un [réseau neuronal récurrent](https://analyticsindiamag.com/implementing-a-recurrent-neural-network-rnn-from-scratch/) est un type d’ANN qui est utilisé lorsque les utilisateurs souhaitent effectuer des opérations prédictives sur des données séquentielles ou basées sur des [séries](https://analyticsindiamag.com/guide-to-implementing-time-series-analysis-predicting-bitcoin-price-with-rnn/) chronologiques. Ces couches d’apprentissage profond sont couramment utilisées pour des problèmes ordinaux ou temporels tels que [le traitement du langage naturel](https://analyticsindiamag.com/how-to-identify-entities-in-nlp/), la traduction automatique neuronale, [les tâches de sous-titrage d’images](https://analyticsindiamag.com/hands-on-guide-to-effective-image-captioning-using-attention-mechanism/) automatisées et autres. Les appareils d’assistance vocale modernes d’aujourd’hui tels que Google Assistance, Alexa, Siri sont intégrés à ces couches pour offrir des expériences sans tracas aux utilisateurs.

Dans RNN pour former les réseaux, nous revendiquons dans le temps et à chaque pas de temps ou boucle, le gradient est calculé et le gradient est utilisé pour mettre à jour les poids dans les réseaux. Maintenant, si l’effet de la séquence précédente sur le calque est faible, le gradient relatif est calculé petit. Ensuite, si le gradient du calque précédent est plus petit, cela réduit les poids à attribuer au contexte et cet effet est observé lorsque nous traitons des séquences plus longues. En raison de ce réseau n’apprend pas l’effet des entrées antérieures et provoque donc le problème de mémoire à court terme.

Pour surmonter ce problème, des versions spécialisées de RNN sont créées comme LSTM, GRU,...

# ****Fonctionnement de LSTM****

**Qu’est-ce que la mémoire à long terme ou LSTM?**

La mémoire à long terme en LSTM court est un type spécial de RNN capable d’apprendre des séquences à long terme. Ils ont été introduits par Schmidhuber et Hochreiter en 1997. Il est explicitement conçu pour éviter les problèmes de dépendance à long terme. Se souvenir des longues séquences pendant une longue période de temps est sa façon de travailler.

La popularité de LSTM est due au mécanisme Getting impliqué dans chaque cellule LSTM. Dans une cellule RNN normale, l’entrée à l’horodatage et l’état masqué de l’étape de temps précédente sont passés à travers la couche d’activation pour obtenir un nouvel état. Alors que dans LSTM, le processus est légèrement complexe, comme vous pouvez le voir dans l’architecture ci-dessus à chaque fois, il prend l’entrée de trois états différents comme l’état d’entrée actuel, la mémoire à court terme de la cellule précédente et enfin la mémoire à long terme.

Ces cellules utilisent les portes pour réguler les informations à conserver ou à jeter lors du fonctionnement de la boucle avant de transmettre les informations à long terme et à court terme à la cellule suivante. Nous pouvons imaginer ces portes comme des filtres qui suppriment les informations sélectionnées et non pertinentes indésirables. Il y a un total de trois portes que LSTM utilise comme porte d’entrée, porte d’oubli et porte de sortie.

Porte d’entrée

La porte d’entrée décide quelles informations seront stockées dans la mémoire à long terme. Il ne fonctionne qu’avec les informations de l’entrée actuelle et la mémoire à court terme de l’étape précédente. À cette porte, il filtre les informations des variables qui ne sont pas utiles.

### **Oublier la porte**

L’oubli décide quelles informations de la mémoire à long terme doivent être conservées ou rejetées et cela se fait en multipliant la mémoire à long terme entrante par un vecteur d’oubli généré par l’entrée actuelle et la mémoire courte entrante.

### **Porte de sortie**

La porte de sortie prendra l’entrée actuelle, la mémoire à court terme précédente et la mémoire à long terme nouvellement calculée pour produire une nouvelle mémoire à court terme qui sera transmise à la cellule lors de la prochaine étape temporelle. La sortie du pas de temps actuel peut également être tirée de cet état caché.

Il s’agit donc du mécanisme de LSTM pour réaliser cela avec une mise en œuvre pratique. [Ici,](https://colab.research.google.com/drive/1-c36k29ptuMUXUrQzH5SrQ-24t5pFZZL?usp=sharing) j’ai démontré le cas d’utilisation LSTM dans lequel vous pouvez vérifier les séquences d’entrée et de sortie avec leur forme.

# ****Fonctionnement du GRU****

**Qu’est-ce que l’unité récurrente fermée ou GRU?**

Le flux de travail de l’unité récurrente fermée, en abrégé GRU, est le même que celui du RNN, mais la différence réside dans le fonctionnement et les portes associées à chaque unité GRU. Pour résoudre le problème rencontré par le RNN standard, GRU intègre les deux mécanismes de fonctionnement de la porte appelés porte de mise à jour et porte de réinitialisation.

### **Porte de mise à jour**

La porte de mise à jour est chargée de déterminer la quantité d’informations précédentes qui doivent être transmises à l’état suivant. C’est vraiment puissant car le modèle peut décider de copier toutes les informations du passé et d’éliminer le risque de disparition du gradient.

### **Porte de réinitialisation**

La porte de réinitialisation est utilisée à partir du modèle pour décider de la quantité d’informations passées à négliger; en bref, il décide si l’état de la cellule précédente est important ou non.

Tout d’abord, la porte de réinitialisation entre en action, elle stocke les informations pertinentes de l’étape de temps passée dans un nouveau contenu de mémoire. Ensuite, il multiplie le vecteur d’entrée et l’état caché avec leurs poids. Ensuite, il calcule la multiplication par élément entre la porte de réinitialisation et le multiple d’état précédemment masqué. Après avoir résumé les étapes ci-dessus, la fonction d’activation non linéaire est appliquée et la séquence suivante est générée.

Il s’agit du fonctionnement du GRU, les exemples pratiques sont inclus dans les cahiers.

# ****Quelle est la différence entre GRU et LSTM ?****

Les quelques points de différenciation sont les suivants:   
Le GRU a deux portes, LSTM a trois portes  
Gru ne possède pas de mémoire interne, ils n’ont pas de porte de sortie qui est présente dans LSTM Dans LSTM  
, la porte d’entrée et la porte cible sont couplées par une porte de mise à jour et dans la porte de réinitialisation GRU est appliquée directement à l’état caché précédent. Dans LSTM, la responsabilité de la porte de réinitialisation est assumée par les deux portes, c’est-à-dire l’entrée et la cible.

# Conclusion

Grâce à cet article, nous avons compris la différence fondamentale entre les unités RNN, LSTM et GRU. À partir du travail des deux couches, c’est-à-dire LSTM et GRU, GRU utilise moins de paramètres d’entraînement et utilise donc moins de mémoire et s’exécute plus rapidement que LSTM alors que LSTM est plus précis sur un jeu de données plus grand. On peut choisir LSTM si vous avez affaire à de grandes séquences et que la précision est concernée, GRU est utilisé lorsque vous avez moins de consommation de mémoire et que vous voulez des résultats plus rapides.