Le mécanisme d'attention

Simple astuce ou principe universel de l'IA ?

Pirmin Lemberger
Directeur Scientifique

9 janvier 2018

onepoint.
beyond the obvious

Résumé

Dans Lorsque nous regardons une scène ou une image pour comprendre ce qui s'y déroule, nous focalisons instinctivement notre attention sur certains objets, certaines personnes ou certaines actions car notre expérience nous a appris que c'est là que se trouve l'essentiel de l'information. Des systèmes de traduction automatique ou des systèmes de description d'images parviennent aujourd'hui à exploiter un simulacre de ce mécanisme pour améliorer considérablement leurs performances. On l'appelle le mécanisme d'attention (MA). Dans certains cas le MA apporte par ailleurs une aide appréciable à l'interprétabilité de modèles prédictifs jusque-là considérés comme des boites noires opaques. L'objectif de cet article pédagogique est de présenter comment fonctionne le MA dans deux contextes évoqués : traduction automatique et description d'image. Au-delà de ces deux cas particuliers et des gains de performances et d'interprétabilité nous examinerons brièvement l'hypothèse selon laquelle le MA pourrait jouer un rôle central en IA.

1. Les bons conseils de la nature

Aussi loin que l'on remonte dans l'histoire de l'IA on retrouve la même question : est-il judicieux d'imiter la nature ou alors est-il préférable de s'en détacher entièrement pour concevoir des solutions purement artificielles ? L'utilisation, aujourd'hui très populaire, de réseaux de neurones (RN) artificiels semble accréditer la première option à condition de ne pas être trop exigeant sur l'étroitesse de l'analogie. Le principe d'apprentissage le plus couramment utilisé aujourd'hui en IA, celui du **machine learning** (ML) **supervisé**, est en revanche assez éloigné de la manière dont les humains acquièrent des connaissances puisqu'il consiste à entraîner un algorithme au moyen de très nombreux exemples du phénomène qu'il s'agit « d'apprendre », un sujet qui était l'objet de ce précédent article.

Le sujet du présent article, le **mécanisme d'attention** (MA), peut clairement être assimilé à un mécanisme librement inspiré du fonctionnement de notre propre cortex cérébral. Lorsque nous analysons par exemple une image pour la décrire, notre attention se focalise instinctivement sur quelques zones que nous savons receler une information importante. Nous ne regardons pas chaque partie de l'image de la même acuité. Ce mécanisme s'apparente donc à un moyen d'**économiser des ressources** de traitement face à des données complexes à analyser. De manière similaire, lorsqu'un interprète traduit un texte d'une langue source dans une langue cible, il sait par expérience quels mots dans une phrase source sont associés à un certain terme dans la phrase traduite.

Figure 1 A droite : focalisation de l'attention sur la partie de l'image qui correspond au mot "frisbee".

Ce mécanisme d'attention fait désormais partie intégrante de la plupart des solutions d'analyse sémantique moderne et a été identifié par de nombreux experts en IA comme l'une des principales **tendances de la R&D en 2017** [4DLT]. Non content de booster les performances de plusieurs algorithmes de Deep Learning il offre par ailleurs un début de solution à l'épineux problème de l'interprétation du fonctionnement des RN réduits jusque-là à des simples boites noires insondables.

Dans les reste de cet article nous décrirons dans les grandes lignes deux architectures de RN qui exploitent un MA et ceci dans les deux contextes déjà évoqués : celui de la **traduction automatique** (section 3) et celui de la **description d'image** (section 4). Enfin, nous examinerons brièvement une idée audacieuse qui envisage le MA comme un **mécanisme plus général** à même de se substituer aux architectures existantes utilisées pour l'analyse sémantique complexe (section 5).

Dans le souci de rendre cet article largement accessible, la section 2, assez longue, décrit les principales intuitions derrières les briques élémentaires qui entrent dans la conception de ces systèmes (*word-embedding*, *FC–layer*, *softmax*, *RNN*, *LSTM*, *encoder–decoder*). Le lecteur familier avec ces concepts pourra passer directement à la section 3.

Les briques de base

Pour fixer les idées, considérons le problème de la traduction automatique (NMT = Neural Machine Translation). Si chaque mot d'un **dictionnaire** est numéroté, une phrase de n mots pourra être représentée par une suite $(x_1, x_2, \dots, x_n)$ de n entiers x_j (ou token) compris entre 1 et K avec $K \cong 100'000$ désignant la taille du vocabulaire dans une langue donnée.

Pour traduire une phrase, il faut par conséquent construire une fonction de traduction (très compliquée), nommons-la τ, qui à chaque phrase à traduire $(x_1, x_2, \dots, x_n)$ associe une liste de token $(x'_1, x'_2, \dots, x'_m) = \tau(x_1, x_2, \dots, x_n)$ d'une traduction possible avec, en général, $m \neq n$. La conversion inverse des token vers les mots de la langue cible utilise un second dictionnaire (cf. figure 2).

Notre objectif est de décrire dans les grandes lignes l'architecture d'un tel système, en d'autres termes comment on peut combiner quelques **briques de base** pour construire une telle fonction τ. Il faut envisager chacune de ces briques élémentaires comme une fonction compliquée avec un grand nombre paramètres ajustables. La fonction τ qui les combine (d'une manière que nous décrirons) dépend par conséquent de la réunion de tous ces paramètres. Pour trouver les valeurs optimales de ces paramètres, on utilise une approche classique par **ML supervisé** qui consiste à les ajuster pour minimiser[1] le taux d'erreur des phrases traduites par τ sur un corpus bilingue de référence.

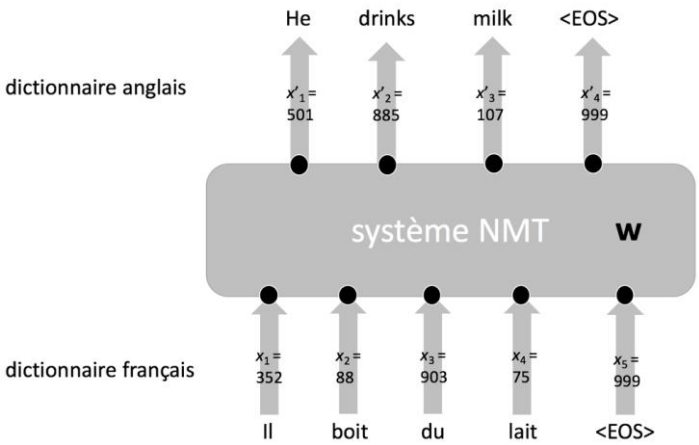

Figure 2 Un système de traduction automatique convertit une suite de token qui représente une phrase source en une autre suite de token qui représente la phrase traduite. Les paramètres **w** du système sont appris par ML supervisé.

Les Word-Embeddings – (WE)

La représentation d'un mot par un token comporte hélas l'inconvénient de ne pas rendre compte de la proximité sémantique entre deux termes. Nulle raison en effet pour que les entiers associés à « ville » et à « cité » soient proches.

Le rôle d'un module de **Word Embedding** (WE) consiste à associer à chaque mot d'une langue un vecteur $\mathbf{x}_j$ dans un espace de grande dimension, p.ex. d = 500, et ceci de telle manière que la proximité sémantique se reflète dans la distance (euclidienne) entre leurs vecteurs représentatifs $\mathbf{x}_1$ et $\mathbf{x}_2$. L'ensemble des K=100'000 vecteurs représentatifs d'une langue forme donc une gigantesque matrice dont les éléments sont les

[1] L'algorithme de recherche de ce minimum est standard pour tous des RN et ne nous concerne pas directement. C'est une descente de gradient stochastique avec un gradient sur les paramètres calculés par une version moderne de la rétropropagation comme le *Reverse-Mode Autodiff*.

paramètres ajustables **w** de ce module WE. Aussi surprenant que cela paraisse une telle association s'avère possible en pratique.

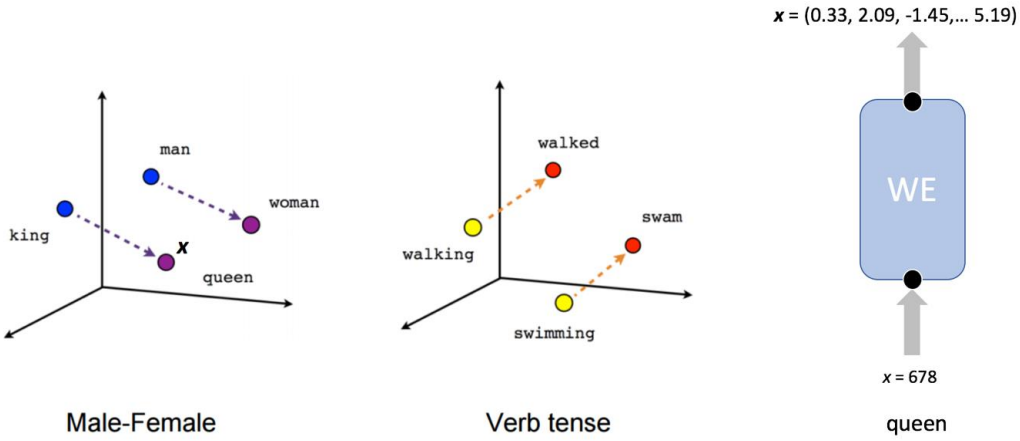

Figure 3 L'opération de word-embedding associe un vecteur (de grande dimension bien qu'ici d=3) à chaque mot. L'expérience montre que ces embedding manifestent d'intéressantes propriétés géométriques, source.

Il existe des matrices d'embedding prêtes à l'emploi dont les éléments de matrices ont déjà été pré-calculés. GloVe est un exemple. L'expérience montre par ailleurs que de telles immersions présentent des propriétés remarquables de parallélisme qui reflètent correctement des différences sémantiques ou grammaticale comme l'illustre la figure 3.

La couche complètement connectée – (FC)

Le module (ou couche) FC constitue une brique de base standard d'un RN dont on peut empiler plusieurs exemplaires pour créer des fonctions aussi complexes qu'on le souhaite. Peu importe ici la structure interne[2] d'un module FC, il suffit pour nous de savoir qu'il définit une fonction (avec des paramètres ajustables **w**) qui transforme un vecteur en entrée $\mathbf{x}=(x_1, x_2, \ldots, x_n)$ en un autre vecteur $\mathbf{h}=(h_1, h_2, \ldots, h_m)$ en sortie. Ce vecteur **h** pourra à son tour jouer le rôle de vecteur en entrée d'une seconde couche et ainsi de suite comme l'illustre la figure 4. Les vecteurs intermédiaires entre deux telles couches sont généralement appelées des **variables cachées**.

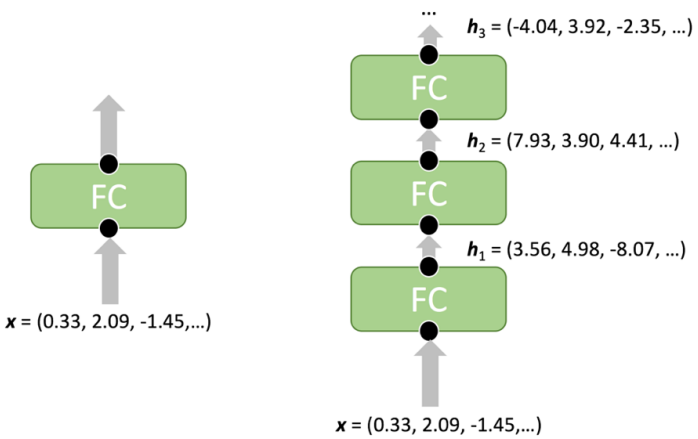

Figure 4 Un module FC et une pile de 3 module FC

[2] FC est l'abréviation de Fully Connected qui désigne un réseau de neurones complètement connectés qui combine une transformation linéaire et une non-linéarité définie par une fonction d'activation.

La couche softmax

Pour des raisons techniques[3], la fonction τ que nous cherchons à construire ne calculera pas directement la valeur de chaque token de la phrase à traduire. Elle calcule une **distribution de probabilité** sur les *K* mots du vocabulaire parmi lesquels on retiendra le plus probable. Ainsi pour une phrase donnée le token n° 357, correspondant au mot « *milk* », aura par exemple une probabilité de p_{357}=0.72 alors que le token n° 503, correspondant au mot « *wacky* », aura une probabilité de p_{503}=0.0001 etc…

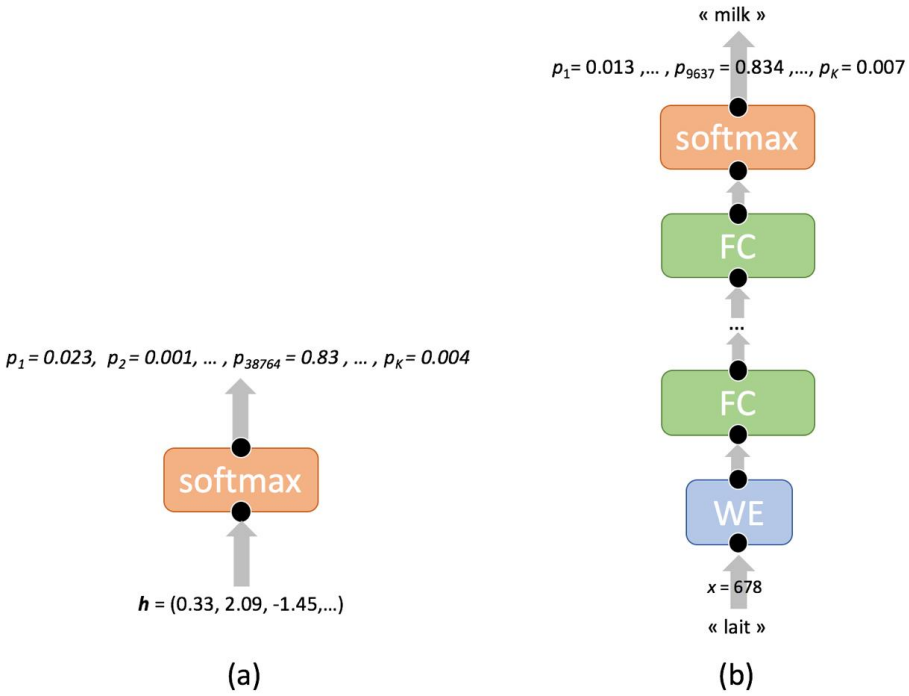

Figure 5 (a) Une couche softmax convertit un vecteur en une distribution de probabilité sur les K token d'un vocabulaire. (b) Elle constitue le dernier maillon d'un système de traduction mot à mot.

Il nous faut par conséquent une brique qui transforme le vecteur $\mathbf{h}_j$ en sortie du dernier module FC en une distribution de probabilité p_1, p_2, … p_K (dont la somme vaut 1) sur les entiers compris entre 1 et *K*. C'est ce que fait une couche **softmax** (figure 5 (a)) qui comporte, comme les autres, des paramètres ajustables qui seront « appris » lors de l'entraînement du modèle complet. Avec les 3 modules que nous venons de décrire nous pourrions déjà construire un système de **traduction** rudimentaire, fonctionnant **mot à mot** comme l'illustre la figure 5 (b).

Remarque sur l'entraînement du modèle

Lors de l'entraînement du modèle on lui présente des mots en entrée (et par la suite des phrases) ainsi que leur traduction correcte. Chaque traduction correcte d'un mot peut être envisagée comme une distribution qui attribue une probabilité 1 au mot correcte et 0 à tous les autres (one-hot encoding). Il s'agit par conséquent de comparer, pour chaque mot du vocabulaire source, cette distribution correcte à celle prédite par le système NMT. Entraîner le système revient dès lors à chercher des paramètres **w** tels que, en moyenne sur tous les mots du vocabulaire source, ces deux distributions soient aussi proches[4] que possible.

[3] La technique d'entraînement des RN utilisée aujourd'hui, la rétropropagation, exige que la sortie de la fonction τ soit différentiable.

[4] La bonne notion de proximité entre deux distributions de probabilité est la notion de cross-entropie.

Les réseaux récurrents – (RNN)

Le système de traduction mot à mot de la figure 5 (b) n'a évidemment pas grand intérêt pratique car on pourrait tout aussi bien lui substituer un dictionnaire bilingue ! Pour aller de l'avant il nous faut élaborer une structure capable de prendre en compte le **contexte** d'un mot. C'est ce que permettent les **réseaux** (de neurones) **récurrents** (RNN) comme l'illustre la figure 6 où RN désigne soit une couche FC, soit une couche LSTM que nous décrirons un peu plus loin.

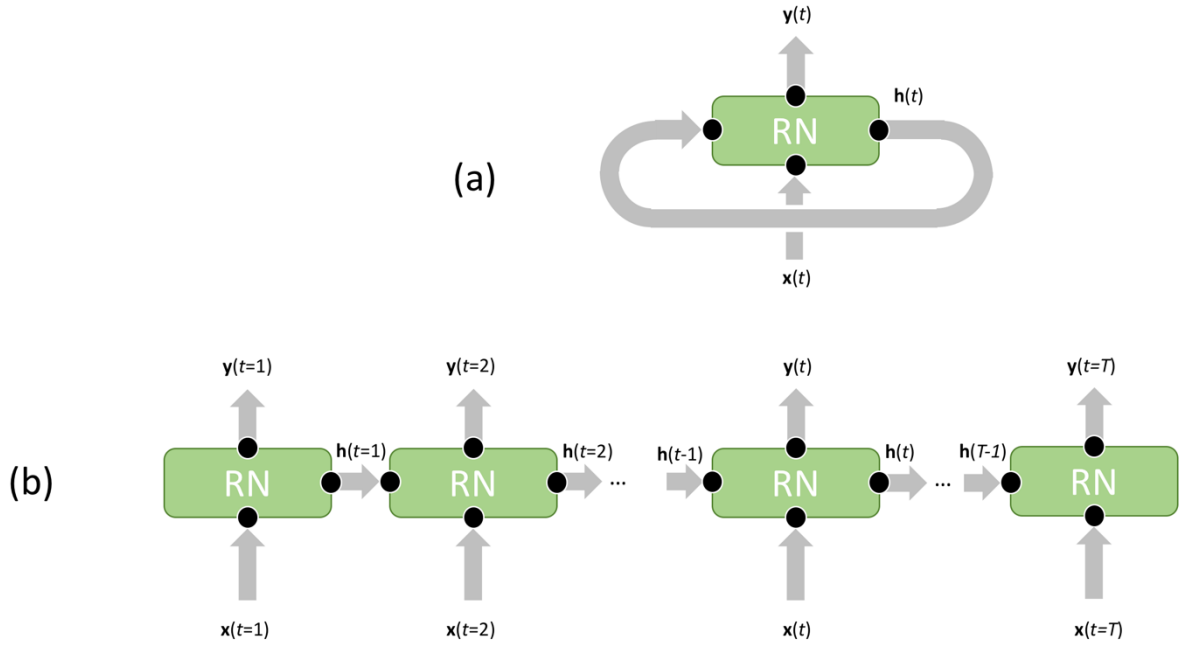

Figure 6 : (a) Dans un réseau récurrent où RN désigne par exemple une couche FC, la sortie **y**(*t*) *dépend à la fois de l'entrée* **x**(*t*) *et de variables cachées* **h**(*t*-1) *définies à l'instant précédent. (b) Le même réseau en vue développée dans le temps, chaque bloc RN est en fait le* ***même*** *bloc avec les* ***mêmes*** *paramètres qui agit à des instants successifs.*

Un RNN permet de transformer une **suite de vecteurs** de longueur *T* en une autre suite de vecteurs de même longueur. A chaque instant *t* un **même module**, nommé ici RN, est réutilisé. Le vecteur en sortie **y**(*t*) et les variables cachées **h**(*t*) sont des fonctions de l'entrée **x**(*t*) et des variables cachées **h**(*t*-1) à l'instant précédent.

Un RNN est donc une structure récurrente où la sortie **y**(*t*) *générée à l'instant t dépend non seulement de l'entrée* **x**(*t*) *à cet instant précis mais également de toute l'histoire passée* **x**(*t*-1), **x**(*t*-2),…, **x**(*0*). *C'est ce qui va permettre de prendre en compte le contexte qui précède un mot dans une phrase.*

Dans sa forme la plus simple le module RN est constitué d'un module FC dont les variables cachées **h**(*t*) et la sortie **y**(*t*) coïncident et dont l'entrée est formée par concaténation des deux vecteurs **x**(*t*) et **h**(*t*-1). En pratique le module RN est plus sophistiqué, comme nous le verrons au paragraphe suivant.

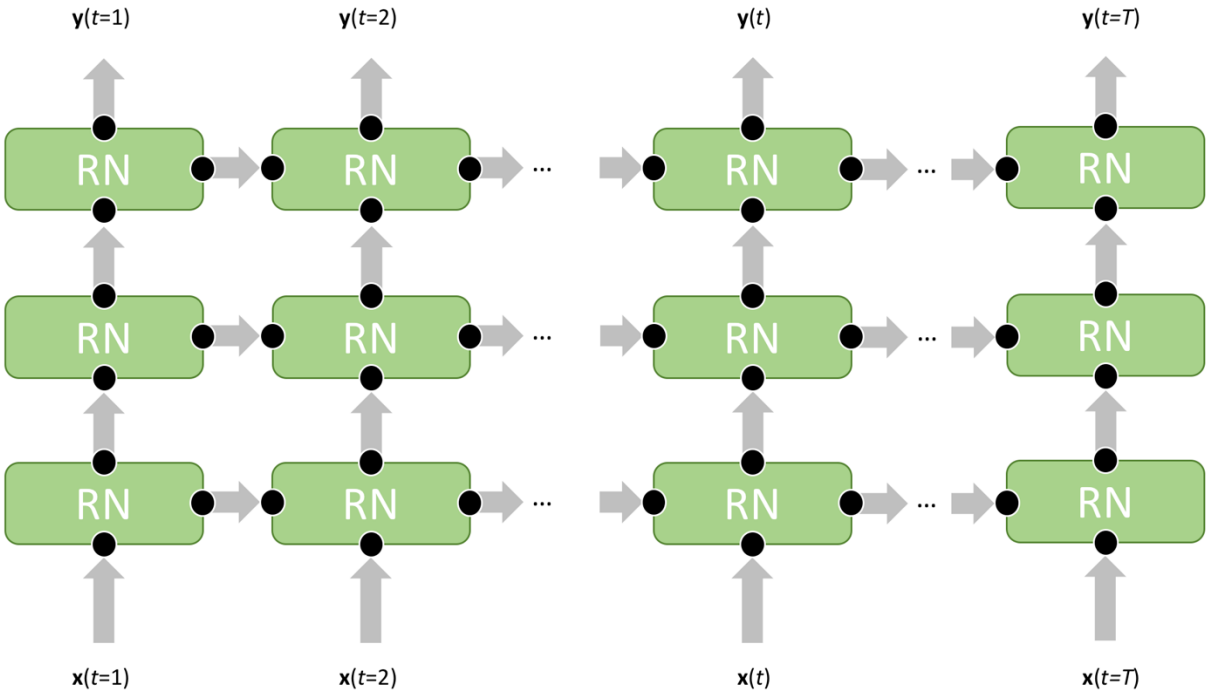

Figure 7 : Un empilement de RNN pour calculer des transformations complexes entre suites de vecteurs.

De la même manière qu'il est possible d'empiler des modules FC pour apprendre des fonctions complexes, on peut empiler des RNN pour apprendre des transformations complexes de suites de vecteurs, comme l'illustre la figure 7.

Les réseaux avec mémoire longue – (LSTM)

L'expérience montrent que le RNN de la figure 7 fonctionne mal lorsque RN est un module FC élémentaire. En fait le réseau a tendance à **perdre la mémoire** au sens où un événement **x**(*t'*) qui survient dans un passé lointain (*t'* << *t*) d'un événement **x**(*t*) n'a que très peu d'influence sur la sortie **y**(*t*) ce qui est évidemment un handicap pour la traduction de phrases longues par exemple.

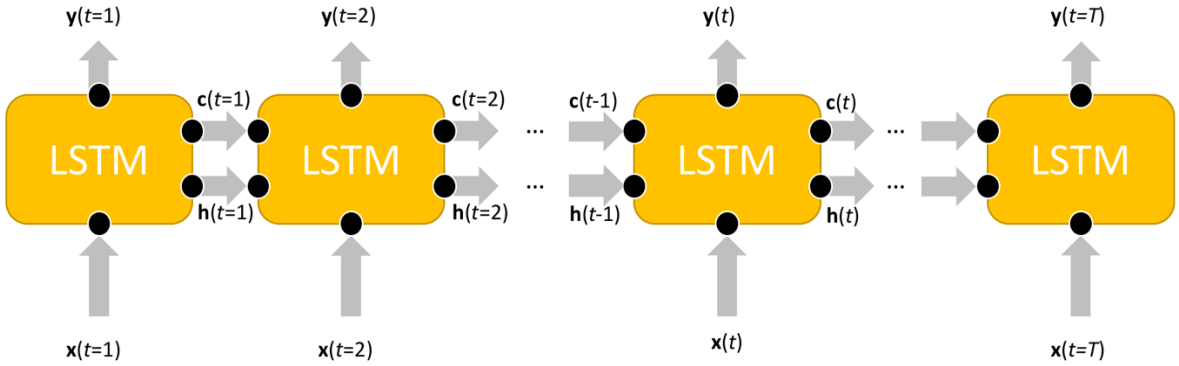

Figure 8 : Les LSTM cellules transmettent se transmettent de proche en proche deux types d'information. Une mémoire à court terme avec **h**(*t*) et une mémoire à long terme avec **c**(*t*).

Pour compenser ce phénomène les chercheurs ont conçus plusieurs cellules améliorées qui ne perdent *pas* la mémoire. La plus utilisée à ce jour est la cellule LSTM (Long Short Term Memory). En plus des informations passées d'un instant *t* à l'instant suivant *t*+1 via les variables cachées **h**(*t*), que l'on peut considérer comme une **mémoire à court terme**, ces cellules se transmettent également le contenu d'un registre mémoire **c**(*t*) que l'on peut envisager comme une forme de **mémoire à long terme**. La cellule LSTM contient des paramètres qui lui permettent d'apprendre (lors de l'entraînement du système) quelle partie de l'information dans **c**(*t*) il convient d'oublier, d'ajouter et de transmettre pour créer **c**(*t*+1). Les LSTM (ou leurs variantes) sont des éléments indispensables pour permettre l'analyse et la synthèse de longues suites de données.

Le schéma « encodeur-décodeur »

Une limitation importante des RNN tient au fait que la longueur des suites en sortie est toujours égale à la longueur des suites en entrée du système comme l'illustrent les figures 7 et 8. Une solution simple pour résoudre ce problème dans le cadre d'un système NMT pourrait consister à définir d'une part deux symboles <EOS> et <PAD>, désignant respectivement la fin d'une phrase et un symbole vide, et à choisir pour durée *T* celle de la plus longue des phrases en entrée ou en sortie. Le système complet est représenté sur la figure 9.

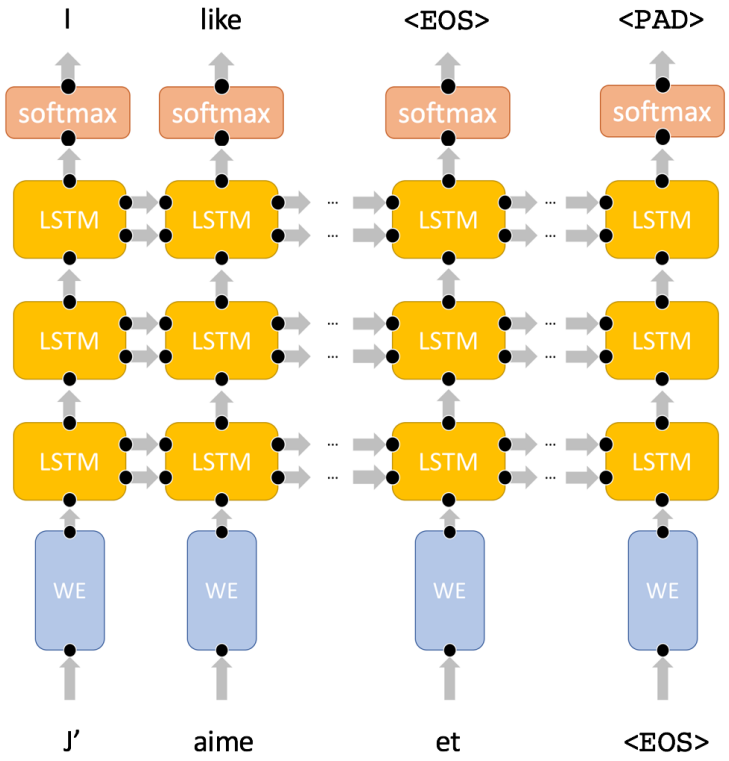

Figure 9 Un système NMT primitif qui exploite un RNN multicouche.

L'expérience a toutefois montré que cette architecture **fonctionne mal** car les mots en sortie du RNN restent trop corrélés aux mots en entrée au même instant *t*, conduisant à une traduction de piètre qualité.

Une meilleure solution pour traiter le problème de la longueur variables des phrases consiste à utiliser une **architecture** dite **d'encodeur-décodeur** telle qu'illustrée dans la figure 10. Une première section du RNN joue le rôle d'**encodeur**. Pour certaines langues il s'est avéré préférable d'inverser l'ordre des mots. Les sorties des LSTM ne sont pas utilisées et seule l'information contenue dans les variables cachées situées tout à droite

(flèche verte) est transmise au **décodeur**. Ces informations contiennent en principe le sens complet de la phrase à traduire dans un seul vecteur de grande taille ! L'encodeur a ensuite la charge de convertir ce vecteur en une suite de mots qui correspondent à la phrase traduite. Lors de l'entraînement on « aide » le modèle en lui présentant les mots de la phrase correctement traduite. En revanche, lors d'une traduction d'une nouvelle phrase chaque mot fourni en entrée du décodeur à l'instant *t* correspond au mot trouvé en sortie à l'instant précédent *t*-1.

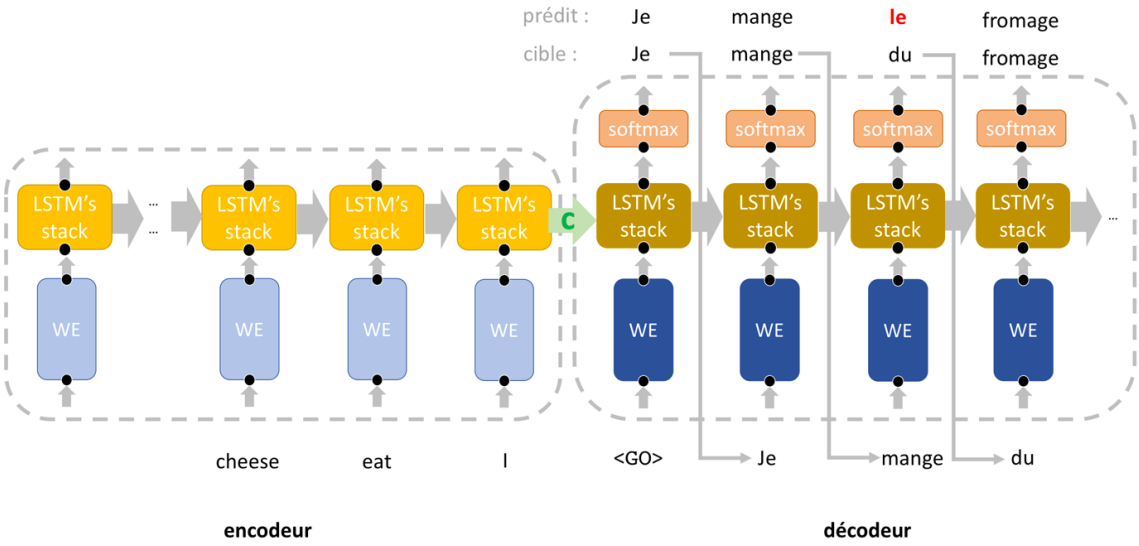

Figure 10 L'architecture encodeur-décodeur pour un système de NMT sans mécanisme d'attention. Le contenu du vecteur **c** *(en vert) de variable cachée en bout de chaîne de l'encodeur contient en principe tout le sens de la phrase.*

Les réseaux de convolution – (CNN)

L'analyse d'images au moyen de RN utilise un type d'architecture que l'on appelle les **réseaux de convolution** (CNN). Il est indispensable en l'occurrence de décrire brièvement comment sont interconnectés les neurones (à penser comme de simples variables numériques) qui constituent un tel CNN pour en comprendre le fonctionnement.

A chaque pixel de l'image on associe une variable en entrée du CNN (trois si l'image est en couleurs). Ces variables sont rangées dans une matrice à deux dimensions, plutôt que dans un vecteur, de sorte que la proximité entre pixels coïncide avec celle entre neurones. Une propriété que l'on attend de ces RN est qu'ils soient grossièrement insensibles à la position des structures qu'ils ont à détecter dans une image, une propriété que l'on qualifie d'**invariance par translation**.

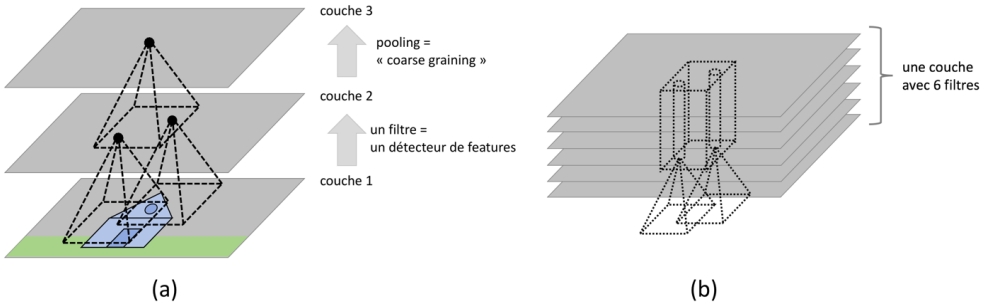

Figure 11 Structure d'un réseau de convolution. (a) un filtre de convolution entre les couches 1 et 2 et une opération de pooling entre les couches 2 et 3. (b) Une couche avec 6 filtres.

Pour réaliser cette invariance on exploite deux schémas d'interconnexion entre couches successives que l'on appelle respectivement les **filtres de convolution** et les opérations de **pooling** et qui sont représentées toutes deux dans la figure 11 (a).

Un filtre de convolution entre les couches n°1 et n°2 interconnecte selon un schéma où toutes les neurones situés dans un carré de la couche n°1 sont connectées au même neurone de la couche n°2, les poids des interconnexions étant reproduits à l'identique par translation d'un carré à un autre (qui se chevauchent). L'activation de chaque neurone de la couche n°2 est alors calculée[5] à partir des activités des neurones du carré de la couche n°1 auquel il est associé. Un tel filtre opère comme un **détecteur de structures**, ainsi pourrait-il par exemple détecter des lignes horizontales, des zones de contrastes etc. L'opération de pooling entre les couches n°2 et n°3 dans la figure 11 (a) est une opération de zoom-out : on partitionne la couche n°2 en carrés de 4 neurones (par exemple) et on ne transmet à la couche n°3 que l'activation du neurone le plus actif de chaque carré. La couche n°3 comportera ainsi quatre fois moins de neurones que la couche n°2. En pratique on utilise plusieurs filtres simultanément entre deux couches successives, si bien qu'une couche doit plutôt être envisagée comme un **mille-feuilles** comme l'illustre la figure 11 (b). En accumulant successivement des opérations de convolution et de pooling et en terminant par une série de modules FC et un softmax comme l'illustre la figure 12 on parvient, l'expérience le montre, à construire des systèmes capables par exemple de classer des images.

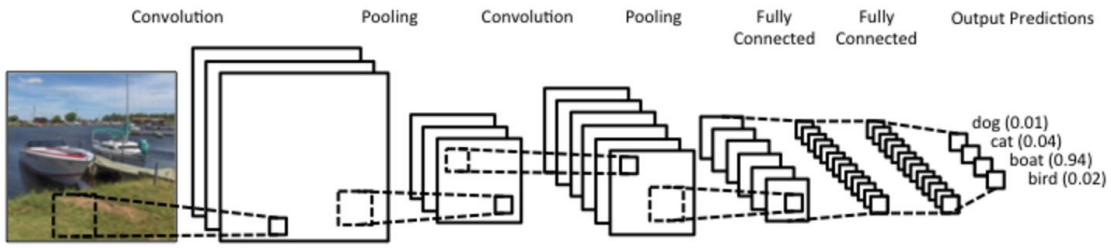

*Figure 12 : Un réseau de convolution fonctionnant en classifieur d'images empile une succession d'opération de convolution et de pooling puis se termine par quelques couches FC, **source**.*

[5] Plus précisément l'activité du neurone de la couche 2 s'obtient en calculant la somme pondérée des activités des neurones du carré de la couche 1 et en appliquant une fonction d'activation non-linéaire à ce résultat.

2. Le mécanisme d'attention pour la traduction automatique

Le système de traduction de base représenté sur la figure 10 repose sur l'hypothèse que le contenu sémantique de la phrase à traduire se laisse correctement encapsuler dans un seul vecteur de contexte **c** de dimension fixe en sortie de l'encodeur (cf. la flèche verte dans la figure 10). On peut concevoir un tel vecteur comme une forme de traduction en une **langue universelle** (que l'on pourrait baptiser comme étant du « **neuralais** » par exemple) intelligible par un RN décodeur fonctionnant comme un interprète. On conçoit donc que le sens d'une phrase longue soit plus difficile à encapsuler dans un tel vecteur **c**, qui constitue un **goulet d'étranglement**, que celui d'une phrase courte. L'expérience confirme que la qualité de la traduction se dégrade effectivement à mesure que la longueur des phrases augmente.

Plutôt que d'alimenter le décodeur avec un unique vecteur **c** de taille fixe, il serait par conséquent plus judicieux d'introduire un vecteur de contexte dynamique **c**(*t*), différent à chaque instant *t* du processus de décodage. L'idée est que le contenu d'un tel vecteur **c**(*t*) devrait correctement refléter le contexte sémantique des mots les plus pertinents dans la phrase source selon le mot à générer par le décodeur à l'instant *t*. C'est là précisément le cœur d'un **mécanisme d'attention** (MA).

Concrètement, un MA nécessite d'intercaler juste avant le softmax une couche (FC), en vert dans la figure 13, qui opèrera la fusion entre les informations de contexte **h**(*t*) locales au décodeur et le vecteur de contexte **c**(*t*) dynamique calculé à partir de la source.

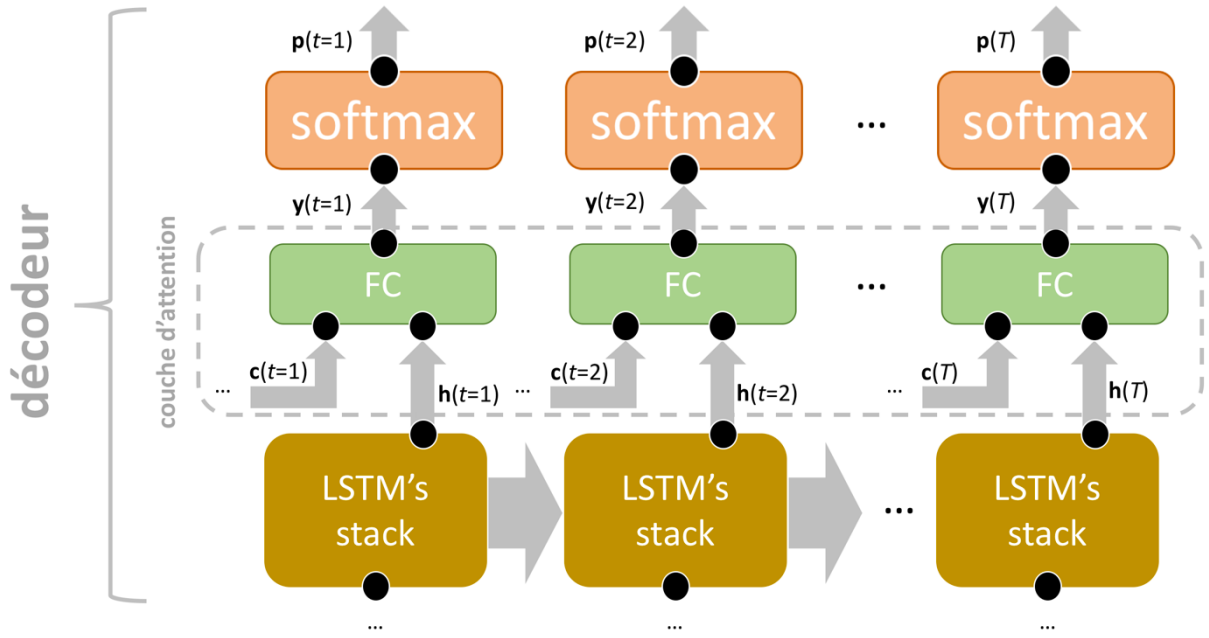

Figure 13 : Une couche d'attention intercalée entre la dernière couche de LSTM et les softmax permet de tenir compte à la fois du contexte **h**(*t*) *local au décodeur et d'un contexte* **c**(*t*) calculé à partir de la phrase source comme le décrit la figure 14.

Il existe de nombreuses variantes du MA qui calculent chacune différemment le vecteur de contexte **c**(*t*). La figure 14 illustre une variante à la fois simple et performante [LUO].

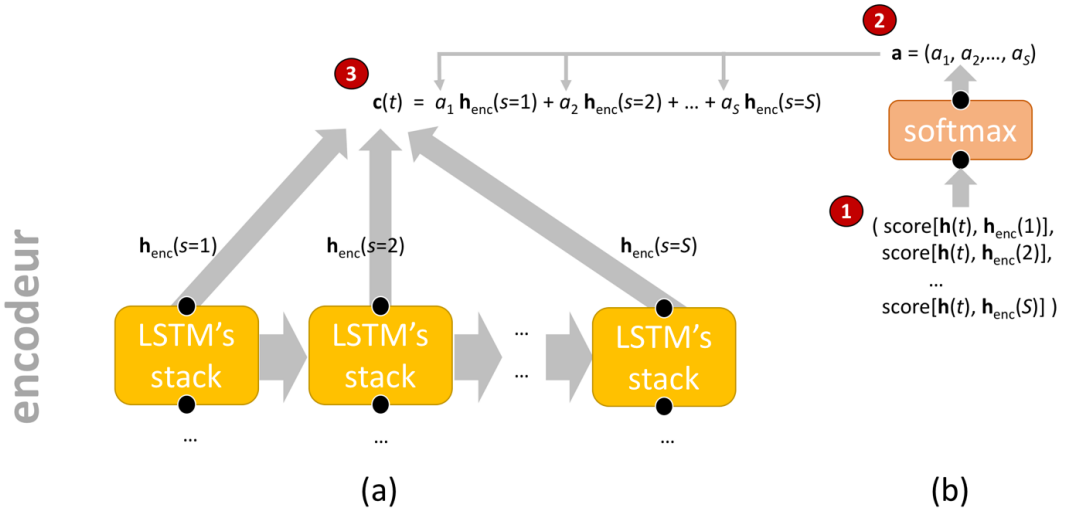

*Figure 14 : Le vecteur de contexte **c**(t) utilisé dans la figure 13 est une somme pondérée des S vecteurs de contexte côté encodeur, il est calculé en 3 étapes : (1) on calcule des scores de proximité entre le contexte **h**(t) coté décodeur et les S contextes $\mathbf{h}_{enc}(s)$ associés à chaque mot côté encodeur, (2) on transforme ces scores en probabilités a_s avec un softmax et (3) on utilise ces probabilités a_s pour calculer **c**(t) comme une somme pondérée des S contextes $\mathbf{h}_{enc}(s)$.*

Le vecteur de contexte $\mathbf{c}(t)$ est construit comme une somme pondérée de vecteurs de contexte $\mathbf{h}_{enc}(s)$ côté encodeur. Les pondérations a_s sont estimées à partir de scores (appris lors de l'entraînement) qui représentent la proximité sémantique entre le contexte cible $\mathbf{h}(t)$ et chacun des contextes sources $\mathbf{h}_{enc}(s)$. L'expérience montre qu'un simple produit scalaire $\mathbf{h}(t)\cdot\mathbf{h}_{enc}(s)$ peut très bien faire l'affaire !

Un mécanisme d'attention apprend donc à traduire séquentiellement une suite de mots et, simultanément, à aligner les mots cibles avec les mots sources les plus pertinents.

En résumé on peut dire que le NMT « sait » sur quels mots fixer son attention car il a appris à le faire sur des millions d'exemples en évaluant l'efficacité de chaque alignement essayé durant l'apprentissage. La figure 15 illustre la répartition de l'attention (les poids a_s) pour une traduction français-anglais.

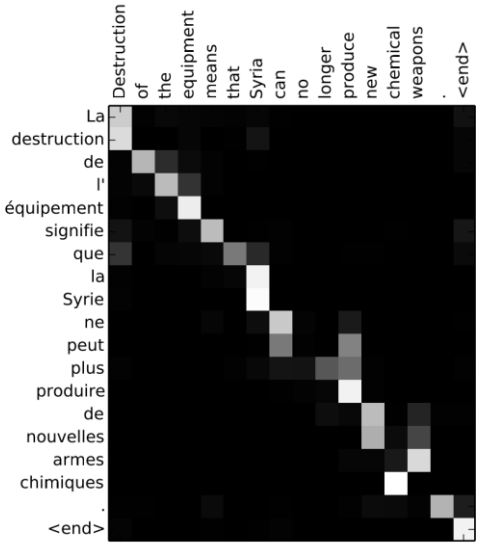

*Figure 15 Une représentation graphique des zones d'attention. Les carrés clairs sont ceux pour lesquels l'attention est importante c.à.d. que le poids a_j est proche de 1, **source**.*

3. Le mécanisme d'attention pour la description d'images

Nous savons désormais comment construire un système NMT capable de traduire une phrase d'une langue dans une autre, avec ou sans MA. On peut dès lors se poser la question, en apparence naïve, de savoir s'il est possible d'utiliser une méthode similaire pour construire un système capable de « traduire » une image en texte clair. Aussi surprenant que cela puisse paraître, la réponse est « oui » ! La figure 16 illustre le cœur d'un tel mécanisme dans une version *sans* MA.

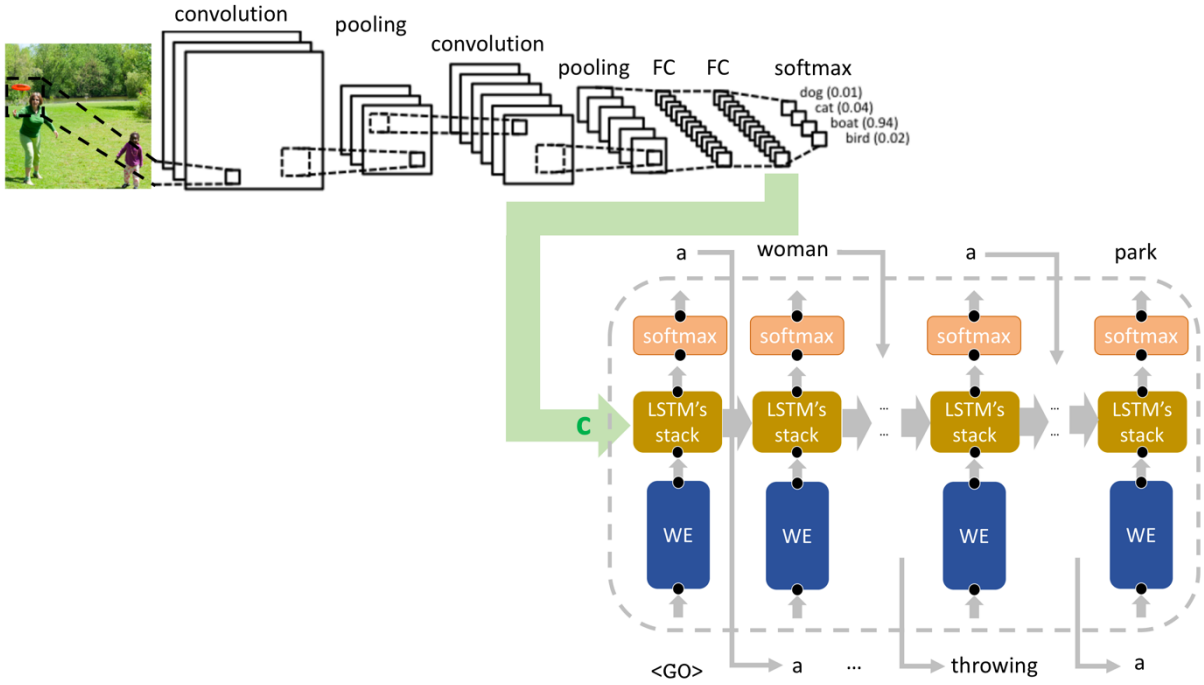

Figure 16 Un système de description d'image utilise comme encodeur un CNN dont on extrait une information descriptive au niveau de la dernière couche FC. Le décodeur est quant à lui similaire à celui utilisé dans un système NMT.

Le principe consiste à remplacer l'encodeur du NMT par un CNN capable d'encoder le sens d'une image dans un vecteur sémantique **c** de grande dimension avec lequel on alimente le décodeur. L'expérience prouve que ce dispositif fonctionne mieux que ce qu'il est raisonnablement permis d'espérer ! En pratique on utilise pour le CNN l'un des RN qui ont fait leur preuve dans l'analyse d'image, comme le Oxford VGGNet dont on récupère la sortie de la dernière couche FC (de dimension 4096).

Un tel système de description automatique d'image n'utilise aucun détecteur d'objet et est par conséquent à même d'identifier également des concepts et des actions.

L'utilisation du verbe « to throw » dans la scène du frisbee illustre bien ce dernier point.

Introduire un MA dans l'architecture de la figure 16 revient à permettre au système d'apprendre à focaliser dynamiquement son attention sur une partie de l'image en fonction du prochain mot à générer. Plutôt que d'alimenter directement le décodeur avec un seul vecteur **c** encodant toute l'image, on va générer plusieurs vecteurs $\mathbf{h}_j^{(conv)}$, associés à différentes parties *j* de l'image, en exploitant l'une des couches profondes du convolution du CNN. La 4ème couche de convolution du réseau VGGNet (qui en compte 5) comporte 14 x 14 = 192 cellules (correspondant à autant de parties *j* d'image) et associe à chacune d'elles un vecteur $\mathbf{h}_j^{(conv)}$ de

dimension 512. La figure 17 illustre la situation. Ces **vecteurs d'annotation** sont les équivalents des vecteurs de contexte $\mathbf{h}_{enc}(s)$ du système NMT de la figure 14.

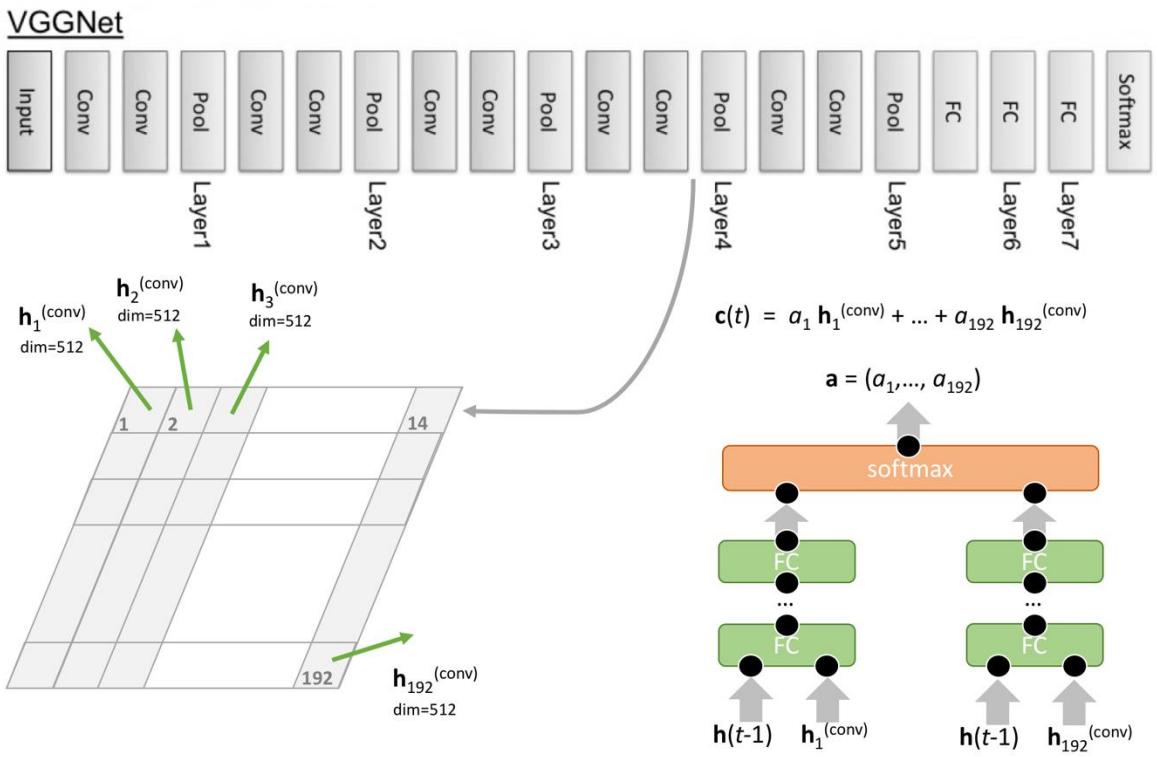

Figure 17 Les 14 x 14 = 196 vecteurs d'annotation $\mathbf{h}_j^{(conv)}$ de dimension 512 sont extraits à la sortie de la 4ème couche de convolution d'un réseau VGGNet. Voir le texte pour les détails.

Comme pour le système de NMT, un **vecteur de contexte** dynamique **c**(*t*) est construit pour indiquer au décodeur sur quelle partie de l'image il doit focaliser son attention à un instant *t*, cf. figure 13. Concrètement **c**(*t*) est construit comme une combinaison des 192 vecteurs d'annotations. Les poids a_j sont appris au moment de l'entraînement via plusieurs couches FC, pour évaluer la proximité entre le contexte sémantique **h**(*t*-1) côté décodeur et le contenu de la partie n°*j* de l'image encodé dans l'annotation $\mathbf{h}_j^{(conv)}$, cf. image 17.

La figure 18 illustre quelques descriptions correctes et les alignements correspondant entre les mots et une partie de l'image.

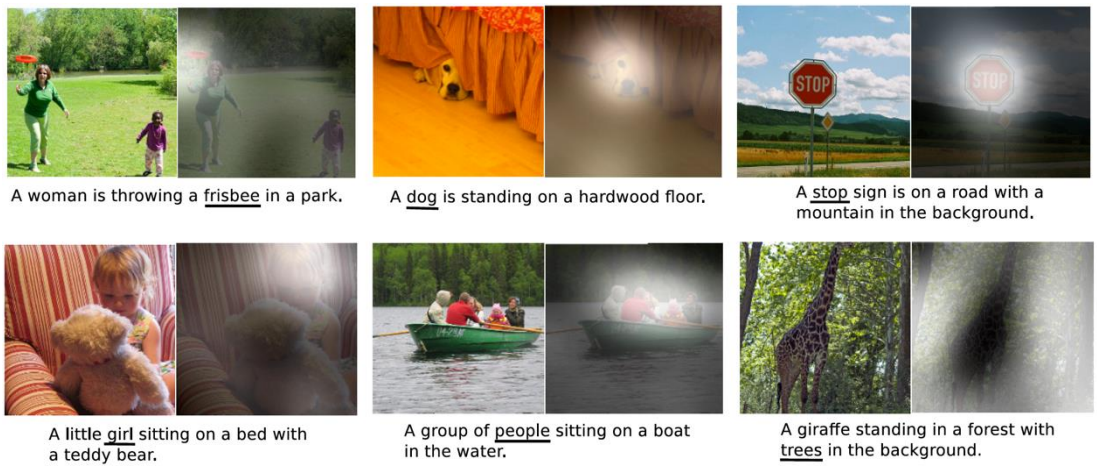

*Figure 18 Exemple de MA à l'œuvre dans un système de description automatique d'images, **source**.*

L'un des intérêts d'un MA pour un système de description automatique d'images est qu'il permet de comprendre l'origine de certaines confusions comme l'illustre la figure 19.

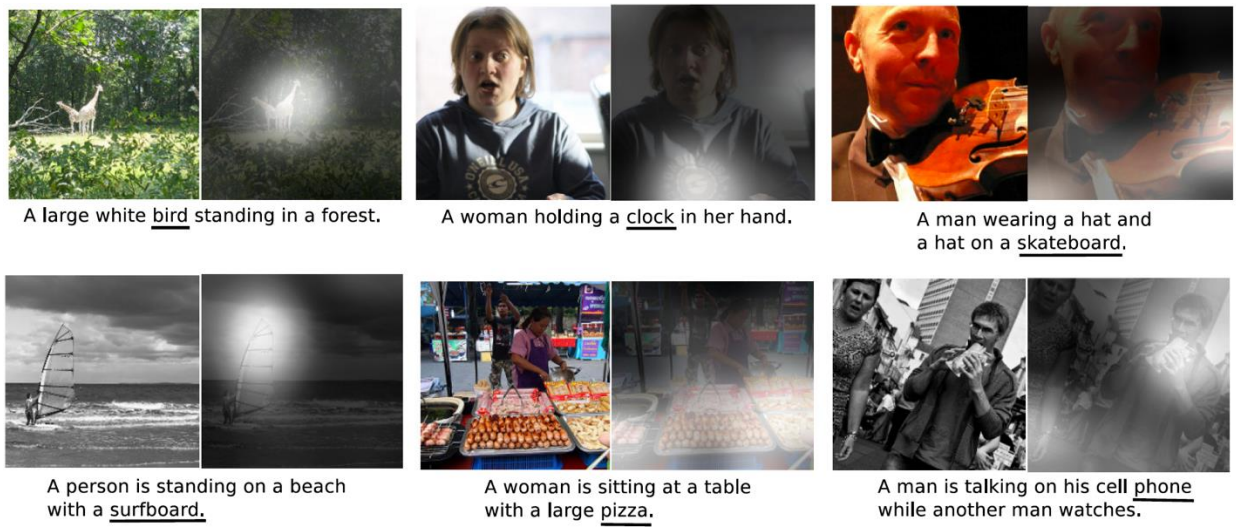

Figure 19 : Un mécanisme d'attention permet de comprendre l'origine de certaines erreurs d'interprétation du système, ***source****.*

4. Le mécanisme d'attention comme principe de base

Une définition générale

En nous reportant aux deux exemples précédents de MA, illustrés respectivement dans la figure 14 pour le système de traduction et dans la figure 17 pour la description automatique d'image, nous pouvons tenter de saisir ce qui fait l'**essence d'un MA**.

Dans un cas comme dans l'autre, il s'agit de construire dynamiquement une information contextuelle **c**(*t*) côté décodeur. Pour cela, on part d'une liste de contextes locaux côté encodeur, notons les $\mathbf{k}_i$, i=1,…,n (**k** pour « key ») et on évalue leur proximité avec un contexte **q**(*t*) (**q** pour « query ») à l'instant *t* côté décodeur au moyen d'un score de compatibilité score($\mathbf{k}_i$, **q**(*t*)).

La fonction de score est plus ou moins complexe selon les cas. Pour le système NMT de la figure 14 un simple produit scalaire $\mathbf{k}_i \cdot \mathbf{q}(t)$ fait l'affaire. Le système de description d'image de la figure 17 utilise quant à lui un score plus complexe définit par plusieurs couches FC.

Le contexte dynamique **c**(*t*) côté décodeur est alors construit comme une somme pondérée de certains vecteurs $\mathbf{v}_i$ (**v** pour « value ») où les pondérations $a_i(t)$ sont les scores $a_i(t)$ = score($\mathbf{k}_i$, **q**(t)). En clair : $\mathbf{c}(t) = a_1(t)\,\mathbf{v}_1 + \ldots + a_n(t)\,\mathbf{v}_n$.

Dans les deux exemples que nous avons examiné les vecteurs $\mathbf{v}_i$ et contextes $\mathbf{k}_i$ étaient en fait identiques mais il s'avère utile [AYN] d'envisager un mécanisme un peu plus général où les vecteurs $\mathbf{v}_i$ et les contextes locaux $\mathbf{k}_i$ pourraient être différents.

Un MA construit un vecteur de contexte c(t) construit à partir d'une requête q(t), d'une liste de couples clé-valeur [(k1, v1), … , (kn, vn)] et de scores de compatibilité entre une clé ki et une requête q(t).

Eviter la récurrence

Les RNN (cf. figure 6) sont très efficaces pour le traitement ou la génération de suites de données mais ce caractère séquentiel empêche hélas de les faire passer à l'échelle en les parallélisant. Récemment, une équipe de chercheurs [AYN, TRA] a mis au point une nouvelle architecture de NMT appelée le **Transformer** qui exploite simultanément plusieurs MA en tout en évitant le caractère récurrent des RN utilisés jusque-là. L'importance de ce travail qui a contribué à définir un nouvel état de l'art en NMT et son titre, « *Attention is All You Need* », particulièrement bien trouvé, lui ont assuré une grande visibilité dans la communauté du Deep Learning durant l'été 2017.

Nous n'entrerons pas dans les détails assez complexes de l'architecture du Transformer mais remarquons qu'une autre motivation de cette équipe était de pouvoir traduire correctement des phrases du type :

The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigué.

The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.

Figure 20 Deux phrases qui nécessitent de lever une ambiguïté pronominal avant traduction.

Une traduction correcte exige en l'occurrence de lever l'ambiguïté liée au pronom « it » dont on doit au préalable déterminer s'il se réfère à l'animal ou à la route. Dans le Transformer un MA de **self-attention** identifie des corrélations au sein des mots d'une même phrase et permet de résoudre cet épineux problème. Pour l'instant ce problème est encore mal géré par Google Translate mais il est correctement pris en compte par DeepL ! La figure 21 illustre les corrélations intra-phrase découvertes par le MA de self-attention.

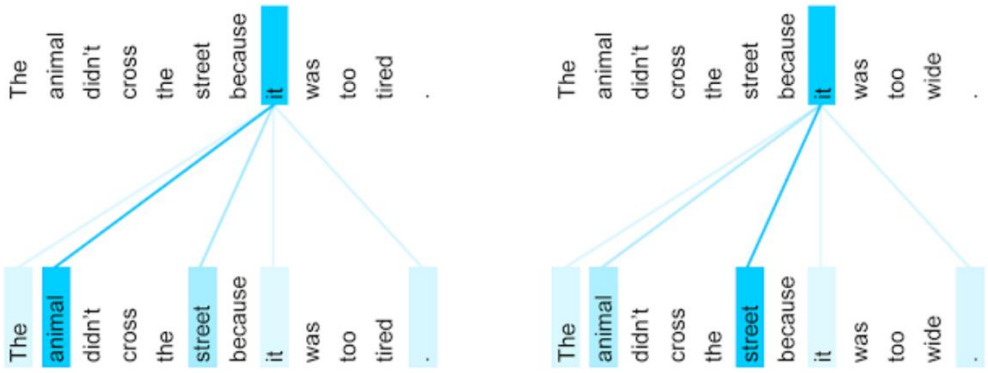

*Figure 21 Corrélations entre le pronom "it" et les autres mots de la phrase détecté par l'architecture du Transformer, **source**.*

Ce succès du MA face à l'un des problèmes difficiles de la traduction automatique pourrait indiquer que l'idée d'un MA est plus qu'un simple artifice de performance mais un principe plus fondamental utile dans beaucoup d'applications de l'IA qui impliquent des suites de symboles, de sons ou de mots.

5. Conclusion

Le recours à un MA était originalement motivé, rappelons-le, par le souhait de surmonter le goulet d'étranglement que constitue un vecteur sémantique **c** de taille fixe transmis par un encodeur à un décodeur (cf. figure 10). Un MA permet de construire un vecteur de contexte dynamique **c**(*t*) en fonction du mot, ou plus généralement d'un élément, à générer à l'instant *t*. Un tel vecteur **c**(*t*) contient une information beaucoup plus pertinente qu'un vecteur statique **c** ce qui se concrétise par des phrases générées plus fluides.

Le MA apporte en outre plusieurs de **bonus** appréciables :

- Il permet de traduire des **phrases plus longues** que ne pourrait le faire un système NMT dépourvu de MA.
- Le modèle devient partiellement **interprétable** car le MA aide à localiser quelles parties de la source (texte ou image) contribuent à quelles parties de la phrase générée, comme l'illustrent les figures 18, 19 et 21.
- A terme, le MA pourrait s'avérer d'une **utilité plus générale** que celle envisagée initialement.

A PROPOS DES PERFORMANCES D'UN MA

Si l'intuition associe un MA avec une forme de parcimonie dans le traitement de l'information, il faut bien prendre conscience qu'en réalité, l'entraînement d'un modèle avec MA, sera plus coûteux en ressources que le même modèle dépourvu de MA. Il s'agit en effet d'évaluer tous les scores de compatibilité entre une requête **q**(*t*) et tous les contextes locaux $\mathbf{k}_i$. Comme pour les humains, **il en coute au système d'apprendre à focaliser correctement son attention**. Cependant, une fois cette aptitude acquise, la génération des phrases s'en trouve améliorée. Certaines variantes du MA sont toutefois conçues pour passer à l'échelle :

- Dans [LUO] on définit un MA qui calcule une fenêtre de contexte de taille fixe autour d'une position centrale que le système apprend à prédire.
- Dans [AYN] plusieurs MA sont utilisés en parallèles mais les dimensions des requêtes **q**(t), des clés $\mathbf{k}_i$ sont ajustées de manière à ce que le nombre d'opérations soit indépendant de la taille des phrases.

UN PRINCIPE GENERAL PLUTOT QU'UN MODELE DE CONCEPTION

Le MA n'est **pas un pattern de conception** au sens strict du terme mais **plutôt un principe général** qui s'est avéré très bénéfique pour certains modèles où il s'agit d'apprendre à générer des phrases ou, plus généralement, des suites d'éléments. Toutefois, dans certaines applications comme les systèmes de résumé automatique de documents, le MA s'est avéré plutôt décevant [4DLT]. Par conséquent il est sans doute prématuré pour affirmer que le MA et le NMT constitue une panacée incontournable de l'IA. Peut-être aussi ne faut-il pas jeter aux orties trop rapidement des décennies de R&D en analyse sémantique classique qui ont précédé l'ère glorieuse du Deep Learning et envisager plutôt la construction de systèmes hybrides qui concilient le meilleur des anciennes et des nouvelles approches.

Références

- [NIC] **Neural Machine Translation by Jointly Learning to Align and Translate**, *D. Bahdanau, K. Cho, Y. Bengio*, arXiv preprint 2016.
- [LUO] **Neural Machine Translation**, *Minh-Thang Luong*, PhD Dissertation Stanford University 2016.
- [4DLT] **Four deep learning trends from ACL 2017**, *Abigail See*, a Deep Learning and NLP blog.
- [SAT] **Show, Attend and Tell: Neural Image Caption Generation with Visual Attention**, *K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, Y. Bengio*, arXiv preprint 2016.
- [AYN] **Attention Is All You Need**, *A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin*, arXiv preprint 2017.
- [TRA] **Transformer: A Novel Neural Network Architecture for Language Understanding**, *J. Uszkoreit*, Google Research Blog.
- [IVA] **Interpretability via attentional and memory-based interfaces, using TensorFlow**, *G. Mohandas*, O'Reilly AI 2017.